User-Centred Artificial Intelligence for Game Design and Development with GAGeTx: Graphical Asset Generation and Transformation

A Thesis Submitted for
the Degree of Doctor of Philosophy

By

Kaisei FUKAYA
Brunel Design School, Brunel University London

January 29, 2025

Abstract

In an increasingly digitalised world, visual media is utilised in a wide array of forms. This visual content is made up of many individual elements, referred to as graphical assets. A wide variety of well established and nascent methods, referred to as graphical asset generators (GAGs), can be used to automate the production of graphical assets. Video games are a popular and growing application of graphical assets, requiring copious amounts of 3D and 2D visual content. The aim of this thesis is to examine how generative methods can be applied to the creation of graphical assets for games, and to discover how game designers and developers choose to utilise them. This is achieved through the pursuit of 5 research objectives: first, collating and examining the state-of-the-art of GAGs in the literature; second, developing a framework for using, implementing and evaluating GAGs (GAGeTx); third, obtaining user needs and preferences through a user experiment; fourth, developing a proof-of-concept prototype tool, serving to validate GAGeTx; fifth, refining the framework through further user experimentation using the prototype tool. Contributions of this thesis include: the GAGeTx framework; a systematic literature review state-of-the-art GAG methods; empirical findings on user needs and requirements; a novel, game engine-integrated framework and prototype tool for sword generation; and a method for dataset creation, tailored to unsupervised deep learning for GAG tasks.

The GAGeTx framework offers a comprehensive categorisation and conceptualisation of GAG methods, which allows researchers and practitioners to identify or create the most appropriate GAG methods given their needs and requirements, through a step-wise process built on empirical findings. This is supported by the integration of GAG evaluation metrics and the consideration of user pipeline applications. The systematic literature review consolidates fragmented research from various domains into a unified taxonomy, identifying key aspects of GAGs. This facilitates cross-over between domains and provides a valuable entry point for both new researchers and practitioners in the field of GAG research. Empirical findings from user studies provide guidelines for integrating GAG tools into game design and production pipelines with minimal friction and facilitating the adaptation of GAG research into practical tools. In addition, they identify the appropriate metrics for evaluating the strength and utility of GAG tools based on their technique, further aiding in the benchmarking and improvement of GAG methods. The prototype, named Swordgen, allows users to generate varied sword assets for games via several generative techniques at different levels of user initiative, providing and validating a configurable and extendable framework for game-engine integrated GAG tools. The dataset creation method enables the creation of bespoke content and style specific datasets for training unsupervised deep-learning-based GAGs. Through selecting specific data sources, including concept-art from the user's current project, users can control the design constraints of a GAG model, without compromising on dataset size.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr Damon Daylamani-Zad for his valuable guidance and advice throughout my research, and for always being available to provide feedback and encouragement. Our consistent meetings kept me grounded and on track. In addition, I would be remiss to not highlight his enduring patience in helping to design and format the copious diagrams and tables throughout this work, for which I am also grateful. I would also like to express gratitude for the help and guidance of Dr Harry Agius, my second supervisor, and for his editorial expertise.

I would like to thank both of my supervisors for providing many opportunities to gain additional professional and research experience during my doctoral studies. Furthermore, I am greatly appreciative of the user study participants, who put aside their time to test and provide valuable data and feedback. Most of all, I am grateful to my parents for their endless and unconditional love, support and encouragement throughout my academic journey.

Publications

Fukaya, K., Daylamani-Zad, D. and Agius, H., 2024. Evaluation metrics for intelligent generation of graphical game assets: a systematic survey-based framework. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Bishop, D.T., Daylamani-Zad, D., Dkaidek, T.S., **Fukaya, K.** and Broadbent, D.P., 2023. A brief gamified immersive intervention to improve 11–14-year-olds' cycling-related looking behaviour and situation awareness: A school-based pilot study. Transportation Research Part F: Traffic Psychology and Behaviour, 97, pp.17-30.

Under review

Fukaya, K., Daylamani-Zad, D. and Agius, H., Intelligent Generation of Graphical Game Assets: A Conceptual Framework and Systematic Review of the State of the Art. ACM Computing Surveys.

Fukaya, K., Daylamani-Zad, D. and Agius, H., Generative Tools for Graphical Assets: A Study on Game Designers' and Developers' Preferences. Decision Support Systems

Under preparation

Fukaya, K., Daylamani-Zad, D. and Agius, H., GAGeTx: A framework and integrated prototype for graphical asset generation. IEEE Transactions on Human-Machine Systems

Contents

List of Figures

List of Tables

Chapter 1: Introduction

Throughout history, humans have used graphics to represent the world around them, communicate, plan and share stories. In an increasingly virtual world, digital graphics have become a common and expected part of daily life. Many industries that have traditionally relied on hand-drawn graphics, such as architecture and engineering, have adopted digital methods such as building information modeling (BIM) and computer aided design (CAD). This digitalisation streamlines design tasks, and allows for fast iteration, collaboration and sharing. In addition, the ability to incorporate algorithms elevates digital graphics methods far beyond what is possible with hand-drawn methods. This only improves more as the processing capability of computer hardware advances.

One of many applications of digital graphics is in video games. The video games industry is the largest entertainment industry globally, expected to be worth $363.20bn USD by 2027 [447] and by 2022 was worth more than the film an music markets combined [445]. As an audiovisual form of content, video games can require a large amount of graphical data in order to represent worlds, characters, objects, and graphical user interface (GUI) elements. This graphical data takes a combination of artistic skill and technical knowledge to produce and requires careful consideration for data storage and performance. These items, commonly referred to as *assets*, form the building blocks of video game content. Typically, graphical assets are either created by artists and designers during the production stage of game development, or purchased from a third party. The former case can take a large amount of time, while the latter case, though convenient, can be stylistically limiting.

Furthermore, the impact of games technology beyond entertainment is far-reaching. Gamification and serious games present affective methods for training and increasing the awareness of individuals [78, 441]. These methods utilise the immersive and engaging qualities of games to enhance the experience of learning and improve confidence. For example, immersive gamified cycling is shown to improve the on-road cycling of children as well as their self-reported confidence [30].

One way to utilise the computing power afforded by a digital workflow is to automate asset creation through procedural content generation (PCG). PCG is a class of methods and algorithms that output or *generate* content, whether it be game levels, loot or characters. These methods take many forms and can incorporate randomness as well as modifying existing content, they can be constrained by parameters or used as a form of compression [109]. A range of games use PCG as a way to produce new content during gameplay, creating unique experiences for players and endless replay-ability. Alternatively, there are many software tools that utilise PCG algorithms such as speedtree [190] and houdini [421]. These systems illustrate the potential of PCG usage within design and development pipelines.

While large development teams can put aside the necessary resources to create the many graphical assets required for detailed game worlds, smaller teams with lower budgets are forced to limit their scope to match their capabilities. Graphical asset generation has the potential to streamline the asset creation process, reduce the resources required, and democratise game development for those without digital drawing, painting and 3D modelling skill sets.

On top of more traditional and well established methods for graphical PCG, such as grammars [85, 10, 514] and evolutionary algorithms [153, 245], there has been a surge in deep-learning approaches [493, 416, 545, 126]. The current widespread interest in generative artificial intelligence (AI), utilising deep neural network architectures, has given rise to the usage of such methods in design and development tools. With new and emerging commercial technology, such as Muse for Unity [472] and Adobe Firefly [5], there is a clear and valid interest in the use of generative tools to help with creative graphical processes and pipelines.

PCG for games has been thoroughly examined throughout the years, such as in the work of [465, 146, 164], and with further focus on machine learning [294, 450, 225]. Much of the existing literature focuses on high level game elements, such as level structure and narratives, rather than low level elements, such as graphical assets. While graphical assets have a lesser direct impact on game-play, they are a necessary element of any game, serving to visually represent the game state and provide immersion. As such they can be required in copious amounts, demanding a large portion of production time and thus incurring a large cost for game developers. Due to being a low-level element, graphical assets can be applied broadly across many fields, and while the purpose of their creation may differ, the methods used are transferable. For example, a technique for generating buildings in architecture can also be applied to creating buildings in games.

So far, there is no overarching research that consolidates the available methods for these purposes. A centralised understanding of graphical asset generation methods would benefit researchers advancing the generative frontier, companies wishing to provide creative tooling, and individual practitioners seeking to streamline their workflows. Furthermore, while video games are a large part of current society, the plethora of other industries that are augmented by or reliant on digital graphics all benefit from the speed, convenience and utility of generative algorithms.

1.1 Research aim and objectives

The aim of this research is to examine how generative algorithms and PCG can be applied to the creation of graphical assets for games, and to discover how game designers and developers choose to utilise such methods as tools. To achieve this aim, the following objectives are undertaken:

Objective 1: Conduct a comprehensive systematic literature review of the state-of-the-art in methods for generating graphical assets. This will serve as a means to collate the breadth of generative methods used.

Objective 2: Develop a framework for researchers and practitioners on how to use, implement and evaluate graphical asset generators (GAGs).

Objective 3: Establish user requirements and preferences for asset generation systems by conducting a user study with game designers and developers.

Objective 4: Design, develop and integrate a functional proof-of-concept prototype tool that emphasises usage in game design and development pipelines.

Objective 5: Validate the design in objective 4 and refine GAGeTx. Pursued via a user study with game designers and developers.

Objective	Method	Outcome	Duration
Objective 1	Systematic literature review based on PRISMA.	A systematic review of state-of-the-art GAG literature	Oct 2021 - April 2022
Objective 2	Inductive content analysis.	GAGeTx and metrics framework	April 2022 - June 2022
Objective 3	Empirical research, statistical analysis, thematic analysis	User study 1: UX preferences	Jun 2022 - Dec 2022
Objective 4	Rapid application development (RAD)	Functional proof-of-concept prototype.	Dec 2022 - Aug 2023
Objective 5	Empirical research, statistical analysis, thematic analysis	User study 2: Generative technique preferences, Refined GAGeTx framework	Aug 2023 - Aug 2024

TABLE 1.1: Summary of thesis Objectives, Methods, Deliverables and their Duration.

1.2 Research approach

To achieve objective 1, a *Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) based literature search will be conducted in order to collect and analyse research papers from four major databases: ACM Digital Library, IEEE Xplore, ScienceDirect and Springer. The PRISMA protocol [350] is* a set of guidelines for presenting systematic literature reviews. The PRISMA guideline requires that the number of papers identified, screened then assessed be reported along with the final number of included studies. This allows for clarity and reproducibility in the procedures taken in the systematic search. This literature review will yield a pool of papers pertaining to methods for generating graphical assets. While the main body of analysis is to take place between October 2021 and April 2022, additional top-up searches following the same procedure will take place up until August 2024.

In order to achieve objective 2, an inductive thematic analysis (ITA) of the literature obtained through objective 1 was conducted. ITA allows themes to emerge from a body of qualitative data through iterative coding and conceptual grouping [154]. This can be applied to the analysis of literature by coding the described approaches and methods and grouping themes across the literature pool. Through conducting ITA on the literature from objective 1, a classification of the main aspects of graphical asset generators will be derived and a taxonomy of the various categories will be presented. From this a framework for graphical asset generators will be formed.

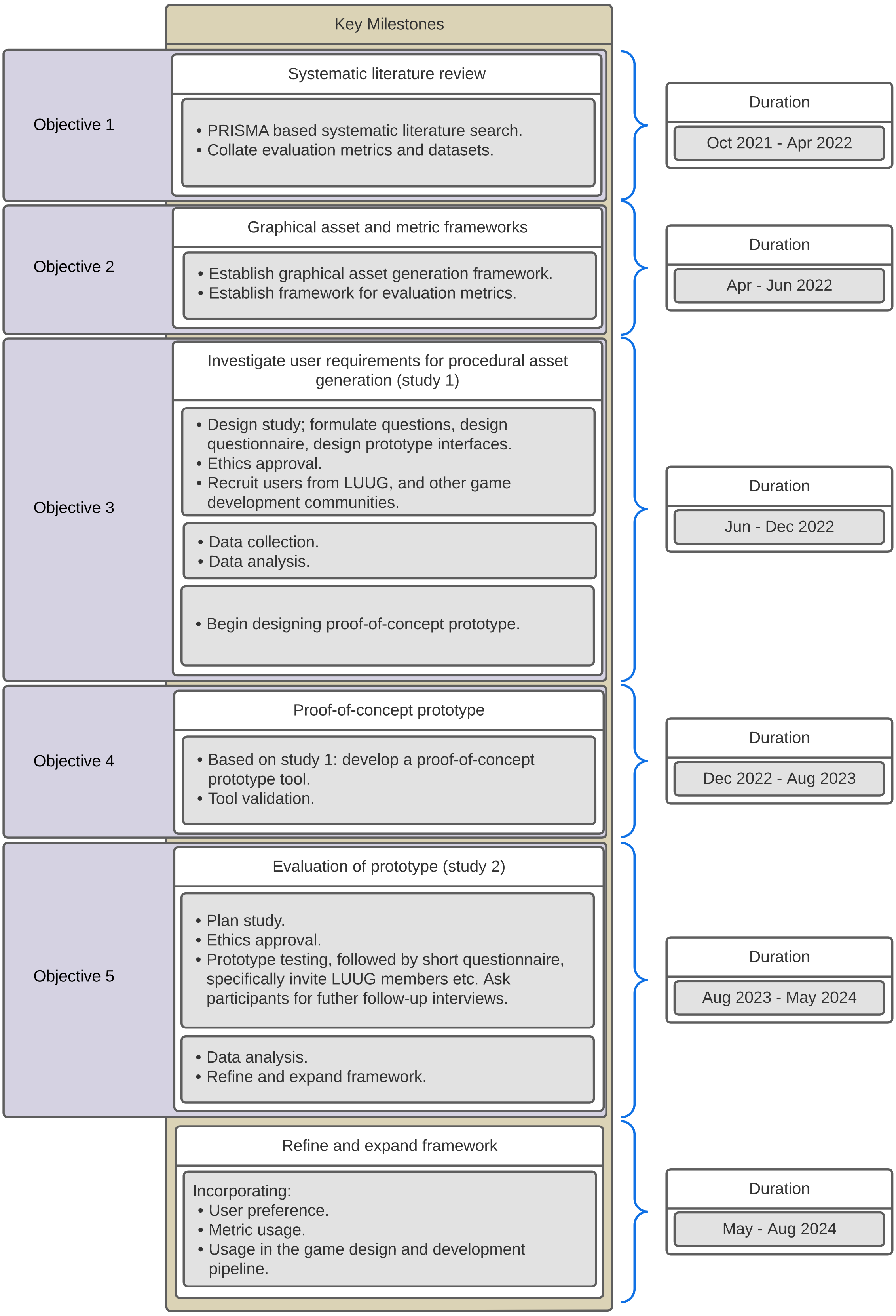
Objective 3 will be achieved via a mixed methods approach [448] examining quantitative data through statistical analysis and qualitative data through thematic analysis. The findings of objective 2 will be used to develop three mock up user interfaces, each representing a different style of interaction, these will be: stand-alone wizard, integrated editor window and integrated editor inspector. These interfaces will implement the key steps and categories of the literature derived framework of objective 2, covering common forms of UI type, spanning stand-alone and engine integrated UI. Via convenience sampling, game designers and developers will be recruited to test each interface and complete a repeated measures questionnaire followed by an optional semi-structured interview. Questionnaire results pertaining to the three mock ups will be analysed via ANOVA and Wilcoxon signed ranks tests to determine significant differences in preferences, and tested for demographic impact via regression. Participants that opt-in for the semi-structured interview will be asked questions aimed at expanding on and extracting nuance with regard to preferences and needs. Notes will be taken during these interviews, then analysed via thematic analysis. Insights from this experimentation will be used to inform the design and user interface of the prototype in the following objective.

To achieve objective 4, a proof-of-concept prototype graphical asset generation tool (with a focus on generating 3D swords for games), will be developed via the rapid action development (RAD) methodology. RAD is a form of agile development, in which software can be developed iteratively through prototyping, with the flexibility of adjusting to evolving requirements [27, 9]. Initial requirements will be determined based on findings from objective 1 and 3. This will be followed by a design phase, applying the framework from objective 2 to select an appropriate generative method. Implementation and testing of the various components, such as the generative method and the game engine-integrated UI, will proceed iteratively. These components will be integrated to form the final prototype tool.

Objective 5 will be achieved via a mixed methods approach, using statistical analysis for quantitative data, and thematic analysis for qualitative interview data. A convenience sampling approach will be used to recruit game designers and developers in order to test the proof-of-concept prototype from objective 4. Quantitative data will be collected via a repeated measures questionnaire for each of the implemented techniques. This will examine core user-perceptible evaluation types derived from the framework of metrics, determining the preference of each technique in relation to their overall perceived usefulness. The questionnaire data will be analysed via Friedman tests and Wilcoxon signed ranks tests to determine significant differences in ratings between techniques. Multiple linear regression will be used to determine the correlation between the various technique ratings and their overall perceived usefulness. This analysis will help ascertain which aspects most significantly impact user-perceived usefulness. Participants that choose to take part in a semi-structured interview will be asked questions aimed at the reasoning and nuance behind these ratings, which will serve to further validate findings. Interview data will be collected via note taking, then analysed via thematic analysis. Insights will then be used to refine GAGeTx from a user centred perspective, incorporating user needs and preferences. By integrating the insights from this objective, along with the contributions of objectives 3 and 4, GAGeTx will both be validated and refined with regard to game pipelines. This will ensure that GAGeTx not only provides a view of all state-of-the-art methods but also aligns with the preferences and needs of its users, thus improving its usability and relevance for tooling in game design and development. This will be the culmination of this research, enabling GAGeTx to provide comprehensive step-wise instruction in building or selecting GAG methods with the full integration of game designer and developer needs and requirements, mapping the design and production pipeline to the strengths of GAG techniques, and defining how to evaluate and compare approaches.

1.3 Planning

To achieve the 5 objectives stated in section 1.1, timelines and milestones were planned, as shown in figure 1.1. Each objective has been achieved in order, with important findings and outcomes feeding into each subsequent objective. This began with the systematic literature review, providing a pool of literature containing graphical asset generation methods from 2016 until present. The core output of this objective was achieved by April 2022, at which point it was possible to formulate the GAGeTx framework. Upon completing the initial framework based solely on the literature at the end of June 2022, work on the first user study commenced between June and December 2023. Following this, it was possible to begin design and development of the proof-of-concept prototype, which was finalised at the end of August of 2023. Between August 2023 and May 2024, the second user study was planned and conducted, results were then analysed and used to inform refinements to the GAGeTx framework along with the results from the first user study between May and August of 2024. During the intervening years, as new literature has been published, the literature review and framework have been updated accordingly.



Key Milestones

Objective 1

Systematic literature review

- PRISMA based systematic literature search.
- Collate evaluation metrics and datasets.

Duration: Oct 2021 - Apr 2022

Objective 2

Graphical asset and metric frameworks

- Establish graphical asset generation framework.
- Establish framework for evaluation metrics.

Duration: Apr - Jun 2022

Objective 3

Investigate user requirements for procedural asset generation (study 1)

- Design study; formulate questions, design questionnaire, design prototype interfaces.
- Ethics approval.
- Recruit users from LUUG, and other game development communities.

- Data collection.
- Data analysis.

- Begin designing proof-of-concept prototype.

Duration: Jun - Dec 2022

Objective 4

Proof-of-concept prototype

- Based on study 1: develop a proof-of-concept prototype tool.
- Tool validation.

Duration: Dec 2022 - Aug 2023

Objective 5

Evaluation of prototype (study 2)

- Plan study.
- Ethics approval.
- Prototype testing, followed by short questionnaire, specifically invite LUUG members etc. Ask participants for futher follow-up interviews.

- Data analysis.
- Refine and expand framework.

Duration: Aug 2023 - May 2024

Refine and expand framework

Incorporating:
- User preference.
- Metric usage.
- Usage in the game design and development pipeline.

Duration: May - Aug 2024

FIGURE 1.1: Key milestones and timeline durations for each objective.

1.4 Outline

The next two chapters (chapters 2 and 3) present the systematic literature review findings, GAGeTx framework, expanded GAGeTx framework and framework for evaluation metrics. Chapter 4 presents the conception, procedure, results and findings of user study 1, examining user preferences regarding generative tools in game pipelines, prompted by hands-on experience with interface mock-ups. In chapter 5, to build on the findings in chapter 4, a proof-of-concept prototype 3D sword generation tool is developed in the Unity engine, and trained on a novel silhouette dataset. This is followed by the planning, data collection and analysis of user study 2 in chapter 6, in which the prototype tool is evaluated and insights are derived on the relationship between technique preference and choice of usage in the design and production pipelines, as well as the perceived quality, speed and controllability. The findings and insights from this chapter and chapter 4 are then applied to the expansion and refinement of the GAGeTx framework. The thesis will conclude in chapter 7, consolidating the findings and contributions throughout the thesis, beginning with a thesis summary, followed by a comprehensive discussion of highlights and contributions, a discussion of research limitations and a discussion of future research directions.

Chapter 2: GAGeTx- Graphical Asset Generation/Transformation

In this chapter, a systematic literature review has been conducted, examining the breadth of state-of-the-art GAG methods. This review informed the development of a conceptual framework for graphical asset generation and transformation (GAGeTx), in which five main aspects have been identified, including input type, technique, approach, target asset type and format. Various categories have been identified within these aspects, while processes for decision making with regard to these aspects have been provided.

2.1 Introduction

On-going improvements in computing technology have expanded the limitations of digital media throughout the years. From websites to movies, video games, virtual reality (VR) and augmented reality (AR) experiences, ranging from serious to recreational, there are many forms of content in constant development. The ubiquity of computing hardware makes digital content available to a wide audience of users. As such, the demand for digital content is ever growing.

The task of producing this content is typically handled by highly skilled artists and designers, requiring creativity, technique and an understanding of user needs. However, production of content takes time, which is further exacerbated by the amount of content required. At scale, this can be counteracted by involving a large number of artists and designers, but this incurs large production costs that studios with smaller budgets may not be able to afford. Software for producing digital content, such as the Autodesk products [16], Blender [32], Unity engine [475] and Unreal engine [105], provide tooling that aims to streamline the technical aspects of production workflows. The creative aspect, however, is still very much the domain of the designer or artist.

Procedural content generation (PCG) is an area of research that involves the application of algorithms and computational processes to the production of digital content. For many years the film and video game industries have made use of PCG approaches to save time on large scale tasks such as producing realistic and varied trees with SpeedTree [190], or simulating large crowds with MASSIVE [385]. Video games in particular, consist of many varied types of content, much of which can be procedurally generated, as seen in Dwarf Fortress [23], that generates entire worlds and their histories, or No Man's Sky [162] with generated planets and creatures.

Hendrikx and Meijer [164] categorise these content types, in pyramid form, from low-level *"game bits"*, such as vegetation, buildings and sounds, up to game spaces, such as maps; and at higher levels, systems, scenarios and overall game design are considered. In this model each layer may derive content by combining elements from layers below. Togelius et al. [465] introduce key terminology, such as *search-based*, *online* and *offline* PCG. *Offline* content generation can be defined as content generated during the development of a product, which allows for it to be curated and edited by a designer. *Online* generation is content generated at runtime, and thus cannot be curated by a designer. Typically, *online* generation approaches are presented as key gameplay features, such as loot and level generation in games such as Diablo 2 [33] and Path of Exile [148]. Search-based PCG involves approaches that seek out quality content by searching a content space and evaluating the results.

Interest in the use of machine learning (ML) for content generation is shown widely in recent research, with the impact of generative adversarial networks (GANs), approaches to style transfer [134], and sketch to image generation such as GauGAN [355]. Summerville et al. [450] discuss the concept of procedural content generation via machine learning (PCGML), whereby ML models learn to generate novel game content from existing content. It is clear from such research that many PCG approaches, while effective, still require human supervision or involvement to produce content that would be sufficient for commercial products. Humans can work with PCG systems, as described by Liapis, Smith and Shaker [280] with the concept of mixed-initiative content generation, whereby machine and user co-create content.

So far, focus has been placed on content with a functional purpose, i.e. the upper elements of Hendrikx and Meijer's [164] pyramid model; or covered game content broadly, for example the work of Liu et al. [294] which examines the uses of deep learning for PCG, focusing on game content with varying functionality constraints, from narratives down to textures. But less focus has been placed on approaches applied to generating *"game bits"* with lower functionality requirements, i.e. lower-level elements, such as: 3D models of vegetation and buildings, or 2D textures and icons. It can be argued that these elements form a large proportion of the content required in a game, and thus are a large part of the development time cost. This chapter will examine the visual forms of this content, referred to as *graphical assets*, and the state-of-the-art in generating them, with emphasis on *offline* approaches.

Furthermore, this work focuses on generative methods that demonstrate a level of "initiative", defined as those that apply logic, reasoning or autonomous decision-making in the production of assets, whether through rules that are built-in, such as in a grammar, or learnt through data, as in deep-learning. This includes any method that embodies decision making that could be expected from a human, based on given inputs. This definition is derived from the usage of the term in existing literature [280, 48].

In this chapter, state-of-the-art GAG methods have been aggregated with the goal of answering the following research questions:

RQ1 What generative methods demonstrating *"initiative"* are in use for producing graphical assets?

RQ2 What are the forms of graphical assets that have been produced through generative methods demonstrating *"initiative"*?

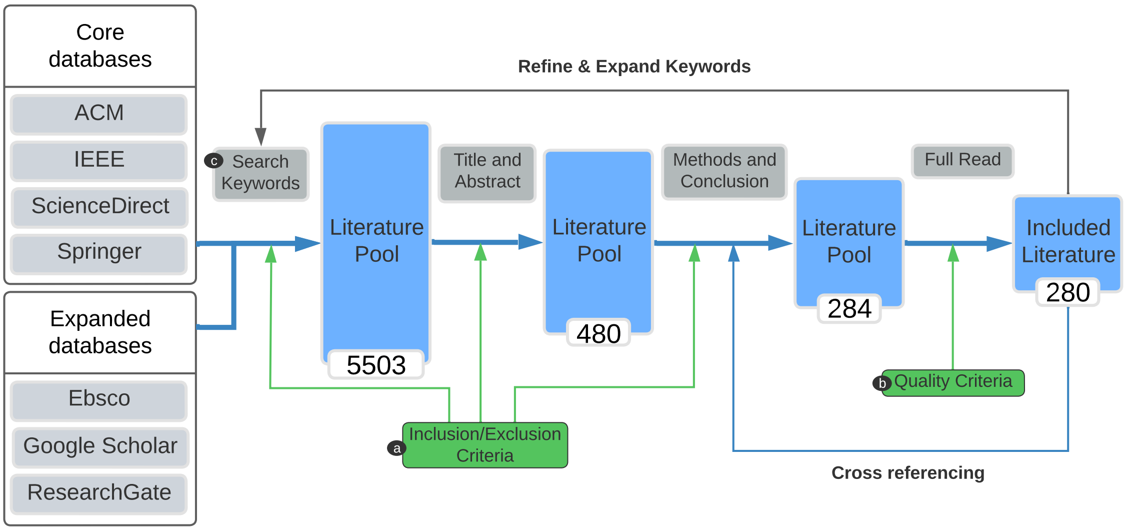
RQ3 Can methods for graphical asset generation map onto the requirements and purposes of the end-user and how?

Within this chapter, the goal is to discover the methods used in the current body of literature and the asset types these apply to (RQ1, RQ2). Furthermore, through the lens of using generative methods as tools for creating graphical assets, the aim is to elucidate the relationship between the requirements of the user and the multitude of methods available (RQ3).

2.2 Literature search

To obtain relevant literature, a systematic literature search inspired by PRISMA and the approach of Hughes et al. [183] has been employed, incorporating a series of screenings as shown in figure 2.2. The initial search examined literature published in four main databases: ACM Digital Library, IEEE Xplore, ScienceDirect and Springer. An initial assortment of keywords was established based on general terms in the PCG literature, alongside variations and synonyms of the word "generation" or "creation", and terms "asset" and "content". These words, figure 2.2.*c*, were separated into three semantic groups.

Query strings were formed by combining terms within each group using "OR" operators and combining across groups using "AND" operators. For example: *"(Environment OR Terrain OR Layout) AND (Graphic OR Asset OR 3D OR Mesh) AND (Generation OR Synthesis)"*. These queries took two forms: broad queries with many search terms, and smaller specific queries that included a single search term from the first group of terms, paired with smaller sets of terms from group 2 and 3.



Inclusion Criteria	Exclusion Criteria
• Methods for generating graphical assets (GGA).	• Not distinctly graphical assets e.g. text or animation.
• Comparisons of methods for GGAs.	• Functional requirements rather than visuals.
• Combinations of methods for GGAs.	• Non-procedural methods.
• Latest version where multiple iterations exist.	• Methods not demonstrating *"initiative"*
• Published between 2016 and 2023.	• Review or survey papers, posters, courses.

(a)

Quality Criteria	
• The method is validated.	• The method is peer reviewed.

(b)

Group 1			Group 2	Group 3
Procedural*	"Deep Learning"	Grammar	Graphic*	Generation
Algorithmic*	Inverse	Deep	Asset	Synthesi*
"Machine Learning"	Stochastic	Parametric	3D	Modeling
ML	Furniture	Vehicle*	"3D Art"	Modelling
Car*	Building*	Cloud*	Content	Creation
Environment	Road*	Tree*	"3D Model"	Design
Terrain	"Normal map"	"Texture map"	Mesh	Production
Layout	"Height map"	Character	Shape	Assemb*
Face	Hair	Organ	“Text-to-image”	
Sprite	2D			

(c)

FIGURE 2.1: Systematic literature review process: a) inclusion and exclusion criteria, b) the quality criteria applied to the literature search, c) The search terms used to query the chosen databases with expanded search terms in grey.

Using the inclusion and exclusion criteria seen in figure 2.2.*a*, results from these queries were first selected based on their titles and abstracts. The pool was then reduced by examining their methods and conclusions. Then the full text of each paper was examined, applying the quality criteria, as seen in figure 2.2.*b*. The results that passed these criteria formed the pool of accepted literature. The process of evaluation, for each query, was continued until the query was exhausted, that is, once each result had been evaluated. It was necessary to deem a search exhausted once a full page of results had not passed the criteria due to the impracticality of assessing every page. However, smaller targeted queries aimed to fill any gaps from broad searches. Furthermore, the ordering of each set of results was subject to the databases' own relevance ordering. The number of results per page was also variable, and thus recorded. Queries that were too long for the database search systems were split into smaller strings. The number of results per page for each database, as well as the number of pages completed before queries were exhausted have been provided in appendix 7.1.3. As common classes of graphical asset emerged, these were added as search terms, and the literature search process resumed with queries incorporating the new terms. In the accepted literature, related work was cross-referenced and evaluated against the criteria, and additional supplementary queries were performed on the databases: Ebsco, Google Scholar, and ResearchGate to ensure completeness. Pre-prints have been considered and discussed, but have not been included in count tables or figure 2.2.

As shown in Figure 2.2, a total of 5503 papers were initially assessed. This number is ascertained by multiplying the number of entries provided per page by the number of pages accessed for each database. From this, 480 papers passed title and abstract screening, 284 of which were accepted following the analysis of the methods and conclusions. After a full read of each paper, the final accepted literature count was 280.

2.3 GAGeTx: A Framework for graphical asset generation/transformation

The conceptual framework, GAGeTx, was developed through an inductive thematic analysis (ITA) [154] of the accepted pool of literature, detailed in section 2.2. The purpose of this analysis was to establish a categorisation of the key components of graphical asset generation methods, based on the literature. This iterative, inductive approach allowed categories to emerge from the literature itself. ITA was conducted and coordinated using a spreadsheet containing entries corresponding to each accepted paper. Analysing the content of each paper, entries were tagged with codes as they emerged. Thematically grouping the emergent codes, a structure of categories was formed. As more literature was analysed, the categories were iteratively refined by splitting and merging categories and associated text extracts. Each paper was then classified based on the established categories. During this process, certain thematic distinctions became clear regarding graphical asset types. To ensure that the literature search was thorough in regard to the range of asset types (RQ2), the search terms were expanded to include these graphical asset types before conducting further rounds of searches, which subsequently expanded the literature pool. At this stage, the final refinement of the category arrangement and naming took place, resulting in the categories and sub-categories of GAGeTx.

Five key aspects emerged as differentiators amongst generative methods, these are: input type, technique, approach, target asset type and format. Within these aspects, various options were observed and categorised. The structure of GAGeTx was formed by ordering these aspects based on their prerequisites, starting with the base assumption that the most important aspect is the asset type of the intended artefact. As such the asset type would be chosen first, and subsequent decisions would follow in a logical order based on previous steps. For example, the *approach* can only be decided once the *technique* and *input type* are determined, which is only possible when the intended asset type is known.

The primary purpose of a generator is to automate or assist in a creative process. For graphical assets, a breadth of approaches may be applied to the task, depending on the desired output, level of control and available data. The GAGeTx framework, presented in figure 2.2, conceptualises the task of building a generator, which requires an understanding of the desired outcome. For instance, the user may desire variations of an existing asset, to digitise a real object via photographs, turn sketches into 3D art, or obtain quick creative inspiration, which can be decomposed as the type of asset required and the technique for producing it. Technique determines the level of control and input type of the generator. Techniques for graphical asset generation fall under two categories: *conceived* and *synthesised*.

Conceived techniques allow for the conception of new content either *internally* from prior learning such as from a random seed or *externally* by transforming human creative input such as text prompts, photos or sketches. *Synthesised* techniques construct new content by combining existing data provided at the time of generation, which is useful for re-configuring, or creating variations of existing content. Examples include object placement within an environment, interpolation between different designs and style transfer. The balance of creative initiative between user and generator is pertinent to the formulation of a useful generator. While conceived techniques may require large datasets, synthesised techniques may require many pieces of data from which to constitute new content. As such, the choice of technique may be constricted by the availability of data. If the *technique* can be seen as the task, the *approach* can be seen as the solution i.e. the way in which the *technique* is achieved. Different graphics formats, such as 3D meshes, point-clouds and voxels, or 2D bitmaps and vector graphics may be required for different purposes. To maximise the applicability of generative methods in cases where a different format is required, conversion methods can be applied as a final step.

The following sections will examine each step of GAGeTx in detail while reviewing the state-of-the-art. Section 2.3.1 will discuss asset types, then section 2.3.3 will examine techniques. Following this, section 2.3.7 will examine approaches, while discussing the bulk of the literature. Generating and converting assets will be examined in sections 2.3.14 and 2.3.15 respectively.

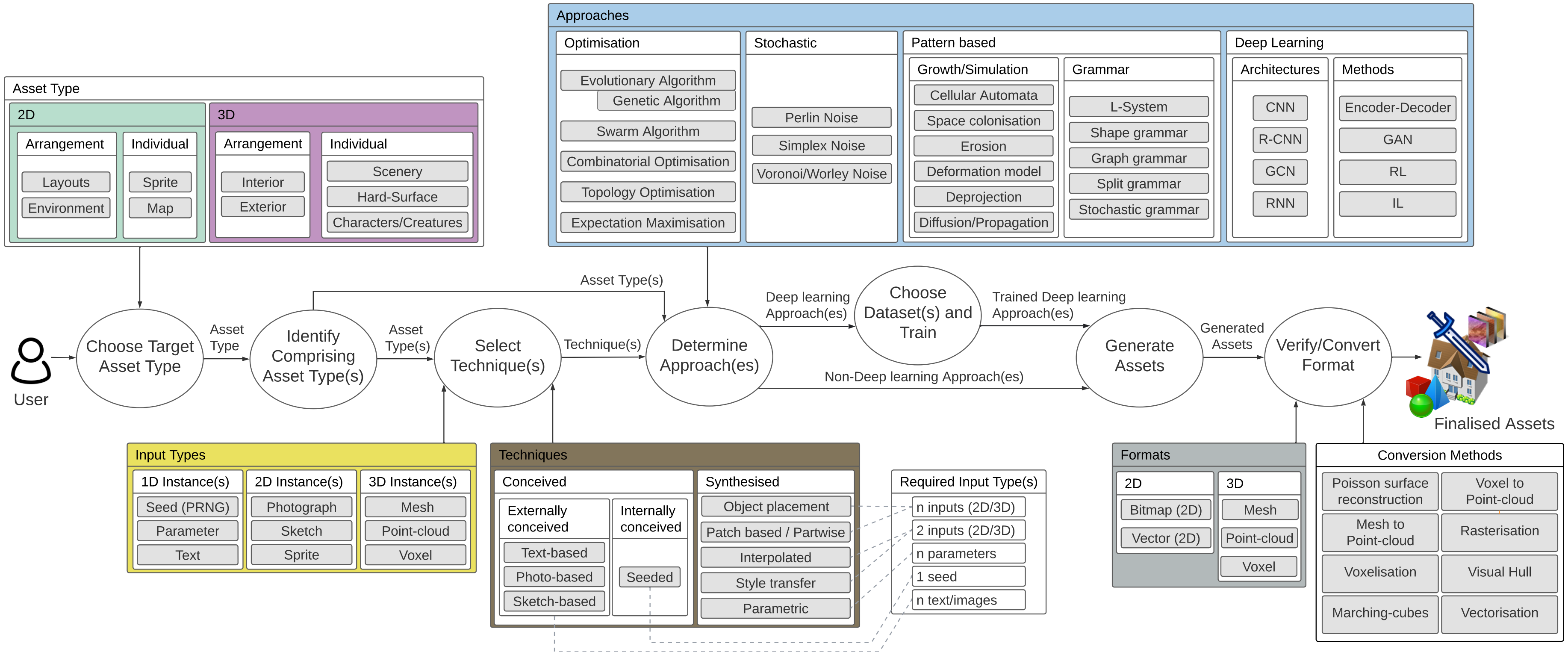


FIGURE 2.2: GAGeTx framework proposed for graphical asset generator

2.3.1 Asset type

Each type of *graphical asset* requires distinct considerations when it comes to generation. For example, structured grammar based approaches are favoured for 3D hard-surface assets, while stochastic, growth or simulation methods may be applied to scenery. This section will introduce the 21 types of asset examined in the literature and the task of selecting a target asset type within the framework. The asset categories found in the literature are shown in figure 2.3 and count tables 2.1 & 2.2 provide their distribution within the literature (many papers feature multiple times as they demonstrate the capability of producing multiple asset types). It is evident there have been more efforts toward 3D than 2D asset generation.

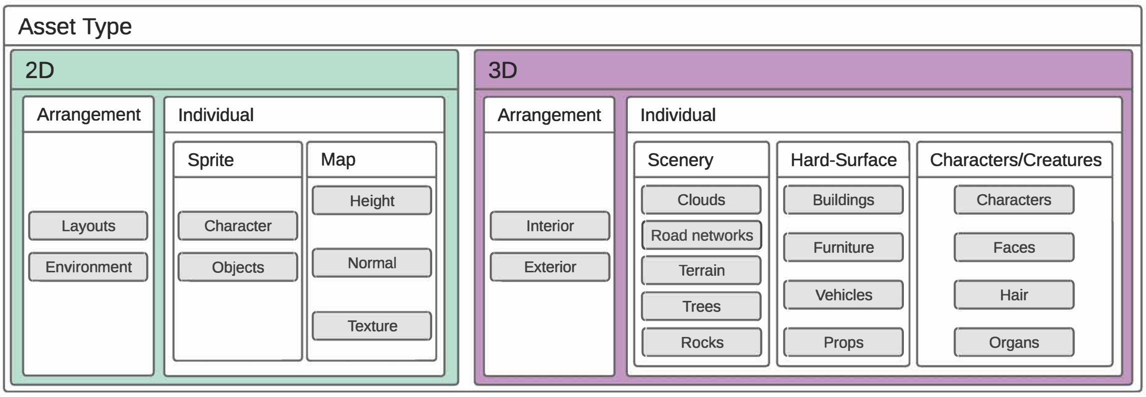


FIGURE 2.3: Asset types categorisations, populated with types observed in the literature.

2D

2D assets have applications in user interfaces (UI) for web, print and games; presenting game worlds and characters as sprites; or augmenting 3D assets as texture, normal or height maps. With the popularity of convolutional neural networks (CNNs) in interpreting 2D data, and the development of GANs, deep learning approaches have become a large contributor in the area of 2D asset generation. In particular, the Pix2Pix framework [192] has had a large impact, allowing for the translation of one form of image to another. This has formed the foundation for many approaches that seek to produce content via user sketches in particular.

TABLE 2.1: Count and breakdown of 2D asset types within categories.

Asset Category	Asset Type	N
Arrangement	Layout	5
	Environment	2
Individual (Sprite)	Character	5
	Objects	35
Individual (Map)	Height	8
	Normal	3
	Texture	15

2D Arrangement: Sprites and interface elements can be arranged on screen to form 2D environments or present information to a user. In web and UI design, graphical elements are not strictly visual. In many cases elements are interactive, which adds more complexity to the task of arrangement. Unlike sprites or 3D environments, these arrangements must be precise in order to capture the attention of an audience or user, and convey information succinctly.

Sprites: Sprites are standalone 2D assets, which for the purpose of this review includes general bitmap images that mimic photographs [373], artwork [131], represent 2D characters [384] or scenery [243]. Much like the distinction between objects and environments for 3D graphical assets, sprites can be examined individually, or as part of a larger arrangement. In 2D games many individual sprites may be arranged on the screen at once to form a cohesive game environment; these individual elements may include characters, objects and traversable areas.

Maps: Typically, a 3D asset is expressed as a shape in 3D space. In the case of meshes, this shape consists of vertices that are connected to form triangles. Rendering these triangles as a surface requires a shader, which is an algorithm that rasterises each triangle into pixels that can be displayed on the screen [406]. Here, a shader handles all computation relating to the lighting and appearance of a 3D object. Shaders typically make use of 2D data, in the form of *maps*. These *maps* are images that define how a shader renders the surface of a 3D object. In modern rendering approaches, many types of map may be employed such as height maps [440, 193], which define offsets for mesh vertices, normal maps [449] which determine how light affects the appearance of a model, and texture maps [107, 130], which determine the colour of the surface. While many methods have been developed with the sole purpose of generating maps, there have been some attempts at generating textures alongside models end-to-end [126, 199, 57, 130, 535, 228]. Height maps in particular have been largely used as a representation for terrain data in generative methods, as they efficiently represent detailed height variation in comparison to volumetric based alternatives, such as voxels [193, 20, 496, 97, 112, 440]. Shaders can be used to dynamically highlight points of interest in virtual environments. For example, in an immersive gamified cycling training intervention, shaders were used to indicate dynamically detected hazards in a 360 video [30].

3D

Among 3D assets examined in the literature, there are five main categories: interior arrangements, exterior arrangements, *hard-surface*, *scenery* and *characters/creatures*. These are split into sub-categories as given in table 2.2.

3D Arrangement: There are two distinct categories of 3D arrangement: interior and exterior. Interior arrangement refers to enclosed environments, such as bedrooms or offices; such approaches mainly emphasise object inter-relationship, where items have distinct purposes. Exterior arrangement, however, refers to the placement of objects upon a terrain, such as vegetation or buildings, where the approach to placement is more stochastic or naturalistic.

Buildings: The need for building generation can be found in architectural design tasks [525] and games [535, 147], while in some cases entire cities are generated [485, 230]. Having a simple structure, consisting of walls, doors, windows, and roofs, buildings are suited to approaches

TABLE 2.2: Count and breakdown of 3D asset types within categories.

Asset Category	Asset Type	N
Arrangement	Interior	16
	Exterior	6
Individual (Hard-Surface)	Buildings	29
	Furniture	64
	Vehicles	49
	Props	38
Individual (Scenery)	Cloud	3
	Road network	6
	Terrain	17
	Tree	8
Individual (Characters or Creatures)	Character	20
	Hair	4
	Face	19
	Organ	3

that work by combining simple elementary or parametric components, such as grammars or procedural growth-based algorithms. Some approaches specifically focus on building facades where these same techniques are applied [453, 46].

Furniture: 3D models of furniture, such as chairs, tables, cupboards and shelves are applied commonly in architectural modelling and game worlds. Believable furniture requires a combination of functionality and stylistic consideration, specifically they must meet a functional purpose while following established design conventions. Hence, it can be beneficial to use functionality or structure aware representations when generating 3D furniture. Furniture is also popular category in the ShapeNet dataset [52], used for testing deep-learning based generative models. Therefore many of such implementations have been trained and evaluated on the generation of furniture [489, 289, 351, 163, 265, 129, 216, 278, 523, 92, 292, 455, 493, 130, 327].

Vehicles: 3D digital environments that resemble the modern world are likely to require 3D assets that represent vehicles. The ShapeNet dataset [52] contains many categories of 3D shape, including aeroplanes, buses and cars; due to the popularity of the dataset, there are many generative approaches validated on such vehicle models, including mesh [305, 292], voxel [236, 526] and point-cloud [289] generation.

Props: To keep the list of asset types compact, hard-surface 3D objects that may be used to fill a virtual world are generically defined as props. These may include guns [493], guitars, lamps or bottles [270]. As a popular dataset for generative approaches, ShapeNet [52] contains many types of prop.

Clouds: Clouds are a common depiction in digital environments that aim to portray a realistic, earth-like world. They have a combination of attributes that make them challenging to implement, namely that they have a form but no distinct surface [194, 330, 332].

Roads: Roads, or road networks, are usually designed with functionality in mind [113, 230, 221, 90]. They exist to facilitate transportation throughout an environment, and in most cases can be considered in two dimensions. However, they also exist in relation to a 3D environment or terrain, conforming to a surface. To model a road system that simulates real-world roads it is also relevant to consider the human decision making involved [410, 462].

Characters: Other than objects and terrains, digital 3D environments may be populated with varied characters, and in the case of games, characters may be customised and used as digital avatars [286, 416]. Alternatively, other tasks may require character mesh generation, such as checking clothing fit in online shops [1].

Faces: Faces are a key element of character identity, thus high quality representations encompassing shape and texture are required. Furthermore, in games where character customisation is allowed, the ability to adjust and customise player-character appearance is key. There have been numerous attempts at the reconstruction of digital faces from photos [108, 247, 203, 226, 417, 286, 416, 491, 87, 290]. There is also a present interest in caricature generation, or the creation of faces with exaggerated features [44, 272, 172, 272].

Hair: Hair consists of many individual strands that can be of varying lengths, and flow in different ways. Current research in hair generation aims to achieve realistic hair flow, which necessitates propagation based approaches to modelling [552, 415, 551, 398].

Terrain: 3D terrains have uses in various domains, from simulation, to video games and animated film. Within these domains, terrains serve the purpose of establishing a setting and environment for exploration. In games and film, terrain serves as a foundation for exterior environments, on which a digital world is built. There are two primary approaches to terrain generation, these are: surface displacement via height map [112, 193, 440], and the use of volumetric representations [115, 24, 89].

Trees: Many 3D digital environments make use of tree models, as shown by the popularity of SpeedTree [190]. Trees are the result of a natural growth process, and so tend to be visually unique. In many cases environments may require dozens if not hundreds of trees, necessitating the use of generative methods [312, 488, 383, 260].

Organs: Due to the need for accurate imaging and visualisation, medical fields benefit from reconstructive visualisation/modelling approaches, particularly organs [467, 246, 241, 502]. Though this does not directly relate to other applications mentioned here, it is necessary to include such examples as the approaches could potentially be applied outside of this domain.

2.3.2 Input type

Within the literature, there are a variety of input types applied in the generation of assets, ranging from single value seeds, to fully formed existing 3D assets. The framework, figure 2.2, presents the input types observed in the literature.

Seeds are the simplest input type, and are often pseudo-randomly generated. In such cases, no user involvement is required. This simplicity, however, results in a low degree of control over the output. An example is basic GANs [212, 107, 300, 498].

Parameters provide a greater degree of control than seeds. A number of parameters can be employed, each mapping to a certain aspect of the asset, though this is dependant upon the algorithm's capacity to expose meaningful variables. Parameters can be configured by a user, but may also be pseudo-randomly generated [136], inferred using deep-learning [181], or optimised via evolutionary algorithms [332].

Text as an input has seen recent usage in deep-learning based text-to-image transformation [381, 390, 397], allowing for text descriptions to be interpreted in a meaningful way to generate images. As an input, text is simple and intuitive to produce, but may not be interpreted as the user fully intends. This, in some sense, asks both the user and the generator to be equally creative, or collaborative. Prompt engineering research seeks to make this form of input more controllable [295].

Sketches are a form of input that also requires the user to be creative. Unlike more complex input types, sketches need not be accurate or particularly detailed, requiring minimal time from a user but providing a good amount of creative control. Sketch based input can be interpreted either solely via deep-learning approaches [496, 83, 249] or in combination with procedural modelling [181, 341].

As inputs, point-clouds and photographs require the user to scan or photograph a subject, or otherwise source this data. The amount of control a user has over the output is constricted by the limitations of reality, that is, a subject must exist physically in order to be scanned or photographed. Photographs are largely interpreted via deep-learning [336, 87, 286, 261].

Fully formed assets may also be used as inputs to some techniques. Such inputs, however, require the user to create precursor assets themselves or otherwise source or generate them [84, 85, 136, 153].

2.3.3 Techniques

Techniques present the core functionality and purpose of the generator. As such, there are many possible approaches to the implementation of each technique, as will be discussed in section 2.3.7. There are two major categories of technique: *conceived* and *synthesised*. Techniques are defined by the inputs they require, as well as how the data is manipulated to form a result. Conversely, the availability of inputs determines what techniques are possible. If the technique is chosen first, the input type may be derived from the chosen technique's requirements. This may not always be possible in cases where certain input data is not feasibly obtainable. In such cases, a choice of input type may take precedence and the technique may be derived from this choice. Table 2.3 presents the count of each technique observed in the literature (many papers feature multiple times as they demonstrate multiple techniques), with the most prevalent techniques being photo-based, seeded and parametric.

2.3.4 Conceived techniques

Externally conceived techniques intake meaningful input from an external source, interpreting and producing a result that resembles the input. In other terms, the idea is preconceived, but the algorithm is given creative license to interpret it. Text, photo and sketch-based techniques are considered *externally conceived*, as the onus is on the user to conceive of an idea, either through text prompts, photographs or hand-drawn sketches.

Text-based asset generation is explored primarily in the 2D domain, with CLIP and Diffusion based deep-learning approaches [381, 397, 390], though text-based generation is achieved in 3D applications, such as with texture and displacement maps [123], or full model generation [299]. Alternatively, photo-based 3D asset generation has seen considerable exploration, with single-view [367, 336, 286, 387, 261], multi-view [261], and scan based generation of 3D assets [132, 263], and many utilising depth data from RGB-D images [552]. Photo-based asset generation allows for the digitisation of real-world objects, though with this comes the creative limitation that the object must exist to be photographed. In contrast, sketch-based generation allows for the creation of novel 3D [83, 531] or 2D [124, 384] assets through hand-drawn designs, though different methods vary in the level of detail required from a sketch.

Seeded generation is considered *internally conceived* as it requires no meaningful input from a user, instead producing outputs determined by a single, usually randomised value, that maps to a range of possibility. As such, the algorithm conceives the output internally without meaningful input or intervention. In deep-learning, asset generation approaches commonly involve learning a latent space from a given data distribution. It is common to randomly sample from the latent space, in order to produce novel outputs [130, 398, 459]. In essence this random sampling is a result of noise, which is seeded using pseudo-random generation. Alternatively, some asset generation approaches may be initialised using a seed, such as in noise based algorithms for generating terrains [112, 424].

2.3.5 Synthesised techniques

In opposition to *conceived* techniques, *synthesised* techniques aim to produce results that are consistent with the inputs provided them. In other words, they perform a logical service on the given inputs and do not provide creative input.

Object placement involves the logical placement of pre-existing assets within a space or *environment*. All *arrangement* type assets are produced via *object placement*, whether for 2D layouts [267, 266], 3D interiors [268, 369] or 3D exteriors [549, 442]. *Patch-based/Partwise* asset generation involves the piecing together of existing components into novel configurations. For instance, [462] build road networks out of pre-defined patches, while Krs et al. and Guan et al. [245, 153] piece together and morph existing meshes to form new shapes.

Interpolation in the context of asset generation is the process of producing a result that is visually *in-between* two given examples. For instance, Wang et al. [488] generate tree shapes using this kind of technique. Many generative deep-learning approaches that successfully learn a latent space are in-turn capable of interpolation. Therefore, meaningful interpolation within the latent space is commonly used as a way of testing a GAN or VAE model for its ability to generalise [321], where successful examples show consistency in their mappings [278, 129]. Hence, these approaches may be implemented for the purpose of *interpolation* based generation.

Style transfer involves the application of the style of one item to the content of another. Popularised by the impactful work on neural style transfer for images [134], many works have followed [218, 22, 131], including approaches to caricature [172] generation and photo cartoonisation [420]. Furthermore, the concept of style transfer is extended into three-dimensions with mesh texturisation [165] and functionality preserving stylisation [307].

Parametric methods take direct numerical inputs that have meaningful effects on the output. For example, in video game character customisation a type of morphable mesh may be used [417]. This may take continuous values for characteristics such as "height", "eye size" and "jaw width". By configuring these features randomly or through user input, many variations of the initial model can be generated. Alternatively, methods based on noise may take numerical inputs which directly impact the results of the noise generation [112, 401]. Some methods aim at converting existing meshes into *procedural* models [136, 84, 85], while others use these models as mediums for photo-based reconstruction [417, 1].

2.3.6 Select Technique(s) stage

This stage encompasses the selection process for GAG techniques, derived from the chosen target asset type, and input complexity. Regardless of the algorithm chosen for the task of asset generation, an input will always be required, whether this is provided by the user directly, or randomly initialised. The choice of input type primarily depends on the level of involvement and time investment the user is comfortable with. Hence the process of choosing the technique and input type is highly dependant on user choice. Algorithm 1 presents the process for selecting a technique and input, in which the input type $input\_type_s$, and technique $technique_s$ are selected from the pool of all generative techniques T and inputs IN. The input type is either determined by a choice of technique, or chosen first to determine the technique. Each technique has required inputs, as seen in figure 2.2. If the technique is not the priority, the input type may be chosen first, in which case the inverse limitations apply. The choice of input type requires a compromise between the user's control over the output, and the work required.

There are three methods for obtaining input data: sourcing existing data, creating data or automating the creation of data. As shown in figure 2.5, when data instances already exist, such as 3D meshes within ShapeNet [52] or photographs found on the internet, and they are of acceptable quality and relevance, they may be used as inputs. If not, inputs can be created by the user. This provides the user with full control over the input at the cost of time and effort. Figure 2.4 presents the input types ordered by their complexity, with the least complex at the bottom and the most complex at the top. As the complexity of input increases the effort required in order to create, automate or source the inputs also grows.

Algorithm 1 Selecting a Technique and Input Type

```
procedure SELECT_TECHNIQUE(a, IN[], T[], c, u)                                   ▷
INPUT: asset type, all input types, all techniques, chosen input complexity, user choices
    if $a = Arrangement$ then                      ▷ Arrangement assets require object placement
        $technique_s \leftarrow Object\_placement$                                ▷ s stands for selected
        $IN \leftarrow [\forall inp \in IN \mid inp.type = dimensions]$                              ▷
Filter input type choices by asset type "dimension"
    else
        i=0
        while $i < |T| \wedge T[i] \neq u.technique$ do          ▷ Allow user to choose an available option
            $technique_s \leftarrow$ T[i]                                   ▷ technique chosen by user
            $required\_input\_types = [\forall inp \in IN \mid inp \in technique_s.input\_types]$
            $IN \leftarrow$ required_input_types   ▷ Filter choices by input types required by technique
            i = i +1
    if $|IN|$ = 1 then                                              ▷ Only one choice is available
        $input\_type_s \leftarrow IN[0]$
    else
        i=0
        while $i < |IN| \wedge IN[i] \neq u.input\_type$ do      ▷ Allow user to choose an available option
            $input\_type_s \leftarrow IN[i]$
    return $technique_s$,$input\_type_s$          ▷ Pass selected technique and its input type to next step
```

Technique category	Technique	N
Externally Conceived	Text-based	19
	Photo-based	70
	Sketch-based	26
Internally Conceived	Seeded	45
Synthesised	Object placement	23
	Patch based / Partwise	4
	Interpolated	13
	Style transfer	12
	Parametric	29

TABLE 2.3: Count and breakdown of techniques within categories.

2.3.7 Approaches

After selecting a target *asset type*, *technique* and *input type*, an appropriate *approach* is determined. This is the specific set of algorithms or processes that perform a specific technique, generating a target *asset type* using particular *inputs*. For improved presentation within the framework (figure 2.2), approaches have been grouped and taxonomised under four main headings: *optimisation*, *stochastic*, *pattern based* and *deep learning*. Table 2.4 presents the count of each approach within the literature (many papers feature multiple times as they demonstrate multiple techniques). Though in many cases a combination of approaches are employed, the prevalence of grammars and deep-learning for the task of generating graphical assets is made evident. In particular, many instances of shape grammars, encoder-decoder networks and GANs are observed.

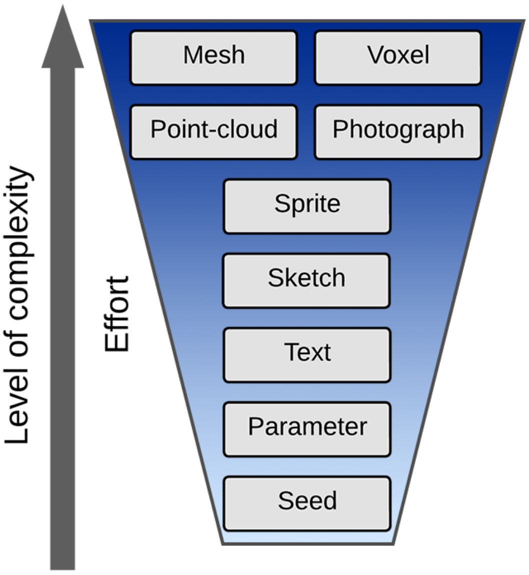


FIGURE 2.4: Input complexity and effort.

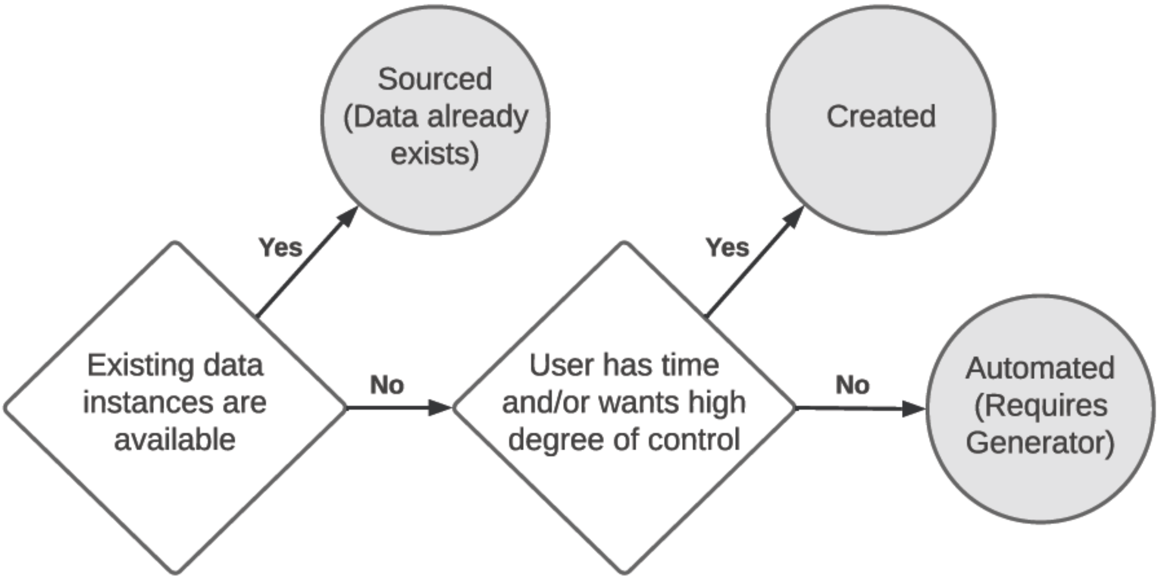

Existing data instances are available
Yes
Sourced (Data already exists)
No
User has time and/or wants high degree of control
Yes
Created
No
Automated (Requires Generator)

FIGURE 2.5: Decision process for obtaining inputs.

2.3.8 Optimisation approaches

Optimisation based generative approaches include *evolutionary*, *genetic* and *swarm* algorithms, as well as *combinatorial* and *topology* optimisation.

Evolutionary algorithms iteratively refine generated examples in accordance to a fitness function. At each generation, candidates with the highest fitness score are combined (through *crossover*) or altered (*mutated*) to produce a new generation of candidates. Over multiple generations, the

TABLE 2.4: Count and breakdown of approaches within categories.

Approach category	Approach	N
Optimisation	Evolutionary Algorithm	2
	Genetic Algorithm	6
	Swarm Algorithm	1
	Combinatorial Optimisation	4
	Topology Optimisation	1
	Expectation Maximisation	1
Stochastic	Perlin Noise	4
	Simplex Noise	1
	Voronoi/Worley Noise	1
Pattern based (Growth/Simulation)	Cellular Automata	2
	Space colonisation	4
	Erosion	2
	Deformation model	3
	Deprojection	11
	Diffusion/Propagation	7
Pattern based (Grammar)	L-System	6
	Shape grammar	14
	Graph grammar	1
	Split grammar	1
	Stochastic grammar	2
Deep Learning (Architectures)	Convolutional Neural Network (CNN)	37
	Regions with CNN features (R-CNN)	4
	Graph Convolutional Network (GCN)	10
	Recurrent Neural Network (RNN)	1
Deep Learning (Methods)	Encoder-Decoder	72
	Generative Adversarial Network (GAN)	80
	Reinforcement Learning (RL)	2
	Imitation Learning (IL)	1

fitness of candidates will improve, resulting in stronger (high fitness) candidates. The Procedural Iterative Constrained Optimiser (PICO) framework [245] is centred around a graph that represents a flow of parameterised operations that generate a 3D shape. An evolutionary algorithm is used to generate and optimise this graph, incorporating user-constraints. This is used to generate a variety of 3D assets, including trees, chairs and terrains. Functionality-Aware Model Evolution (FAME) [153] evolves novel shapes in a functionality-aware manner. An evolutionary algorithm is applied to a set of models by performing crossover between groupings of parts. Users can set functionality constraints on this system, or guide evolution by selecting preferred results. *Genetic algorithms (GA)* are a popular class of evolutionary algorithm, employed in the generation of buildings [535, 318], vehicles [19], props [233, 318] and clouds [332]. A CNN has been used to learn a fitness function for the optimisation of cloud shapes, scoring generated clouds based on how real they appear [332]. Alternatively, *GA* is applied in object placement [442]. In this method, an optimal scene layout is generated based on fitness to a set of positional rules, defined by the authors. GA have also been used for camera parameter estimation [318].

Interactive genetic algorithms (IGA) integrate user input as part of the fitness calculation, which allows a user to influence the development of the asset. The 3DCSS framework [19], for example, successfully integrates *IGA* into the car design process. Alternatively, Yoon and Kim [535] attempt 3D building generation with textures using *IGA*. Though, in a user study, their *IGA* approach to texture generation was rated poorly, the generation of building models was effective. In the method of Kiptiah Binti Ariffin et al. [233], *IGA* is combined with L-systems to produce abstract 3D shapes.

Swarm algorithms employ the use of many agents that behave independently while influenced by the group. For example, in particle swarm optimisation (PSO), a global optimum can be found in a search-space via a combination of individual search and group knowledge [223]. 3D asteroid meshes have been generated based on real data for the purpose of simulating traversal of such terrains [275]. Here, PSO is used to set optimal parameters, ensuring that the shape and surface texture of the asteroids are realistic.

Combinatorial optimisation seeks to find optimal solutions to problems in vast but finite search spaces. The work of Lun et al. [307] uses *tabu search* to perform shape style transfer that preserves object functionality. In this method, tabu search is applied in the combinatorial optimisation of the shape such that functionality is preserved and style adaptation is maximised. This is achieved by efficiently searching through the possible modifications that can be made to the shape.

Topology optimisation aims at producing an optimal shape within a design space based on physical constraints. This is employed in the work of Kazi et al. [220], wherein 3D solutions are generated based on user provided sketches and constraints using the level-set method of Allaire et al. [11]. This sketch based framework is effective at helping a designer to explore solutions to their specifications, though slow computation times make it less feasible for fast design iteration. A form of *expectation maximisation*, first introduced by Kwatra et al. [250] is applied in 3D cloud generation using photographs [194].

2.3.9 Stochastic approaches

Stochastic approaches primarily involve the manipulation of noise in forming randomised yet controlled shapes and designs. Though noise underpins a large proportion of generative approaches, including many deep-learning architectures, this section will discuss methods that focus on its usage. There are many noise algorithms in common use, each with their own characteristics, including: *Perlin noise* [363], *Simplex noise* [364] and *Voronoi/Worley noise* [516]. Usage of noise can be *seeded* and *parametric* depending on the implementation or number of variables exposed to the user. Some approaches based on fractional Brownian Motion [311], for example, have parameters such as *octaves*, *lacunarity* and *gain* that the user may adjust.

A common use case for noise is in the generation of terrains via height maps. Height maps provide a two-dimensional representation of land height which can be applied via mesh surface displacement. For example Fischer et al. [112] employs *Simplex noise* in height map terrain generation as part of a multi-step pipeline for 3D environment generation, while Li et al. [275] use *Simplex noise* to create surface variation on asteroid models and Satyadama et al. [401] use

Simplex noise for surface variation when generating volumetric caves. Alternatively, Sin and Ng [424] use 3D *Perlin noise* to generate consistent height mapping around a spherical surface, and Dey et al. [89] initialise a volumetric terrain using *Perlin noise*.

Many methods combine noise with other approaches as a means to reflect the roughness and variation found in nature. As an alternative to height map generation, Becher et al. [24] introduce a method and pipeline for generating terrains using feature curves in volumetric space. The application of the feature curves concept to 3D volumetric space is an extension of previous applications to 2D height maps [171]. Extending to volumetric space allows for overhangs and tunnels in terrain. Montenegro et al. [330] propose a method, based on the work of Lipus and Guid [293], which combines the use of implicit modelling and noise in generating clouds. This implementation performs in real-time, allowing for rapid iteration on ideas. 2D content can also be generated using noise. For example, texture maps for colouring the walls of caves have been generated using *Perlin* and *Worley* noise [115], and sprites for 2D game environments with *Perlin noise* [243].

2.3.10 Pattern based approaches

Pattern based approaches use serialised logic to solve generative tasks. Examples include growth or simulation algorithms such as: *cellular automata*, *space colonisation*, *erosion* and *diffusion/propagation*. As well as grammars: *L-systems*, *shape grammars*, *split grammars*, *graph grammars*. *Cellular automata* has long existed, with early work of Von Neumann [484] and Conway's Game of Life [133]. Cellular automation works on the basis of adjacency rules that determine the value of a cell in a discrete grid. Evaluating each cell in the grid at each step allows for natural growth or formation of shapes and volumes. Cellular automation is an effective approach to content generation, given the appropriate rules. For example, while using a constrained-growth approach to generate floor plans, Green et al. [147] use cellular automata to arrange the placement of windows on walls. Cellular automata is also applied in combination with L-systems for the generation of caves [13]. Here, cellular automation is used to refine and smooth out the cave formations. *Space colonisation*, attempts to mimic a natural growth process for branching tree-like shapes. For example, Ratul et al. [383] use space colonisation for the real-time generation of trees, and Guo et al. [155] combine multi-view depth data with a rule-based system to perform space-colonisation. Stylised, plant-like designs, based on existing meshes, have also been generated using a form of space colonisation [553], and space colonisation is used in the generation of road networks [90], where it is used as a flexible method for generating organic looking road layouts that conform to user defined constraints.

Realistic terrain details can be generated using *erosion* simulation. AutoBiomes [112] is a pipeline for generating 3D environments with varying biomes. Using a combination of climate simulation, biome refinement and asset placement, an initial noise-based terrain is built upon to create a complex environment. This climate simulation models temperature, wind and precipitation. Franke and Müller [115] generate cave geometries using simulated physical properties, such as water flow and erosion. This produces a voxel-based volume given a set of parameters. A surface is formed from this volume via marching cubes, and textures are generated using a combination of Perlin and Worley noise.

Deformation model is used for generating creatures and trees. In the work of Dvoroznak et al. [100], users are able to draw or trace from reference material on a 2D canvas to create creatures. Users build semantic layers of the subject and order them based on depth. These layers are then inflated to form a 3D shape, aided by a deformation model. In the work of Wang et al. [488], graph representations are used for interpolation of tree models from existing examples.

Deprojection approaches have been applied in the reconstruction of props and scenes. Fedorov et al. [110] propose single-image 3D mesh generation using edge detection in combination with user demarked guides. This method also generates textures by modifying the input image content.

With the current availability of stereo cameras and the Kinect, RGB-D data can be produced readily at a much lower cost than traditional scanning methods. This added depth information can be instrumental in reconstructing 3D objects [132]. Effective RGB-D scanning pipelines are also suggested by Slavcheva et al. and Niemirepo et al. [428, 339], performing on par with other state-of-the-art photogrammetry approaches, while Kim et al. [228] introduce a framework for simultaneously texturing and reconstructing scenes using RGB-D data. Alternatively, Chen and Rosenberg [54] suggest a view-dependant texturing approach to real-time rendering.

In a traditional game development pipeline, 3D art may be produced with *concept art* as reference. It is also common for 2D designs to present different views of the object, typically the front, top and side. In the manual 3D modelling process, these references allow for the accurate reconstruction of designs. This process can be automated by projecting multi-view concept art onto a voxel volume [423], refining and converting the result to a mesh via marching-cubes [303].

Propagation approaches involve the progressive growing of shape through a space. For example, hair generation is primarily achieved by growing strands through a 3D volumetric flow field, representing the directional flow of hair through space [398, 552, 551, 415]. Gao, Yao and Jiang [132] segment individual object meshes from a scene by propagating user assigned labels, capturing the full shape of each object in the scene, while Dijkstra's algorithm is used to traverse a terrain and form natural height variation [140].

Patch-based generation is applied to road networks and web design. In the method of Teng and Bidarra [462], main roads are built using a graph growing algorithm. The spaces between the main roads are then populated with semantically tagged road patches, propagating inwards from the main roads. Instead, Mockdown [306] learns layout constraints for positioning web elements.

Deriving from the work of Chomsky [66], *generative grammars* operate as formalised rules and structures from which instances can be built. There are many variations of grammar employed in the generation of assets, including: *L-systems*, *Shape grammars*, *Graph grammars*, *Split grammars* and *Stochastic grammars*. Devising a grammar that encompasses the aspects of a reconstruction target, while mapping an input to said grammar is challenging. Li et al. [264] propose a probabilistic context-free grammar (PCFG) for reconstructing buildings from images, learning rules from existing models, and Martinovic and Van Gool [315] introduce an approach to grammar learning which focuses on facades. Alternatively, [46] attempt to reconstruct facades from low-resolution images, while Jesus et al. [200] employ a layered approach using grammars.

The challenge of generating buildings with curved surfaces can be addressed with the use of different coordinate systems, allowing for the same grammars to be applied on flat and curved surfaces [101]. Demir, Aliaga and Benes [84] introduce a method, capable of inferring a grammar from an existing building model. Building on this research, a framework has been established for the proceduralisation of existing 3D models [85]. Grammars have also been applied in the generation of ancient Roman and Greek style structures [240]. Nishida, Bousseau and Aliaga [341] reconstruct 3D models of buildings from single-view images, using a series of CNNs that output shape grammar parameters. This is adapted as a web tool [28], in which users interact with a web-based UI and the bulk of the computation is completed remotely. A sketch-based approach is also explored, allowing users to draw in aspects of a building [342].

Grammars have also been applied in interior layout generation. For example, a stochastic grammar with Spatial And-Or Graph (S-AOG) is introduced Jiang et al. [205]. This method allows for a large degree of user control, while adhering to rules and characteristics that are present in pre-existing data. This can be used to synthesise data for training or validating deep-learning methods. Freiknecht et al. [118] introduce an algorithm for generating full building assets including interiors and textures. Alternatively, a Scene Grammar Variational Autoencoder (SGVAE) approach is introduced by Purkait, Zach and Reid [369], which encodes indoor scene layouts via a grammar.

To create a logo, designers must spend time developing ideas, and creating many variations to find the ideal design. Li, Zhang and Li [276] attempt to alleviate the amount of manual exploration for designers, introducing a framework that augments the logo design process with the use of shape grammars. The Procedural Shape Modeling Language (PSML) [514] allows users to express 3D shapes via code. This language integrates shape grammars and object-oriented programming to allow object structures to be expressed hierarchically, with adjustable parameters.

Generative approaches may be used for design ideation, where users are not necessarily interested in polished outputs, but rather unique ideas that can be refined. The approach of Alcaide-Marzal et al. [10] uses a generative grammar system that produces variations of products by combining pre-defined design elements and applying transformations to them. Alternatively, shape grammars have been applied to large scale industrial designs, where parallels are drawn between engineering specifications and generative rules [94]. Geometric graph grammars (*GGG*s) are applied to the generation of road networks [113]. The *GGG* extends the concept of a graph grammar by encoding geometric data alongside topology.

Volumetric terrain generation can be achieved via voxel grammars [89]. As a form of shape grammar, voxel grammars define rules that are applied to a given starting point. Alternatively, Raies and Von Mammen [377] combine grammars with a swarm algorithm to generate entire environments consisting of terrain, vegetation and bodies of water. *Swarm grammars* [483], are devised for each of these aspects, interacting with one another to produce a natural environment.

Lindenmayer first introduced L-systems, a form of recursive re-writing grammar, as a way to model the natural growth of plants [291]. L-systems have since been adapted to three-dimensions [169] and applied to tree reconstruction from photos [155]. They have also been applied in road generation [410], and combined with IGA for novel 3D shape generation [233].

2.3.11 Deep learning approaches

For the purposes of distinguishing high-level structures from network specifics, generative deep-learning approaches are categorised by their *methods*, and the *architectures* that they employ. In line with the rest of this review, this section will focus on the high-level strategy of each approach. There are two dominant generative deep-learning strategies: GANs, and encoder-decoder networks. These strategies are combined in the form of adversarial auto-encoders (AAEs), and to a lesser extent, deep generative reinforcement learning (RL) and imitation learning (IL) have also been attempted.

Generative deep-learning aims to extract patterns from large datasets, in order to derive novel content. GANs, first introduced by Goodfellow et al. [141], are a generative unsupervised learning method that pits two models against each other, such that both models learn through competition. The adversarial strategy has been highly popular in generative approaches throughout the years, with simple image generation [107, 300, 212], sketch-based techniques [249, 559] and text-to-image generation [381, 397, 542, 405, 361, 373].

Basic GAN based implementations have been applied in the generation of spell icons [212] and textures for games [107], aiding the process of design ideation [300] and generating images of indoor scenes [498]. Such generation is *seeded*, as content is produced by randomly sampling the extracted feature space in order to find novel content. GANs have been successfully applied in style-transfer tasks [131, 22, 420, 541], as well as image generation from sketch-input [249], art colourisation [559], and caricature generation [172]. An influential framework for such approaches Pix2Pix [192], employs a conditional GAN (cGAN). cGANs take additional inputs, allowing the end user to specify the kind of result generated. For example, [500] conditions a face image generator on high level attributes, such as age, gender and hair colour. CONGAN [161] provides an alternative method of input for GANs, in which the user provides photo constraints to the generator, causing it to generate results more like, or less like other images. Instead, StyleGAN [213] introduces a style based GAN architecture to great effect, producing high quality images of various types.

LayoutGAN [267] achieves 2D layout generation by learning to produce a feasible layout from a given input. This is further developed with the addition of attribute conditioning [266]. User-sketch based cGAN approaches have also been applied to the generation of height map based terrains [353, 496, 440]. For example, Sketch2Map [496] allows designers to draw simple maps that represent terrains, while a similar sketch-based approach, [97], allows users to sketch sections of terrain that are seamlessly joined. Zhang et al. [561] introduce a GAN based method for combined sketch and text based generation, in which the user sketches the shape of the object they wish to depict, then describe its features and colours in text. Existing GAN based attempts at generating novel images using text inputs include [542] CAGAN [405], SAM-GAN [361], CycleGAN [574], MirrorGAN [373], LeicaGAN [372] and ControlGAN [262]. These approaches extract semantic meaning from input-text and apply appropriate transformations to an image via GAN based architectures.

DALL-E [382] and CogView [91] demonstrate that high quality results can be obtained by scaling the number of parameters and training data to a large degree. Recent methods, such as DALL-E 2 [381] Imagen [397], make use of diffusion and CLIP [375] mechanisms to generate high fidelity images from text. Stable-diffusion [390] successfully applies a diffusion based approach which can be conditioned on text, images and semantic maps.

The usage of GANs also extends to the creation of 3D assets, where they have been successfully applied in the reconstruction [237, 236, 455, 305, 567], generation [270, 278, 419] and interpolation [419, 545, 567] of new mesh, point-cloud and voxel assets. Furthermore, diffusion models have been applied in 3D shape generation [185, 545].

The Sphere as Prior GAN (SP-GAN) [270] is capable of generating point-clouds in a structure-aware manner, while SG-GAN [278] and HSGAN [277] generate point-clouds in topologically and hierarchically aware manners respectively. Voxel generation is also achieved via GANs [489, 425, 439], and cGANs [346]. cGAN has also been applied to generating varied voxel-based rock shapes with user defined boundaries [248].

Hertz et al. [165] introduce an approach to shape texture transfer. Given a reference and target mesh, this method is capable of outputting new geometry that applies the texture of the reference to the form of the target mesh, improving on the results of OptCuts [269], which instead makes use of 2D displacement maps.

3D model conception can be a combined effort between user and machine. Davis et al. [77] introduce a VR based co-creative AI which allows users to generate 3D models by exploring and iterating upon ideas. Deep generation of 3D meshes is a difficult task due to limited data availability. This challenge can be avoided with the use of differentiable rendering. Differentiable renderers allow for self-supervision in 2D to 3D tasks, removing the need for 3D ground-truth data. This has successfully been applied to single-view reconstruction [216, 163, 360] and 2D to 3D style-transfer [216], improved upon with use of normal maps [521], and applied in game character face generation from photographs [417]. With a similar approach, GET3D [126] achieves high quality textured meshes with complex typologies over the full range of 3D asset types.

In the task of building generation, effective GAN implementations have been presented for internal room layouts using graphs [338], and façade image generation [453]. These methods are centred around the 2D domain, however, may still be applicable in combination with other methods for generating 3D buildings. Alternatively, [98] present an approach that is capable of generating fully textured 3D building models by chaining multiple GANs together.

Encoder-decoder network structures allow for a mapping, and therefore, translation between input and target domains. Such networks constitute an encoder network, that learns to condense an input, and a decoder network that learns to interpret an output from this embedding [63, 325]. Autoencoders are a form of encoder-decoder that aims to produce outputs that are identical to the input, they are applied primarily in de-noising or compression tasks where the encoder discards irrelevant information [481]. U-net is a popular form of encoder-decoder [391]. As a fully convolutional architecture, it is primarily used for working with images, and as such, U-nets have seen use in diffusion based generative models [390, 397], image translation [192], and interpreting sketches [82, 249]. In general, encoder-decoder networks have been successfully applied in photo-based [494, 216, 336, 261, 367, 531], sketch-based [82, 412, 83, 531, 76] and text-based 3D shape generation tasks [299, 45, 327, 400, 196, 295, 122, 539].

Simple encoder-decoder networks do not learn a consistent latent space that can be sampled from directly to generate new content. Variational autoencoders (VAEs) address this by employing regularisation during the training process, allowing for smooth interpolation and parametrisation of inputs. VAEs have been used to successfully generate 3D assets including furniture [206, 207, 129, 130, 61, 530, 271], textures [130], characters [459] and hair [398].

Jones et al. [206] introduce a method for generating primitive based objects using a VAE. The model is trained to produce programs in an intermediary language called ShapeAssembly, whereby 3D models are encoded as a list of operations applied to simple cuboids. This is expanded upon with the introduction of a method that is capable of learning macro-operations from existing ShapeAssembly programs [207].

SDM-NET [129] employs VAEs in learning the structure and geometry of objects, producing high-quality generated and interpolated results. This is developed further with TM-NET [130], which produces textured 3D models. A VAE is applied in the generation of novel object and character poses [459], this incorporates the rotation-invariant mesh difference (RIMD) data representation [127], allowing the VAE to learn surface deformation.

Expanding on encoder-decoder style models, generative transformer networks have quickly risen in prominence since their introduction in 2017 [479]. The self attention mechanism of these models allow for more effective sequence to sequence learning. In addition U-net based diffusion models have been successfully employed in text-to-3D [368, 527] and image reconstruction tasks [297, 371] via multi-view image generation. Mescheder et al. [319] introduce occupancy networks as a method of representing 3D shapes in continuous space. This is achieved by predicting an occupancy function from which surfaces can be extracted at arbitrary resolutions. This results in high quality outputs with lower memory overhead than voxel, point-cloud or mesh based representations. This was later improved by incorporating convolutional operations [362]. Similar implicit representations have also been used such as IM-NET's implicit fields [60], and DMTet [414] which applies the implicit function to a tetrahedral grid, that is then converted into mesh format using marching tetrahedra; a method similar to marching cubes.

The usage of diffusion models for text-guided image generation has also been extended to the generation of textures. Methods such as TexPainter [548], MaPa [556], and TEXTure [389] each produce textures for 3D mesh inputs, guided by text prompts. Additionally, the methods of Chent et al. and Metzer et al. [57, 320] both successfully generate 3D mesh models with textures simultaneously, using diffusion based models.

Deep-learning has been applied in the estimation of hair flow fields. Single-image hair reconstruction is achieved in this way using VAE [398] or GAN [551]. DeepSketchHair [415] instead utilises a sketch-based approach, where users sketch the outline of a hairstyle, and draw lines within to indicate the flow of hair. The user is then capable of refining their design by providing more sketches at different angles, while viewing the result. Alternatively, [552] generate hair using multi-view RGB-D images.

Single-view reconstruction allows for the generation of highly specific 3D content with minimal user input. This is largely a task of inferring a whole shape from a single viewing-angle, which can be achieved through prior knowledge of similar objects. Many methods apply deep-learning to the task, such as [292, 526, 235, 336, 184].

Pixel2Mesh [493, 492], uses a CNN in combination with a graph convolutional network (GCN) to deform a base mesh with the goal of matching an input image. As this approach manipulates vertices directly, each vertex can also have an associated colour value, which enables the generation of coloured meshes. This architecture has been successfully expanded with a graph attention mechanism [92] and for multi-view reconstruction [508].

There are many other approaches that incorporate the template deformation concept. For instance, Image2Mesh [367] encodes an input image using a CNN, finds the closest base model, then deforms it via free-form-deformation [407] to match the input. A similar approach uses free-form-deformation to generate lung models from single-view images [502], while template mesh deformation is applied to liver [467] and heart [241] reconstruction.

EasyMesh [455], also takes a deformation approach, while processing input images into silhouettes to gain consistency in training. Similar approaches use deformation to generate meshes [265, 274, 352], and point-clouds [543]. These approaches are limited to deforming existing topologies. Instead, Mesh R-CNN [139] is capable of reconstructing varying topologies from single-view images. This approach expands om Mask R-CNN [158] by adding mesh prediction. The template meshes typically used in such methods are *genus-0*. This covers a wide range of possible geometries, but is not sufficient for all shapes. Pan et al. [351] circumvent this limitation by introducing topology modification modules which remove faces from the mesh to form holes where they are needed. Alternatively, a surface can be constructed out of elementary patches of mesh data. For example, Adaptive O-CNN achieves patch-based image reconstruction by building octrees of patches using a CNN [494].

One core challenge with single image reconstruction, is the lack of information about the parts of the object that are occluded or facing away from the camera. This issue is addressed in the framework of Lu et al. [305], which utilises generated multi-view silhouette data. Yang, Li and Yang [531] disentangle shape and viewpoint in their encoder-decoder architecture by decoding using separate shape and viewpoint transformer networks.

Generalisation is another challenge, often necessitating copious amounts of data to achieve. One way to circumvent this is to employ a few-shot approach, where a network is capable of generalising to new classes from a few samples [501]. Lin et al. [289] make use of few-shot learning in single-view point-cloud reconstruction by separating class-specific and class-agnostic features.

Voxel representations have also been used in reconstruction tasks. Due to their structure, they can be interpreted with CNNs, but typically require a large amount of memory to work with at high resolutions. Furthermore, small errors in generation can result in noise. Xie et al. [523] address this latter issue with a novel weighted voxel representation.

Deformation of template meshes is also applied to photo reconstruction for characters [557, 480, 538], and hands [309]. Additionally, human body meshes have been reconstructed from photographs for the purpose of testing clothing fit in e-commerce [1]. While RODIN [495] achieves the generation of 3D avatars from image or text input using a diffusion based model. Parameter based face generation is a common technique in games, allowing users to adjust features of a character, this is often achieved using 3D Morphable Models (3DMMs). Deep-learning approaches have succeeded in translating photographs to these parameters [417, 286]. Furthermore, the approach of Lin et al. [286], is capable of extracting texture from an input image, and applying it to a reconstructed face model. Fan et al. [108] introduce a pipeline for full head and face reconstruction based on 3DMM, and, [203] use a Siamese encoder-decoder architecture for face reconstruction. While 3DMM models represent face shape well, they typically lack fine detail. This is addressed in the method of Khan et al. [226], where meshes are refined via displacement, and Kuang et al. [247], where a GAN produces a depth map for a given image. Face generation can be achieved via GAN, as with the work of Kuang et al. [247], or PGAN [262], which generates faces with the use of geometry images [151]. The approach of Shamai et al. [409] employs geometry images and a GAN architecture based on progressive growing GANs [214]. StyleGAN [213], has also been expanded upon for learning 3D aware face generation [349]. In the method of Li et al. [272], a caricature mesh and texture are extracted from a single input photograph and combined. The texture is extracted using a GAN, and facial reconstruction is performed using the method of Deng et al. [87]. Facial landmarks may be used to improve the accuracy of facial reconstruction methods. Cai et al. [44] introduce a method for automatically detecting these landmarks for caricatures. Some approaches take sketches as input, for example, in the method of Delanoy et al. [82]. This method struggles with reconstructing thin structures, due to the resolution of the voxel space. This is addressed by combining voxels with normal maps [83], where an additional normal prediction network, based on Pix2Pix [192], produces normal maps from the input sketches. The normal maps are projected onto the voxel representation, refined, and used to generate a mesh that is far smoother than the previous results. Yang et al. [532] introduce a method for generating human body meshes from sketches, and a CNN is applied in converting sketch data to procedural model parameters [181].

Some research attempts to extract maximal information from a scene, attempting to either understand a scene holistically or identify and extract individual items. CDMD3DM [263], for example, reconstructs small scale indoor scenes using RGB-D data as an input, and Jeon et al. [199] produce accurate texture maps for RGB-D based scene generation. Full scenes have been reconstructed from single-view images, using cGAN architectures [237, 236] and CNN based approaches [509]. Some methods conceive scenes and objects using GAN based approaches, including point-clouds of outdoor scenes [418], entire cities from single-view images [230], and voxel-based scenes that are segmented by object class [425]. Alternatively, full scenes and individual asset classes can be successfully generated by utilising large language models (LLMs). 3d-gpt [454] achieves this by using LLMs as decision making agents that select a sequence of functions and inputs for the procedural modeling framework Infinigen [378]. This is shown to be effective at interpreting text descriptions from users and producing 3D results.

The placement of furniture within indoor spaces requires an understanding of the functional relationship between furniture types. Approaches to indoor layout generation largely apply deep-learning [268, 560, 201, 490] and a form of combinatorial optimisation referred to as case-based reasoning to the task [434, 435, 80]. SceneHGN takes a hierarchical approach to indoor scene generation, recursively breaking down the task into room, functional regions, objects and object parts [128]. In addition, transformer models have been successfully applied to interior arrangement tasks, such as ATISS [357] and SceneFormer [499], which are both faster and more versatile than previous approaches. DiffuScene applies a diffusion based model to the task of interior object placement, achieving better symmetry and diversity than ATISS in scene re-arrangement and completion tasks [460]. Floor plans can also be generated based on layout graphs via graph neural networks, as presented in Graph2Plan [176]

AAEs combine the encoder-decoder concept with the the adversarial mechanic of GANs. For example, Zhao et al. [563] apply AAE to style transfer, in which the latent representations of content and style images are learned within an encoder-decoder, evaluated by a discriminator. AAE have also been applied to interior object placement where an encoder-decoder generator is trained in an adversarial manner against a scene discriminator and image discriminator [560]. The image discriminator takes top-down views of the generated and real scenes, providing an extra visual based assessment.

Reinforcement learning involves learning a policy for a given task. This is achieved by placing agents within an environment and rewarding or punishing their actions in order to guide the policy. Some attempts at asset generation through reinforcement learning have been made, with approaches such as *double deep Q learning* (DDQN) and *deep deterministic policy gradient* (DDPG). For example Lin et al. [285], trains agents to reconstruct 3D objects by performing actions similar to human creators, placing primitives and refining geometry with the goal of matching a target model. To set an initial policy, IL is used in the form of *dataset aggregation* (DAgger). This research shows promise, though more work is needed to achieve the generation of detailed models. DDPG is applied to 2D layout generation [174]. In this approach, the network attempts to find an optimal layout for a randomised set of elements.

2.3.12 Determine Approaches stage

This stage entails the selection of a GAG approach, given a selected target asset type, technique and input type. To determine an approach for a given task, the pool of existing generative approaches are labelled based on:

1. the asset type they seek to generate,

2. the technique they implement, and
3. the type of input they take.

Given that A, in algorithm 2, represents the pool of possible generative approaches, the task of determining valid approaches is described as a process of filtering A in accordance to the three attributes above. Then, the remaining valid approaches are selected based on user preference.

It is possible for a single approach to not be sufficient for some tasks, particularly when the goal is to produce a large-scale system, with many inter-related outputs. For a hypothetical task of generating unique buildings that have interior layouts consisting of procedurally generated furniture, there may not be an existing approach that can achieve this on its own. Yet, when the task is broken down into object placement, building, and furniture generation, for which there are existing approaches, a combined solution can be formulated; by considering where an output of one approach can be the input of another, a pipeline can be formed.

Algorithm 2 Determining Approaches

```
procedure DETERMINE_APPROACHES(a, t, in, A[], u)    ▷ INPUT: asset type, technique, input type, approaches, user choices
    $A = [\forall app \in A \mid app.input\_type = in \wedge app.technique = t \wedge app.asset\_type = a]$    ▷
Filter choices by input type and technique, f stands for filtered
    if $|A| = 1$ then
        $approach_s = A[0]$
    else
        i=0
        while $i < |A| \wedge A[i] \neq u.approach$ do    ▷ Allow user to choose approach
            $approach_s \leftarrow A[i]$
            i = i + 1
    return $approach_s$    ▷ Pass selected approach to next step
```

2.3.13 Choose Datasets and Train stage

This stage details the selection process for datasets and training when using a deep-learning GAG approach. Due to the prevalence of deep-learning approaches in the literature it is necessary to consider data requirements in regard to training a model appropriately. Supervised and unsupervised deep-learning approaches typically require large labeled or unlabeled datasets, of which many exist publicly, such as ShapeNet [52], PartNet [329] or MPI FAUST [35]. The use of an established dataset can facilitate benchmarking and comparison between similar methods. Generative methods themselves can be used to produce synthetic datasets for other methods [227, 205]. Such datasets benefit from the additional control and variation that a PCG algorithm can facilitate, though at the same time, such datasets will inherit any biases in the generator. Summervile et al. [450] identify the difficulty in producing datasets for game assets, as a result of a lack in available data. There is also a lack of specific standardised benchmark datasets, as a result of how broad game content is.

Alternatively, reinforcement learning requires hand-crafted training environments and reward functions. Though usage of reinforcement learning as an approach to asset generation is largely unexplored, Lin et al. [285] propose an RL network that successfully learns 3D modelling policies.

This approach allows the RL agent to edit a 3D model using typical 3D modelling actions, rewarding it based on the similarity of the result to a target shape. With the usage of target shapes, this approach requires pre-existing data, much like supervised and unsupervised approaches.

Figure 2.6 presents the proposed process for choosing or creating a dataset in order to train a chosen deep-learning approach. If a supervised or unsupervised learning approach is chosen and the approach is already validated on the target asset type, then the original authors may have provided existing trained weights that can be used. Alternatively, if this is not the case, the original authors may have provided the dataset used to validate their approach. The applicability of existing weights or datasets can be determined by observing their outputs and comparing them to the target result. If they suffice then they may be used. However, if they are close to matching the desired result, a small dataset may be collected and used to fine-tune the model from the existing weights. If no existing weights or datasets are provided, then it may be necessary to obtain an alternative dataset that matches the data requirements of the chosen approach.

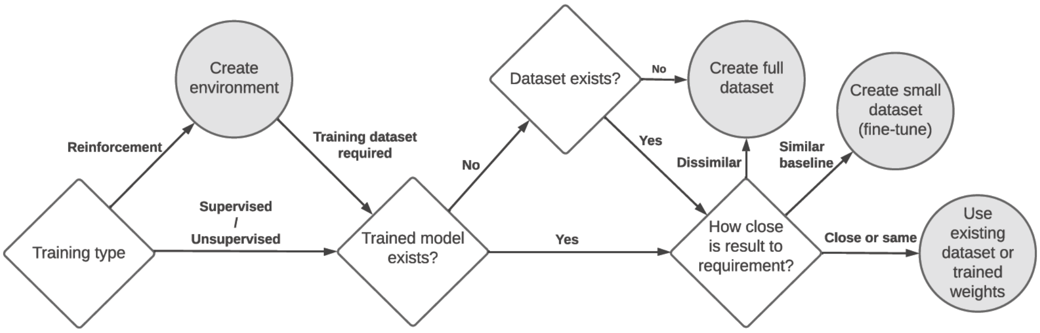

Training type
Reinforcement
Create environment
Training dataset required
Supervised / Unsupervised
Trained model exists?
No
Dataset exists?
No
Create full dataset
Yes
Yes
How close is result to requirement?
Dissimilar
Similar baseline
Create small dataset (fine-tune)
Close or same
Use existing dataset or trained weights

FIGURE 2.6: Approach to choosing or creating a dataset.

2.3.14 Generate assets stage

At this stage, the method for generating graphical assets is formulated and ready for implementation. When multiple sub-components are required, multiple approaches are needed. These approaches must be formed into a generative pipeline, with consideration for the necessary order of operation. This is achieved by chaining the approaches, mapping output to input. The order of approaches can be determined by considering each generative approach's pre-requisite data. For example, if the task was to generate a building with an interior containing furniture, the system may begin with a building generator and furniture generator, which both feed into an interior layout generator, as interior object placement requires both a defined environment and objects to place.

Once implemented the generative method should produce assets of the type defined by the user, given the required inputs. Depending on the approaches used, generated assets will be in 2, or 3 dimensions depending on asset type chosen, however there are multiple formats for presenting 2D and 3D data. The user's required format may differ from the format of the output. This can fortunately be rectified using algorithms that convert from one format to another. In the next section the final step in the framework, format conversion, will be discussed.

2.3.15 Formats and conversion

Format is an important aspect of a graphical asset, determining how the asset may be used, manipulated and presented on-screen. Each possible format has limitations and benefits. In the framework, figure 2.2, graphical asset formats are presented under two categories: *2D* and *3D*.

3D data can be presented as volumes, or a surfaces. Voxel representations are volumetric. This means that they determine the space that an object occupies. They are the primary format for presenting 3D shapes by volume, allowing for overhangs and tunnels in terrains [89, 424]. It is also common to convert voxel data to mesh data using marching cubes [303] or surface nets [138]. Surface representations such as meshes or point-clouds instead directly represent the outline of an object. Meshes are a common format for 3D content in games and other visualisations. A watertight mesh is a mesh that has a complete, connected surface [219, 77]. This is often a requirement for 3D content in real-time rendering and 3D printing. Alternatively a polygon soup lacks full connectivity [153]. Point-clouds, much like meshes, represent surfaces of objects or environments, and are used in the perception of real-world space, being applied to computer vision among other tasks [345]. Such data is obtained in abundance, due to being the natural output of 3D scanning technology. While point-clouds provide excellent spatial information, they lack structure, and are not typically used directly within digital media applications. Nonetheless, there is a great deal of research into generating point-cloud data [419, 345, 543, 283, 278].

With developments in deep-learning, novel internal representations of 3D form, lighting and texture have been developed, including neural radiance fields (NeRF) [324], and the nascent 3D gaussian splatting approach [224], further explored in the work of Liu et al. [296]. These representations are, however, reliant on the prediction and internal rendering of a model that has been trained on the subject, which makes them un-transferable without necessary conversion to more ubiquitous formats, such as voxel and mesh. Signed distance fields (SDF) have also seen increasing use as 3D data representations [62, 428, 567]. SDF represent 3D shapes by defining a function that returns the distance of a point from the nearest surface. The returned distance is negative when inside a surface and positive when outside a surface. This representation lends itself to deep learning approaches, as the shape representation itself is a function. This is opposed to direct mesh generation, which often requires a template to manipulate. Unlike voxels, which model occupancy on a fixed grid and can result in a blocky appearance when converted to mesh format, SDFs allow for smooth representation of arbitrary surfaces. SDFs require their own specialised rendering methods, such as ray marching, which differ significantly from the traditional mesh rendering pipeline commonly used in game engines. Therefore, direct use of SDFs for real-time rendering in games is uncommon outside of shaders and effects [473]. SDFs can be converted to voxel, point cloud and mesh formats [344].

2D assets take the form of either bitmap or vector graphics. Depending on the application and art-style, one form may be chosen over the other. Vector graphics benefit from being procedural, thus constituting comparatively smaller file sizes, and limitless levels of detail, though they are constrained to more simple or block-colour art-styles as a result. Bitmaps, however, allow for more expression in terms of art-style, and benefit in particular from CNN based generative approaches.

Though some approaches make use of implicit shape representations [184, 185], or signed distance fields (SDF) [567, 349, 428, 126], these forms are intermediate representations that must be converted into common formats, such as meshes or point-clouds, to be usable in rendering engines for games and 3D editing tools, and are therefore not considered graphical *asset* formats.

Data Conversion Methods

The output format of a particular generator may not match the user's desired format. In such cases, output data can be converted to another format using a one-to-one conversion method. These methods, unlike the generative approaches previously discussed, aim to translate data from one format to another with minimal loss of information.

The framework, figure 2.2, presents the conversion methods. These are drawn from conversion methods observed in the literature, though it is acknowledged that other methods may exist. Some methods are named, while others, such as voxelisation and rasterisation refer to general approaches that may vary in implementation, depending on the use case.

Marching cubes is a popular method for converting voxel data into a complete surface mesh [303]. The inverse conversion, mesh to voxel, is achieved through voxelisation [217]. Similarly, point-cloud voxelisation is achievable [170]. Conversion from mesh to point-cloud can be achieved through random point sampling [68], and conversion of point-cloud data to mesh data can be achieved using Poisson surface reconstruction [219].

For 2D formats, the process of converting bitmap data to vector data, vectorisation [93, 433], and the inverse, rasterisation [406], can be applied. Conversion from 3D to 2D can also be achieved by rasterising the asset at a single viewing angle, though the reverse of this cannot be achieved without a photo or sketch-based generative technique, as additional data must be inferred. Instead, conversion from 2D to 3D can be performed through visual hull [255], which requires multiple images at different viewing angles. Alternatively, deep-learning generative approaches such as [412] achieve format conversion, though results are less reliable. Conversion methods may also be used when multiple approaches are employed within a generator, for example one approach may produce a voxel output but the following approach may require a mesh as input. A conversion method may be used in this case to convert the voxel output to a mesh before it is passed to the second approach.

2.3.16 Verifying and Converting the Data Format

Figure 2.7 presents the process for selecting a conversion method. First the user's desired graphical asset format should be determined. If a conversion method exists between the output format of the approach and the desired format, this may then be used. If the output format of the approach matches the desired format, then a conversion method is not necessary. In the scenario where no conversion method is applicable, the user may consider selecting an alternative approach, or reconsidering the desired format.

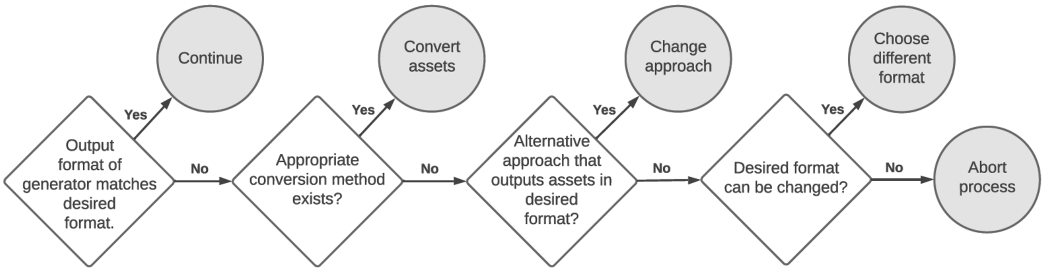

Output format of generator matches desired format.
Yes
Continue
No
Appropriate conversion method exists?
Yes
Convert assets
No
Alternative approach that outputs assets in desired format?
Yes
Change approach
No
Desired format can be changed?
Yes
Choose different format
No
Abort process

FIGURE 2.7: Process for selecting a conversion method.

Figure 2.8 presents the mapping between formats and conversion methods. Within each set of formats, 2D and 3D, data can be freely converted using a single method. When converting from 3D to 2D, however, it is suggested that the asset be converted to mesh format so that it may be rasterised. Likewise, when converting from 2D to 3D using visual hull [255] any vector graphics must be converted to bitmap.

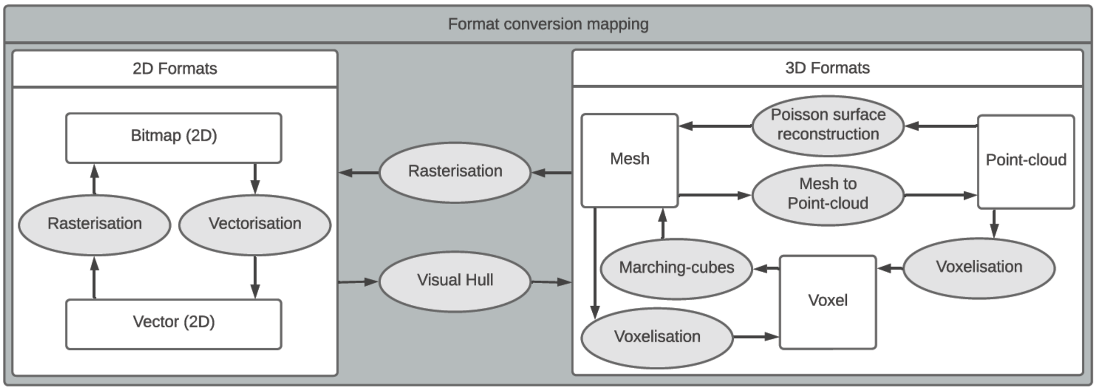


FIGURE 2.8: The mapping between conversion methods and formats.

2.4 GAGeTx: Expanded with Modular Techniques for Multimodal and Unimodal Generators

In recent years, more and more generative tools have begun incorporating multiple input types as a means to improve their versatility, and to allow users to specify what they want more precisely. Tools such as GPT-4o [347] and Google Gemini [142] bring such features to a world-wide general user base with the capability of reasoning across textual, image, video and audio mediums. Each type of input has its own strengths and weaknesses in presenting information. By integrating data from multiple input types, users can compensate for potential misinterpretations of one type and remix or alter ideas more effectively. Creative tools such as Adobe Firefly [5] and Midjourney [323] exemplify this form of interaction for graphical content creation.

This paradigm of generative deep-learning is referred to as *multimodal* for its integration of multiple modalities. Typically, multimodal models are generative models that simultaneusly interpret multiple modalities, that is, they are trained to understand different modalities of data at once. This requires advanced techniques in training to achieve. Multiple modalities may also be incorporated without centralising the interpretation, this was the initial approach to creating multimodal systems, where different modules were incorporated to handle different input types [18]. It is clear that the various techniques discussed in this chapter can each be utilised to achieve different results. While the present developments focus on singular models with multimodal capabilities, these are exclusively deep-learning based and specifically engineered for simultaneous multimodal reasoning. The breadth of generative approaches explored in this chapter, many of which not being deep-learning based, individually present useful capabilities that can be used in conjunction to produce graphical assets in useful ways. To support this latter form of multimodal generation and fully explore the potential of generative methods, it is necessary to refine the GAGeTx framework accordingly.

The techniques previously identified are delineated by the inputs they require and the purpose they fulfill. The approach then determines the implementation of the chosen technique, with said implementation determining the output type and format. If the output of one technique matches the input requirements of another, it is conceivable that the former could feed into the latter. Techniques can be chained together to allow users to introduce data of different modalities, or form a logical pipeline akin to the manual design and production process. This requires that techniques be re-framed as modular elements.

2.4.1 Multimodal generation

Multimodal methods are defined by their simultaneous interpretation of different input types, via a single process. These methods have the potential to reduce the difference between what users envision and the output they receive, thus reducing frustration. By providing opportunities to guide a generator through different modalities or input types, users can express their intentions more closely, with the option to supplement this with additional details that cannot be expressed in a single modality. For example, SDFusion [62] is a multimodal 3D asset generation method built to make asset production easier for users that are less experienced at asset creation. This method leverages text, image, and 3D input conditions to allow users to, reconstruct, generate and complete shapes. This is made more configurable by the inclusion of weighting between modalities, facilitating the prioritisation of one input over another. Baltrušaitis, Ahuja and Morency [18] identify two types of multimodal representation: *Joint* and *Coordinated*. In a joint representation, inputs from different modalities are internally merged into a joint space, while coordinated representations have individual tracks for each modality that are connected by some constraint. For example, Swamy et al. [457], take a modular approach by using modality specific encoders and decoders that can be connected to an internal embedding.

This is in contrast to unimodal techniques, which interpret a single input type or modality. These are the most common configuration of graphical asset generator in the literature. The many existing unimodal and multimodal methods applied in the literature, may be combined to form overall systems that incorporate different input types. This is relevant in complex tasks, in which multiple specialised processes feed into each other to form a *compound* generative system. For example, Nostalgin [209] tackles the challenge of reconstructing cities from historical images by forming a pipeline consisting of image adjustment, inpainting, and 3D model construction, accepting additional input parameters along the way. Alternatively, Du et al. [98] generate 3D buildings by chaining together specialised GAN models. Each GAN in the chain focuses on separate tasks in the creation of building models, from wall and roof textures, to the overall 3D shape.

Figure 2.9 presents the expanded GAGeTx framework, which includes a re-categorisation of techniques and a mapping of input types onto the corresponding interaction types. Here, techniques comprise two components: *Interaction* and *Process*. *Interaction* refers to the input modality or means of control for a given technique, while *process* refers to the way in which the method interprets inputs to produce an output. Under this framework, unimodal techniques can be expressed as a combination between a single *interaction* type and a single *process*, while multimodal techniques are expressed as multiple *interaction* types combined with a single process. Section 2.4.2 will discuss the modularity of techniques and present the taxonomy for technique *interaction* types and *processes*.

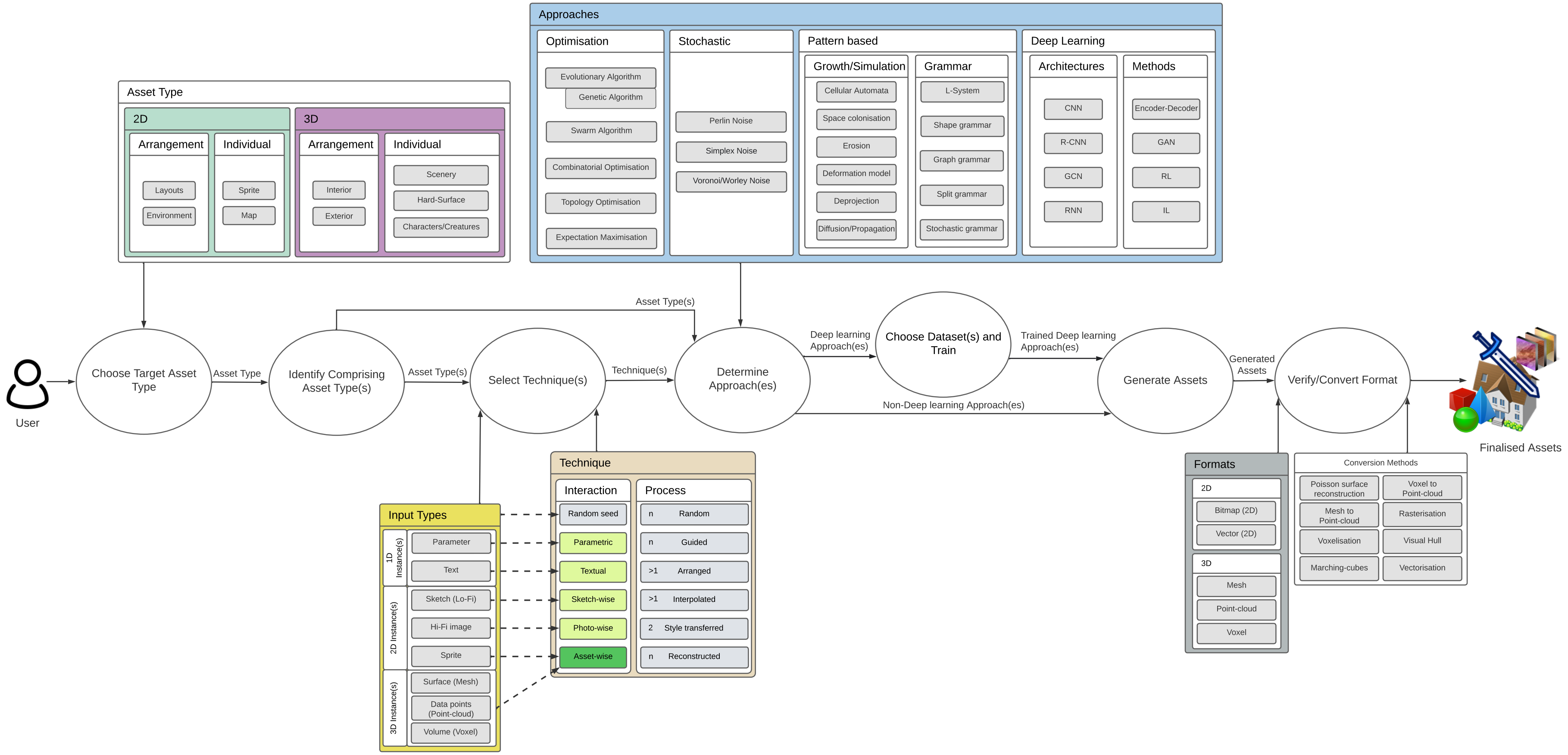


FIGURE 2.9: The GAGeTx framework with techniques re-categorised for multimodal applications.

2.4.2 Techniques

It is clear that within the literature and among emergent commercial products, interest is growing for combined generative methods that make use of a number of different input modalities for maximum control over generated artefacts. The categorisation of *techniques* within GAGeTx lack nuance in line with these needs and interests.

In many cases, multiple techniques can be applied in sequence to produce a single intended artefact [209]. This is even necessary in cases where the user wants to provide multiple types of input data. Figure 2.10 shows how data flows through a unimodal technique, which is comprised of one interaction type and one process. Output data from one technique can be passed as an input into another technique, or taken as a final artefact. This structure provides a large degree of control over the interactions and processes that comprise a generator.

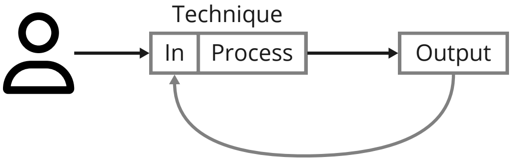


FIGURE 2.10: The data flow for unimodal techniques. Outputs of a technique can be used as inputs for another, therefore forming a "daisy chain".

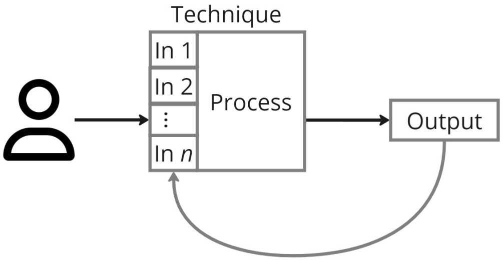


FIGURE 2.11: The data flow for multimodal techniques. Multiple interaction types connect to one process. This may also be chained together, as with unimodal techniques.

Figure 2.11 shows the data flow for multimodal techniques, which can be comprised of multiple interaction types, connected to a single process. This may also be chained together with other unimodal or multimodal techniques. Techniques have two key components that have relevance to an end user, these are: "how it is interacted with", and "what it does". The previous categorisations, while being directly derived from the literature, are rigid and are limited to the what has already been explored. Separating these components provides a more flexible framework for defining and examining generative techniques. Therefore, techniques have been re-categorised as interaction types and process types.

Interaction

The *interaction* type defines the "how it is interacted with" component of a technique. While many inputs map on to these interaction types one-to-one, there is an important distinction between the two. Input types represent forms of data, while interaction types represent the modality and purpose of the interaction. A multimodal technique is characterised by having multiple interaction types connected to a single process, while a unimodal technique will have only one interaction type. The count of each technique interaction type in the literature is presented in table 2.5.

Random seed: A random seed interaction may take data of any structure or format. The purpose of this input is to provide a non-meaningful starting point for a process, allowing for a pseudo random output. Most common associated input types include pseudo random numbers [147] or vectors [270, 212, 273], 2D noise such as Perlin [243] or Worley noise [115, 275] as well as 3D noise [424] in some instances. However, it is also possible for other forms of data to be used as a seed. As this interaction type imparts no meaningful user creativity, it can often be combined with other interaction types without the technique necessarily being considered multimodal, such as in the case of cGANs [500, 248].

Parametric: A parameter based interaction provides the user with the ability to specify exact attributes of their intended output. Parameters can take the form of floating point or integer numbers, booleans and vectors. This type of interaction is most commonly paired with other interaction types, as it can serve as a means for defining constraints within an algorithm. For this reason, as with random seeds, it can be combined with other interaction types without the technique necessarily being considered multimodal [248]. Alternatively, some systems use parameters as the sole input, such as in parametric models [206] and grammars [94, 10].

Textual: Text based interaction may take the form of a prompt, command, or request using alphanumeric characters. Most recently, development in large language models (LLMs) has brought this form of interaction to the forefront. Such deep-learning models interpret text prompts to produce outputs such as images [405, 91, 390] and 3D models [123, 368, 527, 126]. Textual interaction provides a natural way for users to specify or express ideas outside of graphical artistic skills. This makes for both a natural unimodal form of input [262, 405, 373] as well as a powerful interaction type for multimodal applications [62, 457].

Sketch-wise: Sketch-wise interaction takes in loose or incomplete ideas in the form of low fidelity images, designs or ideas. This form of interaction allows a user to quickly come up with ideas and allow the generative system to contribute its own style or fill in the details [181, 320, 415, 58]. This is not limited to 2D sketch inputs, as simple 3D inputs can also be considered sketches [414, 82].

Photo-wise: As opposed to sketch-wise interaction, photo-wise interaction takes refined or complete inputs. This includes what can be considered high fidelity images or scans, such as photographs[455, 492, 508, 60], concept art [423], RGB-D [339, 263, 552] and LiDAR scans [544, 418]. Here, the balance of initiative is shifted more toward the user, with the assumption that the input is a representation of how the output should look.

Asset-wise: Asset-wise interaction takes complete assets or parts of assets as input, usually for the purpose of arranging the former within an environment [267, 268, 369, 112], or piecing together the latter into new variations [153, 245, 84]. Here, the purpose is to take fully formed artefacts and form a new artefact out of them. In 2D, inputs may be sprites, while in 3D, inputs may be surface, data-point or volume based instances. As this interaction type requires finished assets as input, it is the user's initiative that most affects the quality of the output artefact.

Interaction type	N
Random seed	60
Parameter	43
Textual	47
Sketch-wise	32
Photo-wise	110
Asset-wise	42

TABLE 2.5: Count of techniques interaction types. Many methods demonstrate multiple interaction types.

Process

The *process* type defines the "what it does" component of a technique. In other words it encompasses the job that the technique performs on given inputs, via interaction types. These can, within reason, be mixed and matched with different interaction types depending on user needs. Where multiple interaction types are applied to a single process, the resulting technique is considered multimodal. Processes also have constraints when it comes to the number of inputs they expect, as shown in figure 2.9. The count of each technique process type in the literature is presented in table 2.6.

Random: This technique aims to convert an input, usually a seed, into an output without any specific user guidance. Therefore the creative task and conception of the artefact is achieved internally. This is usually used for the purpose of creating new ideas or inspiration [410, 401, 270, 243].

Guided: In this technique, the user provides inputs as a way to influence the generated outcome. Guidance can be of varying levels of complexity or detail, however the technique is expected to produce new content given these inputs [299, 181, 389, 245, 495]. This is in contrast to reconstructed techniques, where the object is to match the input content.

Arranged: This technique's job is to place existing assets within an environment, in either 2D or 3D space. Examples include graphical/UI design [266, 174], web design [306], furniture placement [499, 435, 201], room layouts [176] and object placement on terrains [442, 549, 112]. Arrangement also applies to part-wise shape completion [278, 153, 271, 62, 327]. These processes may incorporate elements of randomness as well as logic surrounding the usage of the input elements, or the overall balance of the layout.

Interpolated: This technique involves finding a point between two or more points within a space [488, 126, 130, 60]. For graphical asset generation, this usually means sampling a point that sits between chosen points in the regularised latent space of a deep-learning model [545, 345, 229, 459]. This process allows for blending between two or more examples to create variations or results that mix the characteristics of said inputs.

Style transferred: In this technique, the form of one input is combined with the style of another. This process inherently takes exactly two inputs. This may be used to quickly produce assets of a consistent style, or alternatively remix and alter existing assets to find variations or explore ideas. The most well known usage is in image style transfer [134, 420, 218], however this has also been attempted in 3D with texture [165] and shape [307, 119] transfer .

Reconstructed: In this technique, the goal is to accurately match or re-represent a given input, usually in a different modality or format. The input is therefore reconstructed, or converted to another modality or format. This type of process aims to predict or fill in missing data with minimal artistic liberty or contribution on the side of the software. Thus, the user takes creative initiative. Examples of use include reconstructing 3D building models [28, 387, 341], faces [286, 416, 87] or props [339, 297, 351] from images.

Process	N
Random	45
Guided	101
Arranged	25
Interpolated	22
Style transferred	16
Reconstructed	103

TABLE 2.6: Count of technique processes. Many methods demonstrate multiple technique processes.

2.4.3 Discussion

In this section, the GAGeTx framework has been expanded to accommodate growing trends in the usage of multimodal generative systems. To achieve this, the categorisation of techniques for graphical asset generation has been refined to better represent the nuances of generative tools and the usage of multiple input types. Techniques are comprised of two elements, *interaction types* and *processes*. This delineation accommodates both unimodal and multimodal configurations. Furthermore, compound generative systems can be formed by chaining together techniques, given that outputs of one technique can serve as inputs to another. Although this taxonomy presents the techniques that can be used for generating graphical assets, it does not provide guidance on which interaction types and processes are appropriate for different circumstances.

To expand on this, techniques can be seen to afford users a certain level of expressibility or creative freedom. As such the level of user initiative and input complexity can be described as the "idea fidelity", with no fidelity representing "no existing idea to work with", and maximum fidelity as "the asset is already fully designed". As the expressibility of a technique increases, so too does the complexity and onus on the user to pre-conceive an idea before the generator's involvement. Figure 2.12 shows the user idea fidelity ranges that each technique interaction type and process applies to. Technique processes are grouped under four headings: *Ideation*, *Guided design*, *Partial design* and *Full design*.

Random seed interactions can be used as the sole interaction type if there is no existing idea to base outputs off of. Whereas, parametric, textual, sketch-wise and photo-wise interaction can be used to guide a process, while affording the generator a wide range of variation and freedom. However, sketch-wise and photo-wise interaction types do require more specificity and effort at a minimum level. Arranged, interpolated and style transfer processes require inputs with more definition due to the direct inclusion of the inputs, or parts of the inputs in the output artefact. Interpolated processes are considered *partial design*, as the output is a combination of the inputs. This is dependent on how the generator interprets the features, thus the output cannot be fully determined by the user. Sketch-wise interaction's lo-fi nature limits it to basic forms of these processes. These are superseded by asset-wise interaction as higher *idea fidelity* is needed. This is also where the inclusion of random seeds becomes less relevant, particularly for interpolation and style transfer, as users typically define exactly how the inputs are to be combined. Reconstructed techniques are then used when the design is already complete, and the intended output is clearly envisioned by the user. This requires inputs that encapsulate the specific details of the intended artefact, such as exact parameters, detailed textual descriptions and detailed photo-wise inputs or assets. Here, the graphical asset generator acts as a method for converting these designs to a modality, dimensionality or format that the user desires.

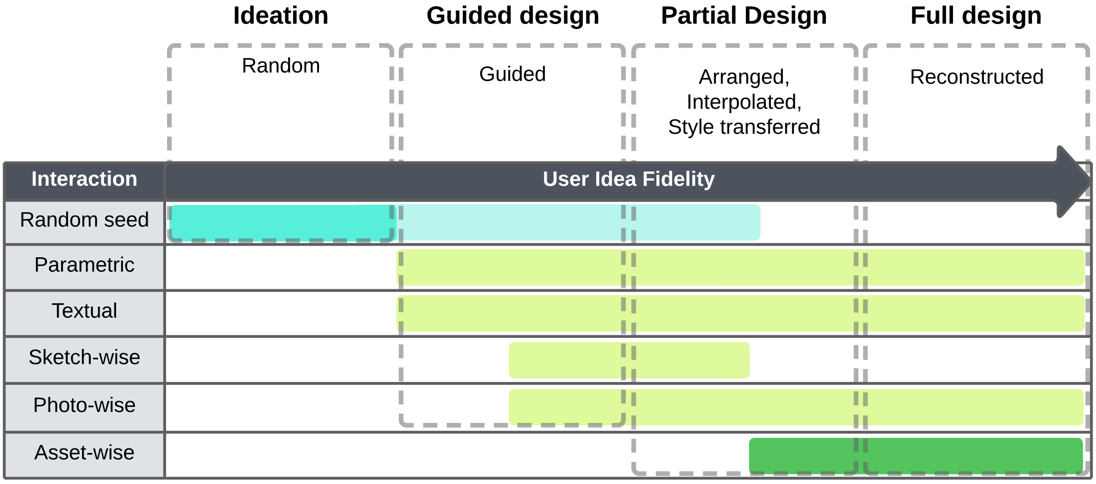


FIGURE 2.12: The user idea fidelity range that each interaction type and process inhabits. The lighter shade on random seed represents the range in which random seed can supplement but not be the sole interaction type.

2.5 Summary of chapter

In this chapter, a systematic literature review has been conducted, examining the existing research regarding graphical assets. From this literature has emerged key categories surrounding graphical asset types, generative techniques, input types, generative approaches, formats and conversion methods. These aspects have been ordered based on the prerequisites of each, and formed into a framework named GAGeTx. In recent years, approaches to multimodal generative systems have become more popular. As such GAGeTx has been expanded in light of this new research. The framework accommodates mixed and multimodal methods with a re-taxonomisation of techniques for modularity.

Chapter 3: Evaluation metrics framework

In the previous chapter, a systematic literature review of GAG methods was conducted, resulting in the formulation of the GAGeTx framework. While this framework categorises the five main aspects of GAGs, there is no way to compare the effectiveness of each specific implementation and thus determine which is best for a given purpose.

Quantitative testing and evaluation is paramount to the development and dissemination of effective software tools. While appropriate metrics are applied in assessing the various individual methods present in the GAG literature, there is no unified framework for metrics applicable across the gamut of methods. While deep-learning approaches have a well developed collection of applicable metrics [38] others are fragmented or otherwise context specific. Thus, in this chapter the literature from chapter 2 will be examined through the lens of evaluation processes to build a centralised understanding of evaluation metrics applicable to GAGs.

3.1 Evaluation metrics

There are various metrics that can be applied to GAG methods, depending on the type or format of the asset, or the generative technique employed. These metrics are categorised into three main classifications: *Operation*, *Artefact Validation*, and *Artefact Quality*.

Methods for generating graphical assets take many forms. These methods largely vary by target asset type and overall technique employed. Figure 3.1 shows the taxonomy of asset types and techniques found in the literature, introduced in section 2.3; related with the high-level metrics framework, contributed here. Methods for generating 2D and 3D assets work with vastly different forms of data. 2D data may consist of bitmaps made up of pixel values, or vector data that represents points in 2D space. Whereas 3D data may consist of meshes, point-clouds or voxels. The technique represents the general task that the generator completes, but also relates to a particular arrangement of input and output data types for a given method. Selecting the correct metrics for evaluating a generative method requires that the asset type and technique is known. For example, a method that uses a sketch guided technique to produce 3D mesh assets will take 2D bitmap or vector data as an input, and will output mesh data. With this knowledge, appropriate *operation*, *artefact validation* and *artefact quality* evaluation metrics can be selected. Low-level diagrams in figure 3.2, figure 3.4 and figure 3.5 show the expanded list of metrics found in the literature. Items are placed under 3D and 2D groupings, and all ungrouped metrics have

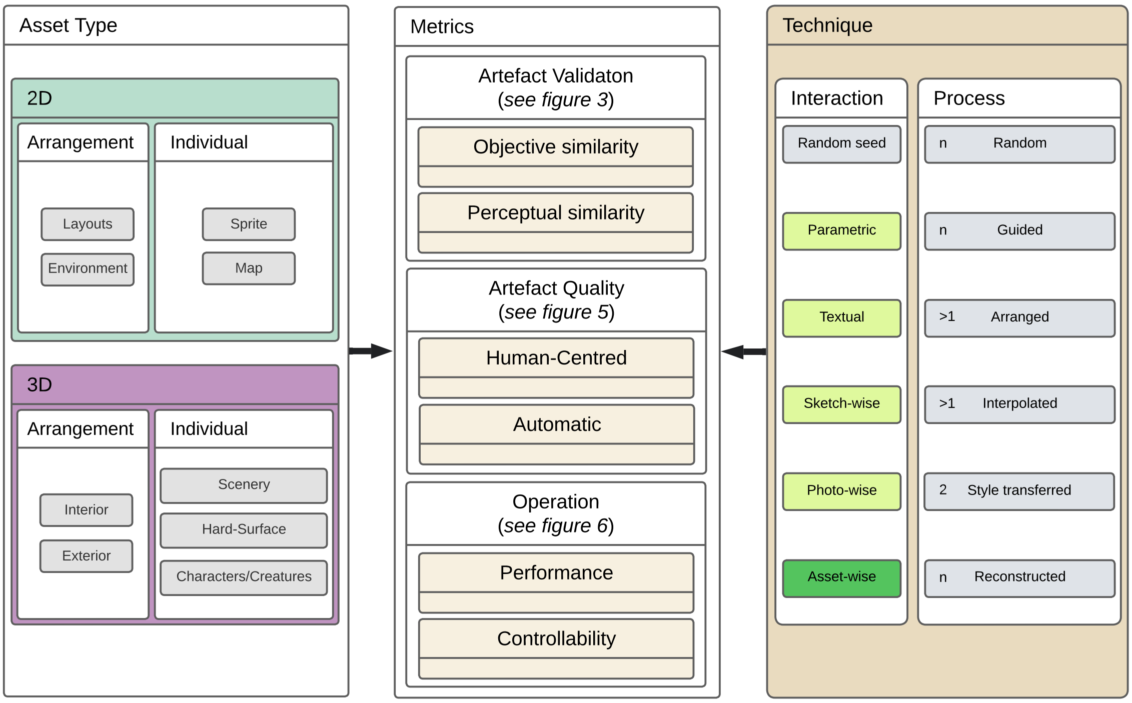


FIGURE 3.1: A high-level view of the metrics framework, with a taxonomy of asset types and techniques found in the literature.

been used in both 3D and 2D use cases. Exceptions relating to specific approaches or formats are marked with tags which are defined at the bottom of the figures. Metrics that occur only once in the examined literature are marked white, and metrics that occur more than once in the examined literature are marked grey.

Togelius et al. [464] put forth 5 desirable properties of PCG methods in games. These are: *Speed*, generation time; *Reliability*, the meeting of baseline expectations; *Controllability*, the user's ability to specify or steer content; *Expressivity/Diversity*, the variety of possible artefacts; and *Creativity/Believability*, the quality of artefacts. While it would be invaluable to have the capability of addressing each of these properties via quantitative means, only 4 of these properties are covered by the methods found in the literature. *Artefact validation* metrics assess the *Reliability* and to some extent the *Expressivity/Diversity* of the method via statistical aggregation. *Artefact quality* metrics assess the *Creativity/Believability* of the the artefacts, and operation evaluation, namely performance metrics, cover the *Speed* and scalability of the method. On the other hand, *Controllability* is harder to quantify, and thus no metrics were observed in the literature. This could be mapped to the method's technique classification by means of user degrees of freedom, though it is difficult to determine what constitutes a meaningful degree of freedom. For example, a sketch-based system affords the user a large amount of control over the outcome in exchange for a moderate degree of effort; whereas a seeded approach may require close to zero user input and thus affords very little control. A parametric approach may have varying degrees of freedom depending on the parameters exposed to the user, but whether or not these are meaningful controls is not objectively identifiable. In figure 4 of GAG input types have been ranked in terms of complexity.

Artefact validation consist of *objective* and *perceptual* similarity metrics. *Objective similarity* metrics assess the similarity between outputs and corresponding ground-truths and are often used in aggregate form for evaluation purposes. For these metrics to be applied, the ideal (i.e. ground-truth), corresponding data must exist for each output in the test. As such these metrics are largely seen in deep-learning generative approaches where this data is available as a matter of course. This naturally limits the relevance of objective similarity metrics to *externally conceived* methods, where the generator's job is to reliably interpret data of one form to another e.g. text-to-image or sketch-to-mesh. Conversely, *objective similarity* metrics are counter-productive if at all possible in cases where the generator should have creative agency, as they punish variation and reward conformity. Some common objective similarity metrics include, Mean squared error (MSE) [367, 400, 440, 459], Root mean square error (RMSE) [341, 87, 205, 342, 46] or Intersection over union (IoU) [574, 278, 336, 369].

Perceptual similarity metrics offer an alternative approach to validating artefacts, requiring less correspondence to ground-truth data. This is achieved either through human perception, or standardised deep-learning models that behave as a proxy for human perception such as Inception score (IS) [399] and Frechet inception distance (FID) [167].

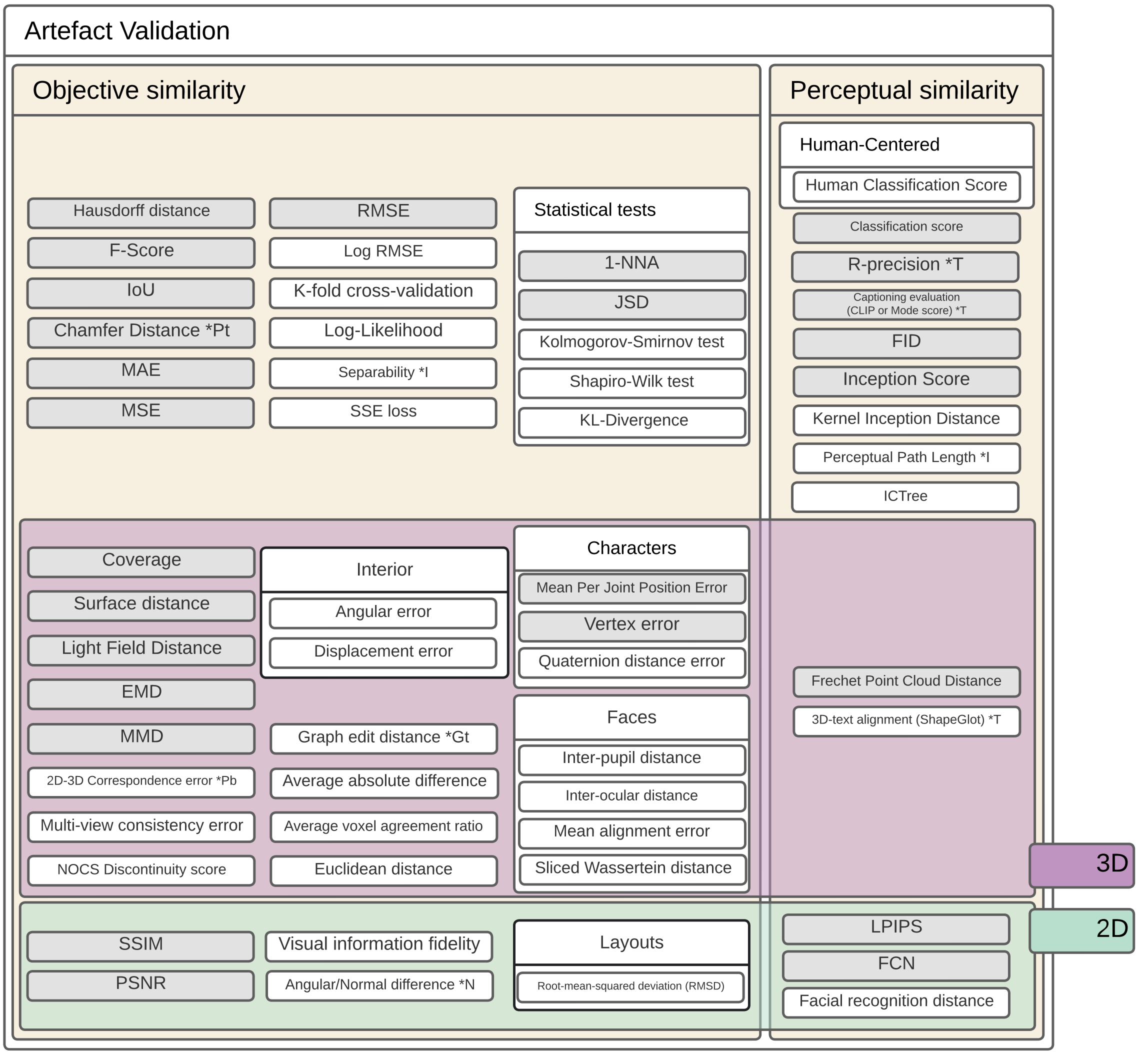
Artefact quality metrics assess the generator's ability to produce high quality outputs. This includes *human-centered* measures, such as questionnaires and rating systems [286, 397, 539, 91], and *automatic* metrics, measuring the particular characteristics of assets [20, 206, 514].

Operation evaluation includes *performance* and *controllability* measures. Performance metrics assess resources, such as memory [553, 94, 494] or time cost [77, 268, 153, 94, 494] of a method. While these metrics are important during development and testing, they are particularly important when applications go into production and release and run in real-time, as they have direct implications for user-experience and usability.

It is in the best interest of researchers and practitioners to evaluate the *validity* and the *quality* of artefacts as well as the *operability* of their generative methods, in order to assess and present the capability of their approach, and compare it with existing alternatives. Sections 3.1.1, 3.1.5 and 3.1.9 will cover artefact validation, artefact quality and operation metrics respectively.

3.1.1 Artefact Validation Metrics

A wide breadth of established metrics may be applied in validating the capabilities of generative methods as shown in framework figure 3.2. These artefact validation metrics assess the degree to which generated artefacts match the intended asset type. This can be achieved through objective or perceptual similarity measures, which are presented in section 3.1.2 and 3.1.3 respectively. Section 3.1.4 will present relevant procedures.



Key	Value
*V	Only for Voxels.
*N	Only for Normal maps.
*Pt	Only for Asset formats that consist of points, i.e. Mesh, Point-cloud and Vector graphics.
*G	Only for Grammar-based approaches.

Key	Value
*T	Only for Text-based Techniques.
*Gt	Only for Graph-based data e,g. roads or meshes.
*I	Only for Interpolated Techniques.
*Pb	Only for Photo-based Techniques.

FIGURE 3.2: The low level metric evaluation framework, focusing on operation metrics, and the various contexts in which they are applied. Keys at the bottom indicate their corresponding value within the diagram. White coloured boxes indicate metrics that only occur once within the examined literature, while grey coloured boxes indicate metrics that have more than one occurrence in the examined literature.

3.1.2 Objective similarity metrics

Objective similarity testing assumes that there is an exact intended output for every input. Therefore only applying to methods, often deep-learning based, that have the purpose of reliably re-interpreting an input in some way. However, this is not always a complete limitation. For example, with variational autoencoders (VAEs), such metrics can validate how well the method captures the desired features of a dataset, with the method itself still being capable of producing novel and varied outputs by sampling or interpolating between data points [130, 369, 530]. Many games follow a specific art direction, with requirements for the style of assets. Altogether, validating and training a VAE on data that fits these requirements would help ensure style consistency while still producing varied results. In other words, *objective similarity* can help ascertain in these cases, whether a method has successfully captured a target search space.

Objective similarity metrics are applied in the comparison between corresponding data points, such as mean absolute error (MAE) [553, 226, 384], mean squared error (MSE) [367, 400, 440, 459], root-mean-squared error (RMSE) [341, 87, 205, 342, 46] and sum of squared errors (SSE) [230]. MAE aggregates the absolute error of data-points, that is the positive difference between corresponding values. The squared values of SSE and MSE make them more sensitive to outliers [102]. SSE aggregates via summation while MSE aggregates via mean. The resulting values are in squared units, however. RMSE negates this effect by taking the root of the resulting value [38].

Intersection over union (IoU) measures the overlap between two volumes or regions, measuring the difference in shape and positioning between two assets. This has be applied in 2D for evaluating segmentation tasks [574] and layouts [278] or in 3D for comparing shapes [336] or bounding boxes [369]. Chamfer distance (CD) compares the difference between two sets of points by averaging the difference between each point and its closest point in the other set, thus not requiring a defined pairing or matching set sizes. This can be applied to all point-based asset formats such as point-clouds [543], and meshes [508]. Also used for point-based formats, Hausdorff distance finds the greatest difference between two sets of points [502, 240]. The F-score is a measure that combines the precision and recall for generated and ground-truth counterparts. This is used to score shape similarity between the two, therefore measuring the reconstruction quality of the method. This has been used to evaluate single-view [305, 265, 492] and multi-view [508] reconstruction approaches. For methods of interpolation, a separability score can be obtained to measure how disentangled the latent space is [213]. To test how well a deep-learning model generalises, irrespective to the set of training data, K-fold cross-validation may be employed [332]. This method splits the dataset into K subsets. For each subset, the model is trained on all other subsets, and tested using the subset in question. The mean and variance across tests can then be reported in the chosen evaluation metric. Zhao et al. [563] utilise log-Likelihood analysis in assessing an adversarial autoencoder's ability to capture the distribution of the training data.

Statistical tests may be necessary in the case of large deep-learning models, providing insight into the capability of the method through analysis over output distributions. These tests are purely statistical analysis such as analysis of means and standard deviations, and nearest neighbour classification and regressions. 1-nearest neighbour accuracy (1-NNA) assesses the similarity between two distributions using a 1-nearest neighbour classifier. This classifies each sample as belonging to one of the two groups, thus identical distributions should converge on an accuracy of 50%, therefore the closer the value is to 50% the better [529]. This approach takes into account the similarity in both the quality and the variation or diversity of the two sets, and has been used for 3D deep-learning approaches [567, 185, 545, 61]. Jensen–Shannon divergence (JSD) is used to measure the divergence between a ground truth and output distribution. For example, this is applied in point-cloud evaluation [270, 277]. Ivanov et al. test that their results follow a normal distribution, using Kolmogorov-Smirnov and Shapiro-Wilk tests [193].

Many similarity metrics are specific to 3D assets. For example, earth mover's distance (EMD) is used to measure the difference between an output and a ground truth distribution. This is used for point-clouds [345], voxels [119], and meshes [351]. Coverage scores the amount of similarity between two sets of point-clouds or meshes, for example, between generated results and a reference set [283, 545, 126]. Minimum matching distance (MMD) [126, 545, 271] instead gives a better representation of difference between the two sets, matching items by minimum distance and yielding the average of these distances [3]. Mesh reconstruction similarity can be calculated using point-wise euclidean distance between target and generated meshes [467, 455].

Surface distance metrics densely compare 3D points as a measure of surface similarity [237, 367, 423]. Light field distance metrics use multi-view renderings of the 3D assets to calculate shape similarity invariant to rotation [185, 126]. Whereas multi-view consistency error observes the distance between the same points at different viewing angles [261]. Here, a normalised object coordinate space (NOCS) representation is used. A NOCS discontinuity score is also introduced, to measure the connectivity of the surface [261]. Öngün and Temizel [346] introduce average absolute difference (AAD) and average voxel agreement ratio (AVAR) metrics, which measure the agreement between paired voxel shapes at different angles. For assessing data in graph form, graph edit distance [2] can be used. This represents the minimum amount of change required to transform one graph to another, and thus how similar their topologies are. A key aspect when generating characters is the joint angle. Mean per joint position error (MPJPE), vertex error and quaternion distance error may be used to measure this [557]. For interior object placement, bounding box displacement and angular errors have been used as metrics for correct positioning and rotation [369].

In the task of reconstructing caricature faces in 3D, inter-pupil and inter-ocular distance metrics have been used [42]. While sliced Wasserstein distance is used to compare the difference in ground-truth and generated face patch distributions [409], and mean alignment error may be used to find the average difference between vertex positions [203].

Alternatively, some similarity metrics are specific to 2D assets. For example, peak signal-to-noise ratio (PSNR) can be used to determine the strength of noise within an image, and therefore can be a measure of visual quality [286, 390, 108]. Structural similarity (SSIM) measures the perceptual quality based on high-level structure, comparing the output to a ground-truth image [505]. This has seen widespread use across image and texture generation approaches [286, 390, 440, 130]. Yadav et al. [528] use the visual information fidelity (VIF) metric [411] to assess the visual quality of their outputs. VIF measures the difference in visual information for the output image by assuming the ground-truth image to be a perfect signal [411]. Alternatively, when evaluating methods for generating normal maps, an angular or normal difference metric may be applied [449, 521]. These are similar to other pixel-wise error metrics, in that distances in pixel-wise values between two images are assessed. Angular error is reported as angles in degrees, as each pixel on a normal map represents a direction [449], while the normal difference metric reports non-angular values [521].

Root-mean-squared deviation (RMSD) has been applied in evaluating 2D layout generation by computing the discrepancy between the positioning of elements in generated and ground truth layouts [306].

3.1.3 Perceptual similarity metrics

While objective similarity is not always possible due to requiring one-to-one ground-truth data, perceptual similarity can be used to classify or score an artefact based on *perceived similarity* to a visual reference. Perceptual similarity in its simplest form may involve human-centered classification of artefacts. Here, a human evaluator will be given visual references to compare generated artefacts with, and tasked with judging whether the artefact is "real" or "fake" [130, 554], or whether it belongs to the target classification [307, 286], as mentioned in section 3.1.8. Alternatively, a variety of automatic perceptual similarity evaluators may be used. The *Inception* CNN model has been shown to perform well at image classification and detection tasks [458]. Since its introduction, subsequent versions of the *Inception* model have been used for evaluating generative models, such as generative adversarial networks (GANs). Metrics such as inception score (IS) [399], frechet inception distance (FID) [167] and kernel inception distance (KID) [29] each make use of the inner layers of the *Inception* model to compare latent similarities. Though IS uses this to evaluate the perceptual similarity of images with their expected class, FID and KID compare a distribution of generated artefacts with a ground-truth distribution. As a result, IS is limited to the categories that *Inception* is trained on, (typically ImageNet ILSVRC [394]) but requires no ground-truth data. However, FID and KID are not limited to assessing pre-defined categories but require a reference or ground-truth dataset for comparison. FID utilises Frechet distance between the two image distributions, and assumes that the two are Gaussian. This is applied to 2D assets [212, 76, 131, 420, 213, 218, 453], 3D assets via 3D classifiers [400, 298] or rasterisation to 2D form [123, 338].The Frechet point cloud distance extends FID for applications in assessing the similarity of point-based 3D shapes [419]. This has been used to evaluate many deep-learning based 3D point-cloud [270, 419, 270] and mesh [567] generators. KID instead utilises the maximum mean discrepancy between the image distributions, and does not assume a Gaussian form [29]. This has been used for evaluating faces in 2D [349] and for evaluating 3D assets rendered in 2D form [163]. IS however, does not take into account the statistical distribution of the data. It has been used primarily in the evaluation of 2D approaches [500, 107, 91, 405, 349], though it has been adapted for 3D as the 3D Inception Score, which uses a 3D classification network instead [489]. The 2D IS has also been applied to 3D assets that have been rendered in 2D form [163].

For the generation of trees, the ICTree metric can be used [365] as an automatic measure of perceptual realism, as an alternative to human evaluation. For VAE based interpolation, a perceptual path length metric is proposed [213]. This metric measures the perceptual distance at points along a path in the latent space, determining how smooth the interpolation is. For text-based generation methods, CLIP score [166] or R-precision may be used to measure the alignment between the output and the text prompt used to produce it. The latter is calculated using a Deep Attentional Multimodal Similarity Model (DAMSM) or CLIP-R-precision [354]. Extending this concept to 3D, the ShapeGlot dataset [4] can be used to train a similar alignment score model for text-3D generation methods [122]. A similar metric for faces is proposed [409], making use of features from a facial recognition network. For 2D tasks, learned perceptual image patch similarity (LPIPS) can be applied. This metric is trained on the Berkeley-Adobe Perceptual Patch Similarity (BAPPS) dataset, which consists of human perceptual judgements across many sets of images [555]. In a similar way, some approaches use alternative pre-trained fully convolutional networks (FCNs) as perceptual measures for 2D images [575, 192].

3.1.4 Evaluation Procedures

For methods that contain multiple sub-processes, e.g. deep-learning approaches. Ablation studies are a common procedure for assessing the effectiveness of these sub-processes or components [563, 369, 126, 412, 139, 417]. In an ablation study, components are systematically assessed for their contribution to the effectiveness of the method. This is achieved by removing the component and evaluating the rest of the method without it.

When developing generative AI systems, developers require a way to evaluate the quality and performance of their iterations against each other and against other existing methods. Benchmark datasets provide a controlled input to objectively compare outputs and performance. The challenge of benchmark datasets is that it might be difficult to always find a dataset that closely matches the intended input/output of the method being developed. Hence, may require slight adjustments to how the method functions. For methods that require image data, a popular dataset to use is ImageNet [111]. While general 3D datasets include: ShapeNet [52], ModelNet [519] and Pix3D [456]. Part-wise 3D methods may use PartNet [329]. Datasets are also available for certain specific object types, such as shoes (UT Zappos50K [536]) and birds (CUB-200-2011 [486]). Figure 3.3 shows the ground-truth datasets found in the literature, separated into 2D and 3D data types, and grouped by asset type.

When conducting objective similarity validation, the dataset must meet two requirements: 1) it must contain data that can be used as an input into the generative system and 2) it must contain corresponding ground-truth data in the same form as the artefacts. When conducting validation via perceptual similarity, wherein most methods compare 2D image data, as long as the data can be arranged as such it can be used. For example, mesh data could be rendered to create 2D data for perceptual comparison, as seen in the work of [567]. There are exceptions to this approach, namely with captioning evaluation and 3D-text alignment in which artefacts are scored on closeness to text descriptions. Alternatively, when using IS no dataset is required, because scoring is based on a pre-trained classifier.

TABLE 3.1: Usage of 2D metrics in literature, organised by generative technique process used.

Metrics		Random	Guided	Arranged	Interpolated	Style/ Transferred	Reconstructed
Perceptual similarity	LPIPS	[130]	[358, 390, 528, 53]		[130]	[218]	[297, 371, 130]
	FCN					[575, 192]	
	Facial recognition distance	[409]					
Characteristic	Visual balance			[174]			
Objective similarity	SSIM	[440, 130]	[390, 286, 520, 528, 542]		[130]	[384]	[130, 228, 297, 108]
	PSNR		[390, 286, 520, 528]	[430]			[371, 228, 297, 108]
	Visual information fidelity		[528]				
	Angular/Normal difference		[449]				[521]
	Overlap			[266, 267, 174, 242]			
	Alignment			[242, 266, 267]			
	RMSD			[306]			

TABLE 3.2: Usage of 3D metrics in literature, organised by generative technique process used.

	Metrics	Random	Guided	Arranged	Interpolated	Style/ Transferred	Reconstructed
Perceptual similarity	Frechet Point Cloud Distance	[270, 277, 419, 530]	[299, 298]		[277, 419, 530, 567]		[567]
	3D-text alignment (ShapeGlot)		[122]				
Characteristic	Flood extent		[20]				
	Stability	[206]			[206]		[206]
	Rootedness	[206]			[206]		[206]
	Symmetry score	[489]					
	Mesh intersection ratio						[265]
	CNR	[462]					
	Density	[462]					
Objective similarity	Coverage	[185, 270, 279, 271, 277, 278, 283, 419, 530, 126, 129]	[126]	[518]	[545, 345, 277, 278, 283, 419, 530, 126, 129, 567]		[345, 129, 235, 182, 61, 567]
	Surface distance						[367, 423, 237]
	Light Field Distance	[126, 184]	[126]		[184, 126]	[184]	[184]
	EMD	[185, 231, 279, 271, 98, 277, 278, 419, 530]	[152, 299]	[518, 128]	[545, 345, 277, 278, 419, 530]	[119]	[345, 289, 531, 351, 502, 493, 61, 412]
	MMD	[185, 279, 271, 98, 126]	[400, 126]	[518]	[545, 126, 567]		[61, 567]
	Correspondence error						[261]
	Multi-view consistency error						[261]
	NOCS Discontinuity score						[261]
	Graph edit distance		[338]				
	Average absolute distance	[346]					
	Average voxel agreement ratio	[346]					
	Euclidean distance		[152]				[467, 538, 455]
	Angular error		[449]	[369]			
	Displacement error			[369]			
	Mean Per Joint Position error		[532]				[557, 538]
	Vertex error	[459]			[459]		[309]
	Quaternion distance error						[557]
	Inter-pupil distance		[44]				
	Inter-ocular distance		[44]				
	Mean alignment error						[203]
	Sliced Wasserstein distance	[409]	[178]				

TABLE 3.3: Usage of metrics that do not require 2D or 3D data specifically, within the literature, organised by generative technique process used.

Metrics		Random	Guided	Arranged	Interpolated	Style/ Transferred	Reconstructed
Performance	Memory usage		[179]	[549]			[494, 94]
	Speed	[260, 147]	[416, 417, 548, 389, 59, 284, 140, 24, 304, 179, 44, 245, 153, 77, 415, 13, 89, 113, 342, 115, 383, 275, 220, 20, 147]	[268, 435, 434, 80, 266, 490, 306, 112, 499]			[82, 341, 83, 494, 199, 339, 428, 423, 132, 387, 272, 94, 514, 264, 28]
	Running cost		[59, 304]				[28]
	Encoded size		[113]				
	Grammar precipitate						[264]
Feedback	User score		[389, 59, 179, 397, 539, 286, 400, 373, 372, 45, 338, 91]			[575, 192]	[58, 290]
	Layout accuracy			[435]			
Perceptual similarity	Classification score		[498]	[490, 460, 242]			[543]
	Captioning evaluation		[539, 91]				
	FID	[213, 212, 349, 97, 126, 163]	[416, 123, 548, 556, 390, 397, 539, 550, 400, 520, 527, 495, 528, 122, 298, 338, 91, 453, 500, 172, 361, 405, 53, 382, 213, 126]	[128, 460, 242, 357]	[213, 126, 567]	[218, 420, 131]	[163, 76, 240, 567]
	Inception Score	[489, 163, 107]	[382, 405, 361, 542, 500, 91, 262, 372, 373, 390, 299]				[163, 272]
	Kernel Inception Distance	[349, 163]	[556]	[460]			[163]
	Perceptual Path Length	[213]	[213]		[213]		
Characteristic	Lines of code			[36]			[514]

Objective similarity	Hausdorff distance		[312, 62, 327]				[62, 327, 241, 502, 240]
	F-Score	[98]	[62, 327]				[62, 327, 265, 524, 362, 508, 305, 502, 492, 274]
	R-precision		[299, 195, 304, 368, 373, 372, 196, 361]				
	IoU		[299, 358, 400, 53, 327]	[369, 242, 430, 267, 176]	[545]	[575]	[292, 367, 562, 521, 92, 509, 319, 216, 362, 246, 336, 508, 305, 538, 297, 523, 502, 236, 526, 236, 82, 316, 547, 285, 544, 182, 398, 327]
	Chamfer Distance	[98, 277, 278, 419]	[152, 179, 62, 126, 327]	[518]	[345, 277, 278, 419, 545, 530, 126, 567, 184]		[345, 412, 543, 352, 139, 292, 92, 336, 289, 531, 502, 351, 492, 274, 494, 285, 62, 362, 362, 414, 58, 508, 538, 297, 182, 61, 327, 567, 184]
	MAE		[17]	[549]		[384]	[173, 226]
	MSE	[440, 459]	[400, 415, 17]		[459]		[367]
	RMSE		[312, 342, 230, 17]			[384]	[87, 491, 226, 182, 247, 341, 263, 46]
	Log RMSE			[205]			
	K-fold cross-validation	[332]					
	Log-Likelihood					[563]	
	Separability	[213]	[213]		[213]		
	SSE loss		[230]				
	1-NNA	[185]		[518]	[545, 567]		[61, 567]
	JSD	[98, 271, 277, 278, 418, 283, 129]			[345, 277, 278, 419, 283, 129]		[345, 129]
	Kolmogorov-Smirnov test	[193]					
	Shapiro-Wilk test	[193]					

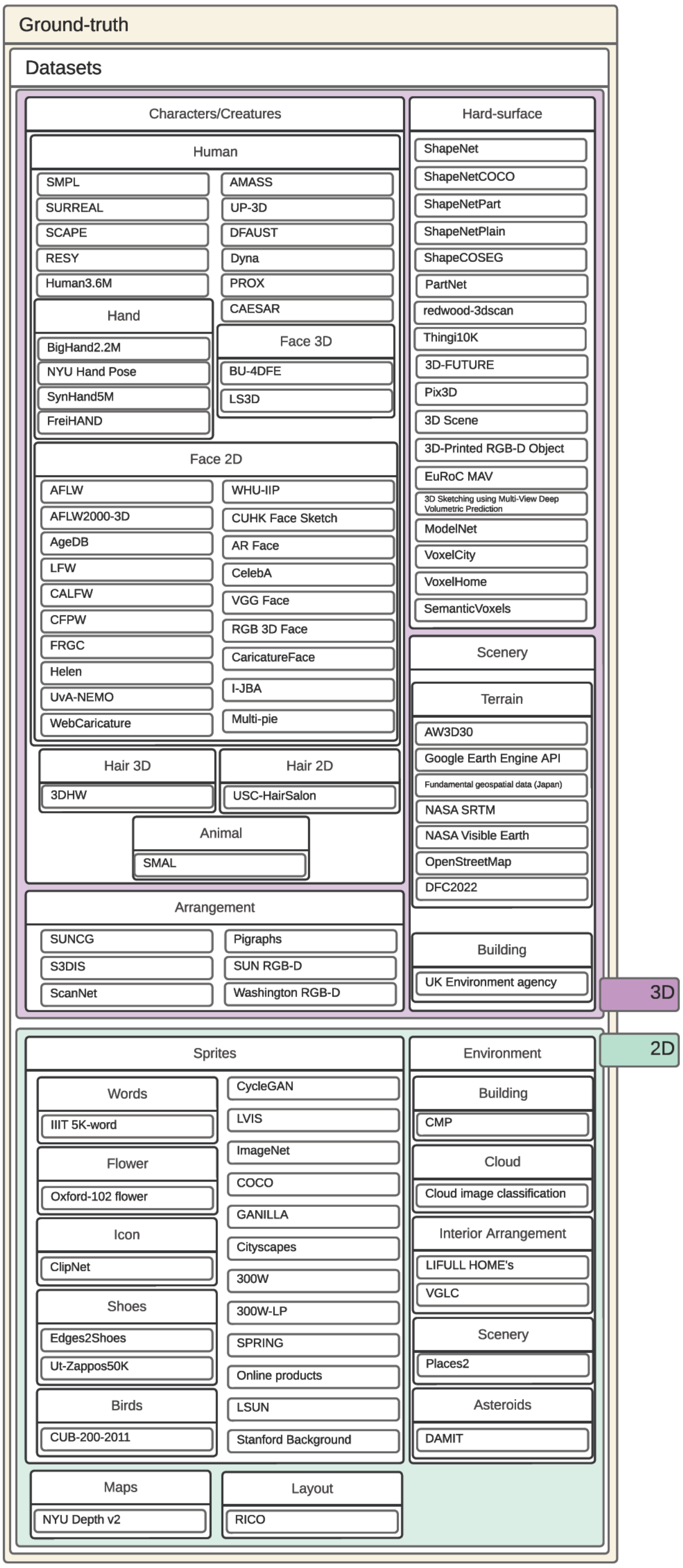


FIGURE 3.3: Ground-truth datasets from the examined literature, arranged by asset type.

3.1.5 Artefact Quality

Artefact quality evaluation methods are categorised as *human-centered*, or *automatic quality metrics*, shown in framework figure 3.4. *Human-centered* quality metrics rely on human assessment, while *automatic quality metrics* assess objective characteristics of assets. The following subsections 3.1.6 and 3.1.7 will cover human-centered and automatic quality metrics respectively. Section 3.1.8 will cover relevant evaluation procedures.

3.1.6 Human-centered quality metrics

Human perception and opinion can be used to measure asset quality. Such metrics are particularly valuable given the purpose of graphical assets, and their integration into games or other digital media where end-user experience and immersion are paramount. However, these measures can be inconsistent, and dependent on many factors. Thus, are best paired with more objective forms of evaluation i.e. characteristic metrics. Human perception is a common approach to evaluation in text-guided generative methods for example [397, 539, 91].

A preference metric may be used, where human evaluators choose the result that they prefer out of a selection of examples. These examples can consist of outputs from the method being evaluated and outputs from existing alternative methods. These questions can be posed as a general preference [286], or preferences for certain aspects of the result [397]. Users may alternatively rate or score the method on a scale [263, 561].

In assessing interior room layouts, [435] introduce a layout accuracy metric, which observes the number of furniture pieces a human evaluator chooses to move in a generated layout. In other words, this is the number of placements that the human evaluator is unsatisfied with.

3.1.7 Automatic quality metrics

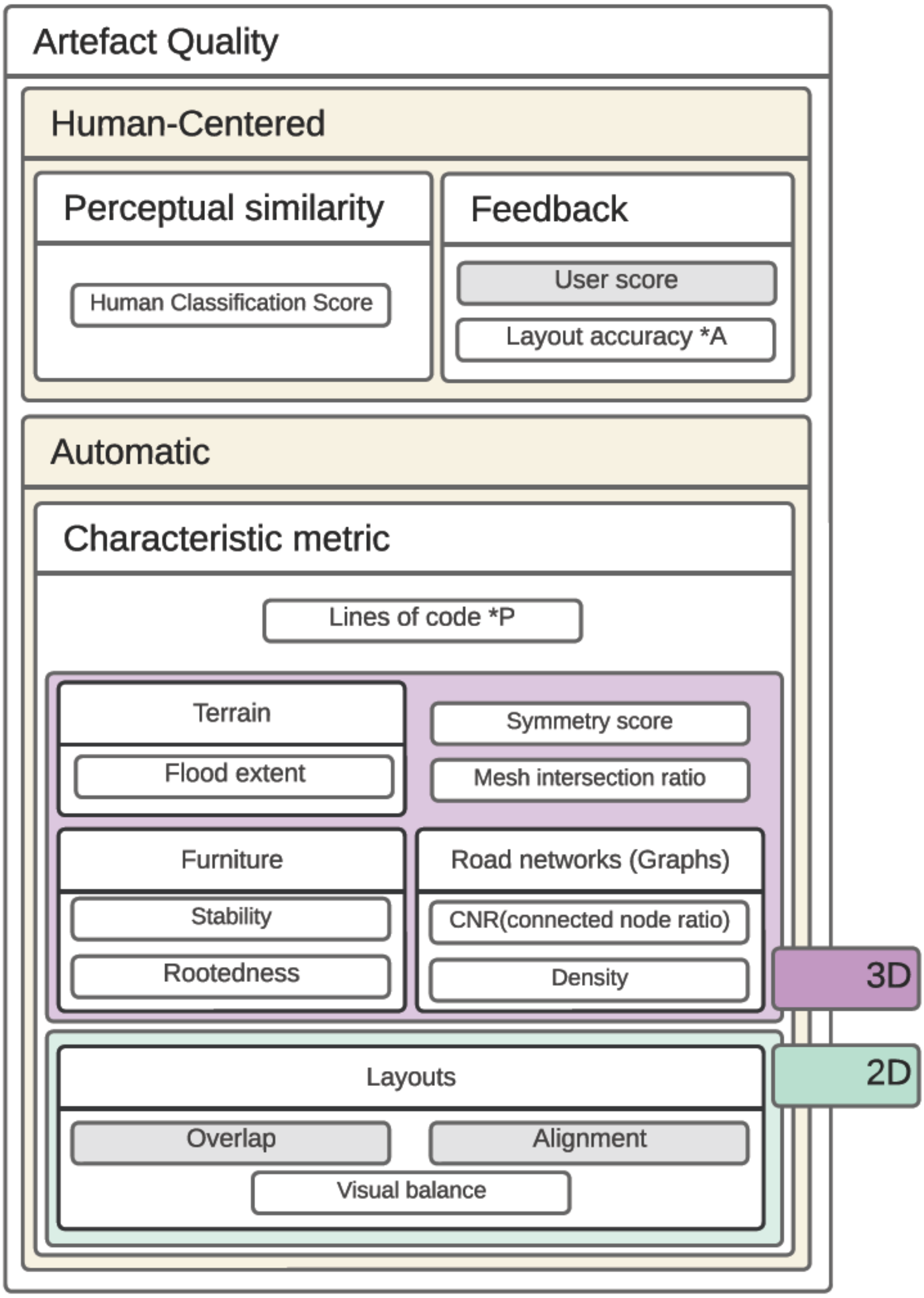
Automatic characteristic metrics may be applied in examining particular characteristics of generated assets. These metrics tend to be specific to the type of asset and format used. To choose or develop a characteristic metric, desired characteristics of an artefact must first be determined. This will be be dependent on the intended use case, and whether the characteristic can be objectively measured. For example, symmetry score [489] and mesh intersection ratio [265] are applied in quantifying the symmetry and self-intersection of 3D assets respectively. For road networks, or graphs in general, connected node ratio and density metrics [462] can be employed to examine the properties of the networks. Jones et al. [206] introduce stability and rootedness metrics for evaluating generated furniture assets, which use physics interactions such as gravity and pushing forces to test the generated shapes. While a measure of flood extent is used for measuring the realism of terrains, as natural formations tend to have a degree of drainage [20]. For 2D layouts a measure of visual balance has been used; considering the distribution of elements across the layout [174]. While the overlap and alignment of elements have been used as measures for layout quality, where the positioning of text and visuals are key. Here, a good layout will have minimal overlap between elements and maximal alignment [278, 267]. Where programming languages for shape creation are concerned [514], the number of lines of code has been used as a metric to determine the the language's efficiency or writing speed. Interestingly, the literature did not include any instances of symmetry score in relation to 2D assets.

Depending on the use case, there may be certain desirable quality requirements for the generated artefacts. For example, to consider a 3D model "game ready", it may need to meet geometric standards, such as having low self-intersection [265]. If the realism or functionality of a design is a concern, metrics similar to stability or rootedness will be of more relevance [206].

3.1.8 Evaluation Procedures

For the task of collecting human feedback or observing human perceptions, questionnaires are the primary method employed. When receiving quantitative feedback from users, it is common to obtain a scoring or ranking from participants. This is obtained through a Likert-scale for example [372]. In many cases Likert-scales are used to collect ordinal ratings, or binary choice questions may be used for classification. For text-to-image generation methods, DrawBench [397] may be used to benchmark and compare the performance of one method with another, providing a systematic list of prompts. For example, an experiment may ask users to compare two images generated via the same prompt, from different methods, and rate them in terms of fidelity or image-text alignment [397].

When developing a questionnaire for evaluating a generative method, consider general questionnaire design principles [244] and practices for implementing Likert scale questions [438]. As an alternative to rating, a questionnaire may present participants with a reference asset (i.e. image or 3D model) and ask them to choose between a number of assets based on similarity to the reference [307, 286], where one image is the output of the chosen method, and others are from comparative methods. Similarly, a set of images from various methods can be ranked [561]. A human classification score, or fooling rate may also be obtained by presenting participants with a "real" asset and a generated result, and asking them to select the one that is "real" [130, 554]. Here, the "real" asset does not necessarily have to be a photograph or an exact match to the generated result. The purpose is to see if the quality of the artefact can fool the participant into believing that it was not generated. This also acts as a form of artefact validation as, if the participant believes the artefact is a "real" example of a particular asset type, then it must be identifiable as such.



Key	Value
*T	Only for Text-based Techniques.
*A	Only for Arrangement asset types.

Key	Value
*I	Only for Interpolated Techniques.
*P	Only for Parametric Techniques.

FIGURE 3.4: The low level evaluation metrics framework, focusing on artefact quality metrics used in the literature, and the various contexts in which they are applied. Keys in the bottom-most table are used to indicate their corresponding value within the diagram. White coloured boxes indicate metrics that only occur once within the examined literature, while grey coloured boxes indicate metrics that have more than one occurrence in the examined literature.

3.1.9 Operation Metrics

Figure 3.5 presents the operation metrics used in the literature. These consist of various *performance* metrics and controllability. There are four performance metrics that can be applied to most methods, these are: memory usage, speed, time complexity and running cost. Alternatively, for grammars specifically, there are two performance metrics that can be applied, *encoded size*, and *grammar precipitate*. Memory usage and speed can be observed by monitoring the hardware usage or efficiency during runtime. In general, these metrics are dependent on the relevant specifications of the machine used, so these should be reported in any performance evaluation. It is often necessary to measure the speed of an implementation. A faster approach can allow for more content to be produced in a shorter amount of time. An example of speed testing can be found in the work of Delanoy et al. [82], where the method's performance with different numbers of inputs, and different hardware are compared. A common finite hardware resource is volatile memory such as dynamic random access memory (DRAM) for CPU based computation, and video random access memory (VRAM) for GPU based computation. These have been used as evaluation metrics [494, 553, 94]. Memory usage provides a benchmark for the minimum system memory required to run the method, determining the type of device it may operate on. The scalability of a system can be assessed by evaluating the time complexity of a method given the inputs [54]. This provides an indication for the speed of a method depending on its scale or number of inputs.

Running cost may be relevant for commercial projects or content generation services, for example, Bhatt et al. [28] report potential running cost of their generative method as a cloud-based tool. For grammar based methods, an encoded size [113] or grammar precipitate metric [264] may be used, determining the efficiency of the grammar encoding, and the versatility of extracted rules respectively.

Characteristics such as parallelizability, distributability and access to intermediate results [77], are contributing factors to performance. Though the review did not yield any quantitative measures for these, they should be considered per user need and expertise. Memory usage, speed, time complexity and running cost can all be impacted by these characteristics. Hence their impact can be measured through these metrics. While not observed in the literature, the controllability of a method could be measured in user degrees of freedom. While more degrees of freedom does not necessarily equate to more control, it could suggest more variability of input. But more controls can come at the cost of usability [388]. Instead, controllability could be indicated via user studies or indirectly through general usability assessment, such as System Usability Scale (SUS) [39].

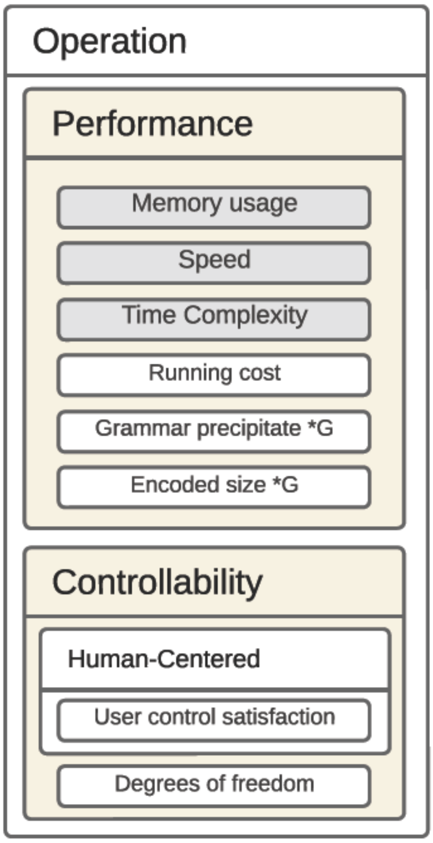


FIGURE 3.5: The low level metric evaluation framework, focusing on operation metrics used in the literature. White coloured boxes indicate metrics that only occur once within the examined literature, while grey coloured boxes indicate metrics that have more than one occurrence in the examined literature. *G indicates metrics that are specific to grammar based methods.

3.1.10 Selecting metrics

This section will present how to select appropriate evaluation metrics from those found within the literature. Figures 3.2, 3.3, 3.4 and 3.5 show the categorisation of validation metrics, ground-truth datasets, quality metrics and operation metrics, while tables 3.1, 3.2 and 3.3, show the literature organised by metric and technique used for 2D, 3D and shared metrics respectively. It is evident that some metrics are very popular for evaluating specific techniques, while others are popular regardless of technique. For example, PSNR has been used in 8 instances for evaluating image-based 2D guided and reconstruction tasks, while FID has been used in a total of 46 instances evaluating random, guided, arranged, interpolated, style-transfer and reconstructed generation tasks. There are many metrics that, due to being context specific, are only used in one or two instances, e.g. flood extent, stability, rootedness, inter-pupil and inter-ocular distances.

Figure 3.6 presents the selection procedure for operation, validity and quality metrics. This process is about finding available methods to consider for all three metric types. In this process, relevant metrics are first narrowed down by category based on various limiting factors, and then the dimensionality of the asset type. Visiting the relevant figure a list of metrics can be obtained from headings related to the asset type, and more general metrics: figures 3.2, 3.3, 3.4 and 3.5. These metrics can then be found in the relevant tables, where the choice can be narrowed down based on the technique and by observing their application in existing examples cited: tables 3.1, 3.2 and 3.3. Metrics applied to the same technique may be considered more relevant. If the intention is to compare with existing methods, then a popular metric for that technique should be chosen. To identify the technique a method's inputs and functionality should be considered, as discussed in section 3.1. For artefact validation metrics, human-centered or automatic approaches can be used.

Algorithm 3 Algorithm for selecting metrics.

```
procedure SELECTING EVALUATION METRICS(asset_type, input_type, AIM)    ▷ Prioritise metrics applied to same technique/input_type, consider relevance of metrics based on the example uses in the literature.
    if AIM = Does the generated asset look like what I wanted? then
        return Dataset, ValidationMetrics ← E1(asset_type, input_type)
                                                         ▷ Choosing a validation metric.
    if AIM = Are these assets "good"? then
        return QualityMetrics ← E2(asset_type, input_type)
                                                         ▷ Choosing a quality metric.
    if AIM = How operable is this approach? then
        return OperationMetrics ← E3(asset_type)
                                                         ▷ Choosing an operation metric.

function E1: ARTEFACT VALIDATION(asset_type, input_type)
    Dataset ← E1C1(asset_type, input_type)
    ValidationMetrics ← E1C2(Dataset, asset_type, input_type)
    return Dataset,ValidationMetrics

function E1C1: CHOOSE GROUND-TRUTH(asset_type, input_type)
    if Relevant dataset in figure 3.3 for given asset_type then
        if input_type matches dataset input type then
            return Dataset ← dataset matching asset_type and input_type
        return Dataset ← dataset matching asset_type
    if Ground-truth dataset can be created or has been used in training then
        return Dataset ← created or unused portion of training dataset

function E1C2: CHOOSE VALIDATION METRIC(dataset, asset_type, input_type)
    if there is an expected output for every input_type then
        if dataset contains asset_type and input_type then
            if asset_type is 3D then
                Metric options ← Relevant 3D Objective similarity metrics
in figure 3.2 based on asset_type.
                Metrics ← Find metric options in tables 3.2 and 3.3.
            else if asset_type is 2D then
                Metric options ← Relevant 2D Objective similarity metrics
in figure 3.2 based on asset_type.
                Metrics ← Find metric options in tables 3.1 and 3.3.

    if dataset matches the intended general appearance of asset_type then
        if asset_type is 3D then
            Metric options ← Relevant 3D Perceptual similarity metrics
in figure 3.2 based on asset_type.
            Metrics ← Find metric options in tables 3.2 and 3.3.
        else if asset_type is 2D then
            Metric options ← Relevant 2D Perceptual similarity metrics
in figure 3.2 based on asset_type.
            Metrics ← Find metric options in tables 3.1 and 3.3.
    if consider human participation then
        Metrics ← Human Classification Score (table 3.3)
    if asset_type belongs to an ILSVRC [394] category then
        Metrics ← Inception Score (table 3.3)
    if appearance of asset_type can be put into words then
        if asset_type is 3D then
            Metrics ← ShapeGlot (table 3.2)
        else if asset_type is 2D then
            Metrics ← ClipScore (table 3.1)
    return Metrics

function E2: ARTEFACT QUALITY(asset_type, input_type)
    if consider human participation then
        Metrics ← Human classification or feedback (table 3.3)
    if asset_type has measurable characteristics then
        if asset_type is 3D then
            Metric options ← Relevant characteristic metrics in figure 3.4 based on asset_type
            Metrics ← Find metric options in tables 3.2 and 3.3.
            Metrics ← Devise new characteristic metric/s based on use case.
        else if asset_type is 2D then
            Metric options ← Relevant characteristic metrics in figure 3.4 based on asset_type
            Metrics ← Find metric options in tables 3.1 and 3.3.
            Metrics ← Devise new characteristic metric/s based on use case.
    return Metrics

function E3: OPERATION(asset_type)
    Metrics ← Memory usage, speed and time complexity (table 3.3)
    if consider human participation then
        Metrics ← User control satisfaction
    if intended to be run as a service then
        Assess running cost (table 3.3)
    if asset_type is grammar based then
        Assess grammar precipitate and encoded size (table 3.3)
```

The purpose of artefact validation is to assess whether artefacts are of the intended asset type. Artefact validation via objective similarity is only possible when ground-truth data is available. In most cases, evaluation of a method via ground-truth datasets is only possible with reconstruction method in which there is a known intended result for a given input e.g. conversion from sketch-to-mesh. Many of such methods, often deep-learning based, will already use this form of validation as a means to optimise the generative model itself. Naturally, applying these metrics to the final generative system using data previously unseen to the model will assess its ability to perform the reconstruction task. To compare a method with existing approaches a benchmark dataset may be used, though the availability of a relevant dataset depends on how popular the asset type is for other generative systems. For example a generic 3D reconstruction task has many options for benchmark datasets, while a method that produces a highly specific type of artefact may require a bespoke dataset. Perceptual metrics require less direct conformity to a ground-truth. Here, artefacts can be compared with datasets that represent the general intended appearance of the artefacts. Metrics such as FID can be used to assess visual similarity between generated artefacts and a reference dataset without one-to-one correspondence. Alternatively, IS does not require any ground-truth data, but can only be used if the asset type belongs to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) categorisation [394]. Human-centered validation is achievable with any method, though it may not always be feasible due to the time it takes to recruit and access participants. The creator of a method designs or trains their system based on their conception of what attributes define the asset type. Human-centered perceptual similarity testing may be used to confirm if others deem the artefacts to meet their own conceptions. Typically a description, example or reference will be provided as a point of comparison for the assessor. This is not an assessment of quality, but a subjective assessment of whether artefacts are perceived as what they should be, i.e. is the method for generating chairs producing what can be deemed a chair. Where possible, both automatic and human validation metrics should be applied, though the former is only possible with externally conceived methods.

Artefact quality evaluation metrics can either be automatic or human-centered. These can be selected based on relevance to a use case. Automatic quality evaluation can be achieved through characteristic metrics. These are metrics that apply to specific characteristics of an artefact, and will be dependent on the asset type, data format or end use case of the artefact. For example, assessing the stability of furniture under physics simulation [206], or the quality of a mesh by the amount that it intersects with itself [265]. These make assumptions about what makes a quality asset e.g. a good chair will remain upright when exposed to gravity, and a good mesh surface will not intersect with itself. As such, it is difficult to systematically choose characteristic metrics as it is highly context specific. Alternatively, human-centered feedback through scoring can provide a more subjective, opinion based assessment of the quality of artefacts. Though finding participants and collecting this data will take more time than automated testing. Both should be used where possible.

Operation evaluation includes performance and controllability measures. Memory usage, speed and time complexity can be applied to any generative algorithm. These measures are beneficial for determining the resources required to use a given method. Additionally, in the case of generative methods as a cloud service, running cost may be assessed [28], while grammar based methods may benefit from assessing grammar precipitate [264] and encoded size [113]. Controllability should be considered based on the context of use. While there are no direct quantitative measures, users may be asked to indicate their satisfaction with the controls, whether they are too limited to achieve their goals, or too complex to comfortably use.

In most cases, multiple metrics of each category will be relevant. Where feasible, applying every relevant metric will provide the most precision in evaluating a method. This is not always possible due to time constraints, therefore, in such cases individual judgement should be used to prioritise which metrics are used.

Algorithm 3 presents the process for selecting all three types of metric based on the artefact asset type, the input type of the method, and the evaluation aim. To illustrate the use of this algorithmic approach, the following scenario is presented. Game developer A has produced a method for generating 3D models of castles using a GAN architecture. The method is using a random technique process with a randomised input vector. Firstly, game developer A validates if the outputs of the method are in fact "castles". They choose a dataset and validation metric (E1) by first examining if there are any ground-truth datasets available for buildings (E1C1). Looking at buildings in figure 3.3, there is LiDAR data from the UK environment agency, which has been used for finding roof shapes [544]. This is not so relevant to their use case. Not finding an existing similar dataset, they can use an unused portion of the dataset they used for training. They proceed with selecting a validation metric (E1C2). As they use a *random* technique process, there is no clear expected output for every input. Once they validate that their dataset matches their intended artefact appearance, they consider general or 3D perceptual similarity metrics as their asset type is 3D. Due to the challenges of human participation, they decide not to use human classification. Therefore, they move on to other perceptual similarity methods which includes Inception Score, where "castles" is a category of ILSVRC, figure 3.2 and table 3.3. Then, developer A proceeds to to assess the quality of their artefacts (E2). As before, they reject human participation and use an automatic characteristic metric. They choose to measure mesh intersection ratio based on figure 3.4 and table 3.2, as self-intersecting geometry can look bad and waste resources. Finally, to evaluate the operability of their method (E3), developer A assesses the memory usage, speed and time complexity. They once again reject human participation, so do not assess controllability. They run their GAN solution locally, not on cloud, and their method does not use grammars, therefore they do not assess running cost or grammar metrics.

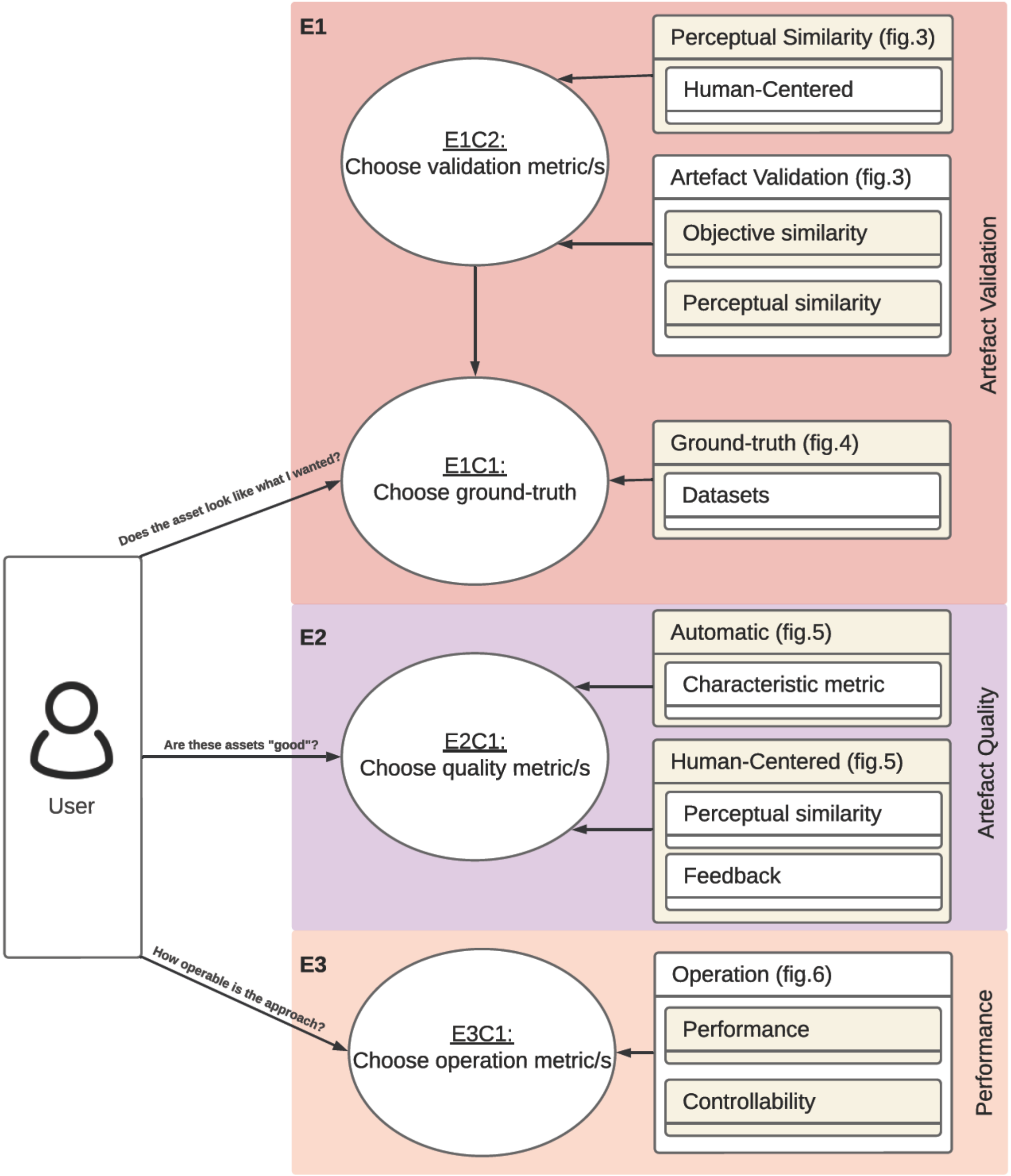


FIGURE 3.6: The process for selecting appropriate operation, validity and quality metrics. Box colours match counterparts in figures 3.2-3.5.

3.2 Chapter summary

In this chapter a framework has been developed pertaining to the selection of metrics for GAGs. This framework, based on the same body of literature as chapter 2, presents and categorises the range of evaluation approaches and metrics in use. GAGs can be evaluated through *artefact validation*, *artefact quality* and *operation* metrics, evaluating whether generated artefacts pass as the intended asset type, how "good" generated artefacts are and how controllable or performant the implementation is, respectively.

While this chapter examines methods for evaluating the effectiveness of GAG methods themselves, practical tools also require appropriate user interfaces (UIs) and user experiences (UXs). The following chapters will address these aspects such that GAG tools can be effectively evaluated and appropriately presented to users. Chapter 6 will revisit and improve the application of the metrics framework, by exploring how users prioritise certain evaluable aspects of GAGs.

Chapter 4: Game Developer and Designer UX preferences

In the previous two chapters, the existing literature pertaining to graphical asset generation has been systematically reviewed, resulting in the development of a conceptual framework dubbed GAGeTx. Furthermore, the evaluation metrics used in the same body of literature has been examined to form an additional framework for the evaluation of graphical asset generators. While these frameworks lay out and taxonomise the existing approaches and techniques, the rationale behind the selection of said approaches and techniques is loosely defined as a user "choice" or "preference".

In this chapter a user study has been conducted with 16 game designers and developers in order to shed light on the greater context in which asset generators can be used, as well as observing preferences for their usage and implementation. In this study, three mock-up graphical asset generator systems were implemented and tested via repeated measures. Data collection was achieved and guided through a web form. Through statistical analysis, four key insights were observed.

4.1 Background

While PCG methods are a popular topic, with many practitioners and researchers suggesting their value in "offline" use cases, there are few existing works that examine generative tools in game development from the perspective of game designers and developers, i.e. the users of such tools. It is apparent that different generative methods and techniques are designed with different use cases in mind, each attempting to solve a problem or streamline a process within a larger creative pipeline. Game designers and developers use many tools during game production to create video game products, from editor tools for game engines, to asset editing software such as Photoshop [7] and Blender [32], or integrated development environments such as Visual Studio [322]. These tools each have their place within the workflows and development pipelines of game creators. While these workflows look different from company to company or from one individual to another, there exists three ubiquitous stages of development: prototyping/design, production and testing [379]. All graphical assets are formulated and produced during the prototyping, design and production stages. Within this, there are many ways that PCG can help, from early inspiration and creating rough placeholders to creating fully fledged assets or remixing existing assets for variety. Each of these use cases, naturally, will have different requirements for the complexity of input, method of interaction and quality of output. However, no research has yet explored the purposes for which designers and developers prefer to use PCG tools.

Mixed initiative PCG (MI-PCG) is a common instance of offline PCG. The user study of Walton et al. [487], examines user opinions of an MI-PCG level creation tool using a mixed-methods approach. The findings suggest that the MI-PCG method stimulates user creativity, and helps to inspire new ideas. They discover that the inclusion of qualitative data from participants is important for gaining the full picture, as it provides important context to the quantitative data.

The concept of designer modeling extends MI-PCG by considering the adaptability of a system to user needs [281]. In such a system, the tool aligns itself to the designers style choices, or may even help break design fixation by producing results that are different to what the user is accustomed to, thus providing inspiration.

In a qualitative study examining expectations of Finnish game developers regarding their tools, Kasurinen et al. [215] present some key insights. Across the seven organisations examined, developers were largely satisfied with the tools they have, though, when it comes to assets, many preferred to purchase rather than create in-house. It is found that the companies *'expect their tools to allow easy prototyping and ability to design while implementing'* [215]. Many of the companies relied on third-party game engines, and thus compatibility with these engines was a large part of the consideration when selecting new tools.

While there is an extensive body of literature pertaining to MI-PCG there are limited case studies which examine their use and fit in design and development pipelines [251]. While it is clear that MI-PCG has the potential to improve game development asset pipelines, the opinions and preferences of game designers and developers remain, in general, unheard. Individual MI-PCG systems may be validated with user feedback [267, 415, 412], but feedback is limited to specific purposes of use, and are not contextualised within the larger scope of the development pipeline. The goal of this study is to obtain insights into where and how graphical asset PCG is most useful to game designers and developers.

4.2 Applications of GAGs in game design and development pipelines

The creation of graphical assets occurs during the prototyping, design and production phases of the game design and development process. Graphical asset generation may be utilised throughout each of these phases to streamline or augment the creation of games. Through observing existing generative methods, tools and pipelines in the literature, seven potential usages for graphical asset generators have been identified:

Generating inspiration
Creative block and design fixation [197] are widely experienced in creative fields, including game production. Generative methods for graphical assets can be used to help circumvent this issue by providing unexpected or varied outputs that can serve as inspiration. In this context, the quality

of the generated content is less important than its ability to spark the user's creativity, serving as a starting point or trigger for further design iterations.

Exploring ideas
During early stages of design, iteration is key for expanding on initial inspirations and producing finalised designs. During this process, designs are flexible, thus ideas can be quickly tested and evaluated before committing extensive resources and time. The speed and configurability of some generative methods such as [397] may help with exploring ideas and potential design directions for this use case.

Creating placeholder assets
Placeholder graphics are assets that are used as a stand-in for future intended game elements. They are used during the prototyping phase to allow for the testing of game features and mechanics. As the intention is for these assets to be replaced during later stages, quality and detail is unnecessary. However, to serve fast prototyping, placeholder assets must be fast to produce.

Creating variations of existing (complete) assets
In games, asset variation can help to immerse players and reduce the repetitiveness of content throughout a digital environment. Many generative methods facilitate the ability to create similar variations of assets. This is most commonly seen with methods that rely on parametric modelling [207, 136], or with the usage of VAEs [459]. Therefore, there is potential in the usage of generative tools for increasing the diversity and richness of content available, making the digital environment more engaging and varied for players.

Creating assets from existing (complete) designs
During the design phase, the appearance of an asset may be determined and planned through the use of concept art. This concept art is later used as reference for creating the final asset during the production phase. A graphical asset generator may be used to achieve the latter task, by converting initial designs and plans into finalised assets. For example, [423] use orthographic concept art from multiple views to generate 3D assets, even utilising the initial art for texturing.

Creating assets from scratch
Some generative tools, such as SpeedTree [190], encompass a full pipeline for designing, configuring then producing assets. In these cases, the process covers the creation of assets from scratch.

Player (or user) made content
A common application of PCG, beyond its use in generating graphical assets, is for *online* content generation. This typically refers to the generation of game elements such as levels and loot to produce unexpected and re-playable experiences. With regard to graphical assets however, some games such as Spore and Dreams provide tools that allow players to create their own designs within a constrained creation framework. In a sense, many of these tools are constrained versions of development tools, there is therefore potential for the application of generative tooling within similar systems. Generative systems can also be used for avatar personalisation, such as through the reconstruction of faces from photographs [286].

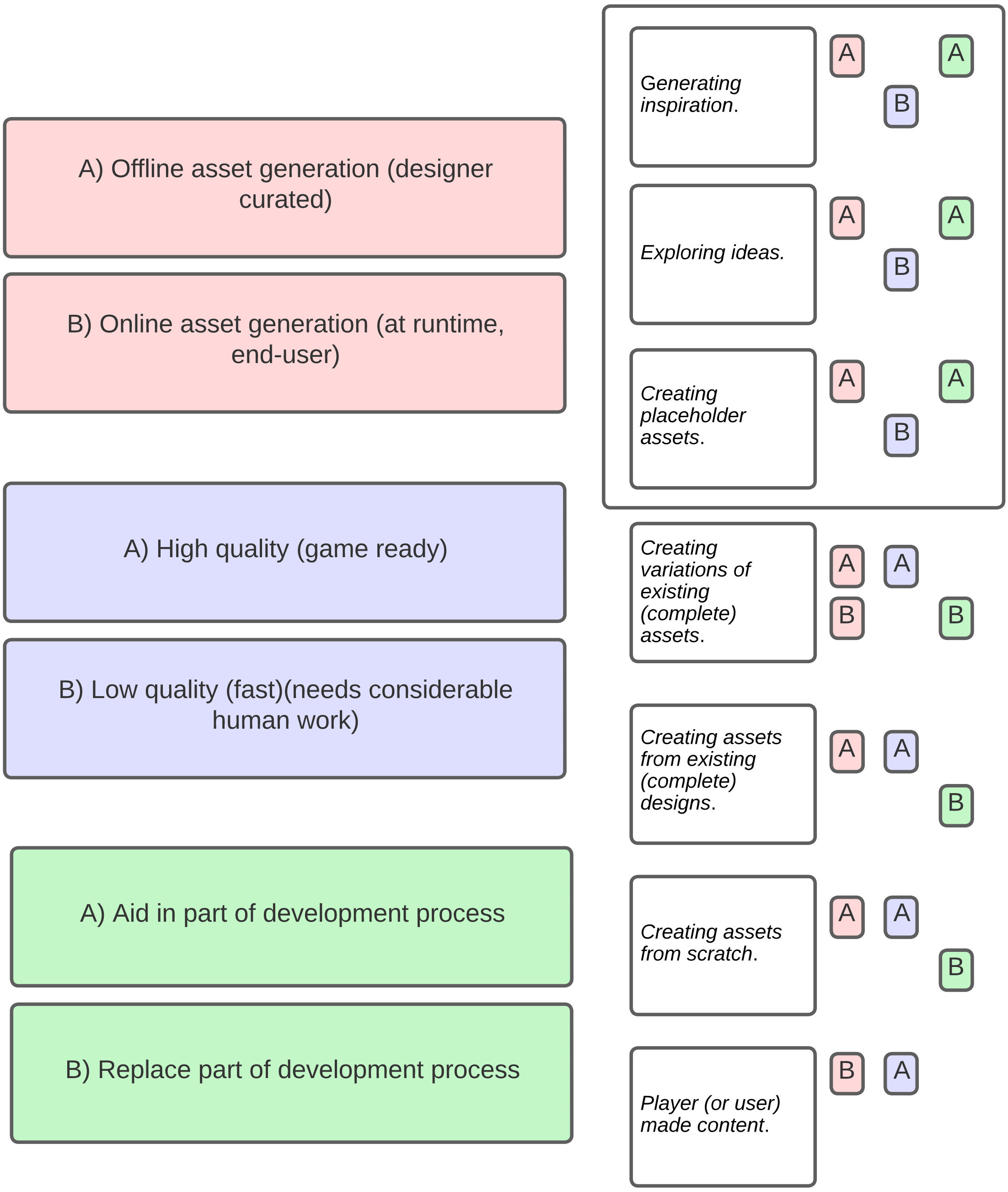

A) Offline asset generation (designer curated)
B) Online asset generation (at runtime, end-user)
A) High quality (game ready)
B) Low quality (fast)(needs considerable human work)
A) Aid in part of development process
B) Replace part of development process
Generating inspiration.
A B A
Exploring ideas.
A B A
Creating placeholder assets.
A B A
Creating variations of existing (complete) assets.
A A B B
Creating assets from existing (complete) designs.
A A B
Creating assets from scratch.
A A B
Player (or user) made content.
B A

FIGURE 4.1: The key characteristics of design and production uses. *Generating inspiration*, *Exploring ideas* and *Creating placeholder assets* are grouped together as they share the same characteristics.

Figure 4.1 presents the key characteristics of each game centred use case. As shown, graphical asset generators may be applied *offline* during the production of a game product, or *online* during run-time. Uses may also necessitate high quality outputs, such as in *creating variations of existing designs* and *creating assets from scratch*, or low quality outputs with an emphasis on speed, such as in the case of *generating inspiration*. These uses may also aid in streamlining part of a development process, such as with *exploring ideas*, or replace an entire process, as with *creating assets from* *scratch*.

These characteristics represent the underlying acceptable qualities of the usages, helping to map different generative methods to specific stages in the development pipeline. For instance, during the early design phase, a generator prioritising speed over quality can help by rapidly generating ideas, thereby encouraging creative exploration. Alternatively, high-quality generators are more suited for final asset production, where precision and detail is necessary.

4.3 Research questions

The purpose of this study is to examine if game designers and developers would find generative systems for graphical assets useful, *what* their requirements are for such as system and *where* in the pipeline they will find most use in it. There are five core questions this study seeks to answer, shown below in order of importance.

CQ 1 Would a generative system for asset generation be useful to game designers/developers?
CQ 2 Where in the design/development pipeline would game designers/developers find value in such as system?
CQ 3 What are the expectations regarding speed and quality for such a system?
CQ 4 Is this type of system preferred as integrated or stand-alone?
CQ 5 Which type/s of UI interaction do game designers/developers prefer for this type of system?

A mixed methods approach is taken to collect quantitative and qualitative data from participants. This methodology attempts to answer a series of research questions, which expand the above; presented in table 4.1 as RQ1-RQ9, by proving or disproving a relative set of hypotheses, listed H1-H7 in table 4.2 and demographical impact hypotheses IH1-IH3 listed below. To achieve this, three prototypes have been devised, as shown in figures 4.2, 4.3 and 4.4.

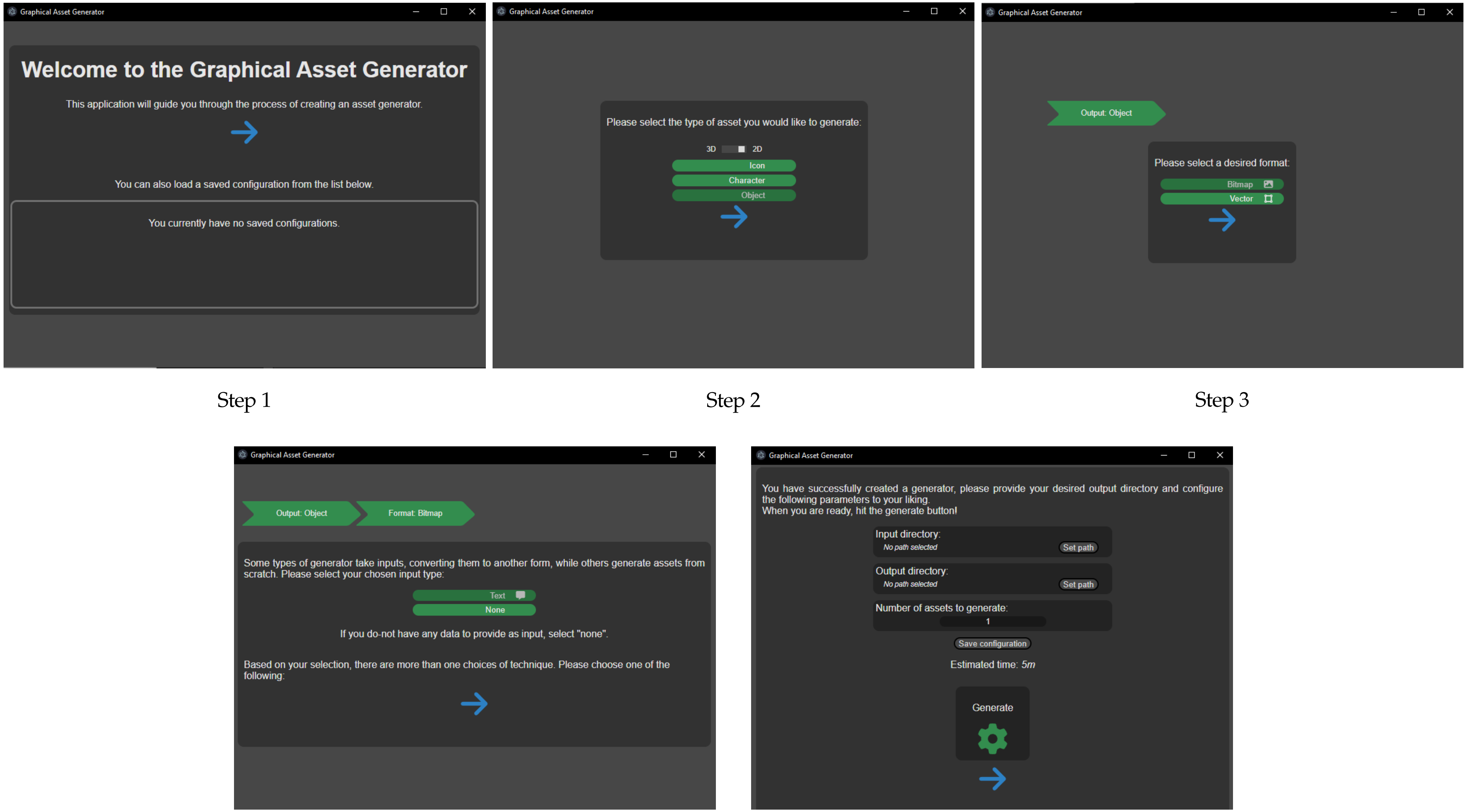
IH1.1 Role impacts preference.
IH1.2 Size of team impacts preference.
IH2.1 Experience impacts preference.
IH2.2 Age impacts preference.
IH3.1 Gender impacts preference.

TABLE 4.1: List of research questions and associated questionnaire questions.

	Research Questions	Survey Questions
RQ1	Which prototype/mock-up (M_i) is deemed the most useful to game designers/developers?	"I would find this software tool that generates graphical assets useful in the projects I work on."
RQ2	Where in the pipeline would game designers/developers find value in M_i?	"Where in your development pipeline would you find value in this software tool?"
RQ2.1	Do designers/developers find value in M_i for Generating inspiration.	
RQ2.2	Do designers/developers find value in M_i for Exploring ideas.	
RQ2.3	Do designers/developers find value in M_i for Creating placeholder assets.	
RQ2.4	Do designers/developers find value in M_i for Creating variations of existing (complete) assets.	
RQ2.5	Do designers/developers find value in M_i for Creating assets from existing (complete) designs.	
RQ2.6	Do designers/developers find value in M_i for Creating assets from scratch.	
RQ2.7	Do designers/developers find value in M_i for Player (or user) made content.	
RQ3	If M_i were to be used in a pipeline, would designers/developers prioritise volume of output or quality of output?	"Considering your answer to the previous questions, would you prefer if this tool generated a large variety of assets at a lower quality, or that it generated a small variety of high-quality assets?"
RQ4	How much time would designers/developers find acceptable for M_i to take in generating a single asset?	"Please indicate the largest timescale you would find acceptable for generating a single graphical asset, if you were to use this software tool in your projects."
RQ5	Do designers/developers prefer M_i as a stand-alone solution or integrated into a game-engine?	"Given the option, would you prefer such a system to exist as stand-alone software, or integrated into your game engine editor of choice?"
RQ6	How important is the ability to configure or modify M_i according to designers/developers?	"Please rate on a scale of 1 (not important) to 5 (very important), how important to you, is the ability to configure or modify such a tool. For example, importing your own bespoke algorithms or trained models?"
RQ7	Which M do designers/developers identify as more useable?	System Usability Scale [39]
RQ8	What are the concerns and perceived benefits of this tool?	
RQ9	What are the desired asset types?	

TABLE 4.2: List of hypotheses and associated research questions.

Hypotheses		Associated RQs
H1	Participants prefer an integrated solution over a standalone	RQ1, RQ5
H2	Participants favour the integrated window interface type over an inspector integrated interface	RQ1, RQ6.1
H3	Participants find the tool valuable in all stages	RQ2
H4	Participants prefer shorter generation times over quality/variety	RQ3
H5	Participants find asset generation times of less than a minute acceptable	RQ4
H6	Participants prefer an open solution i.e., high-configurability	RQ6
H7	Participants identify the solutions useable	RQ7



Graphical Asset Generator

Welcome to the Graphical Asset Generator

This application will guide you through the process of creating an asset generator.

You can also load a saved configuration from the list below.

You currently have no saved configurations.

Step 1

Graphical Asset Generator

Please select the type of asset you would like to generate:

3D
2D
Icon
Character
Object

Step 2

Graphical Asset Generator
Output: Object

Please select a desired format:

Step 3

Graphical Asset Generator
Output: Object
Format: Bitmap

Some types of generator take inputs, converting them to another form, while others generate assets from scratch. Please select your chosen input type:

If you do-not have any data to provide as input, select "none".

Based on your selection, there are more than one choices of technique. Please choose one of the following:

Step 4

You have successfully created a generator, please provide your desired output directory and configure the following parameters to your liking.
When you are ready, hit the generate button!

Input directory:
No path selected

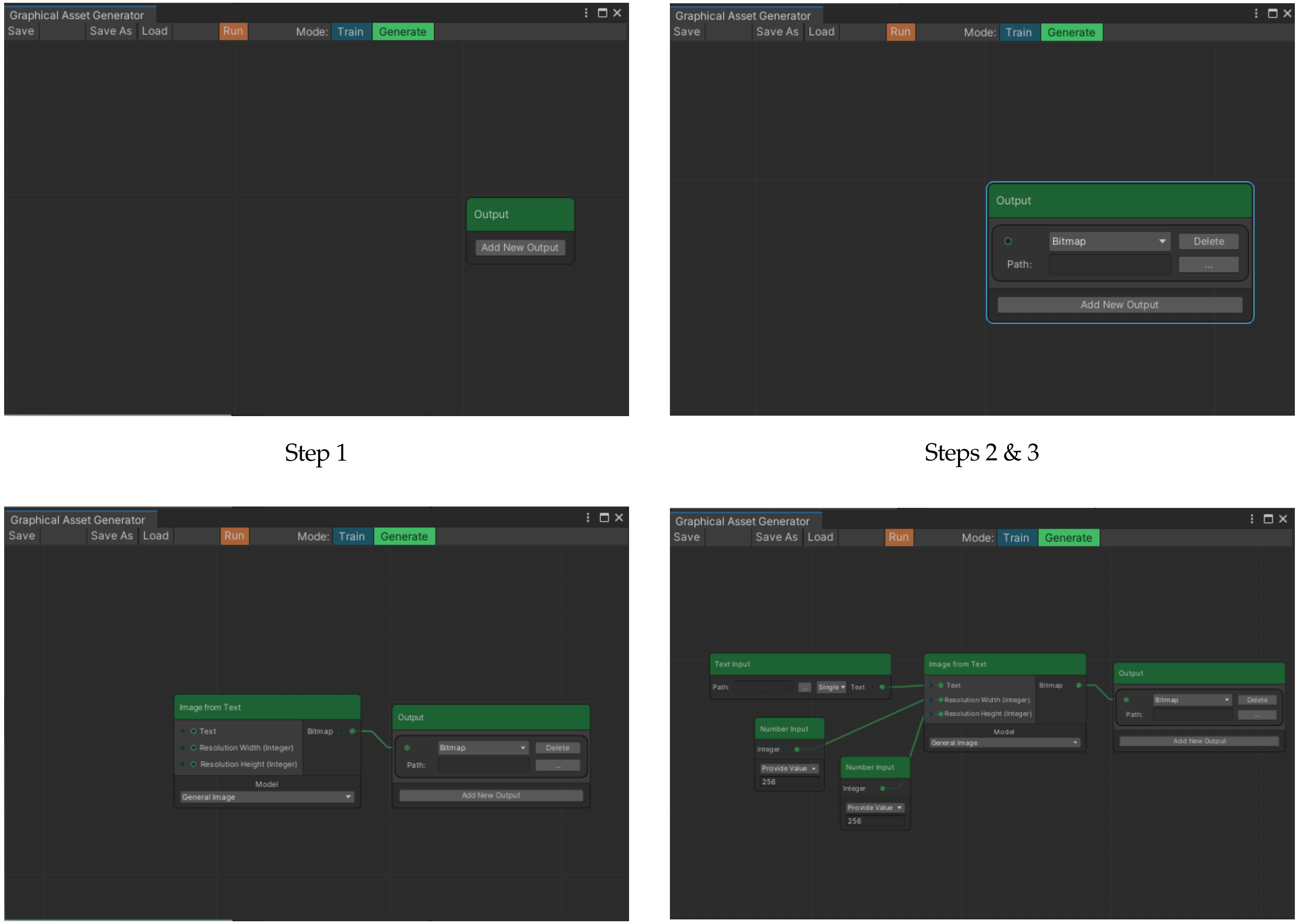
Output directory:
No path selected

Number of assets to generate: 1

Estimated time: *5m*

Step 5

FIGURE 4.2: Step-by-step process for interface M_1.



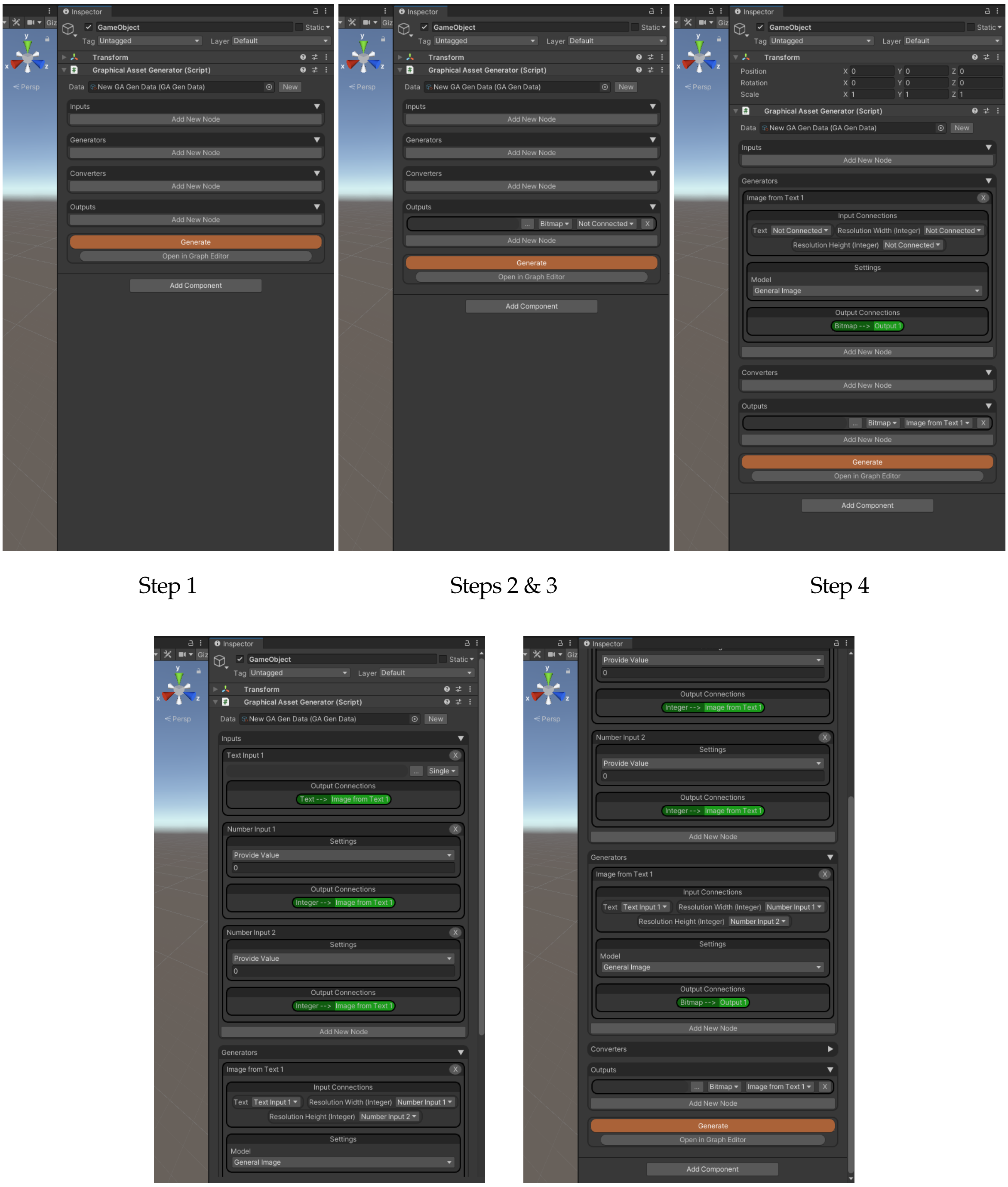
Step 1

Steps 2 & 3

Step 4

Step 5

FIGURE 4.3: Step-by-step process for interface M_2.



Step 1

Steps 2 & 3

Step 4

Step 5 - Part 1

Step 5 - Part 2

FIGURE 4.4: Step-by-step process for interface M_3.

4.4 Mock-ups

In order to obtain game designer and developer feedback, three UI mock-ups have been developed and compared. To ensure that opinions are not shaped by the type of asset, technique or approach to generation; the UI mock-ups are designed to present a fully customisable generative system, where the user decides these specifics. As such, this system is based on the multimodal technique selection presented in GaGeTx.

Each graphical asset generator can consist of any number of techniques, which are in turn comprised of an *interaction* type and a *process* type. Inputs are provided via the interaction type and outputs of the process can be fed as an input to another technique, or as a final artefact. In addition, a user can choose to convert the format of input and output data as needed.

The three mock-ups represent *stand-alone wizard*, *editor window integrated* and *editor inspector integrated* implementations of this system. Figures 4.2, 4.3 and 4.4 show an example usage process implemented in each of the three mock-ups. This process involves the configuration of a generator that takes text as input, and outputs bitmap images.

4.4.1 *M_1: Stand-alone wizard*

The first mock-up (M_1) represents a stand-alone wizard application in which the user answers a series of questions that determine the configuration of the generator. This mock-up was created using the Electron framework which packages designs built for web into an executable form. In addition, this mock-up was hosted as a web-page for users that did not have system permissions to run the application version.

A software wizard is an application designed to guide a user through a series of steps [150]. These steps can be dependent on choices made in previous steps, thus allowing for branching behaviour. This can be used to match the logical process for GaGeTx, with information from each procedure feeding into later procedures.

The goal of this UI was to directly present the decision process proposed within the GaGeTx framework. The UI presents relevant multiple-choice questions that drill-down on a configuration based on the user's needs. The options are dynamic, such that certain answers limit the choices for certain proceeding questions. The result is a save-able generator configuration that the user can edit and return to. A breadcrumb navigation system allows the user to return to earlier choices and make changes as needed.

As shown in figure 4.2, this mock-up introduces the process, asks the user what type of asset they need to generate, the format, and then the input type. The choice of input then determines the possible techniques, in this example, there is only one choice so the user does not receive a selection. This is because there is only one technique for image generation via text. The user then arrives at a screen in which they choose input and output data paths, set the number of assets to generate, and begin the generation process.

4.4.2 *M_2: Integrated in an editor window*

Mock-up M_2 represents an interface that would be integrated within existing game development software, in this case, the Unity engine. This uses Unity's experimental graph view API to render a node-based interface for designing generator configurations. This is incorporated as a separate sub-window of the Unity editor application.

As with the other mock-ups, this mock-up aims to present the GaGeTx framework decision process. For M_2 the order in which decisions can be taken is up to the user. The UI provides four categories of node: inputs, generators, convertors, and outputs. The user can chain nodes from these categories to produce their desired configuration. Each node contains its own parameters that the user may adjust. These configurations can be saved as asset files within the Unity project. This UI also includes two modes: *Generate*, and *Train*. The set of nodes within the graph view persist between both modes, but connections between them can vary between the two. In generate mode you define outputs under a single output node. You then chain together input, generator and convertor nodes with the goal of connecting the final result to an output. In train mode the flow between nodes is indicative of a training pipeline. Alternative input paths can be provided as training data, labeling nodes can be inserted for labeling input data and output paths are disabled. The goal is to present this as a tool in which you design a pipeline, then fine-tune its inner workings. As this UI is an editor window, it is suited to *offline* content generation as it does not directly connect with the active game scene. Instead, it directs outputs to user chosen folder paths from which the user can browse, refine or export results.

M_2 is shown in figure 4.3. Here, the user creates a new bitmap path in the output node, creates an "image from text" node, and then creates an input path node while defining image size parameters.

4.4.3 *M_3: Integrated in an editor inspector view*

Mock-up M_3, also built in Unity, presents the same node based system as M_2, but is instead incorporated within the editor inspector window. This means that the interface is attached to an entity within the game scene, that the user must have selected in order to edit. This has the same functionality as M_2, however it is presented in a vertical, linear format.

The same categories available in M_2 are shown as collapsible sections in M_3. Nodes are added under these categories using the respective "Add New Node" buttons which produce a relative list of nodes to choose from. To connect these nodes the user must select connections at the top of each node UI. An indicator at the bottom of each node will turn green if the node has been connected to, and which node this is. In the Unity Editor, inspector UI applies to a specific entity within a game scene. In a full implementation, this would give this version of the tool the ability to instantly load generated content within the game scene at a position of the user's choosing, facilitating rapid prototyping, variation, or runtime generation.

As presented in figure 4.4, M_3 presents a vertical list of categories which the user can add to. The user adds an output node with the format of "bitmap", adds an "image from text" generator node, and creates the necessary input nodes. These nodes are then bound to each other using drop-downs in the UI.

4.5 Procedure

As stated in the previous section, three mock-ups were developed, each implementing the same underlying system with different forms of UI interaction. Each participant is asked to complete a form consisting of demographic questions, followed by a section for each mock-up. In each section the mock-up is introduced and instructions for installing and using the mock-up are provided. The participant is asked to try the mock-up for as long as they like, then complete a set of preference questions. After all three sections are completed, the participant is provided with an opportunity to book a semi-structured interview to provide qualitative feedback. In analysis, the questionnaire responses are compared between the three mock-ups, and overall. The following subsections will provide more detail on the participants, mock-ups, questionnaires and interviews.

4.5.1 Participants

Participants were recruited via a convenience sampling approach, in which members of the game development communities: LUUG and BCS Animation and Games specialist group, were contacted via group emails through the respective community administrators. Participants were required to be over the age of 18, with professional experience in game design or development. Participants were also required to have Unity engine installed on their personal machines, and experience with the software was preferred. Of the group email respondents, participants were recruited based on the aforementioned criteria. Participation was not incentivised on the basis of financial gain, and was thus voluntary.

4.5.2 Interface prototype mock-ups

In order to explore the preferences of participants, three prototype mock-ups have been devised, as presented in section 4.4. These mock-ups M_1, M_2 and M_3 represent three forms of tool implementation; stand-alone, integrated in an editor inspector and integrated in an editor window respectively. These mock-ups were designed as representations of the flow of interaction and were therefore not functional under the hood. M_2 and M_3 have been implemented in the Unity engine. Figures 4.2, 4.3 and 4.4 show a step-by-step process within each of the three mock-ups, achieving an equivalent result. As the figure shows, M_1 takes a direct approach by asking the user a series of questions that drill down on a configuration based on the user's need. M_2 and M_3 take a more free-form approach, giving the user a palette of options from which to build their generator. Users were provided walk-through videos to watch before using each interface, demonstrating the functionality at each step.

4.5.3 Questionnaires

Participants were given the opportunity to explore each mock-up at their own discretion and complete a questionnaire for each one, as well as a fourth questionnaire for collecting data independent of the mock-ups, such as demographic data. These questionnaires consist of 5-point Likert scales, multiple-choice tick boxes and dichotomous questions. Demographic data, such as age, years of experience and team size, were recorded by range. These surveys were hosted on Microsoft Forms. The research questions and the corresponding survey questions are presented in Table 4.1.

4.5.4 Semi-structured interviews

Participants were scheduled 10-to-15-minute interviews following their completion of the previous step, this was on an opt-in basis. In these interviews, participants were first asked questions to confirm their answers within the questionnaires to ensure validity, then asked to describe any concerns or perceived benefits of such a tool, what asset types they desire to generate, and what additional features they desire. Participants were also given the opportunity to voice any opinions that had not been addressed at any other stage in the study. Notes were taken during these semi-structured interviews for later analysis.

4.6 Results

All quantitative data from the questionnaires was analysed univariately, extracting average scores from Likert scale and single-choice questions, and frequency distributions for multiple choice questions. For this, one-sample Kolmogorov-Smirnov (K-S) [73] tests were used to determine whether results were normally distributed. Wilcoxon signed-rank tests [513], and one-way-ANOVA [392] were conducted for each question, comparing the results from the three questionnaires. These results answer hypotheses H1 to H7, with respect to variance between mock-ups. IH1 to IH3 are assessed via linear regression and independent samples T-Tests [72], observing the relationship between demographic and preference data, and an inductive thematic analysis was conducted on the notes from the semi-structured interview, answering RQ8 and RQ9. During the data collection period, a total of 16 participants tested the mock-ups and completed the questionnaire, 4 of which also completed informal follow-up interviews. The initial mock-up testing phase took participants an average of 63 minutes to complete.

K-S tests were performed for each question to determine if the distributions are normal. RQ7 (the SUS results) and RQ2 frequency of options picked (Table 4.3), are found to have normal distributions with statistical certainty ($P > 0.05$). All other questions meet the null hypothesis ($P < 0.05$).

Descriptive statistics

Overall, responses to RQ1 (usefulness) were moderately positive ($\mu = 2.72, \sigma = .669$). Of the options in RQ2, the most popular choices ($\mu > .5$) were RQ2.1 (generating inspiration), RQ2.2 (exploring ideas) and RQ2.3 (creating placeholder assets). The number of options picked by participants was between 2 and 3 ($\mu = 2.78, \sigma = 1.629$). For RQ3, participants show a clear preference for a high volume of lower quality assets over a lower volume of high quality assets ($\mu = .17, \sigma = .383$). RQ4 responses show an acceptable generation time for a single asset to be between 1 and 10 minutes ($\mu = 1.67, \sigma = .686$). For comparison, existing tools such as Didimo [188] can take 5-10 minutes to produce a result, so this may be an expectation based on what is already available. RQ5 responses point toward preference for a tool integrated within a game-engine

TABLE 4.3: One-sample Kolmogorov-Smirnov test results for RQ1 - RQ7, presenting the mean, standard deviation (SD), and significance scores.

Research Question		Mean	SD	Sig.
RQ1: (0-4)		2.72	.669	.001
	RQ2.1 (0-1)	.67	.485	.000
	RQ2.2 (0-1)	.72	.461	.000
	RQ2.3 (0-1)	.61	.502	.000
RQ2:	RQ2.4 (0-1)	.17	.383	.000
	RQ2.5 (0-1)	.06	.236	.000
	RQ2.6 (0-1)	.17	.383	.000
	RQ2.7 (0-1)	.39	.502	.000
	RQ2 Frequency of options picked.	2.78	1.629	.194
RQ3: (0-1)		.17	.383	.000
RQ4: (0-6)		1.67	.686	.001
RQ5: (0-1)		.78	.428	.000
RQ6: (0-4)		3.50	.985	.004
RQ7: (0-100)		78.75	13.4287	.131

or editor ($\mu = .78, \sigma = .428$). For RQ6, results show that participants considered the ability to configure or modify the tools important ($\mu = 3.50, \sigma = .985$). The overall usability (RQ7) across all mock-ups was considered good ($\mu = 78.75, \sigma = 13.4287$).

Tool Preference

In order to determine the statistical significance in the difference in preference between the three mock-ups, ANOVA and Wilcoxon tests were conducted for the research questions RQ1 to RQ7. As seen in Table 4.4, none of the questions presented statistically significant difference in preference between the mock-ups when tested using one-way ANOVA. This is confirmed in Table 4.5, where the Wilcoxon tests similarly report no statistically significant difference. Therefore, there is no significant difference in preference between the three mock-ups.

Beyond this, there is a slight leaning in the selection of RQ2.7 (user made content) for M2 and M3 over M1 ($Z = -1.414$). RQ2.6 (assets from scratch) is chosen more for M1 than M3 ($Z = -1.414$), and slightly more than M2 ($Z = -1.000$). RQ2.4 (variations of complete assets) is chosen more for M2 than M3 ($Z = 1.1414$) and slightly more than M1 ($Z = -1.000$). Overall these ANOVA and Wilcoxon Test results suggest a slight separation between stand-alone (M1) and integrated (M2 and M3) mock-ups.

Further ANOVA and Wilcoxon tests were conducted between the integrated and stand-alone solutions for RQ1 (usefulness) and RQ5 (preferred as stand-alone or integrated), shown in table 4.6. These also present non significant differences, though ther is a slightly higher mean RQ1 score for integrated mock-ups, and a leaning towards preference for integrated solutions overall (RQ5 $\mu = .78$).

TABLE 4.4: One-way ANOVA significance scores for RQ1 to RQ7, alongside means and standard deviations per mock-up.

Research Question		M1		M2		M3		Overall		
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Sig.
RQ1: (0-4)		2.50	.548	2.83	.753	2.83	.753	2.72	.669	.637
RQ2:	RQ2.1 (0-1)	.50	.548	.83	.408	.67	.516	.67	.485	.521
	RQ2.2 (0-1)	.67	.516	.83	.408	.67	.516	.72	.461	.791
	RQ2.3 (0-1)	.50	.548	.67	.516	.67	.516	.61	.502	.821
	RQ2.4 (0-1)	.17	.408	.33	.516	.00	.000	.17	.385	.342
	RQ2.5 (0-1)	.00	.000	.17	.408	.00	.000	.06	.236	.391
	RQ2.6 (0-1)	.33	.516	.17	.408	.00	.000	.17	.383	.342
	RQ2.7 (0-1)	.17	.408	.50	.548	.50	.548	.39	.502	.439
RQ3: (0-1)		.17	.408	.17	.408	.17	.408	.17	.383	1.000
RQ4: (0-6)		1.67	.816	1.67	.816	1.67	.516	1.67	.686	1.000
RQ5: (0-1)		.67	.516	.83	.408	.83	.408	.78	.428	.761
RQ6: (0-4)		3.50	1.378	3.50	.548	3.50	1.049	3.50	.985	1.000
RQ7: (0-100)		80.00	12.649	78.75	10.694	77.50	18.303	78.75	13.429	.995

TABLE 4.5: Z-values and significance (P-values) for Wilcoxon Signed Ranks Test between each mock-up, for each question.

		M2 ->M1		M3 ->M1		M3 ->M2	
		Z	Sig.	Z	Sig.	Z	Sig.
RQ1		-1.414	.157	-1.414	.157	.000	1.00
RQ2	RQ2.1	-1.000	.317	.000	1.00	-1.00	.317
	RQ2.2	-1.000	.317	.000	1.00	-1.000	.317
	RQ2.3	-1.000	.317	.000	-1.00	.317	1.00
	RQ2.4	-1.000	.317	-1.00	.317	1.1414	.157
	RQ2.5	-1.000	.317	.000	1.00	-1.000	.317
	RQ2.6	-1.000	.317	-1.414	.157	-1.00	.317
	RQ2.7	-1.414	.157	-1.414	.157	.000	1.00
	Total	-1.342	.180	-1.00	.317	-.816	.414
RQ3		.000	1.000	.000	1.00	.000	1.00
RQ4		.000	1.000	.000	1.00	.000	1.00
RQ5		-1.000	.317	-1.000	.317	.000	1.00
RQ6		.000	1.000	.000	1.00	.000	1.00
RQ7		-.406	.684	-.184	.854	.713	.713

Demographic impact

14 of the 16 participants were male, and 13 participants were under the age of 25. Additionally, 12 participants had under 1 year of experience in professional game development, while 2 had between 1 and 4 years, and 2 had between 5 and 9 years. 10 participants typically work solo, and 6 participants typically work in a team size of 11-25. The most common typical roles of

TABLE 4.6: One-way ANOVA and Wilcoxon Signed Ranks tests comparing stand-alone mock-up results (M1) to integrated mock-up results (M2 & M3), for relevant research questions.

	One-way ANOVA							Wilcoxon Signed Ranks	
Research Question	M1		M2 & M3		Overall			M2 & M3 ->M1	
	Mean	SD	Mean	SD	Mean	SD	Sig.	Z	Sig.
RQ1	2.50	.548	2.83	.718	2.72	.669	.334	-1.414	.157
RQ5	.67	.516	.83	.389	.78	.428	.453	-1.000	.317

participants were artist and designer (50%), followed by technical artist and programmer (33%). To examine impact, multiple regression with a confidence interval of 95.0% was conducted for each single choice scale question (RQ1, RQ4, RQ6 and RQ7), against participant demographic variables. Table 4.7 presents the results of the multiple regression for single-choice scale questions, showing their correlation with the four single-choice demographic questions. As RQ3 and RQ5 are binary questions, Independent-Samples T-tests were instead conducted, comparing demographic answers between two groups, where the the groups are defined by the binary answer to the option (chosen or not chosen). This approach was also taken with RQ2, where multiple choice answers were binary, due to being tick-boxes. These results are presented in table 4.8.

RQ2.5 "Creating assets from existing (complete) designs" (shown in grey table 4.8), was only selected once out of all samples. As a result there was not enough variance to report a T-test result. The RQ3 gender T-test could not be completed due to standard deviations of both groups being zero, this is also shown in grey.

The ANOVA on the linear regressions of RQ1 (usefulness), RQ6 (importance of configurability) and RQ7 (SUS results) in table 4.7, are statistically significant (P < 0.05). Of these, RQ7 has a strong correlation with the four tested demographic questions ($AdjustedRSquare = .671$), while RQ1 ($AdjustedRSquare = .305$) and RQ6 ($AdjustedRSquare = .428$) show low correlation. For RQ7 age, years of professional experience and size of team have high negative correlations. That is, the older and more experienced participants, and those that work in larger teams, gave lower SUS ratings. A moderate negative correlation for age, years of experience and size of team is also observed in results for RQ6 and RQ1. Additionally there is an insignificant and low degree of correlation between gender and results of RQ6 and RQ1.

T-test results in table 4.8 show similar levels of impact for the four demographic categories. Gender's impact is insignificant in all cases, while years of professional experience and size of team are significant for RQ2.1, RQ2.2 and RQ3. Positive T values for RQ2.1 and RQ2.2 show that those with more professional experience and larger team sizes choose these options (generating inspiration, and exploring ideas) more frequently. More years of professional experience ($P = .014, T = -2.806$) and a larger size of team ($P = .000, T = -7.483$) both point toward the choice of volume over quality, though a near unanimous selection of volume over quality was made by participants.

While years of professional experience and age are intrinsically linked, they are not found to have equivalent impact in most cases. This suggests that the two are decoupled. Age does not have a significant impact in any of the T-test results, while years of professional experience does. The

TABLE 4.7: Linear regression results for RQ1, RQ4, RQ6 and RQ7 (all single-choice questions scale), reporting Pearson correlation [71], R Square [74], adjusted R Square and ANOVA p-value for each single-choice demographic question. (*) *Experience* stands for *Years of professional experience*

Research Question		Correlation		Model Summary		ANOVA
		Correlation	Sig.	R Square	Adjusted R Square	Sig.
RQ1	Age	-.497	.018	.387	.305	.026
	Gender	-.267	.142			
	Experience*	-.615	.003			
	Size of team	-.604	.004			
RQ4	Age	-.447	.031	.302	.209	.067
	Gender	-.224	.186			
	Experience*	-.546	.010			
	Size of team	-.530	.012			
RQ6	Age	-.701	.001	.495	.428	.006
	Gender	.078	.379			
	Experience*	-.646	.002			
	Size of team	-.492	.019			
RQ7	Age	-.614	.003	.710	.671	.000
	Gender	-.443	.033			
	Experience*	-.815	.000			
	Size of team	-.835	.000			

level of experience and size of team both correlate with lower ratings (RQ1/RQ7) and interest in using the tools for generating inspiration and exploring ideas (RQ2.1/RQ2.2). This is corroborated by the near unanimous selection of volume over quality. It can be concluded that experienced users prefer graphical asset generation in early ideation/inspiration stages of creation. This is also reflected in the choices overall for RQ2.1 ($\mu = .67$), and RQ2.2 ($\mu = .72$).

Interview Results

4 out of the 16 participants agreed to complete a follow-up interview. During these interviews, the participants generally confirmed the results of the questionnaires. There is no clear preference between the three mock-ups but they emphasised compatibility with their existing tools. One participant mentioned that each mock-up was fine, as long as they didn't need a converter or third tool to get the assets into their game. Most participants confirmed that they would prefer to use the tool for ideation and therefore prefer speed and volume over quality.

Overall, the participants were very supportive and excited about content generation tools and use of AI in their pipelines. The main concern they had was regarding the ease of use and prior knowledge of AI or PCG needed to be able to use the tool effectively. for example a participant pointed that whilst they would not necessarily expect tools such as DALL-E and Mid-Journey to be integral part of game design and development but did like the idea of a middle ground so that they can have control but also use the tool relatively easily. There was not a clear preference towards the type of assets they would prefer to generate with such a tool. Most participants mentioned a wide variety of asset types, based on their current project, but also highlighted that any one type would also be useful. Furthermore, with regard to the format used to store these assets, participants expressed that as long as the assets were in a standard, readable format for their game engine of choice, there were no concerns.

With regard to usability, one participant noted that they found it frustrating that they could not undo and redo their changes within the Unity integrated tools. They stated that this was a feature that they expected, considering that the tool appeared to be part of the engine interface that they were familiar with.

Findings

For H1, given answers to RQ5, users prefer an integrated solution over a stand-alone implementation, with a mean value of .78 (σ = .428), where 1 represents an 'integrated solution'. This is confirmed by RQ1, in which the integrated solutions scored higher. With regards to H2, results of RQ1 and RQ6.1 suggest no significant preference between window and inspector integrated interfaces. H4 is confirmed with a preference for volume over quality across the board, with a mean value of .17 (σ = .383), where 0 represents 'volume over quality'. Regarding H5 (RQ4), a mean selection of 1.67 (σ = .686) shows a preference for a maximum generation time between 1 and 10 minutes for a single asset, rejecting this hypothesis. For H6, a mean score of 3.5 (σ = .985) on a 5-point Likert scale, RQ6, suggests a moderate preference for the ability to configure or modify the tool. Answers to RQ7 present an overall mean SUS score of 78.75 (σ = 13.429), therefore confirming H7. The individual mean SUS scores for M1, M2 and M3 were 80.00 (σ = 12.649), 78.75 (σ = 10.694), and 77.50 (σ = 18.303) respectively. Following the adjective rating system provided by Bangor et al. [21], the mean overall score can be classed as "Good" (78.75). Section 4.7 will discuss these findings and provide recommendations, as well as suggestions for future research.

4.7 Concluding discussion

This study has found that there is a preference for all UIs to be integrated into game-engines/editors, and for the system to generate larger amounts of lower quality assets as opposed to smaller amounts of higher quality assets. This is also corroborated by a preference for using such a system in early stages of development to generate inspiration, explore ideas and create placeholders. The ability to configure or modify such a tool is also considered important. The maximum acceptable time scale for generating a single asset is between 1 and 10 minutes. This is not surprising, as existing tools such as Didimo [188] can take 5-10 minutes to generate assets, especially at higher qualities. The other findings suggest that a much faster generation speed would be ideal, in order to provide larger volumes of assets for inspiration and ideation. There is however no statistically significant difference in preference between a stand-alone multi-choice based UI, integrated graph-view UI and integrated inspector UI for generating graphical assets.

Overall, results are in alignment with the findings of Walton et al. [487] and Kasurinen et al. [215]. Users were interested in using the tool for inspiration and prototyping, and favored integration, and thus compatibility with their chosen game engine. Additionally, when observing

TABLE 4.8: Independent-Samples T Test results for RQ2, RQ3 and RQ5 (all binary-choice questions), reporting t-statistic (t) and p-value (sig.). (*) *Experience* stands for *Years of professional experience*

Research Question		T	Sig.
RQ2.1	Age	2.236	.076
	Gender	1.103	.309
	Experience*	3.638	.012
	Size of team	4.536	.000
RQ2.2	Age	2.449	.070
	Gender	1.258	.266
	Experience*	5.932	.002
	Size of team	7.303	.000
RQ2.3	Age	-1.936	.082
	Gender	.947	.368
	Experience*	-1.049	.310
	Size of team	-.323	.751
RQ2.4	Age	1.871	.082
	Gender	-.816	.426
	Experience*	.392	.700
	Size of team	.000	1.000
RQ2.5			
RQ2.6	Age	1.871	.082
	Gender	-1.765	.210
	Experience*	-.392	.700
	Size of team	-1.333	.201
RQ2.7	Age	1.936	.082
	Gender	-.947	.368
	Experience*	1.049	.310
	Size of team	.323	.751
RQ3	Age	1.871	.082
	Gender		
	Experience*	-2.806	.014
	Size of team	-7.483	.000
RQ5	Age	1.441	.235
	Gender	-1.883	.082
	Experience*	1.068	.353
	Size of team	.770	.453

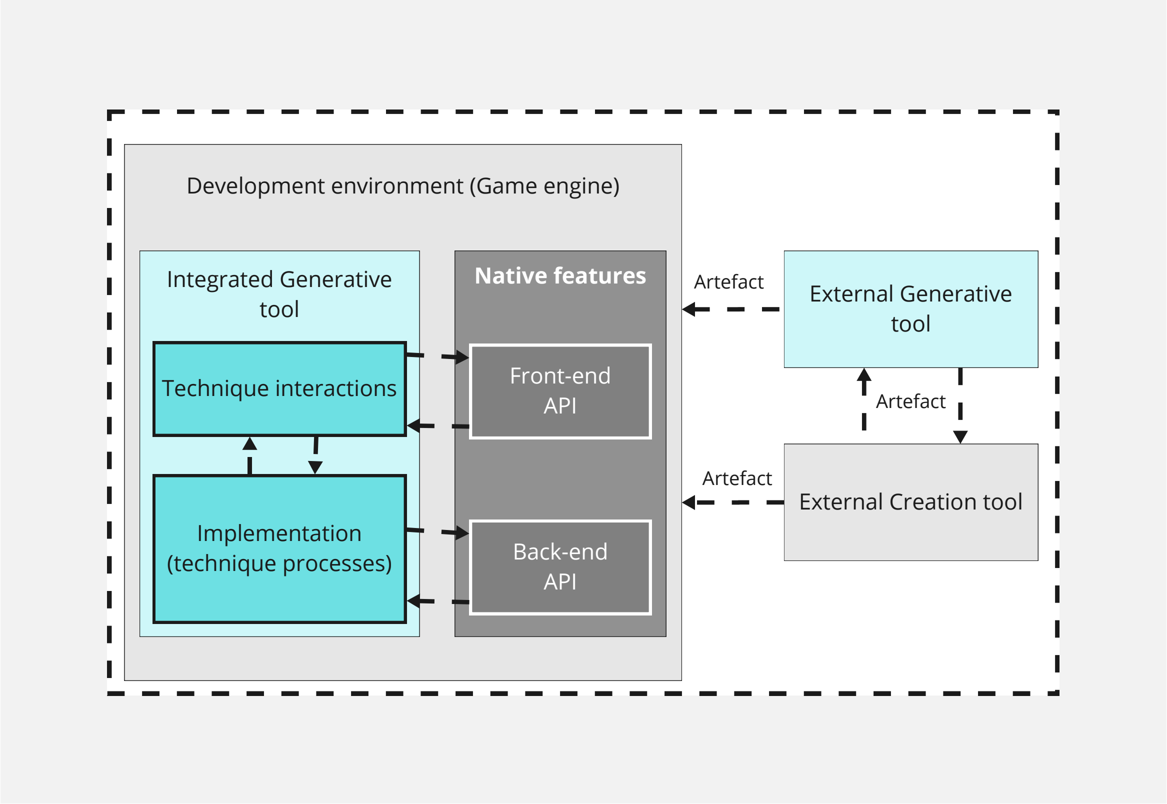


FIGURE 4.5: Graphical asset creation within the game production framework.

the characteristics of the three preferred usages according to the characteristics shown in figure 4.1, in general, users prefer generators for work in an *offline* context, with an expectation for lower quality and to augment, rather than replace an existing process in their pipeline.

Figure 4.5 presents the flow of graphical asset data in both stand-alone and integrated scenarios. As shown, creative tools can either be integrated within the main development environment (game engine) or implemented externally as stand-alone programs. Users interact with these tools via user-interfaces, and in the case of integrated tools, the implementation itself can rely on the native features of the environment through application programming interfaces (API). Stand-alone external generative tools, as with other creation tools such as Photoshop [7], Blender [32] and Visual Studio [322] instead rely on outputting artefacts in standardised formats that most game engines support, such as FBX, OBJ, PNG and JPEG. Integrated tools may implement GAG technique *interactions* and *processes* using game engine UI APIs and native engine features via back end APIs respectively.

Figure 4.6 presents guidelines resulting from the findings of this user study, spanning both the user intent as well as the implementation and integration of graphical asset generators in design/development pipelines. Table 4.9 presents the mapping of these guidelines to the research questions.

The first consideration is user intent with regard to the usage application, what technique interaction and process is used and what the desired outcome is. When considering user intent, based on the findings, there are three main considerations:

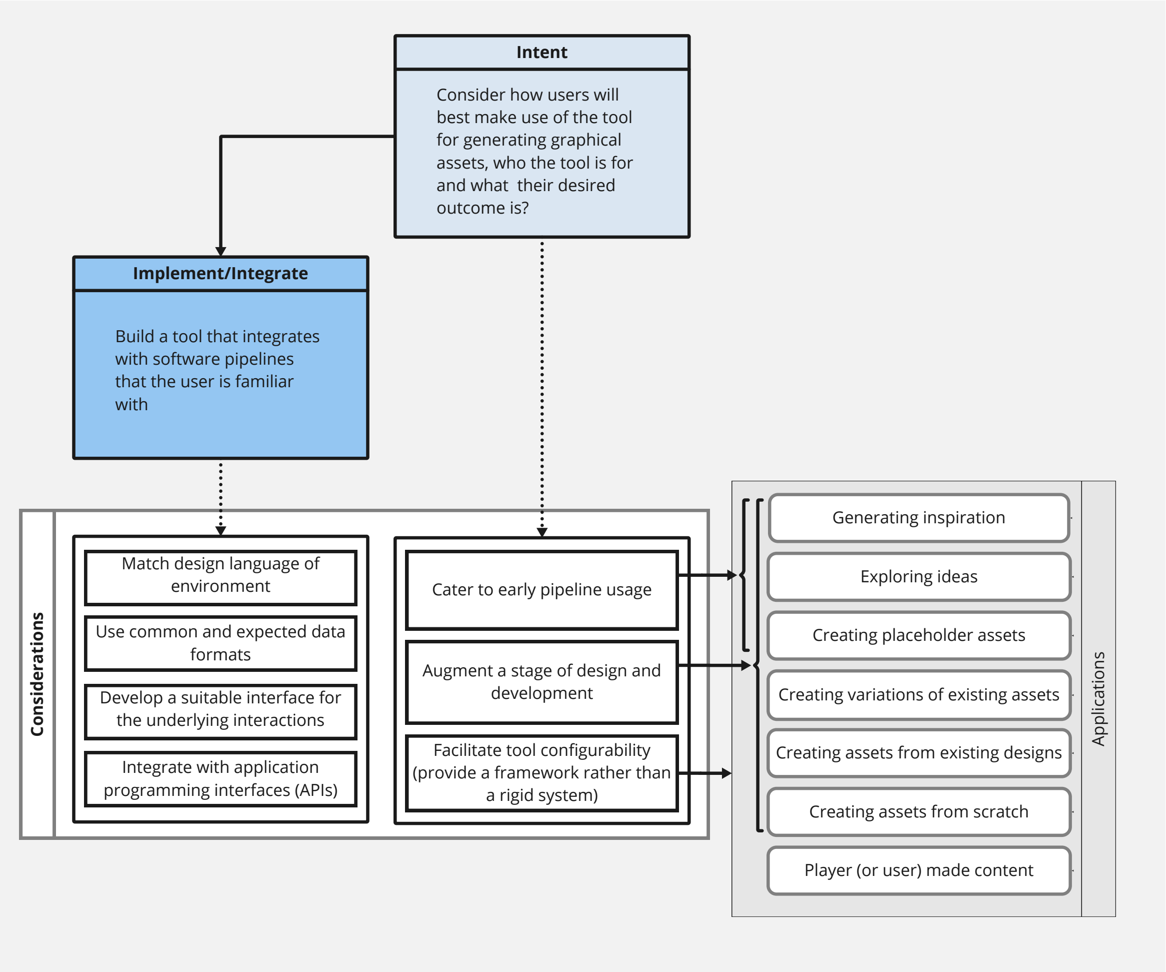

Intent
Consider how users will best make use of the tool for generating graphical assets, who the tool is for and what their desired outcome is?
Implement/Integrate
Build a tool that integrates with software pipelines that the user is familiar with
Considerations
Match design language of environment
Use common and expected data formats
Develop a suitable interface for the underlying interactions
Integrate with application programming interfaces (APIs)
Cater to early pipeline usage
Augment a stage of design and development
Facilitate tool configurability (provide a framework rather than a rigid system)
Generating inspiration
Exploring ideas
Creating placeholder assets
Creating variations of existing assets
Creating assets from existing designs
Creating assets from scratch
Player (or user) made content
Applications

FIGURE 4.6: Guidelines resulting from research findings.

- *Cater to early pipeline usage.* Users favour early pipeline generation for purposes of generating inspiration, exploring ideas, and creating placeholders, as presented in figure 4.6. Furthermore, the system should aim for variety and volume of outputs rather than quality. This is because users prefer not to use the results in a finished product directly, but would rather use them as ideas to build on and refine.

- *Augment a stage of design and development.* The preferred use cases are all supportive of a human-driven design and development process, and do not directly replace the final creation of assets. Rather, they give designers and developers ideas to expand on, break creative block or help to streamline more mundane or inconsequential tasks. This entails the majority of applications, excluding player made content as shown in figure 4.6.

- *Facilitate tool configurability.* Game design and development is a creative endeavor and thus, new, unique forms of asset are part and parcel. A graphical asset generator that has a pre-defined style or rigid way of working becomes a tool with limited applicability. Configurability should be considered, regardless of the chosen application.

Once intent is established, and the GAG technique and approach is determined, the tool must be implemented and integrated within the game design and development pipeline. Here there are four main considerations:

- *Match the design language of the environment.* The less the user has to learn on top of what they are accustomed to within their pipeline, the easier it is to incorporate the tool. If the interface of the tool provides functionality that other native features of the environment have, such as the ability to undo and redo changes, frustration or confusion can be avoided. This can be achieved through appropriate use of front-end UI APIs within the host software, as illustrated in figure 4.5.
- *Use common and expected data formats.* To smoothly integrate the tool, whether it is integrated or stand-alone, the output artefacts must be in a format that the development environment can utilise. Typically these are ubiquitous data formats, such as OBJ or common proprietary formats that are supported due to popularity, such as Autodesk FBX. This is particularly important in the transfer of artefacts from and between external generative tools, creation tools and the users development environment of choice, as shown in figure 4.5.
- *Develop a suitable interface for the underlying interactions.* Design an interface to best interact with the features of the tool, assuming that it provides good usability, there is no preference between step by step options, graph view windows and inspector editors.
- *Integrate with application programming interfaces (APIs).* The tool should be integrated as a part of an existing popular engine or editor, or at the very least should have full compatibility with said application. Technique interactions and processes must interact with each other while appropriately utilising the host game engine's front-end and back-end APIs, as shown in figure 4.5.

TABLE 4.9: Mapping of guidelines to research questions.

Recommendations	Research Questions
Facilitate tool configurability	RQ6
Cater to early pipeline usage	RQ2, RQ3, RQ4
Augment a stage of design and development	RQ2
Match the design language of the environment	RQ8
Use common and expected data formats	RQ8, RQ9
Develop a suitable interface for the underlying interactions	RQ1, RQ7
Integrate with application programming interfaces (APIs)	RQ5, RQ8

4.8 Chapter summary

In this chapter a user study was conducted to examine the user preferences and needs regarding graphical asset generation tools in game production. Three mock up UI designs were developed representing a stand-alone tool, integrated window tool and integrated inspector tool. These mock up UIs were presented to game designers and developers in a mixed methods experiment in which a questionnaire was used to collect quantitative data with repeated measures, and semi-structured interviews were conducted to obtain nuance through qualitative analysis. Resulting from these findings, a set of guidelines for graphical asset generation tools were formulated.

Chapter 5: Swordgen-
A proof-of-concept protototype

In the previous chapter, an initial user study was conducted in order to obtain insights into the preferences of game designers and developers with regard to graphical asset generators as tools. For this purpose, three non-functional mock-ups were implemented and tested. It was found that in general, participants preferred to have an integrated solution, and to use such tools at early stages in their pipeline, such as creating placeholders and generating inspiration. Participants also expressed a preference for volume over quality, in line with the latter point, and compatibility with their existing tools, which is consistent with the preference for integration. These observations are useful, however, without testing a functional system, many nuances and details regarding the moment to moment use and satisfaction with such software cannot be gauged. In this chapter, a proof-of-concept prototype tool was developed, incorporating four different generative techniques.

5.1 Introduction

Leading on from the findings in chapter 4, it is clear that the lack in functionality was a limiting factor on the perception of mock-ups. To add to these findings and obtain a deeper understanding of the potential usage of graphical asset generators, user preference must be observed regarding functional implementations. In this chapter a proof-of-concept prototype has been developed with the goal of representing a range of generative techniques. This prototype was tested with game designers and developers to examine technique preference and observe the relationship between usage and technique choice. Results of this testing confirm a connection between the user pipeline choice and preferred technique, as well as an affinity between technique and assessment criteria.

5.2 Proof-of-concept prototype

The proof-of-concept generator was designed to cover representative techniques from all four *idea fidelity* groupings under a single implementation. This has been achieved using a Variational Autoencoder (VAE) architecture, combined with a procedural shape system. As a result, the final system allows the user to guide and iterate designs via procedural modelling parameters, reconstruct designs based on silhouette images, interpolate between two images, and generate random designs, all via a shared UI.

In the development of this prototype, two key elements had to be considered. The first being the generative method itself, informed by the literature and framework in chapter 2, and the second being the UI and integration, informed by the guidelines in chapter 4. The UI and integration needed to support the features of the generative method, ensuring it can take inputs and provide outputs seamlessly. Additionally, the generative method needed to offer the necessary access while being capable of running in the background alongside all the features of the host game engine. Rapid action development (RAD) is a methodology that emphasizes quick development and iteration of prototypes, allowing for continuous refinement [27, 9]. As such, a RAD approach was applied, to ensure that both elements could be iterated in support of each other.

To achieve this, a common game related 3D asset type was selected as a target for the generative system to output. Here, 3D swords were chosen due to their ubiquity in many genres of video game, e.g. role-playing, fighting and strategy in historical and fantasy settings. In addition, the main characteristics of a sword's form can be easily represented in two dimensions, making them a good candidate for silhouette representations and single view reconstruction. This would allow for more efficient training and a more compact dataset in terms of file size. Requiring a single view per sample would also make obtaining training data easier, as any standalone image of a sword would be compatible.

As such, the VAE is trained on a dataset of 13,728 front profile silhouette images of swords. This dataset was formed of 513 images sourced from the internet, obtained from the Unity Asset store, or generated via StableDiffusion, then expanded upon by variation. The training process made use of differentiable rendering to allow the model to backpropagate from the final mesh result and thus learn to output the correct shape parameters given the input image. The VAE was implemented in PyTorch. Initially, the Unity MLAGENTS LLAPI was to be used as an interface between the training code in python, and the Unity implementation. The result however was very slow and lacked the possibility of differentiation. To achieve efficient training, a one-to-one implementation of the procedural modelling algorithm was instead created in python. As not all parameters of the procedural modelling system were used, and a predetermined number of nodes were used (12), it was possible to create a version of the algorithm that would support GPU processing.

While this system is trained on sword data, the same framework may be applicable to other forms of graphical asset, and potentially trained on general prop shapes, due to the flexibility of the procedural modelling algorithm. Though the effectiveness would be dependent on degree to which the shape can be represented from a single-view perspective.

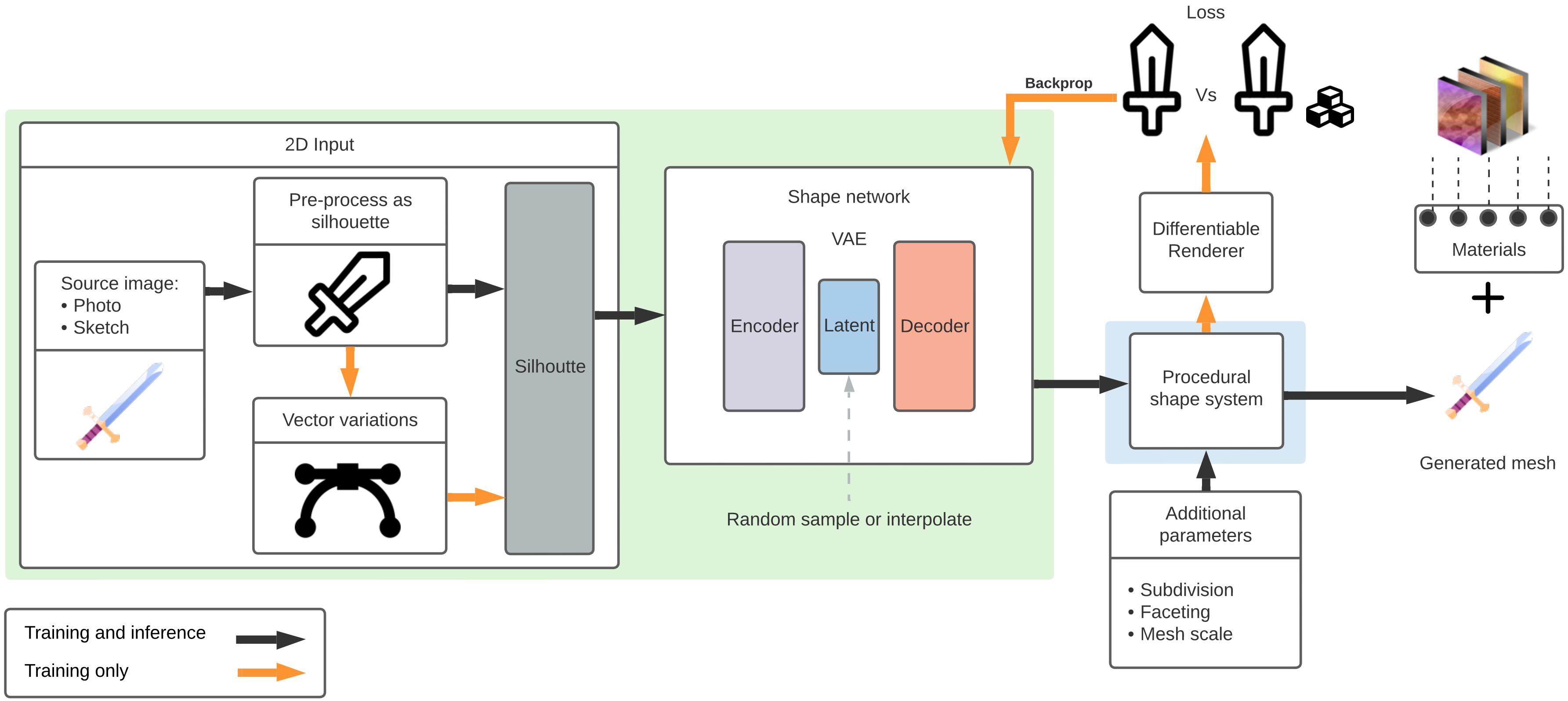


FIGURE 5.1: Overview of the GAG process. Orange lines indicate paths only used in training.

5.2.1 Procedural Shape System

The procedural shape system was designed as a method for making generic sword shapes out of simple constituent parts. In this system, each design consists of individual mesh nodes that are chained together to form a complete mesh. These nodes each have their own parameters that define their form. There are two types of node, segments, and branching segments. Segments can be connected to a maximum of two other nodes, whereas branching segments can have a maximum of 6 connections.

Segments

In 3D mesh modelling, consistent topology is important for producing clean looking shapes. This is reliant on having uniform, even length edge loops across the model. An edge loop is a sequence of vertices, connected by edges, with the last vertex connecting to the first vertex. Edge loops define the contour of a shape, allowing for consistent deformation and subdivision [253].

As presented in figure 5.2, segments are made up of edge loops that run from the bottom of the shape to the top. Each loop has the same number of vertices, equal to the n-th value in the sequence (4, 8, 16, 32, 64, 128,...) where $n = subdivisions\ (2^i)_{i=n+2}^{\infty}$. As each segment can have varying parameters, the end loops are not likely to line up with those of other segments. Therefore, when segments are joined together, the last loop from the first segment is duplicated and connected to the new segment.

Tapering is achieved by linear interpolation between the center loop and top outer loop (forward taper), or center loop and bottom outer loop (backward taper). In which, each loop's relative position is converted to a t-value, and this t-value is used to produce a scale amount that is then applied to the loop.

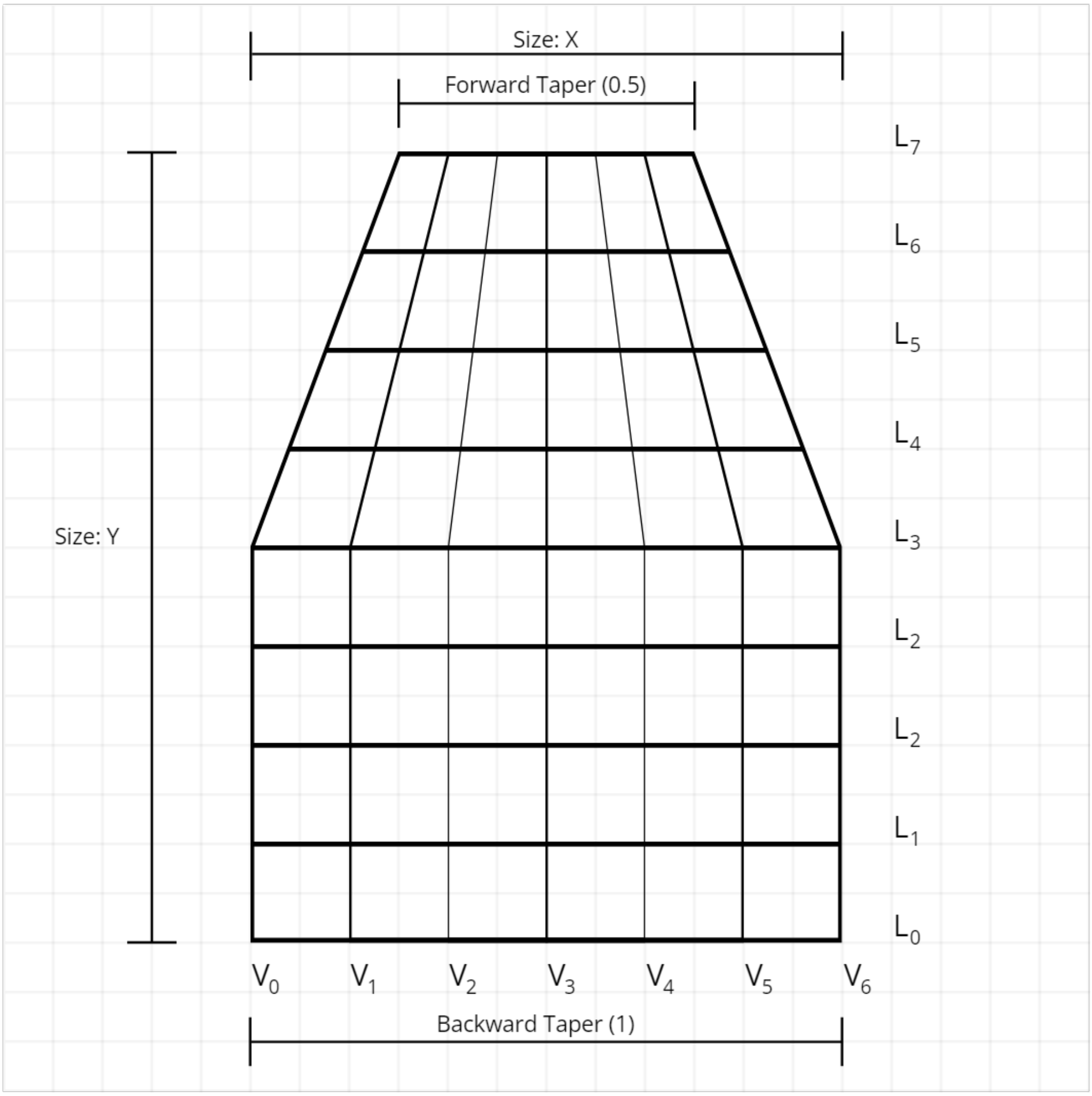


FIGURE 5.2: Representation of segment geometry parameters along the XY plane. Loops $L_0 \ldots L_7$ are each formed of vertices $V_0 \ldots V_6$ as well as an equal number of vertices for three other faces.

Rounding is applied by linear interpolation of the distance (v_d) between the center of the loop (c) and the vertex (v), and the circumradius (d), along the direction, di, where $\hat{di} = (\vec{v} \hat{-} \vec{c})$. This is shown in figure 5.3.

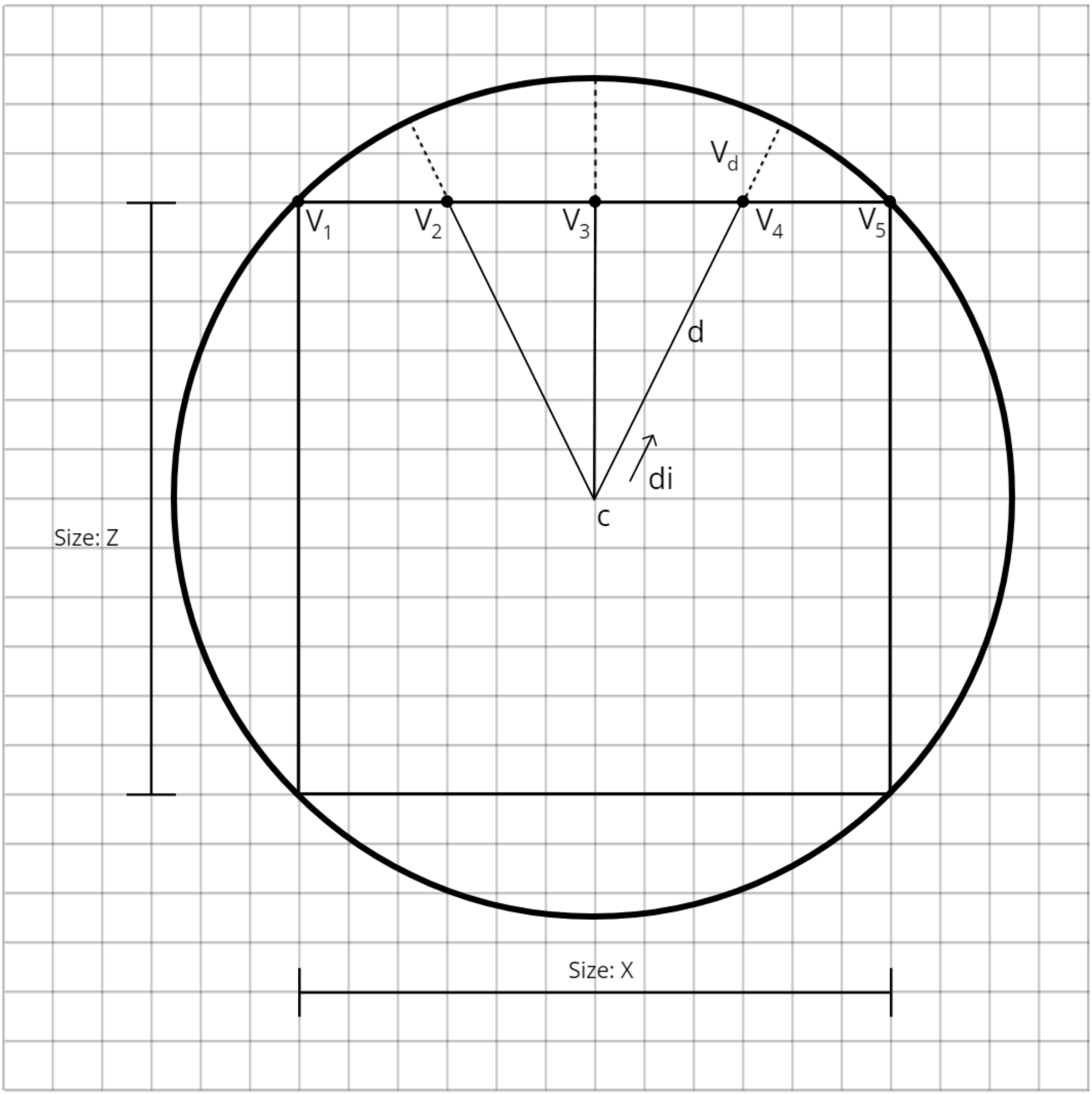


FIGURE 5.3: Top-down representation of segment geometry in regard to the rounding parameter. Vertices $V_0...V_4$ represent points along one face of the shape, which are interpolated along di between initial position V_n and $V_n + Vd$.

Curvature is achieved via quadratic bézier curve: $p0(1-t)^2 + p1(2(t(1-t))) + p2t^2$, where $0 < t < 1$, $p0 = origin$, $p1 = curveoffset$, $p2 = tipoffset$. The center of each loop is positioned by sampling the bézier curve using the loop's associated t-value.

Mesh Joining

A depth first traversal over the list of segments adds each segment's vertices, triangles and UVs to a final mesh. At each point in the stack, a transformation matrix is generated by combining the stored transform matrix of each element in the rest of the stack. This is then applied to the current segment's vertices, causing them to be offset at the correct position and rotation relative to the rest of the shape. In addition, leaf segments get tagged so that end cap geometry can be added to enclose the shape.

A key limitation of this approach is the inability to create cyclic connections within the node graph. Each segment in the graph is defined by its own set of parameters, which independently define its geometry. As a result, aligning the individual segments to form cyclical shapes would require overriding or violating these individual parameters, while not overriding the parameters would result in gaps at connection points between segments. This limitation restricts the algorithm's applicability for a broad range of generic shapes, though it suffices for typical sword shapes.

High level parameters

The subdivision value is applied to all segments generated. This ensures that all segments share the same number of vertices in each loop, allowing for clean geometry when joining meshes. This also provides a very easy control for vertex density and mesh detail without altering the shape itself. This way the designer can adjust the value to achieve a balance between vertex count and mesh detail. Furthermore this feature may be used for producing different levels of detail (LoD) of the same shape, which is a common optimisation technique.

Following the mesh generation algorithm is a post-processing stage in which faceted shading can be applied, and mesh optimisation is performed. The mesh generation algorithm produces meshes with shared vertices. This means that a single vertex can be included in up to eight triangles, this makes manipulation of the shape much more efficient at the cost of having *smoothed* normals. Each vertex has a related normal vector which is the average direction of all triangles the vertex belongs to. To allow shapes to have sharp edges when rendered, additional vertices must be added to allow for separate normal vectors on each triangle. To achieve this a *faceting* algorithm has been implemented. This algorithm iterates over the triangle array, duplicating a vertex each time that it is seen, and updating the indices of the triangle array accordingly. This way, each triangle gets unique vertices, and thus has its own normal.

Algorithm 4 Faceting Algorithm, editing the vertices (verts), texture space coordinates (uv) and triangles (tris) of the mesh data.

```
function FACET(verts, uv, tris, bounds)
    initialize seen ← {}
    originalCount ← COUNT(tris)
    for $i$ ← 0 to originalCount-1 do
        if CONTAINS(seen, tris[i]) then
            ADD(verts, verts[tris[i]])
            ADD(uv, uv[tris[i]])
            tris[i] ← COUNT(verts) - 1
        ADD(seen, tris[i])
    return verts, uv, tris
```

Mesh optimisation accounts for additional loose vertices or redundant triangles that may have occurred as a result of the previous steps. This is achieved by iterating over the vertex and triangle arrays, removing any duplicate triangles and vertices that do not belong to a triangle.

5.2.2 Variational Autoencoder Architecture

The goal of this prototype implementation is to test as many techniques as possible within the same tool. VAEs can be used for reconstruction, randomisation and interpolation tasks, due to their architecture. This section will introduce the architecture design of the VAE.

Given the design of the procedural shape system, the number of segments to produce outputs for had to be defined. A total of 12 segments was chosen to provide a balance between shape variation and model complexity. Alternative segment counts between 4 and 20 were tested. It was observed that with fewer than 12 segments the model struggled to match the input image shapes, while with more than 12 segments, the model had a tendency to underutilise some segments by assigning small size values. It was decided that 12 segments would provide enough control for shape variation, while not introducing too much complexity. An additional benefit of this limitation was that, rather than generating a new set of triangles for each output, a pre-defined set of triangles could be used. This is because the output contains the exact same number and order of vertices each time, when the subdivision is constant.

As the goal of the model was to produce general shapes within the scope of a single silhouette, not all parameters were necessary, and would add to the complexity and parameter count of the model. Of the procedural modelling system's parameters, 7 were chosen for the VAE to train on, based on their affect on the silhouette. These were size (x, y and z), taper top, taper bottom and the curve amount (x and z). A value range also had to be defined, which the outputs of the VAE were scaled to. These were chosen in order to keep results in frame of the differentiable rendering camera. In the final integration, these were once again scaled to an appropriate amount for Unity, with the same ratios as before.

The network architecture consists of three parts: the encoder, the bottleneck and the decoder. The encoder consists of a typical CNN structure, involving four repeating sets of convolution and batch normalisation layers, followed by a final convolution layer. The convolved data is then flattened and passed through a dropout layer, then three linear layers. Input data was first scaled from 128 x 128 pixels to 64 x 64 pixels to reduce model complexity and improve training speed. Convolutional layer output channel counts were in the following order: conv1 = 32, conv2 = 256, conv3 = 256, conv4 = 512, conv5 = 512. As a result of the four batch normalisation layers, the filter sizes go from 64 x 64 down to 2 x 2. Leaky ReLu activations were chosen to improve training efficiency by avoiding issues with *'dying ReLu'* and vanishing gradients [95].

The bottleneck passes the output of the encoder through two linear layers to obtain the mean (μ) and log variance each. These linear layers (four in total) have a size of 512, matching the encoder output. The reparameterization trick, proposed by [232] is then applied to these two values, the output of which is passed to the decoder.

The decoder first progressively increases the dimensions of the input via two linear layers, then passes the result into three separate identical modules, each of which focuses on different sets of shape parameter, these are: the size decoder, bezier decoder, and taper decoder. Outputs of these modules are stacked together then flattened, such that the values are interleaved in the order that the procedural modelling system expects i.e. in a node-wise manner. The Tanh activation function was used for the size and bezier decoder sub-modules so that values were centered around 0 and bound within negative and positive 1. This was necessary for ease of computation in the procedural modelling algorithm. Similarly, the Sigmoid activation function was used for the taper sub-module, as the taper parameter is bound between 0 and 1. Here, the unbounded positive outputs of Leaky ReLU, would not map appropriately.

An alternative decoder was used for phase 1 of training. This decoder directly mirrors the encoder, progressively expanding the dimensionality via transposed convolutional layers until the shape matches the input. The purpose of this decoder was to reconstruct the input images the same way as a typical convolutional autoencoder, such that the encoder could learn to extract features of input images without the asset form conversion that the main architecture imposes.

When integrated into Unity, the interpolation and randomisation functionalities will make use of the separation between the encoder, bottleneck and decoder. When two images are passed into the encoder for interpolation, the two outputs are interpolated to form a single tensor before being passed into the decoder. Randomisation functionality is achieved by producing a new tensor with values drawn from a normal distribution with a mean of $\mu = 3$ and standard deviation of $\sigma = 3$. These values were obtained through tuning and observation of output value ranges from the encoder. The image reconstruction functionality, however utilises all three sections in sequence.

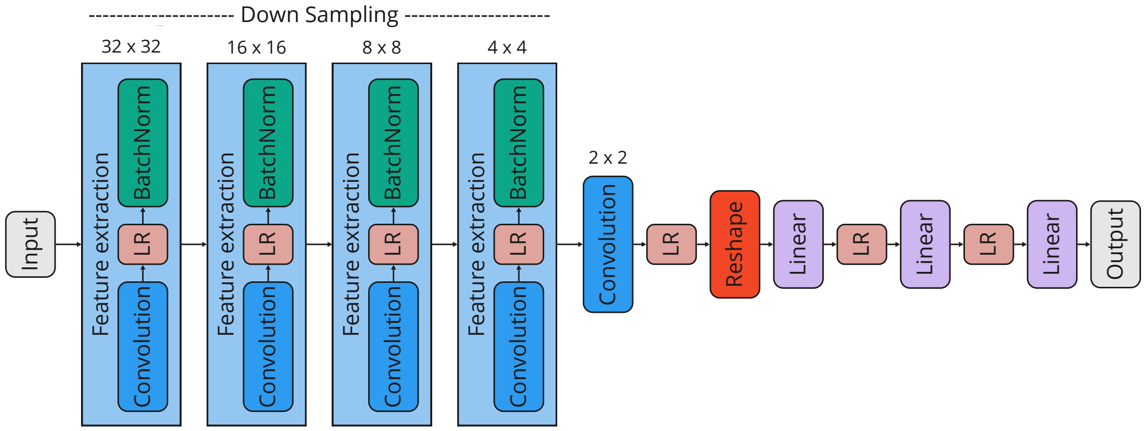


FIGURE 5.4: The final encoder architecture, consisting of downsampling convolutional layers, linear (fully-connected) layers, Leaky RELU (LR), and Batch normalisation (BatchNorm). After convolutional feature extraction, the data is reshaped into a single dimension for the linear layers. Filter dimensions are shown above the layers.

Layer (type)	Output Shape	Parameter Count
Conv2d-1	[B, 32, 32, 32]	832
BatchNorm2d-2	[B, 32, 32, 32]	64
Conv2d-3	[B, 256, 16, 16]	73,984
BatchNorm2d-4	[B, 256, 16, 16]	512
Conv2d-5	[B, 512, 8, 8]	590,080
BatchNorm2d-6	[B, 512, 8, 8]	512
Conv2d-7	[B, 512, 4, 4]	1,180,160
BatchNorm2d-8	[B, 512, 4, 4]	1,024
Conv2d-9	[B, 512, 2, 2]	2,359,808
Dropout-10	[B, 2048]	0
Linear-11	[B, 2048]	4,196,352
Linear-12	[B, 1024]	2,098,176
Linear-13	[B, 512]	524,800

TABLE 5.1: The final encoder summary. *B* represents batch size.

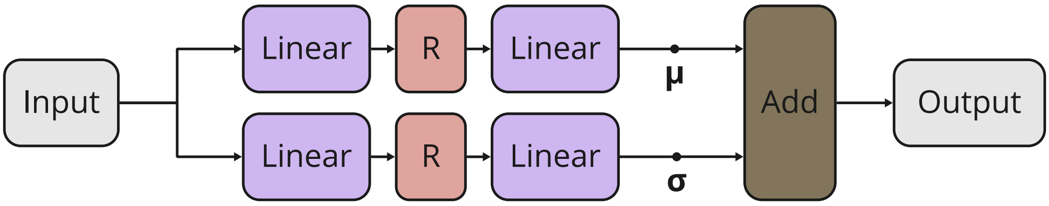


FIGURE 5.5: The final bottleneck architecture, consisting of linear (fully-connected) layers and RELU (R). Mean (μ) and standard deviation (σ) are extracted the encoded features, then added together.

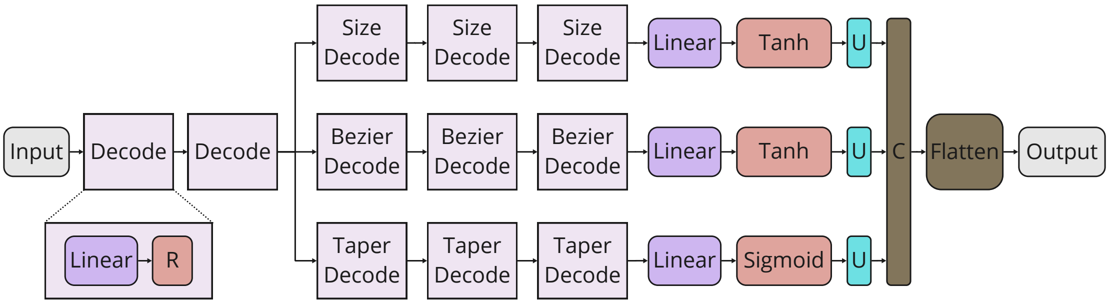


FIGURE 5.6: The final decoder architecture, consisting of linear (fully-connected) layers and RELU (R). Size and bezier decoding modules output via a Tanh function, and the taper decoding module outputs via a Sigmoid function. Outputs are unsqueezed (U) [370] such that they can be concatenated (C) to form a collection of segments, where the corresponding size, bezier and taper parameters are sequential.

Layer (type)	**Output Shape**	**Parameter Count**
Linear-1	[B, 1, 1024]	525,312
Linear-2	[B, 1, 2048]	2,099,200
Linear-3	[B, 1, 1024]	2,098,176
Linear-4	[B, 1, 1024]	1,049,600
Linear-5	[B, 1, 512]	524,800
Linear-6	[B, 1, 24]	12,312
SizeDecoder-7	[B, 12, 2]	0
Linear-8	[B, 1, 1024]	2,098,176
Linear-9	[B, 1, 1024]	1,049,600
Linear-10	[B, 1, 512]	524,800
Linear-11	[B, 1, 24]	12,312
BezierDecoder-12	[B, 12, 2]	0
Linear-13	[B, 1, 1024]	2,098,176
Linear-14	[B, 1, 1024]	1,049,600
Linear-15	[B, 1, 512]	524,800
Linear-16	[B, 1, 24]	12,312
TaperDecoder-17	[B, 12, 2]	0

TABLE 5.2: The final decoder summary. *B* represents batch size.

5.2.3 Supporting training

For a deep-learning model to be trained via backpropagation, the full process from input to output must be differentiable. As the output of the VAE was a set of parameters that had to be first converted to a mesh using the procedural modelling algorithm, then rendered to allow for pixelwise comparison to the input, both the procedural modelling and rendering had to be backpropagatable. This presented two main challenges, differentiable mesh generation, and differentiable rendering.

Differentiable generation was achieved by adapting the procedural shape algorithm from C# to a Python GPU implementation. This allowed the algorithm to be made differentiable via PyTorch autograd, with the added benefit of faster computation using CUDA. For this, the number of parameters was limited to those provided by the final layer of the VAE.

Differentiable rendering was achieved via the PyTorch3D renderer implementation. For this the *Soft Silhoutte Shader* was used with a sigma (edge sharpness) and gamma (opacity fall-off) of 1e-5. The *Rasterization settings* had an image size to match the input resolution (128px), a blur radius of 1e-5 and backface culling was disabled. An orthographic camera was used and positioned such that the 3D origin was positioned at the bottom pixel of the rendered image, therefore causing renders to be consistent with the framing of the input images.

5.2.4 Method for game asset dataset creation

While generalised datasets like ShapeNet [52] enable deep-learning generative systems to learn generic shape reconstruction, their performance varies when tasked with reconstructing asset classes not seen in the training dataset. Games often feature diverse styles and settings, resulting in numerous possible shape classes. Game creators may need generative systems tailored to specific asset types, such as swords, aircraft, or barrels. Additionally, to maintain stylistic cohesion, creators might require non-generic assets that align with their established style. Given this specificity, bespoke datasets are often necessary to achieve effective generative solutions.

Addressing this with a focus on developing shape datasets for unsupervised learning, the following three step method is introduced. Consisting of: *Building a data foundation*, *Preprocessing*, and *Augmentation*.

Building a data foundation

In a typical game design and development pipeline, ideas are first explored via concept art and visual or photographic reference. In the context of game tooling, it could save time to leverage this data, which is already used to inform the creation of assets, as a basis for developing training datasets. Here, stylistically cohesive and relevant data can be created or otherwise sourced to form a small initial dataset. This may also be further supplemented by other GAG generated content.

Preprocessing

In order to use this data in training deep-learning methods, it must first be standardised and preprocessed to ensure smooth training. To begin with, it is necessary to remove un-needed data, such as colour and texture in the case of shape focused generation, as well as background removal. Furthermore, the data should be standardised to ensure consistency across the dataset. This includes scaling images to the same size and resolution and centering shapes within the data frame.

Augmentation

As deep-learning methods require a large amount of data to avoid overfitting during training, it is necessary to expand the dataset through augmentation. Common methods of data augmentation include cropping, rotating and applying filters to the source data. However, these methods do not provide alternate variations of the selected shapes, but rather account for variation in the quality of input. Subtle automated manipulation of the source shapes can instead introduce conceptual variation into the dataset. For example, raster images can be converted into vector graphics, and these vectors can be moved to create variation. Alternatively, source designs can be combined to create new variations through existing generative techniques [99]. Furthermore, the degree of manipulation as well as constraints can be defined to limit the amount of deviation from the initial source material. This can be achieved by converting the initial data into a manipulable format such as vector graphics.

5.2.5 The sword dataset

To train the model, a large dataset of sword silhouettes including a wide variety of different shapes and styles was required. As no publicly accessible dataset of this type existed, a new dataset was created for this purpose.

In accordance with the above method, an initial set of concept designs and assets were collected to form the basis of the dataset. Various sources were accessed including the Unity Asset store [461] and game wikis for Diablo 2 [510], Diablo 3 [511] and Final Fantasy IX [512]. Additionally, CC0 licensed images were accessed from Smithsonian open access [189] and FreeImages [117]. This dataset was supplemented further with a set of sword images generated through Stable Diffusion [390]. The number of images sourced from each location are presented in table 5.3. In total there were 513 images obtained from the above mentioned sources.

Source	Count	Total count
Web CC0	122	513
Unity Asset Store	165	
Stable Diffusion	60	
Diablo 2 Wiki	21	
Diablo 3 Wiki	60	
Final Fantasy IX Wiki	85	

TABLE 5.3: The number of source images obtained from each source to form the dataset.

To pre-process the dataset, source images first had their backgrounds removed via Background Remover AI [334], which uses a pre-trained u2net model to identify and remove image backgrounds. Following this, the output images were reduced to their alpha channel to produce greyscale silhouettes.

To facilitate augmentation of data, the image silhouettes were then loaded into Photoshop and batch processed using the ′actions′ system. First, colour range selection is used to select the shape, the shape is then converted to a path, which is then converted into a vector shape. The original layer is then deleted and the project is saved as a *′.psd′* file. Photoshop is then used to bulk load the *′.psd′* files as layers, then export each layer as an SVG file for the data augmentation stage.

Step	**Details**
1. Color Range	
Fuzziness	125
Minimum	Grayscale: 50
Maximum	Grayscale: 2.81
2. Make Path	
Form	Selection
Tolerance	2 pixels
3. Make Fill Layer	
Using	Fill Layer
Type	Solid Color
Slot Color	Grayscale
Gray	100
4. Select Layer "Background"	
5. Delete Current Layer	
6. Save	

TABLE 5.4: The Photoshop actions for preprocessing.

To augment the SVG data, Python libraries *svgpathtools* and *cairosvg* were used to create variations of the vector silhouettes. This was achieved by randomly shifting the positions of 10% of the vertices in each shape, with the displacement for each affected vertex drawn from a uniform distribution ranging between -20 and +20 pixels along both the x and y axes. 30 variations were created for each vector shape via this method. Control points could have also been randomised via this approach, however this caused the shapes to deviate too far from the original designs, so this was not done. The resulting dataset of 15,390 images was further refined by examining the shape of each output and removing any results that had inverted shapes, i.e. poor topology resulting in folds or misalignment. Furthermore, the final images were rotated so that they aligned with the vertical axis, i.e. pointing from bottom-to-top or top-to-bottom. This ensured that training data could be matched to the output of the trained model, as output segments would be constructed from the bottom of the image frame, upwards. Variation in top-bottom, bottom-top directions was included as both forms are equally achievable given the rendering constraints, and would support inputs from either orientation. This can be observed in Figure 5.8, in which sample 100 is oriented from top-bottom. The final dataset image count after this process was 13,728.

5.2.6 Training

Throughout development, the VAE architecture and training procedure took multiple iterations to obtain the final model. The training process involved two phases. In the first phase, the encoder was paired with a convolutional decoder and the model was trained to match input images via pixel-wise MSE loss. In the second phase the decoder was replaced with a decoder that output a set number of parameters corresponding to the procedural shape parameters of the segment nodes.

Two main varying factors were used to produce the final model configuration and determine the size and complexity of the model. These were the filter counts of the five incremental layers within the encoder, and the size of the bottleneck layer. As the encoder and decoder were mirrored in phase one, filter counts were applied in ascending order within the encoder and descending order within the decoder for this setup. The size of the bottleneck was also required to match the filter count of the final encoder.

The phase two decoder consisted of a series of fully connected layers that progressively increased to 4 times the bottleneck size. The ouput of this was then passed to three sub-decoders. These were the size decoder, bezier decoder and taper decoder. Each of these decoders were structured to have four layers, the first three progressively reducing in dimension until matching the bottleneck size and the final layer reducing this to match the appropriate number of parameters, which in each case was 2 times the node count, as there were 2 size values: x and y, 2 bezier values: tip and center offset, and 2 taper values: forward and backward.

Feeding these outputs into the differentiable procedural shape algorithm, then rendering the shape using the differentiable renderer, the generated results could be compared with the input images and a loss value could be computed. The loss function combined a Sobel loss for edge based comparison, and a KL-divergence to for a variational loss.

The final chosen layer filter configuration was: 32, 256, 256, 512, 512. With the bottleneck size being 512. This yielded the lowest loss given the above loss setup for phase one of training, as shown in figure 5.9. This first phase model was trained for 491 epochs, at which point loss had plateaued. Figure 5.7 shows the progress of pre-training outputs from the 1st epoch to the 400th epoch.

After phase one was complete, the convolutional decoder was replaced with the phase two decoder. The bottleneck and new decoder were then trained with the pre-trained encoder's parameters locked. After 200 epochs of training, the encoder's parameters were unlocked and the full model was trained for a further 200 epochs. For this phase, a learning rate scheduler was configured to reduce the learning rate by a factor of 0.9 each time a plateau was detected across the preceding 20 epochs of training. This allowed the model to continue improving throughout the extended training period. Figure 5.8 shows the progress of outputs from the 1st epoch to the 400th epoch.

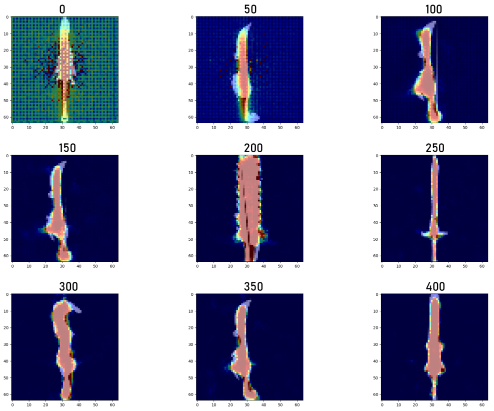


FIGURE 5.7: Output images during phase one of training at 50 epoch increments between the first epoch and epoch 400.

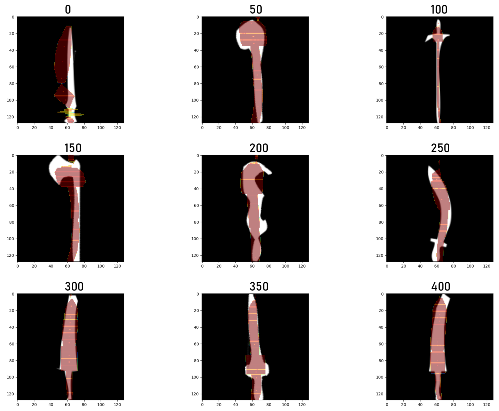


FIGURE 5.8: Output images during phase two of training at 50 epoch increments between the first epoch and epoch 400.

Colour	Layer Configuration (filter count per layer)	Final Loss Value
🟧	1024, 32, 128, 256, 512, 1024	0.01368
🟦	256, 64, 128, 256, 512, 1024	0.01426
🟪	512, 128, 256, 512, 512, 1024	0.02499
🟦	512, 32, 128, 256, 512, 1024	0.01261
🟥	512, 32, 128, 256, 512, 512	0.01159
🟩	512, 32, 256, 256, 512, 512	0.01105

TABLE 5.5: The colour coding of lines on figure 5.9, along with final loss values.

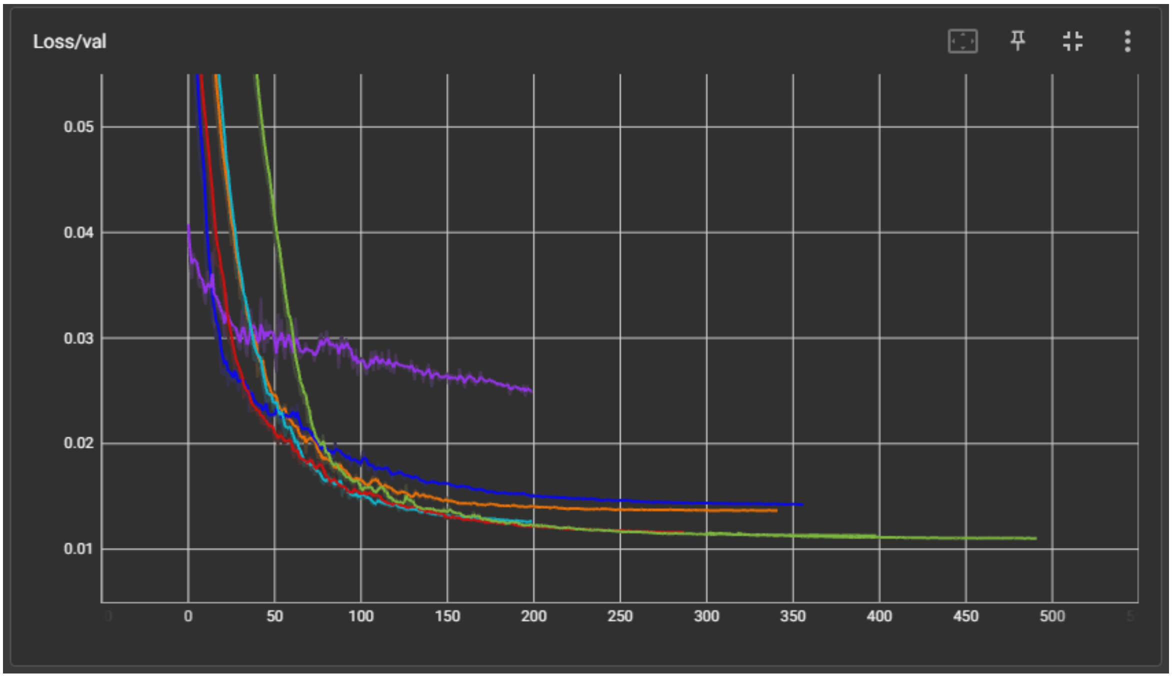


FIGURE 5.9: The step-wise validation loss of different configurations.

5.2.7 Unity integration

The final model was integrated into the Unity Engine and its features were presented through a graph view user interface. To achieve this an Open Neural Network Exchange (ONNX) file of the final model checkpoint was generated, then loaded within Unity via the Barracuda API. In the UX preference study (chapter 3) no statistically significant preference was found between the tested interface types. Instead, the results highlighted a need for good usability of the underlying tool's features. Therefore, a UX should be chosen based on its ability to encompass the necessary controls for the system. Therefore, a graph view interface was chosen as it allows for the assignment of each nodes parameters while also mapping the high-level structure of the shape. This style of interface allows for flexibility in designing shapes as well as the configurability required for manual adjustment of parameters in real-time.

To import user sourced images as inputs, the previous pre-processing method involving the use of the stand-alone background remover tool [334] could not be integrated seamlessly. Therefore, an alpha matting approach was implemented within tool to achieve the conversion to silhouette. This method used the same pre-trained u2net model used within the original tool [334] to produce an alpha matte from an input image.

The graph-view interface, shown in figure 5.10 is situated within a Unity Editor window. Access to generative features are provided in a sub-panel, as well as a real-time preview of the generated mesh. Features such as saving and loading of shape graph data are provided, as well as an export option that outputs the mesh and chosen materials in the serialized Unity GameObject (Prefab) format.

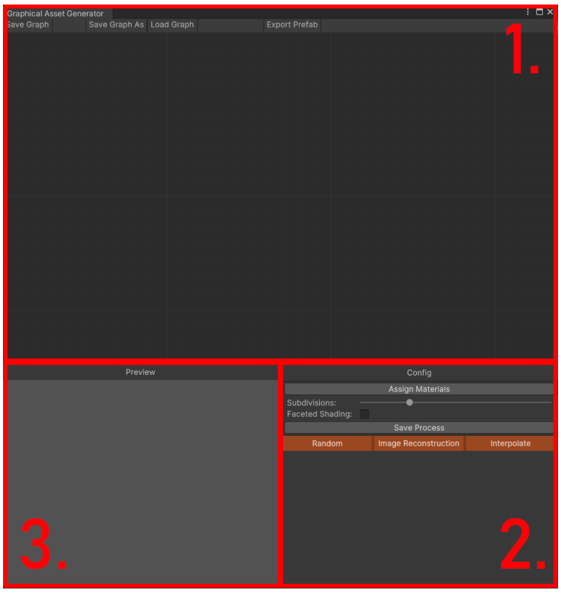


FIGURE 5.10: The main tool user interface, broken down into: graph view (panel 1), config panel (panel 2) and preview box (panel 3)

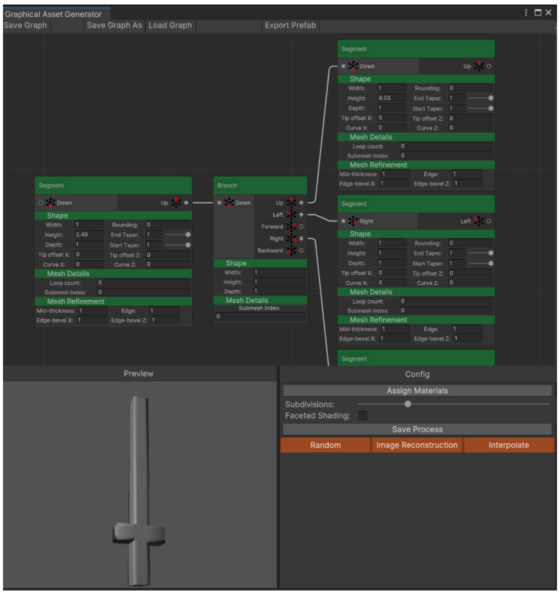


FIGURE 5.11: The tool screen populated with graph data.

Configuration panel

Under the configuration panel, general mesh options can be found, including a slider for the subdivision count, a toggle for facetted shading, and a button that creates a pop-up menu for assigning materials. Below this are three tabs for accessing the generative functionality: *Random*, *Image Reconstruction* and *Interpolation*.

The *random* tab provides a button for full random generation, and three buttons for randomising dimensions, edge shape and curves respectively. Full random generation is achieved by sampling the latent space of the VAE using a normal distribution, via random mean and standard deviation values. These values are fed into the bottleneck, then the decoder to produce the output parameters. The dimension, edge shape and curve randomisation simply applies a pseudo random value to relevant parameters of selected nodes in the graph. The edge shape and curve randomisation applies the same values to all selected nodes such that they share the same randomised form.

The *Image Reconstruction* tab allows the user to import an image of their choosing via a system file-picker. Once selected, the image is automatically pre-processed using the alpha matting solution and dimension resizing. The user is asked to manually adjust the rotation of the image via a slider so that the orientation is consistent with the training data, i.e. aligned on the vertical axis. Upon clicking *generate* the processed image is passed into the VAE and the generated result is loaded into the graph-view.

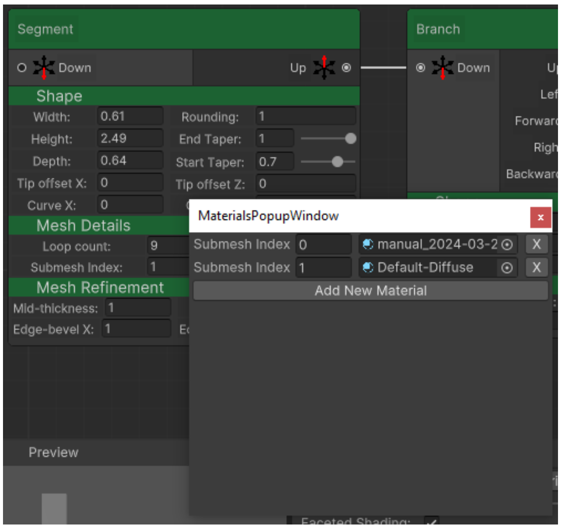
The *Interpolation* tab provides two image input fields, of the same type found in the *Image Reconstruction* tab. These represent the two shapes that will be interpolated between. The user is also provided a slider for inputting the t value of the interpolation between the two provided images. Upon clicking *generate* the encoder processes each image independently, then the resulting tensors are linearly interpolated before passing the result into the decoder. For example, consider two images, A and B. Image A will first be processed by the encoder and bottleneck models, followed by image B. Each corresponding value in the two output tensors from image A and image B are then linearly interpolated with the given t parameter to produce a new tensor with values in-between the two. This is then passed into the decoder model to produce the final output.

For the purposes of data collection, and additional *Save Process* button was added. The user study participants were instructed to click this button at the end of each task to record their data. Upon clicking this button, the current graph is saved along with the processed input images, the assigned materials, generated mesh and overall prefab, along with information on the generative process used. This data is packed into a folder within the *Asset* directory of the Unity project, then compressed as a zip file. This will allow testing participants to share their data easily.

Manual parameter technique

To design swords using manual parameter input, users create nodes within the canvas of the tool's window using context menus and configure their individual parameters, as shown in figure 5.13. Users connect nodes by dragging from the output of one node into the input of another, a connection is then signified by a line within the interface. Icons on node inputs and outputs signify the orientation of the connection along Cartesian axes relative to the global coordinate system. Connected nodes are then rotated based on the direction of the connection, for example connecting a segment to the *left* output of a branch node will cause the output of the segment to also point left globally. This is all represented dynamically by the directional icons. A full sword shape can be created by chaining together segment and branch nodes in this manner.

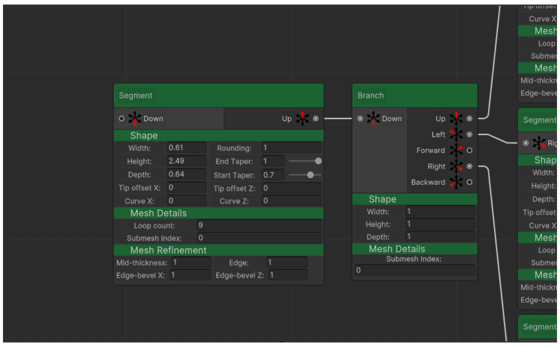
Materials can be applied to a generated shape by matching submesh indices on each node to corresponding submesh indices in a pop-out materials window found within the *config* panel, as shown in figure 5.12. Users can assign any material within their Unity project using this method. Additionally, within the *config* panel, users can adjust the overall subdivision factor of the shape and toggle *faceted shading*, as shown in figure 5.14. The latter applies the algorithm 4 to create sharp edges between rendered faces at the cost of generating more vertices.



Width: 0.61
Rounding: 1
Height: 2.49
End Taper: 1
Depth: 0.64
Start Taper: 0.7
Tip offset X: 0
Tip offset Z: 0
Curve X: 0
Loop count: 9
Submesh Index: 1
Mid-thickness: 1
Edge-bevel X: 1

Submesh Index 0 manual_2024-03-2
Submesh Index 1 Default-Diffuse

FIGURE 5.12: The materials pop up window, that allows users to assign materials to sub-mesh indices.



Width: 0.61
Rounding: 1
Height: 2.49
End Taper: 1
Depth: 0.64
Start Taper: 0.7
Tip offset X: 0
Tip offset Z: 0
Curve X: 0
Curve Z: 0
Loop count: 9
Submesh Index: 0
Mid-thickness: 1
Edge: 1
Edge-bevel X: 1
Edge-bevel Z: 1

FIGURE 5.13: The graph view populated with segment and branch nodes, containing the parameter control UI (panel 1).

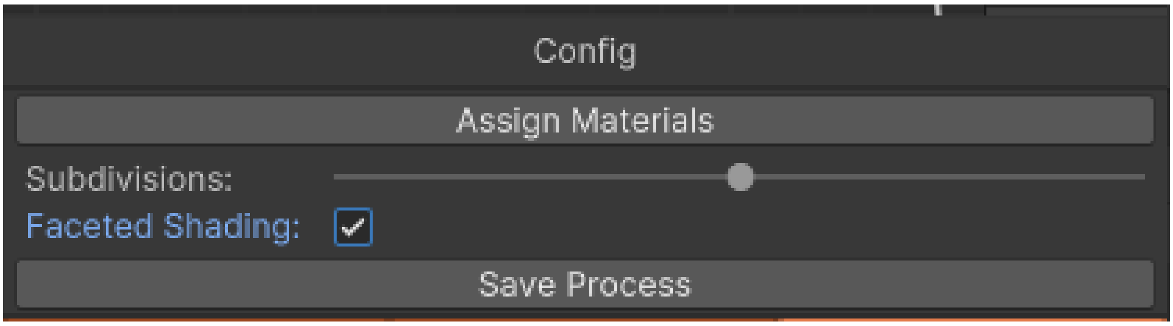


FIGURE 5.14: Config controls including a subdivision slider and toggle for *faceted shading* (panel 2).

Image reconstruction technique

To create sword shapes from images, users select the the *image reconstruction* tab within the *config* panel, as shown in figure 5.15. This provides an image input box with a button for opening a file picker. After clicking the button marked with ellipses, the user navigates to the file location of a *JPG* or *PNG* image file representing a front view of a sword. Once an image is selected, it is automatically processed into a silhouette using the u2net model described in section 5.2.5. The result of the processing is loaded into a preview box, where the user adjusts a slider to rotate the image until the shape is vertical, as shown in figure 5.16. Clicking generate then passes this input through the VAE, and the output nodes are loaded into the graph view canvas.

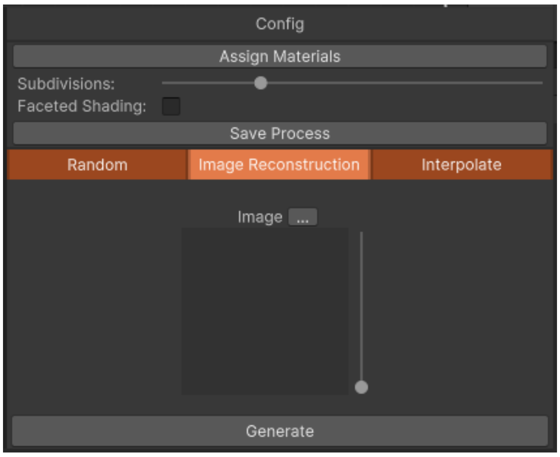


FIGURE 5.15: The image reconstruction tab (panel 2).

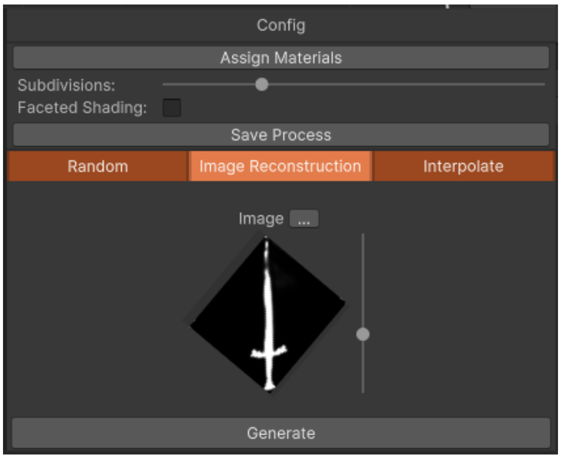
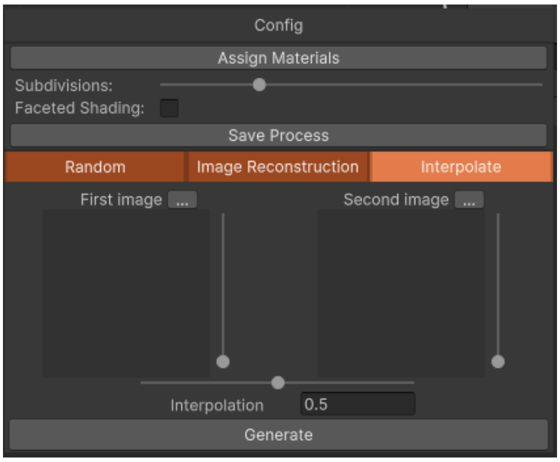


FIGURE 5.16: The populated image reconstruction tab, with the input image correctly rotated (panel 2).

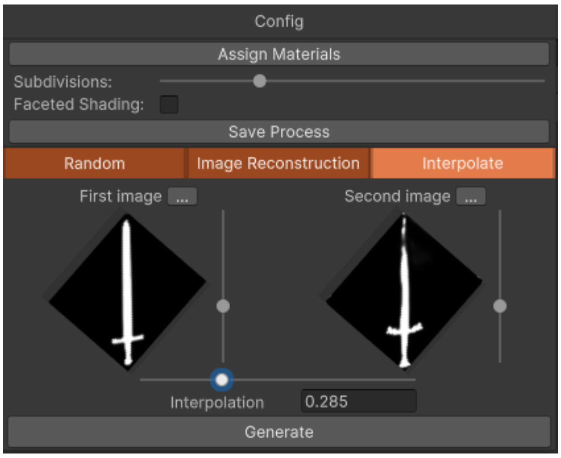
Image interpolation technique

The interpolation technique is used by selecting the *interpolate* tab in the *config* panel, as shown in figure 5.17. The same input method as shown in section 5.2.7 is used to select two images within the UI, and a slider below the two loaded images can then be adjusted to determine the blend of inputs. This slider represents a t value between 0 and 1, shown figure 5.18. Clicking the *generate* button then passes each input through the encoder of the VAE, and a latent vector is formed via linear interpolation between the two encoder outputs before passing this to the decoder. The resulting nodes are then loaded into the graph view canvas.



Interpolation 0.5

FIGURE 5.17: The interpolation tab (panel 2).



Interpolation 0.285

FIGURE 5.18: The populated interpolation tab, with input images correctly rotated (panel 2).

Randomisation technique

Randomisation is accessed via the *random* tab within the *config* panel, shown in figure 5.19. Clicking the generate button will produce a random outcome from the learned latent space of the VAE model. This is done by generating a random vector consisting of values from a normal distribution, and passing this vector into the decoder. The output is then loaded into the graph view canvas, where it can be further randomised by selecting segments within the graph view canvas and clicking the *randomise selected dimensions*, *edge shape* or *curves buttons*. Users can apply these modifications one by one. Multiple modifications can be applied repeatedly or sequentially.

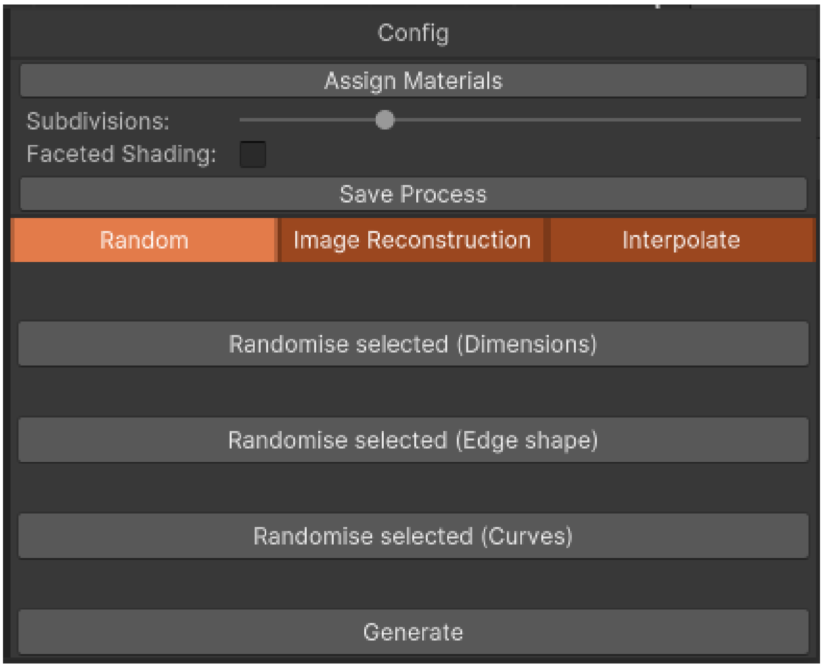


FIGURE 5.19: The random tab (panel 2).

5.3 Chapter Summary

In this chapter, a proof-of-concept prototype sword generation tool was developed and integrated into the Unity engine, incorporating a total of four techniques representing the four *idea fidelity* groupings. To achieve this a method for creating datasets for game asset shapes was developed and utilised to create a novel sword silhouette dataset. This dataset was used to train the GAG system which consisted of a VAE trained to convert image silhouettes to procedural shape parameters.

Chapter 6: Empirical Study

In the previous chapter, a proof-of-concept prototype was developed, implementing representative techniques from the four *idea fidelity* groupings first introduced in chapter 2. This prototype was integrated within the Unity engine editor, and utilises the Barracuda API to interface with a VAE trained on a novel sword silhouette dataset. A method for developing asset shape datasets for unsupervised training tasks, which was used to create the sword dataset, was also presented.

In this chapter, the proof-of-concept prototype is tested and evaluated via a user study examining the preferences and needs of game designers and developers. This study explores the relationship between perceived quality, speed, and controllability, and how these factors influence the usefulness of each implemented technique, as well as their preferred usage in the development pipeline.

6.1 Introduction and research questions

To assess game designer and developer opinions and preferences regarding the developed proof-of-concept prototype, a user study has been planned and conducted. The focus of this user study is to identify the difference in preference between the four techniques provided by the prototype tool, the perceived usability of the tool, and the relevance and usefulness of such a tool in game development pipelines. Furthermore, the opportunity has been taken to compare the functional prototype tool with the corresponding mock up from the previous study (ST1).

This study is structured as a repeated measures experiment, measuring the user perception of each technique and its interface, followed by an overall rating of usability via SUS [39] and the set of preference questions used to assess the mock-ups in chapter 4. This is followed up with an optional interview in which more nuanced feedback could be provided by participants, while helping to obtain further insights into the quantitative data.

In this study, user perception of techniques is assessed via participant ratings on four main points. These are: *Quality*, *Speed*, *Control* and overall *Usefulness*. In chapter 3 three main forms of evaluation metric are identified: *artefact Validation*, *artefact Quality* and *operation evaluation*. *Artefact Validation* assesses whether the output of the method can be perceived as belonging to the target asset type. Most methods, if tested, have been evaluated via some form of artefact validation metric. While it is variable whether or not *operation evaluation* or *artefact validation* is employed. Therefore the questions instead focus on the perceived quality and operational aspects of the tool. For the former, a general question of quality is posed, and for the latter, performance and controllability are assessed. Speed is chosen as a measure of performance, as it is the only identified performance metric that can be directly perceived by a user, while controllability satisfaction is also directly experienced. An overall usefulness rating is also included to assess whether participants would use the particular techniques of the tool in their own projects.

This study aims to answer 4 core questions surrounding generative techniques in the context of tools for graphical asset creation. ST2-CQ1 and ST2-CQ2 regard the usefulness and core qualities of the techniques respectively, while ST2-CQ3 regards the relevance of such a tool in different parts of the design/development pipeline. Additionally, ST2-CQ4 regards how the tool is perceived or rated differently when functional as opposed to being a mock up. These core questions (ST2-CQ1 to ST2-CQ4) are presented below.

ST2-CQ1 Which type of generative technique do game designers/developers find most useful and why?

ST2-CQ2 Which type of generative technique do game designers/developers find to be fastest, highest quality and most controllable and why?

ST2-CQ3 Where in the design/development pipeline would game designers/developers find value in such as system?

ST2-CQ4 Are preferences different for the functional prototype in comparison with the non-functional mock up?

These core questions have been expanded into a series of research questions (RQs) which have a corresponding questionnaire question presented in table 6.1. Listed below are the research questions ST2-RQ1 to ST2-RQ8. These are ordered similarly to the mock up preference study (ST1) research questions for ease of comparison (ST2-CQ4). There are some key differences in these research questions, ST2-RQ1 is concerned with the usefulness of techniques in place of the mock ups of ST1-RQ1, while ST2-RQ8 focuses on the perception of speed (ST2-RQ8.1), quality of output (ST2-RQ8.2) and controllability (ST2-RQ8.3) for techniques, where technique *i* is represented as T_i, as defined in table 6.3.

ST2-RQ1 Which T_i do game designers/developers find most useful?

ST2-RQ2 Where in the pipeline would game designers/developers find value in the tool?

ST2-RQ2.1: Do designers/developers find value in the tool for Generating inspiration.

ST2-RQ2.2: Do designers/developers find value in the tool for Exploring ideas.

ST2-RQ2.3: Do designers/developers find value in the tool for Creating placeholder assets.

ST2-RQ2.4: Do designers/developers find value in the tool for Creating variations of existing (complete) assets.

ST2-RQ2.5: Do designers/developers find value in the tool for Creating assets from existing (complete) designs.

ST2-RQ2.6: Do designers/developers find value in the tool for Creating assets from scratch.

ST2-RQ2.7: Do designers/developers find value in the tool for Player (or user) made content.

ST2-RQ3 If the tool were to be used in a pipeline, would designers/developers prioritise volume of output or quality of output?

ST2-RQ4 How much time would designers/developers find acceptable for the tool to take in generating a single asset?

ST2-RQ5 Do designers/developers prefer the tool as a stand-alone solution or integrated into a game-engine?

ST2-RQ6 How important is the ability to configure or modify the tool according to designers/developers?

ST2-RQ7 What is the perceived usability of the tool?

ST2-RQ8 Which T_i is rated highest for the following aspects...?

ST2-RQ8.1: Which T_i do designers/developers find to be fastest.

ST2-RQ8.2: Which T_i do designers/developers find to produce the highest quality assets.

ST2-RQ8.3: Which T_i do designers/developers feel in the most of control when using.

10 main hypotheses were formed in relation to the research questions, based on the results of the first study (ST1) and the amount of interaction required for each technique.

This study also aims to observe the impact of participant demographic on the ratings and preferences of the techniques. Four impact hypotheses are presented, mirroring those of the mock up study (ST1) and are listed below.

ST2-IH1 Role impacts technique rating.

ST2-IH2 Size of team impacts technique rating.

ST2-IH3 Experience impacts technique rating.

ST2-IH4 Age impacts technique rating.

ST2-IH5 The rating of speed, quality and controllability each correlate with the rating of usefulness.

TABLE 6.1: Survey questions corresponding to each research question

Research Question	Survey Question
ST2-RQ1	"Please rate on a scale of 1 (strongly disagree) to 5 (strongly agree), how much you agree with the following statement: I would find this particular technique useful in the projects I work on."
ST2-RQ2	“Where in your development pipeline would you find value in this software tool? Please tick all boxes that apply.”
ST2-RQ3	“Considering your answer to the previous questions, would you prefer if this tool generated a large variety of assets at a lower quality, or that it generated a small variety of high-quality assets?”
ST2-RQ4	“Please indicate the largest timescale you would find acceptable for generating a single graphical asset if you were to use this software tool in your projects.”
ST2-RQ5	“Given the option, would you prefer such a system to exist as stand-alone software, or integrated into your game engine editor of choice?”
ST2-RQ6	“Please rate on a scale of 1 (not important) to 5 (very important), how important to you, is the ability to configure or modify such a tool. For example, importing your own bespoke algorithms or trained models?”
ST2-RQ7	System Usability Scale [39]
ST2-RQ8	"For each aspect listed below, please rate the technique you have just used on a scale of 1 (very poor) to 5 (very good)."

TABLE 6.2: List of hypotheses and associated research questions.

RQs	Hypotheses	
ST2-RQ1	ST2-H1	Participants will find T_1 to be the most useful technique
ST2-RQ2	ST2-H2	Participants find the tool valuable in all stages
ST2-RQ3	ST2-H3	Participants prefer shorter generation times over quality/variety
ST2-RQ4	ST2-H4	Participants find asset generation times of less than a minute acceptable
ST2-RQ5	ST2-H5	Participants prefer an integrated solution over a standalone
ST2-RQ6	ST2-H6	Participants prefer an open solution i.e., high-configurability
ST2-RQ7	ST2-H7	Participants identify the solutions useable
ST2-RQ8.1	ST2-H8	Participants will find T_4 to be fastest
ST2-RQ8.2	ST2-H9	Participants will find T_1 to produce the best quality assets
ST2-RQ8.3	ST2-H10	Participants will find T_1 to provide the most control

Technique	Technique code
Parameter guided	T_1
Reconstruction	T_2
Interpolation	T_3
Randomisation	T_4

TABLE 6.3: Implemented techniques and their abbreviations.

6.1.1 Procedure

Data was collected remotely via a single web form hosted on Microsoft Forms. This form guided participants through the process of installing and testing the technique, providing questionnaires between each technique tested. Instructions were provided for participants to install the tool on their computer, and video tutorials were provided to show participants how to navigate the UI and interact with each technique being tested. Participants were presented with a series of tasks contextualised around a design/development scenario. In this scenario the participants were presented with two character personas and asked to design a sword for each of them, for every technique. The personas, shown in figure 6.1, were provided with the following description:

"You are working on a fantasy role-playing game and have been tasked with designing weapons for two rival characters. Your goal is to create weapon designs that match the personalities and attributes of these two characters. You will be making use of generative tools integrated within the Unity Editor to produce your designs."

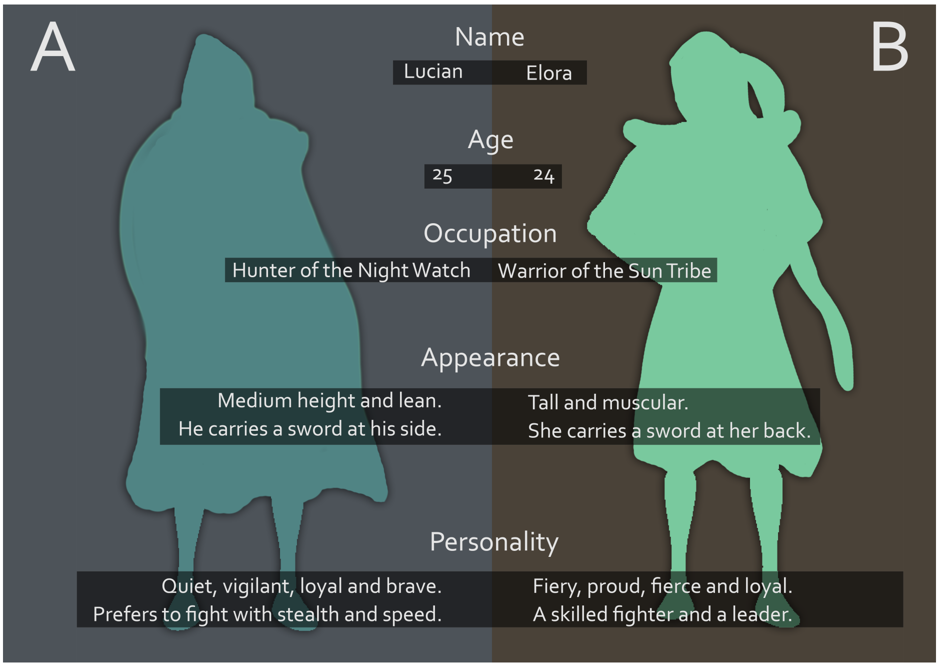

A
B
Name
Lucian
Elora
Age
25
24
Occupation
Hunter of the Night Watch
Warrior of the Sun Tribe
Appearance
Medium height and lean.
He carries a sword at his side.
Tall and muscular.
She carries a sword at her back.
Personality
Quiet, vigilant, loyal and brave.
Prefers to fight with stealth and speed.
Fiery, proud, fierce and loyal.
A skilled fighter and a leader.

FIGURE 6.1: Character personas for the testing scenario.

Participants

A convenience sampling approach was employed in the recruitment of participants. Game development communities such as LUUG and BCS Animation and Games specialist group were contacted through their respective gatekeepers. Participants were required to be over the age of 18 with some amount of professional experience in game design or development. To successfully test the tool, participants were also required to have some experience in the Unity Engine and to have it installed on their machine.

Questionnaires

Prior to testing, participants provided demographic information including their age group, years of experience and size of team they are used to working in. Participants then completed a short questionnaire after testing each of the four techniques. This questionnaire included questions corresponding to ST2-RQ1 and ST2-RQ8, as shown in table 6.1. After the participant had tested the four techniques and completed the associated questionnaires, they completed a further set of questions regarding their preferences for the overall tool. These questions corresponded with research questions ST2-RQ2, ST2-RQ3, ST2-RQ4, ST2-RQ5, ST2-RQ6 and ST2-RQ7, which are presented in table 6.1. These questions match those of ST1-RQ2 to ST1-RQ7 and are intended as a repeated measure, for comparison of the original mock up with the prototype tool.

Semi-structured interviews

Participants were asked to schedule and complete a 10 to 15 minute interview following the completion of the tool testing. Within the web form, participants were provided with a link to book a meeting time at their own convenience. This was on an opt-in basis. During these interviews, participants were asked questions to confirm their answers within the questionnaires to obtain more nuanced explanations of their preferences and reasoning. Furthermore, they were given an opportunity to voice any thoughts or opinions regarding the tool that they could not address through the questionnaires. These interviews were recorded via note-taking.

6.1.2 Results

In total, 18 participants completed the testing process between the approved dates of 24/11/2023 and 31/03/2024, and 3 agreed to a follow-up interview. The following subsections will present the analysis of questionnaire results, starting with technique preference (ST2-RQ1 and ST2-RQ8), then demographic impact analysis (ST2-IH1 to ST2-IH4) and comparisons to the original mock up (ST2-RQ2 to ST2-RQ7), followed by the interview findings.

Technique preference

Technique preference is established along four variables: Usefulness (RQ1), Speed (RQ8.1), Quality (RQ8.2) and Controllability (RQ8.3). To compare the four repeated measures corresponding to the techniques, a Friedman test was first conducted, followed by a set of pairwise Wilcoxon

Signed Ranks tests to assess the statistical significance of any differences in rating. The results of both tests are presented in table 6.4.

For speed (RQ8.1) there is an overall statistically significant difference in ratings between techniques ($P < 0.05$) when conducting the Friedman test. Wilcoxon results show significant differences ($P < 0.05$) between all pairwise comparisons except for the comparison between image reconstruction (T_2) and interpolation (T_3), where $P = 0.083$. Randomisation (T_4) was rated fastest, followed by reconstruction (T_2), interpolation (T_3) then parameter guided (T_1). This result is in line with the amount of interaction and initiative required from users within these techniques, as the randomisation technique is the least involved of the four with the least user initiative. Whereas the parameter guided technique requires the most user input of the four and therefore takes the most amount of time to produce an asset. This result therefore confirms ST2-H8.

For quality (RQ8.2), the Friedman test shows that there is a significant overall difference in ratings between techniques ($P < 0.05$). In addition, all Wilcoxon tests show statistically significant pairwise differences between techniques ($P < 0.05$). The rating for asset quality with the parameter guided technique (T_1) was highest, followed by randomisation (T_4), interpolation (T_3), then reconstruction (T_2). This result is expected, due to the parameter guided technique being entirely dependent on user inputs, thus confirming ST2-H9. Randomisation being the second highest rated technique is likely due to it being sampled from a normal distribution, resulting in generic but believable average results. Interpolation and reconstruction, due to the model's weakness in generalisation, produced results that did not match the inputs in many cases as confirmed by interviews in section 6.1.2.

For controllability (RQ8.3) the Friedman test shows a significant overall difference in ratings between techniques ($P < 0.05$). The Wilcoxon tests show the difference between reconstruction (T_2) and interpolation (T_3), as well as parameter guided (T_1) and randomisation (T_4) to be insignificant ($P > 0.05$), while other pairwise comparisons were significant ($P < 0.05$). The rating of controllability was highest for parameter guided (T_1) and randomisation (T_4), followed by interpolation (T_3) and reconstruction (T_2). These results are in line with the amount of direct controls provided to users in each of these techniques, with parameter guided design having the most, and reconstruction having the least. This confirms ST2-H10.

For usefulness (RQ1) the Friedman test shows that the difference in rating between techniques is statistically significant ($P < 0.05$). The Wilcoxon tests show the difference between ratings for reconstruction (T_2) and randomisation (T_4), and interpolation (T_3) and randomisation (T_4) are statistically significant ($P < 0.05$), while other comparisons are not ($P > 0.05$). Overall, randomisation (T_4) was rated significantly higher than both reconstruction (T_2) and interpolation (T_3). Randomisation (T_4) and parameter guided (T_1) are both rated similarly with a non statistically significant difference in regard to usefulness. The lower ratings of reconstruction and interpolation can be partially attributed to the models weakness in generalising to varied and unseen inputs. However, the interviews discussed in section 6.1.2 suggest there are other reasons for the rating of usefulness regarding parameter guided and randomisation, such as the ability to create 3D models quickly and with little experience. These results confirm ST2-H1.

Users care about the quality of the output and the amount of meaningful control they have over the outcome. Speed is less of a concern due to the tool being faster than the manual methods they already employ. These manual methods are favorable because they provide maximum control and the quality is unlimited. However, the main interest in this tool is not for creating final assets in a game, but rather to explore ideas, create placeholder assets or generate inspiration. Users deemed the randomisation more controllable than reconstruction and interpolation because they could target a specific part of the shape and keep the segments that they prefer.

TABLE 6.4: Friedman and Wilcoxon results for differences in technique ratings (ST2-RQ1 and ST2-RQ8)

Research Question		Wilcoxon sig.	Friedman sig.
RQ1	T_2 -> T_1	.083	.002
	T_3 -> T_1	.277	
	T_4 -> T_1	.335	
	T_3 -> T_2	1.000	
	T_4 -> T_2	.003	
	T_4 -> T_3	.003	
RQ8.1	T_2 -> T_1	.001	<.001
	T_3 -> T_1	.006	
	T_4 -> T_1	.002	
	T_3 -> T_2	.083	
	T_4 -> T_2	.003	
	T_4 -> T_3	.001	
RQ8.2	T_2 -> T_1	<.001	<.001
	T_3 -> T_1	<.001	
	T_4 -> T_1	.020	
	T_3 -> T_2	.001	
	T_4 -> T_2	<.001	
	T_4 -> T_3	.014	
RQ8.3	T_2 -> T_1	.006	.002
	T_3 -> T_1	.048	
	T_4 -> T_1	.335	
	T_3 -> T_2	.083	
	T_4 -> T_2	.006	
	T_4 -> T_3	.014	

Demographic impact

To observe the impact of demographic on the rating of the four techniques, multiple linear regression (MLR) analysis was conducted for RQ1 and RQ8. Checking impact hypotheses ST2-IH1 to ST2-IH4. Table 6.5 shows results of MLR for RQ1, where T_2, T_3 and T_4 have statistically significant ANOVA results. For the rating of usefulness for T_2, age and years of experience both have significant positive correlation. While the rating of usefulness for T_3 is positively correlated with years of experience and negatively correlated with size of team, with the former and latter being statistically significant. The rating of usefulness for T_4 is statistically significant for age and years of experience, and both positively correlated.

TABLE 6.5: Demographic impact multiple linear regression results for rating of usefulness RQ1. (*) *Experience* stands for *Years of professional experience*

Research Question		Correlation		Model Summary		ANOVA
					Adjusted	
		Correlation	Sig.	R Square	R Square	Sig.
RQ1 T_1	Age	.445	.032	.388	.257	.069
	Experience*	.544	.010			
	Size of team	-.349	.078			
RQ1 T_2	Age	.741	.000	.887	.863	<.001
	Experience*	.905	.000			
	Size of team	-.332	.089			
RQ1 T_3	Age	.304	.110	.679	.610	<.001
	Experience*	.603	.004			
	Size of team	-.434	.036			
RQ1 T_4	Age	.782	.000	.909	.890	<.001
	Experience*	.920	.000			
	Size of team	.087	.365			

Table 6.6 presents MLR results for the impact of speed, quality and controllability on the rating of usefulness for each technique. Here, results for T_1, T_2 and T_4 have significant ANOVA results. For T_1, speed and controllability both have significant positive correlations with the rating of usefulness. For T_2, quality and controllability have significant positive correlations with the rating of usefulness. While, for T_4, only quality has a significant positive correlation with the rating of usefulness.

TABLE 6.6: Multiple linear regression results for the impact of the perceived Speed, Quality and Controllability on rating of usefulness RQ1.

Research Question		Correlation		Model Summary		ANOVA
					Adjusted	
		Correlation	Sig.	R Square	R Square	Sig.
RQ1 T_1	Speed	.710	.000	.731	.673	<.001
	Quality	-.134	.298			
	Controllability	.492	.019			
RQ1 T_2	Speed	.186	.230	.468	.354	.028
	Quality	.585	.005			
	Controllability	.643	.002			
RQ1 T_3	Speed	.412	.045	.176	.067	.233
	Quality	-	-			
	Controllability	.343	.082			
RQ1 T_4	Speed	-	-	.920	.909	<.001
	Quality	.958	.000			
	Controllability	.147	.280			

Table 6.7 presents the MLR results for the rating of speed (RQ8.1), where T_3 has a statistically significant ANOVA result. For the rating of speed for T_3 all demographic variables have a

statistically significant impact. Age and years of experience have a positive correlation, while the size of team has a negative correlation. Analysis of T_4 could not be conducted as all participants gave a rating of 5, or *very good*.

Table 6.8 presents the MLR results for the rating of output quality (RQ8.2), where T_1, T_2 and T_4 have statistically significant ANOVA results. No individual correlation is statistically significant for T_1. While for the ratings of T_2, size of team has a statistically significant negative correlation. Ratings of T_4 are positively correlated with age and years of experience. Analysis could not be conducted on T_3 as all participants gave a rating of 5, or *very good*.

Table 6.9 presents the MLR results for the rating of controllability (RQ8.3), where T_1 and T_3 have statistically significant ANOVA results. For T_1 there are no individual statistically significant correlations, while for T_3 age and years of experience both have statistically significant positive correlations.

TABLE 6.7: Demographic impact multiple linear regression results for rating of speed RQ8.1. Highlighted rows indicate a constant result, wherein all participants gave the same answer.

Research Question		Correlation		Model Summary		ANOVA
					Adjusted	
		Correlation	Sig.	R Square	R Square	Sig.
RQ8.1 T_1	Age	.579	.006	.396	.267	.063
	Years of professional experience	.452	.030			
	Size of team	-.060	.407			
RQ8.1 T_2	Age	.342	.082	.394	.264	.065
	Years of professional experience	.100	.347			
	Size of team	.447	.031			
RQ8.1 T_3	Age	.525	.013	.807	.766	<.001
	Years of professional experience	.410	.045			
	Size of team	-.542	.010			
RQ8.1 T_4	Age	-	-	-	-	-
	Years of professional experience	-	-			
	Size of team	-	-			

Table 6.10 shows the MLR results for ST2-RQ3 to ST2-RQ7 where all results have statistically significant ANOVA results. For ST2-RQ3 (preference of volume or quality), age and years of experience have positive correlations, while size of team has a negative correlation. For ST2-RQ4 (acceptable time for generation), and ST2-RQ6 (importance of configurability), no individual correlations are statistically significant. For ST2-RQ5 (preference for stand-alone or integrated solution), years of experience has a negative correlation, while size of team has a positive correlation. For ST2-RQ7 (perceived usability), age has a positive correlation.

Within these results there is a general trend of more experienced participants rating individual techniques and the overall tool higher in many aspects. With more experience, users have had the opportunity to experience a wider range of tools and projects. This makes new tools easier to adopt, and results in the user seeing more contexts in which a tool may be beneficial.

TABLE 6.8: Demographic impact multiple linear regression results for rating of quality RQ8.2. Highlighted rows indicate a constant result, wherein all participants gave the same answer.

Research Question		Correlation		Model Summary		ANOVA
		Correlation	Sig.	R Square	Adjusted R Square	Sig.
RQ8.2 T_1	Age	.192	.222	.773	.725	<.001
	Years of professional experience	-.204	.208			
	Size of team	.240	.169			
RQ8.2 T_2	Age	.028	.457	.807	.766	<.001
	Years of professional experience	.265	.144			
	Size of team	-.868	.000			
RQ8.2 T_3	Age	-	-	-	-	-
	Years of professional experience	-	-			
	Size of team	-	-			
RQ8.2 T_4	Age	.778	.000	.940	.927	<.001
	Years of professional experience	.951	.000			
	Size of team	-.072	.388			

TABLE 6.9: Demographic impact multiple linear regression results for rating of control RQ8.3

Research Question		Correlation		Model Summary		ANOVA
		Correlation	Sig.	R Square	Adjusted R Square	Sig.
RQ8.3 T_1	Age	-.028	.457	.422	.299	.047
	Years of professional experience	-.265	.144			
	Size of team	-.108	.334			
RQ8.3 T_2	Age	.395	.052	.386	.255	.070
	Years of professional experience	.345	.081			
	Size of team	-.387	.056			
RQ8.3 T_3	Age	.766	.000	.625	.545	.003
	Years of professional experience	.598	.004			
	Size of team	.158	.265			
RQ8.3 T_4	Age	.481	.022	.331	.187	.121
	Years of professional experience	.300	.113			
	Size of team	.270	.140			

Furthermore, more experienced participants prioritise lower volume, but higher quality outputs for this tool. The interview results suggest that more experienced users are less concerned with producing larger volumes of assets as they expect to use such a tool to supplement their manual efforts. While those in smaller teams prefer larger volume but lower quality of output. As was found in the interviews, users that work in smaller teams prefer to explore and try different ideas, then refine and add quality when they find something that works.

Those who work in larger teams prefer this tool as an integrated solution. This finding is in line with previous research, as teams have been found to prioritise compatibility with their existing

TABLE 6.10: Demographic impact multiple linear regression results for RQ3-RQ7

Research Question		Correlation		Model Summary		ANOVA
		Correlation	Sig.	R Square	Adjusted R Square	Sig.
RQ3	Age	.570	.007	.635	.586	<.001
	Years of professional experience	.697	<.001			
	Size of team	-.447	.031			
RQ4	Age	-.054	.416	.952	.941	<.001
	Years of professional experience	.384	.058			
	Size of team	-.337	.086			
RQ5	Age	-.051	.420	.240	.192	.039
	Years of professional experience	-.489	.020			
	Size of team	.400	.050			
RQ6	Age	.221	.189	.856	.825	<.001
	Years of professional experience	-.193	.221			
	Size of team	.217	.194			
RQ7	Age	.485	.021	.235	.187	.041
	Years of professional experience	.263	.146			
	Size of team	-.308	.106			

tools [215]. In addition, less experienced participants prefer this tool as an integrated solution. This is likely due to the resulting ease of incorporating such a tool into their pipeline, as opposed to altering their pipeline to accommodate a separate piece of software.

Experienced participants also rate usability higher. This may be due to having more opportunities to experience similar interfaces, thus becoming more accustomed to this style of interaction. Furthermore, more experienced users can be expected to have more confidence with new tools, as opposed to users with less experience, being less comfortable with integrating new tools into their pipeline.

The ratings of usefulness are correlated differently with speed, quality and controllability depending on the technique in use. Speed and controllability correlate with the rating of usefulness for parameter guided design (T_1). This suggests that users do not consider quality a concern when it comes to parameter guided creation as the quality is dependent on the amount of input and initiative they apply. While faster generation speed and more controllability would allow them to work more efficiently within the tool to create designs.

Quality and controllability correlate with the rating of usefulness for image reconstruction (T_2). Here, it is likely that speed is less of a concern, assuming the result meets their expectation for quality, and the tool provides the means to configure their inputs. Quality correlates with the usefulness rating of randomisation (T_4). Here, it is likely that controllability is not a concern as users do not expect to have much control over this technique. A correlation could not be found between speed and usefulness due to the unanimous rating of *"Very good"* for generation speed on this technique. The positive correlation between quality and usefulness rating suggests that better quality outputs are however preferred.

Correlation between pipeline choices and technique usefulness

It is important to this research that the connection between technique usefulness and the choice of location in the design/development pipeline is examined in order to expand on the existing GaGeTx framework. Table 6.11 presents results of a multiple linear regression test between ratings of usefulness for each technique, and the preferences for the locations in the pipeline participants would find the tool useful.

Here, parameter guided (T_1) usefulness ratings are shown to be negatively correlated with the choice of *generating inspiration*, while positively correlated with *creating placeholder assets*, *creating assets from existing designs* and *creating assets from scratch*. Reconstructed (T_2) usefulness ratings are shown to be negatively correlated with *generating inspiration* and *exploring ideas*, while positively correlated with *creating placeholder assets*, *creating assets from existing designs* and *creating assets from scratch*. Interpolated (T_3) usefulness ratings are shown to be negatively correlated with *generating inspiration* and *exploring ideas*, while positively correlated with *creating assets from scratch*. Random (T_4) usefulness ratings are shown to be negatively correlated with *exploring ideas*, *creating variations of existing assets* and *player made content*, while positively correlated with *creating assets from scratch*.

Comparison to previous mock-up

The use of a graph based UI was a choice based on the results of the mock-up evaluation in chapter 3. It is therefore relevant to compare the final graph-based prototype tool with the initial, non-functional mock-up in terms of user perception. As a graph view interface was chosen, the prototype will be compared with the graph view mockup M_2.

A Mann-Whitney U test was conducted on the set of questions asked in both studies, with the mock-up data assigned to group 1 and the prototype data assigned to group 2. Results are shown in table 6.12. The difference in perceived usefulness was found to be statistically significant ($P < 0.05$) with higher mean ranks in the prototype group. This suggests that users perceived the functional tool to be more useful as they were able to observe the results and meaningfully interact with the tool. There is a statistically significant difference ($P < 0.05$) for finding value in using the tool for creating assets from scratch and player made content. The mean ranks show that more participants in the prototype testing group found value in creating assets from scratch and less found value in using it for player made content, while the opposite is true for the mock-up. This suggests that when experiencing the functional tool, users had a better understanding of the quality of output or the complexity of controls, and thus found the tool less valuable for player made content and more valuable for creating assets from scratch.

Output Examples

Participants anonymously uploaded their result data to a secure upload folder by following the instructions in the form, using a generated number as identification both in the the web form and in the upload location. This section will present example results, and inputs where applicable, highlighting key observations regarding each generative technique.

TABLE 6.11: Multiple linear regression results for the impact of technique usefulness on pipeline choices

Research Question		Correlation		Model Summary		ANOVA
		Correlation	Sig.	R Square	Adjusted R Square	Sig.
RQ9 T_1	Generating inspiration	-.521	.013	.993	.991	<.001
	Exploring ideas	-.330	.091			
	Creating placeholders	.929	.000			
	Creating variations of assets	.330	.091			
	Creating assets from designs	.521	.013			
	Creating assets from scratch	.737	.000			
	Player (or user) made content	.330	.091			
RQ9 T_2	Generating inspiration	-.495	.018	.990	.997	<.001
	Exploring ideas	-.589	.005			
	Creating placeholders	.405	.048			
	Creating variations of assets	.037	.442			
	Creating assets from designs	.495	.018			
	Creating assets from scratch	.906	.000			
	Player (or user) made content	.037	.442			
RQ9 T_3	Generating inspiration	-.698	.001	.981	.976	<.001
	Exploring ideas	-.806	.000			
	Creating placeholders	-.077	.381			
	Creating variations of assets	.077	.381			
	Creating assets from designs	.102	.343			
	Creating assets from scratch	.698	.001			
	Player (or user) made content	.077	.381			
RQ9 T_4	Generating inspiration	-.017	.473	.911	.883	<.001
	Exploring ideas	-.535	.011			
	Creating placeholders	.218	.192			
	Creating variations of assets	-.513	.015			
	Creating assets from designs	.095	.354			
	Creating assets from scratch	.488	.020			
	Player (or user) made content	-.513	.015			

Table 6.13 shows an example of two designs via parameter guided creation (T_1) from a single participant. Here, many of the features and parameters have been utilised to create two distinct sword designs, including the use of the material editor, curves, edge properties and rounding. The results corroborate with the higher ratings of quality within the questionnaires compared to the other techniques.

TABLE 6.12: Mann-Whitney test results for independent samples comparison between the graph based mock-up and final prototype, presenting the Mann-Whitney U, Wilcoxon W, mean ranks for group 1, mean ranks for group 2 and significance scores.

Question	Mann-Whit. U	Wilcoxon W	Mean rank group 1	Mean rank group 2	Sig.
SUS	117.0	288.0	21	16	.152
"I would find this software tool that generates graphical assets useful in the projects I work on."	58.5	229.5	12.75	24.25	<.001
Find value in: Generating inspiration	135	306	20.00	17.00	.225
Find value in: Exploring ideas	162	333	18.5	18.5	1.000
Find value in: Creating placeholder assets	135	306	17.00	20.00	.225
Find value in: Creating variations of assets.	135	306	20.00	17.00	.225
Find value in: Creating assets from designs.	135	306	17.00	20.00	.225
Find value in: Creating assets from scratch	108	279	15.50	21.50	.036
Find value in: Player (or user) made content.	108	279	21.50	15.50	.036
Total selected pipeline options	157.5	328.5	18.25	18.75	.883
"How important to you, is the ability to modify such a tool?"	121.5	292.5	16.25	20.75	.203

TABLE 6.13: Exemplar user creation using T_1

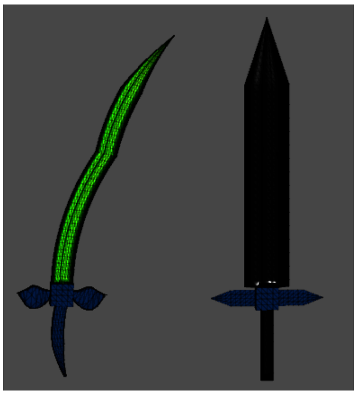
Guided (T_1)		
Original	Processed	Result
Not applicable	Not applicable	

Table 6.14 shows an example reconstruction result (T_2), with the associated input image and pre-processed silhouette. This example demonstrates a common issue regarding the pre-processing stage of the system. In this case, the user has provided an input image that contains multiple different depictions of swords, with various obfuscating effects. This resulted in a partial silhouette,

which then had a large impact on the appearance of the end result.

TABLE 6.14: Exemplar user creation using T_2, including the input data [103], pre-processed input data and the final result outputs.

Reconstruction (T_2)

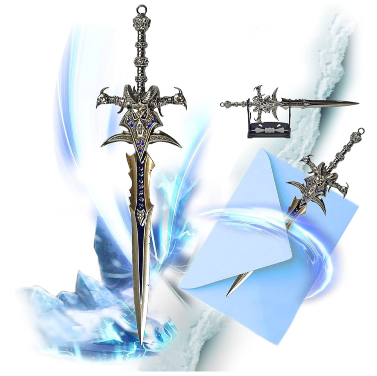
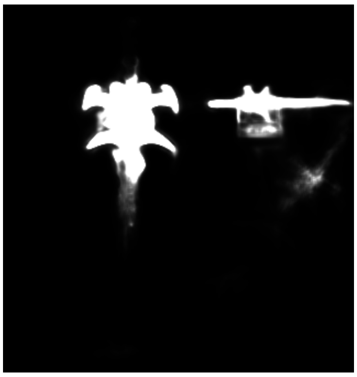
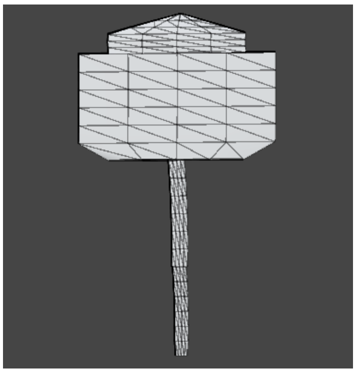
Original	Processed	Result
		

Table 6.15 shows an example interpolation result (T_3), with the two input images and their processed silhouettes, marked *A* and *B*. As can be seen, the user had chosen images of bows, rather than swords. The output appears to not match any of the characteristics of the input images, aside from having a long, thin profile. This is a result of the focus on sword shapes in the training data. This focused approach limits the model's ability to generalise to other types of shape and more unconventional sword designs. User confusion could be avoided by including an object detection layer during pre-processing, to filter out inputs that do not contain swords. If shape generalisation is to be further explored, this may be achieved with a wider variety of shapes types in the training data, and a larger number of trainable parameters.

TABLE 6.15: Exemplar user creation using T_3, including the input data, pre-processed input data and the final result outputs. The two input images, that are interpolated between, are marked *A*[506] and *B*[507].

Interpolation (T_3)

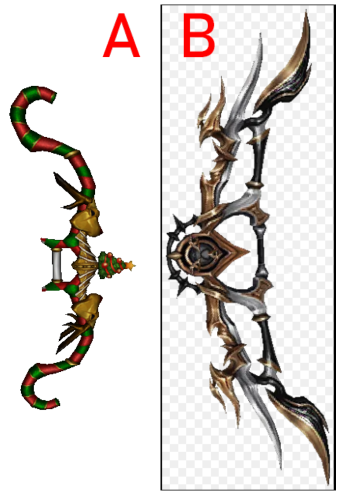
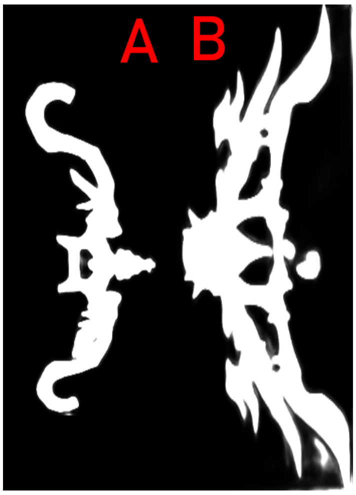
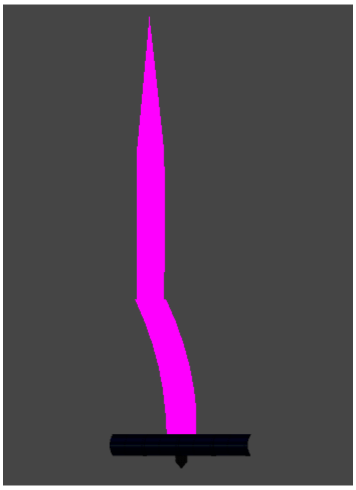
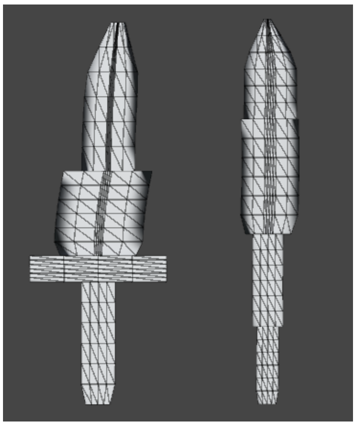
Original	Processed	Result
		

Table 6.16 shows an example of two designs generated by the same participant using the random technique (T_4). It can be observed that results are more simple than that of parameter guided creation, though still somewhat resemble swords. This also highlights that many results of the random generation are unremarkable, due to the random vector generation pulling from a normal distribution.

TABLE 6.16: Exemplar user creation using T_4

Randomisation (T_4)		
Original	Processed	Result
Not applicable	Not applicable	

Interview Results

Of the 18 participants, 3 agreed to a follow up interview. Participants generally confirmed the questionnaire results in the interviews, stating a clear preference for parameter guided creation (T_1) and randomisation (T_4). Interviewees stated that they did not find reconstruction (T_2) and interpolation (T_3) to be as useful due to output not being high enough quality to justify the effort required to find or create the inputs. In many cases, the output designs did not match or have a close enough likeness to the input images for this technique to be useful. On the contrary, interviewees stated that they preferred parameter guided creation for the amount of control it provided, and randomisation for the speed at which they could generate new ideas.

Participants confirmed the analysis of demographic impact shown in section 6.1.2. One interviewee mentioned that they found the tool easy to pick up as they were familiar with graph interfaces from using Shader Graph and Substance Designer in the past. Mentioning that although the results did not meet their standards for final assets, they have experienced contexts in which such a tool would be beneficial, citing "generating inspiration" and "exploring ideas" as applicable use cases. They also suggested that these contexts are not so common as they usually design in 2D before creating 3D models, stating that a similar tool in 2D would be more useful.

One participant stated that they have no experience with 3D modelling tools, and that the parameter guided design gave them a middle ground where they could design 3D shapes without technical know-how. They expressed that this could be improved by having visual controls such as drag-able handles, instead of direct number inputs. The same participant also stated that the low time investment and effort involved in the randomisation technique could help them with *"creative block"*.

When asked to consider how their preference would change if reconstruction and interpolation produced results that closely matched the inputs they provided, participants stated that they would consider these techniques more useful. But between the two, they considered interpolation more versatile as it can be used to achieve the same result as reconstruction.

Some further suggestions were made regarding interaction with the tool itself. One interviewee stated that the tool should have pre-set data for parameter guided creation, so that the user does not have to start from scratch. This way they can alter existing conventional sword shapes, which would speed up the design process.

A participant also expressed the importance of enjoyment in the process of designing, and that the more they enjoy using a system, the more motivated they are to work with it to produce designs, and the more useful they find it. This participant suggested the addition of a feature that would allow the user to preview their designs in a final context, such as in the hands of the character it will belong to. This suggestion is rather specific to the scenario that was provided in the study, however, the underlying point remains that such tools can be made more useful by incorporating features that stoke the user's creativity.

6.2 Concluding Discussion

In this chapter, a proof of concept prototype tool has been developed and tested with game designers and developers. This tool implemented the four techniques of parameter guided generation, image reconstruction, interpolation and randomisation, covering the breadth of generative techniques within GaGeTx. To achieve this a VAE was trained on a novel dataset of 13,728 sword profile silhouettes. The final model was integrated within the Unity game engine [475], then tested with 18 game designers and developers. The focus of testing was to examine user preferences between the various techniques and to obtain insights into how users best utilise such a tool. During testing, data collection was achieved via repeated measure questionnaires for each technique, overall preference questions for the tool, and semi-structured interviews on an opt-in basis.

Results of the testing suggests that users most prefer the parametric guided technique, followed by randomisation, then interpolation and image reconstruction. Though the image reconstruction and interpolation technique implementations were shown to produce poor results, participants, as suggested by ratings for perceived speed and controllability and confirmed within interviews, had lower preference for these techniques for reasons other than output quality. Instead, the rigidity of their control being limited to the input image they provided, and the time it takes to source an image as input both reduced the appeal of these techniques. It is expected that the ratings of all techniques would improve if the tool itself produced higher quality outputs. Future work should re-examine this with improved, generative models to determine the impact output quality has on these perceptions. As suggested by the interview results, users expect to do some level of manual design or refinement when creating assets to include in their final video game products. This naturally shifts interest to the simpler and faster techniques that help users to come up with ideas that they can then refine, as opposed to techniques that produce a close to final output. When comparing the results of questions asked in the initial mock-up study with results from the prototype testing, the latter was rated higher for usefulness. Which is to be expected considering the prototype is functional, while the mock-up consisted of a UI in isolation.

Table 6.17 presents the correlations between the rating of usefulness for each technique and the ratings of quality, speed and control. As shown, each technique correlates differently with the three ratings. The usefulness rating of the parametric guided technique is positively correlated with the ratings of speed and control, suggesting that speed and control assessments are more relevant than quality assessments to the usefulness of the method. For reconstruction, quality and control are more relevant than speed and for randomisation, quality and speed are more relevant that control. Interestingly, usefulness of the guided technique does not significantly correlate with quality, while the usefulness of randomisation does. For the guided technique, this is to be expected, as the output quality is largely dependent on the effort of the user. The more detail and refinement added through parameters the more elaborate the result. For randomisation however, better quality can be beneficial as it reduces the effort required to adapt the asset, and provides greater inspiration. The perceived usefulness of interpolation has a positive correlation with speed, and no significant correlation with quality and controllability. This may be because the output quality is dependent on the quality of the two inputs. Similarly, controllability may be less relevant, as it comes in the form of selecting the inputs, leaving little need for further controls. Figure 6.2 presents these relationships with green lines for relevance and a dashed red line for partial irrelevance.

The above is a valuable insight in determining which metrics to focus on when selecting an approach, given a chosen technique. While it was not possible to observe the importance of each individual metric in relevance to each technique, the importance of the broader categories of quality, performance and controllability are captured through user perceived quality, speed and control.

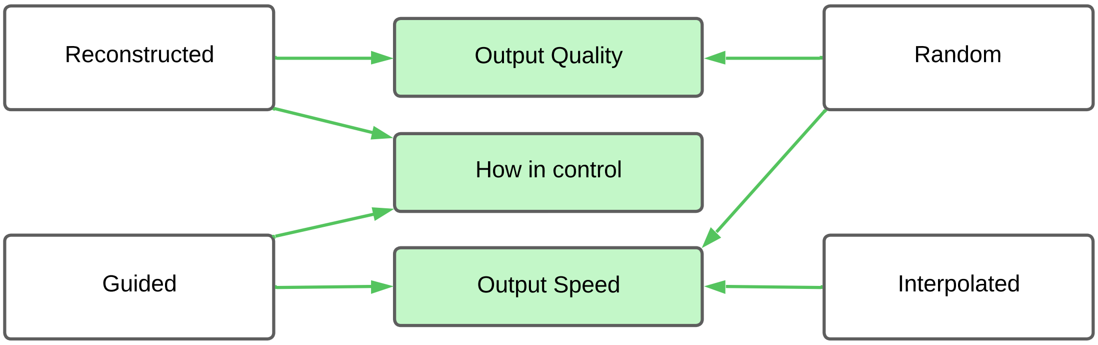


FIGURE 6.2: Relationship between the perceived usefulness of each tested technique and the three rating types: Quality, Control and Speed.

Table 6.18 presents the correlations between the usefulness ratings of each technique and the

Usefulness of:	Quality	Speed	Control
Guided	No relation	Correlation	Correlation
Reconstructed	Correlation	No relation	Correlation
Interpolated	No relation	Correlation	No relation
Random	Correlation	Correlation	No relation

TABLE 6.17: Correlations between the perceived usefulness of each tested technique and the three rating types: Quality, Speed and Control. Cells are labeled "Positive" or "Negative" for statistically significant positive or negative correlations ($P < 0.05$). Cells are labeled "Partial positive" or "Partial negative" for non-statistically significant positive or negative correlations ($P > 0.05$). Cells are labeled "No relation" where significance exceeds 0.1 ($P > 0.1$).

choice of usage in the pipeline. These results are coherent with what would be expected, and are confirmed in part within the interviews. In essence, this maps what the participants want from the tool to the techniques they found more useful. *Generating inspiration* and *exploring ideas* are both use cases in which quality is not necessary. However, as stated in the interviews, the 3D stage of creation is usually past the ideation phase so it is unlikely that this would find real use, thus connecting this with low usefulness ratings. *Creating placeholder assets* requires that to some degree, you know what kind of asset you will need, and *creating assets from designs* implies that the intended artefact is already designed. Higher ratings of the parametric guided and image reconstruction techniques, as expected, did correlate with interest in creating placeholder assets and assets from designs. This is because it is possible to specify the general shape of what you want, using these techniques. While users are not interested in blending or creating random designs in this context.

As expected *creating variations of assets* and *player made content* were two options that were selected with the lowest frequency. None of the four techniques facilitated the ability to create variations of designs and player made content is an uncommon use case and requires a bespoke solution depending on the product. The choice of *creating assets from scratch* positively correlates with the usefulness of each of the techniques due to it being a generic use case, as well as it being possible to create assets from scratch via all techniques.

The results of this study provide insights into how game designers and developers view generative methods for graphical assets, which techniques they prefer and which aspects are of most importance when assessing the usefulness of each technique. Findings both confirm, answering

	Guided	Reconstructed	Interpolated	Random
Generating inspiration	Negative correlation	Negative correlation	Negative correlation	No relation
Exploring ideas	Partial negative correlation	Negative correlation	Negative correlation	Negative correlation
Creating placeholders	Positive correlation	Positive correlation	No relation	Partial positive correlation
Creating variations of assets	Partial positive correlation	No relation	No relation	Negative correlation
Creating assets from designs	Positive correlation	Positive correlation	No relation	No relation
Creating assets from scratch	Positive correlation	Positive correlation	Positive correlation	Positive correlation
Player (or user) made content	Partial positive correlation	No relation	No relation	Negative correlation

TABLE 6.18: Correlations between pipeline choices and technique usefulness. Cells are labeled "Positive" or "Negative" for statistically significant positive or negative correlations ($P < 0.05$). Cells are labeled "Partial positive" or "Partial negative" for non-statistically significant positive or negative correlations ($P > 0.05$). Cells are labeled "No relation" where significance exceeds 0.1 ($P > 0.1$).

some key questions regarding the GaGeTx and metric frameworks, providing the means to expand and give clarity to some of the processes involved. Consequently, this relationship can guide the selection of appropriate metrics for evaluating a method, depending on the technique.

6.3 Expanding GaGeTx

Following the findings in this chapter, the needs and preferences of users have been integrated into the GaGeTx framework. Figure 6.3 shows the expanded framework, including the user goal, which encompasses intent and method priority. User intent represents the general task that the graphical asset generator must fulfil and reflects the point in the design and production pipeline that it will target. This consists of the four applications as discussed in chapter 5. The method priority represents the preference for the generalised user-centric metric types: Speed, Quality and Controllability. While these metric groupings have been shown to have affinities with different technique types in this chapter's findings, this priority represents the individual user's preference, which may deviate from this trend.

User intent is considered at the point of technique selection and is used to determine the appropriate range of technique interaction types and processes that may be applied. Figure 6.4 presents the application of intent to technique choice. Here, user intent is split into four sections, spanning the two relevant stages of design and development: design/prototype and production; and aligned with the four user idea fidelity groupings. These groupings are ordered by the level of pre-design needed, and by definition, the minimum complexity or quality of the required inputs. At the lowest level, wherein there is no pre-design required, random techniques can be used to generate inspiration. In cases where the user has "some idea" about what they want out of the generator, such as when "exploring ideas" or "creating placeholder assets", guided techniques may be used for working with such limited designs or constraints. Here random seed interaction may be used as a supplement to other interaction types in order to help with developing and exploring ideas. As the level of pre-design becomes more specific and complex, the creative load of the technique becomes more specific. When the user is in the production phase of a project, they may have finished or semi-finished assets they wish to create variations of, or arrange in a larger environment i.e. the user has "partial designs" to work with. This level encompasses arranged, interpolated and style transferred techniques. In the final level, users have "full designs" that must be formed into a final or close to final asset. A reconstructed technique should therefore be used, as the intent is to match the input design as faithfully as possible, without randomness, variation or creative expression from the generator.

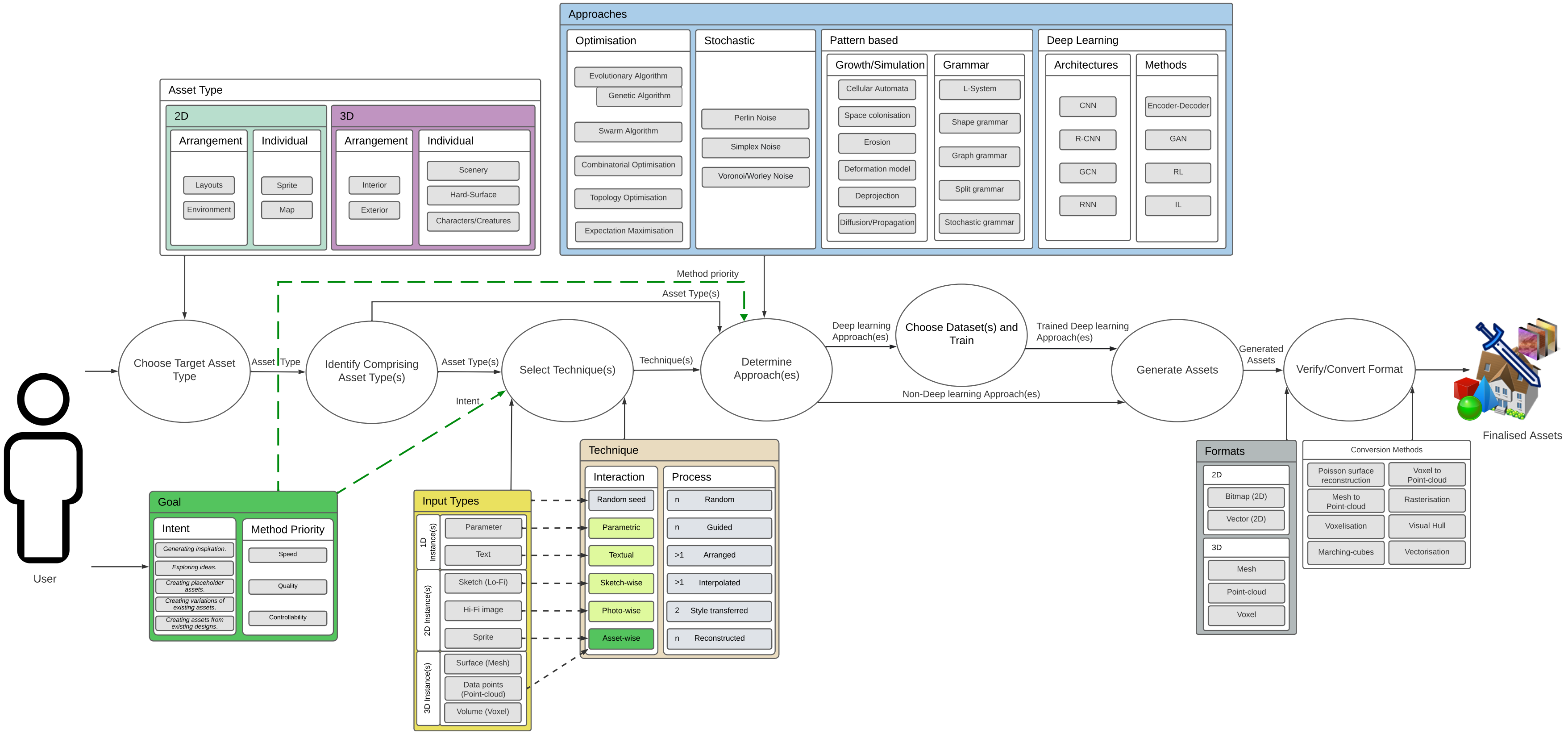


FIGURE 6.3: The expanded GaGeTx framework, incorporating the user goal, including intent and method priority (shown in green).

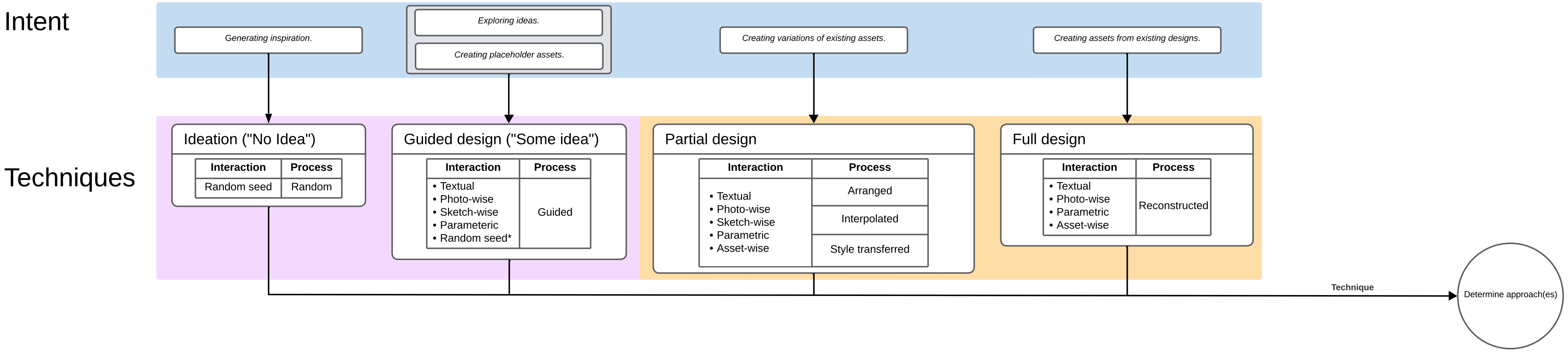


FIGURE 6.4: The mapping of user intent to the choice of technique interaction type and process. (*) random seed interaction cannot be the sole interaction type under a guided process.

As discussed and shown in chapter 2 there are many examples of graphical asset generators in the literature. As current techniques and approaches are explored further, and new techniques and approaches emerge, the number of proposed implementations will only grow. This makes it challenging to systematically select or devise the most appropriate implementation for a given use case, considering user needs or preferences. With the categorisation of metrics in chapter 2, and the discovered affinity between technique process and metric type from chapter 4, it is possible to select the best implementation for the user's needs. Figure 6.5 presents the selection process for the approach and its implementation, based on the chosen technique and user method priority.

Initially, the available approaches are filtered by the chosen asset type and technique. A set of metrics are then used to rank the approach implementations based on user needs, from which the highest scoring approach implementation is selected. Metrics that have been used in the selected approaches from the previous step, form the pool of available evaluation metrics. Of the three main types of metric introduced in chapter 2, two are dependent on user need. These are *artefact quality* and *operation* metrics. In line with the depiction in chapter 4, these are broken down into three groups of metric: quality, control and speed. Quality refers to all forms of quality metric, shown in figure 3.4, while control refers to the *controllability* metrics shown in figure 3.5 and speed refers to the speed *operation* metric shown in figure 3.5. The selected technique process first determines which two of these metric groups are appropriate to assess based on the established affinities from chapter 4. For each metric group, a metric from the approach relevant pool is selected. Approach implementations are then assessed using both selected metrics and ranked. To rank approach implementations using the two selected metrics, a multi-criteria decision-making (MCDM) approach can be used [49].

Artefact validation metrics can only be used to compare methods that use the same datasets. Therefore it is not possible to rank most approach implementations in this way. They are however useful for evaluating every method individually to determine if they produce valid outputs. If a method fails to produce valid outputs, then it is not usable.

6.3.1 Chapter summary

In this chapter a user study was conducted with game designers and developers to observe the perceived usefulness, quality, speed and controllability of each technique in the proof-of-concept prototype from chapter 5. Results of this study found that quality, speed and controllability ratings impact the perception of usefulness differently depending on the technique used. This informed the expansion of the GaGeTx framework by incorporating user needs in the form of *intent* and *priority*, providing more clarity in how decisions are made in the framework process.

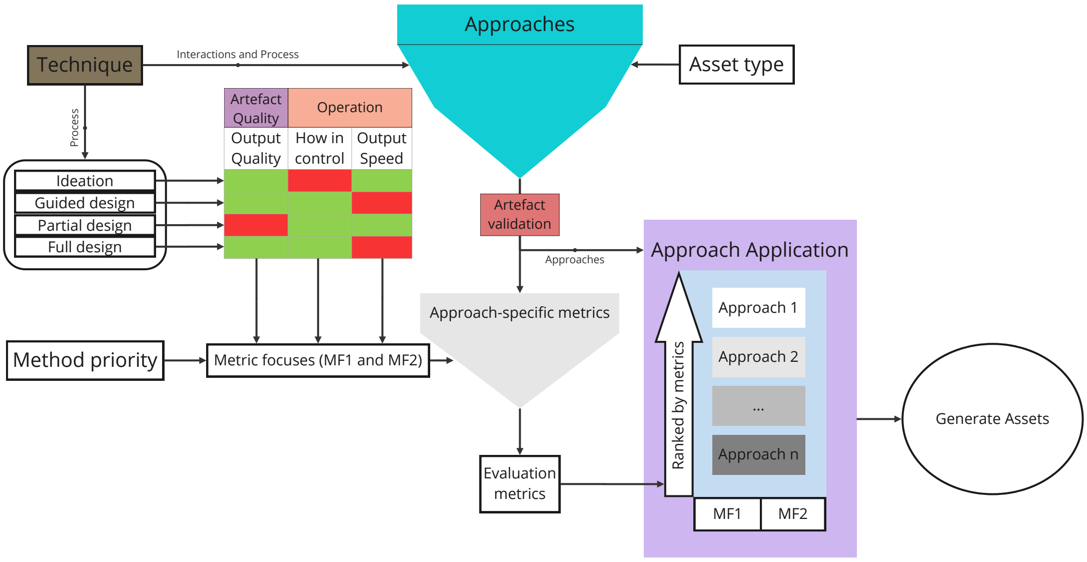


FIGURE 6.5: The process for selecting an approach and implementation. Incorporating metric focus as a result of technique-wise affinity and user method priority. MF1 and MF2 represent metric focus 1 and 2 respectively.

Chapter 7: Concluding discussion

In the previous chapter, a user study was conducted with game designers and developers to observe the perceived usefulness, quality, speed and controllability for each technique implemented by the proof-of-concept prototype tool. Results of this study found that quality, speed and controllability ratings impact the perception of usefulness differently depending on the technique used. This informed the expansion of the GaGeTx framework by incorporating user needs in the form of *intent* and *priority*, providing more clarity in how decisions are made in the framework process. This chapter will conclude the thesis, starting with a summary of chapters, followed by a comprehensive presentation of the highlights and contributions, leading into a discussion of the limitations, and avenues for future research.

7.1 Summary of thesis

Starting with a systematic literature review in chapter 2, where 280 accepted journal and conference papers were examined out of an initial pool of tbd following the PRISMA protocol, thus achieving objective 1. Inductive thematic analysis led to the formulation of the GaGeTx framework (section 2.3), which categorises the main aspects of graphical asset generators and presents a logical process for selecting these aspects, thereby achieving objective 2. Following this, the GaGeTx framework has been expanded to facilitate the nascent interest in multimodal techniques in section 2.4.1.

As identified in chapter 2, the core framework covers the techniques and approaches to asset generation, but not the methods by which to evaluate and compare the many applications presented. Chapter 3 introduces framework the for evaluation metrics that can be applied to graphical asset generators, resulting from a further ITA which targets the methods and metrics used to evaluate generators in the literature.

These findings lead into chapter 4, in which the needs and preferences of game designers and developers were examined through a mixed methods user study in order to achieve objective 3. Through a combination of the statistical analysis of questionnaire data and thematic analysis of interview data, it was found that game designers and developers prefer the use of generative tools in the early stages of design, such as creating placeholders or generating inspiration. Furthermore, a preference for integrated solutions is observed, along with the need for configurability.

Building on these findings to achieve objective 4, a proof-of-concept prototype was designed and developed following the RAD methodology in chapter 5. The prototype tool consisted of a VAE for generating 3D sword assets, which was integrated into the Unity engine using the Barracuda

API [474]. This GAG tool was developed to implement reconstruction, interpolation, guided and random techniques, representing each of the *idea fidelity* groupings introduced in chapter 2.

The prototype developed in chapter 5 was then tested using a mixed methods approach in chapter 6, with a focus on quality, speed and controllability evaluation, derived from chapter 3. The perceived usefulness of each technique is found to correlate with different evaluation types. Suggesting that users associate the usefulness of each technique with specific evaluation criteria. This achieved objective 5. Following this, GaGeTx was refined to include findings from chapters 4 and 6; incorporating the affect of user preference and needs. This resulted in the addition of the *user goals* aspect to the framework, encompassing *intent* and *method priority*. *Intent* represents the chosen application within the design and development pipeline, while *method priority* presents the user's preferred evaluable characteristic. A mapping between *intent* and the technique was developed, allowing for the selection of appropriate techniques based on the *intent*. Furthermore, the findings in the user studies informed a procedure for selecting the best approach given the user needs and priorities.

7.1.1 Highlights and contributions

Throughout this thesis the following contributions have been made via the fulfillment of the 5 objectives laid out in chapter 1. The following subsections will present these contributions in order of prominence.

Contribution	Objective
GAGeTx framework	Objective 2
A systematic review of state-of-the-art GAG literature.	Objective 1
User preference and applications for GAGs.	Objective 3 and 5
Proof-of-concept prototype generative tool.	Objective 4
A method for formulating training datasets for GAGs, and a novel sword shape dataset.	Objective 4

TABLE 7.1: Summary of research contributions.

GAGeTx framework

The GAGeTx framework is the primary contribution of this research, addressing objective 2 by conceptualising the key aspects of GAGs and providing guidance on formulating and selecting GAGs through user derived requirements. The framework serves multiple key roles: it elucidates the varied and complex range of state-of-the art GAG methods, consolidates the concepts, approaches and techniques fragmented across the many research domains that benefit from GAGs, and provides systematic guidance for selecting and developing the most appropriate GAG tools in accordance to user intent and priority. By doing so, GAGeTx offers value to both researchers and industry practitioners, enabling researchers to explore new methods with a cohesive understanding of the field, and allowing game designers and developers to choose, build and apply GAG tools that best meet their project needs.

As GAG methods continue to advance, it is evident that they can provide significant value by streamlining and enhancing game design and production, particularly in the areas of ideation and creative inspiration. With an increasing number of practitioners adopting generative tools for this purpose [471], GAGeTx forms the foundation for understanding and applying GAGs as practical tools, with user needs and requirements at its centre and allowing new GAG methods to be situated within the wider context. As a step-by-step guide, GAGeTx addresses practical questions users can answer, such as: "What type of artefact is needed?", "For what purpose are these assets being generated?", and "Should I prioritize quality, speed, or controllability?". It then connects these considerations to the functional strengths of state-of-the-art methods, providing logical and empirically-based guidance on selecting the appropriate technique, required inputs, and implementation approach. By integrating empirical data from practitioners and users, GAGeTx bridges the gap between GAG methods and their practical application in game design and production pipelines. It also facilitates the prioritisation of user-perceptible metrics, enabling the selection of the most suitable implementation approach for the task. Additionally, the framework highlights the value of non-deep learning methods used for PCG, which can be easily overlooked amidst the growing focus on deep-learning research.

GAGeTx represents the culmination of this research, integrating findings from objectives 1 to 5. It incorporates the GAG categories established by the outcome of objective 1, and has been expanded, refined, and validated through the outcomes of objectives 3 to 5.

The GAGeTx procedure begins with determining the target asset type, followed by selecting a technique based on user needs. The chosen technique is then implemented using a generative approach, which is selected via evaluation based ranking, and format conversion is performed as needed to meet the output requirements. The GAGeTx framework subsequently lays out a logical process for choosing and selecting each aspect of a GAG. GAGeTx has been further refined in two instances. The first being an adaptation resulting from the nascent advancements in multimodal generative techniques, resulting in a more flexible definition of GAG techniques, involving the distinction between technique interaction and technique process. The second augments the decision process with considerations for user needs and preferences, based on insights obtained from user study 1 and user study 2.

Furthermore, effective evaluation is important for both improving and comparing GAG implementations. A comprehensive analysis of the metrics used to evaluate graphical asset generation methods has been conducted, resulting in an additional metric selection framework. This identifies three main types of metric evaluation: Artefact validation, Artefact quality and Operational. Artefact validation metrics are used for the purpose of verifying whether or not generated outputs match the intended asset type. These consist of both objective and perceptual similarity metrics, which require corresponding ground-truth data to calculate. Artefact quality metrics either assess specific characteristics of generated outputs, or utilise human-centred feedback or perception. While operation metrics assess a method's performance, such as speed and memory usage, or controllability from a user perspective. This paves the way for the standardisation of evaluation metrics, which will allow methods to be accurately compared and assessed, aiding in the advancement of GAG solutions across all domains.

A systematic review of state-of-the-art GAG literature

The systematic literature review of state-of-the-art GAG methods serves as a foundational element of this research. Not only does it establish the basis of the GAGeTx framework but it also identifies the shared characteristics or aspects of current GAG methods. Furthermore, highlighting the prevalence of different choices and approaches for different tasks, offering valuable insights into the popularity and applicability of various techniques and approaches.

While existing research broadly covers PCG for games [164], procedural virtual worlds [429] and deep-learning for content generation [294], none so far have endeavoured to consolidate generative methods applied across the full gamut of graphical applications. This left an incomplete view of current GAG methods, and thus a large barrier to entry for researchers and practitioners new to automated asset production. Through the comprehensive identification and categorisation of state-of-the-art GAG methods the systematic literature review addresses this shortfall.

The systematic review of literature resulted in a collection of 280 papers spanning across major databases between 2016 and 2024, a full list of which can be found in appendix A1. GAG methods have been applied in various fields of research and practice, from architecture to medical imaging. All of these methods are aimed at the creation of graphical assets, which, due to their graphical nature, may be applied cross-domain. The literature on graphical asset generation has so far been fragmented across these different disciplines. This review centralises the understanding of the various aspects of graphical asset generators, providing a basis for future research, encouraging the cross-pollination of ideas between the various research domains that benefit from graphical assets, and providing a point of reference for researchers and practitioners considering GAG development and usage. Furthermore, this allows researchers to identify gaps and potential applications of GAG methods within alternate domains, while fostering the development of novel methods, highlighting more overlooked approaches and advancing promising research directions. By identifying shared aspects of GAG methods, researchers and practitioners can easily find and compare alternative methods based on those that share the same target asset type, technique, inputs, approaches, or formats. Consequently informing the integration of appropriate methods within design and production workflows.

User preference and applications for GAGs

In pursuit of objectives 3 and 5, two user studies have been conducted to examine the UX preferences and needs of game designers and developers regarding the use of GAGs in design and production pipelines.

The initial iteration of the GAGeTx framework, being derived from research on specific GAG methods, lacked the nuance necessary to address the user aspect of such tools. While existing studies explore general user sentiments toward tools used in game production [215], and evaluate the usage of generative tools for interactive game content such as levels [487], no prior research has focused on the UX implications for GAG tools. This is integral to the appropriate application of GAGs in the design and production of games. The first study addresses this by examining game designer and developer preferences surrounding the use of GAGs in game design and development applications. The output of this study is a set of guidelines, enabling game designers and developers to incorporate GAG tools effectively within their pipelines catering toward design and production intent, as well as implementation and integration. In addition, this provides a clear direction for further research and development by identifying a key focus on early pipeline applications, such as generating inspiration and exploring ideas, and identifying the importance of augmenting and streamlining the asset design and production pipeline as opposed to completely replacing it via automation.

For user intent, the following insights are presented:

1. *Facilitate tool configurability.*

2. *Cater to early pipeline usage.*

3. *Augment a stage of design and development.*

For GAG implementation or integration the following insights are presented:

1. *Match the design language of the environment.*

2. *Use common and expected data formats.*

3. *Develop a suitable interface for the underlying interactions.*

4. *Integrate with application programming interfaces (APIs).*

These insights and guidelines give GAG researchers and developers a core foundation for integrating GAGs as tools that benefit design and production workflows with minimal friction and user frustration. This also has great value in the adaptation of many existing methods, initially formulated under a research context, into tools that are useful in practical settings.

In pursuit of objective 5, the second user study was conducted. This study established a link between the conceptual GAG techniques and the pipeline applications explored in objective 3, as well as connecting key perceptible metrics to each technique group. In doing so, this also validated the prototype developed in objective 4. The findings provide insights into the measurement of usefulness for each technique type, highlighting which metrics are relevant and which are less applicable in each case. This identifies the appropriate metrics for evaluating the strength and utility of GAG tools, based on their technique, allowing for them to be compared and improved with regard to their usefulness as tools. Furthermore, the insights gained from this study contributed to refining the GAGeTx framework, improving its utility as a comprehensive tool for guiding the selection, application, and evaluation of GAG methods based on user needs and priorities.

To achieve this, user preferences were assessed in the context of the prototype tool, inspecting the differences in favoured usage for the main technique process types depicted in GaGeTx. Users are found to prefer to have full control over the output in the form of parametric modelling or using a random seed technique in which they have minimal control. Users are found to associate the usefulness of different techniques with different combinations of the three evaluation types: quality, speed and controllability.

Proof-of-concept prototype generative tool

Building on the contributions of GAGeTx and incorporating general user preferences from the findings of objective 3, a proof-of-concept prototype sword generation tool named Swordgen was developed in pursuit of objective 4. This tool addresses the limitations of the mock-up testing conducted in objective 3 by offering a functional, interactive implementation that validates the GAGeTx framework. Furthermore, providing a platform for the second user-centered experiment, allowing for a deeper exploration of UX considerations.

Swordgen is a mixed-initiative asset generation tool for *offline* asset creation during game design and production pipelines, implementing GAG techniques at all levels of user-system initiative balance. While this is demonstrated with its successful application to the sword generation task, it is built as a highly configurable software framework integrated within the Unity engine, the structure of which provides a scalable platform for integrated generative tools. At its core, this system has two components, the first being a procedural shape algorithm, and the second being a GAG. The procedural shape algorithm abstracts the low-level asset structure, such as mesh data in this case, into a parametric system that simplifies and constrains the design space for both manual human creation and automated GAG-based generation. This abstraction reduces the complexity of implementing GAGs while embedding asset requirements like mesh topology, optimisation, and the target data format, which are particularly important in the production of game assets. It also facilitates easy user interaction, enabling edits within a unified interface. The interface provides high level controls, access to GAG features and a live asset preview, as well as direct structural and parametric controls for iteration and refinement through a graph view. Both the procedural shape algorithm and the GAG can be swapped depending on the task. For example, the procedural shape algorithm could be replaced with a parametric 2D icon system, while a new GAG could be built or trained for this new purpose.

Furthermore, this is not limited to a single GAG system at one time. While Swordgen demonstrates a range of techniques using a VAE-based GAG for proof-of-concept, further development can scale the system to integrate multiple GAG techniques, each specialised according to the

GAGeTx framework. This specialisation would ensure that each method is optimally suited to its task, leading to the highest possible output quality in line with state-of-the-art standards. In addition, this highlights the potential for specialised generative tools to improve and streamline game design and production pipelines.

This tool implemented representative members of the four main technique process categories via a single implementation. This tool uses a VAE in combination with a novel procedural modelling system for generating 3D sword assets. To achieve this, the VAE is trained on a purpose built sword shape silhouette dataset in combination with differentiable rendering. While the procedural modelling system involves the chaining together of individual parametric segments with shape defining parameters. The trained VAE is capable of taking an input silhouette and converting it into procedural modelling parameters, taking two inputs and interpolating between the two to produce output parameters, and generating parameters via a random latent vector. Representing reconstruction, interpolation and random technique processes, as well as representing guided techniques through the ability to manually input and iterate parameters.

A method for formulating training datasets for GAGs, and a novel sword shape dataset.

In the pursuit of objective 4, a method for formulating new shape-based datasets for graphical assets has been developed to address the need for data in asset specific deep-learning GAGs for games.

While generalised datasets such as ShapeNet [52] are crucial for learning a wide variety of shape classes, such datasets do not cover the often specific shapes and styles required within games. Given that different game genres and settings can require specialised assets, relying on general datasets will result in inconsistent or poor quality artefacts. In these cases, it can be necessary to create datasets tailored to these needs when developing deep-learning based GAGs.

The GAG dataset creation method, validated through its usage in Swordgen, enables researchers and practitioners to generate bespoke datasets with complete control over content style and shape variation, all without compromising on dataset size. This enables the production of large datasets, comprised of thousands of samples, from manually curated sources. Dataset creators can tailor the specificity or breadth of the content to suit their needs, even aligning it with their current project designs by using their own concept art. This level of customisation would otherwise be challenging to achieve due to the limited amount of source data.

The dataset creation method involves sourcing or producing a body of concept art, sample assets, and photographic references, standardising their format, and augmenting them to create large, asset-specific shape datasets. Through this method, a dataset of 13,728 sword silhouettes was generated from 513 source images using data augmentation. This dataset was then used to train the VAE component of the proof-of-concept prototype tool from objective 4. This was instrumental in producing the functionality for reconstructing sword shapes from photos and sketches, and learning sword features for random sampling and interpolation.

7.1.2 Research Limitations

During the creation of the proof-of-concept-prototype it was not feasible to include all technique process and interaction types due to time constraints, as achieving this would require multiple approaches and implementations. Instead, representative techniques from each *idea fidelity* group were included. Covering all categories would provide more granular insights with regard to which techniques users prefer, and how they can be used in the development pipeline. In addition, it was necessary to develop and fulfill these techniques through a single core implementation. Therefore, an approach that was capable of all techniques was selected. Provided enough time, each individual technique process should be fulfilled by an approach selected specifically for that purpose. This would ensure the maximum quality and validity of generated artefacts; thus, more faithfully representing state-of-the-art capabilities.

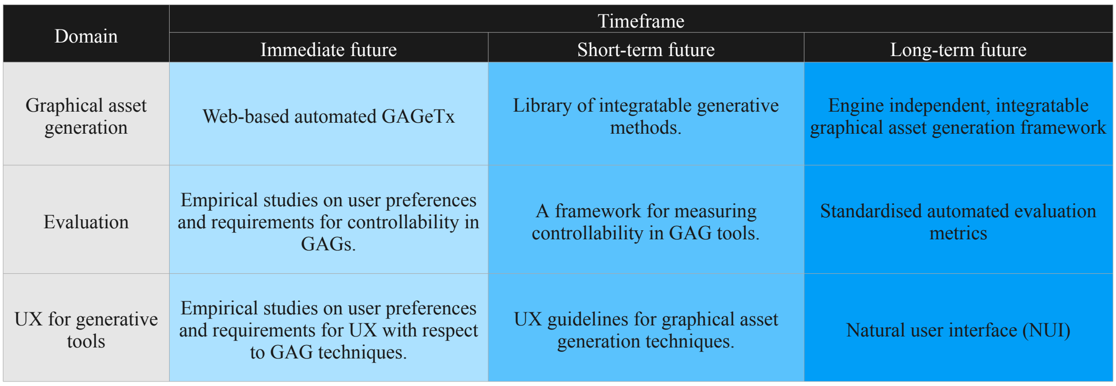
This research would benefit from accessing users from other game engine user-bases. While the Unity engine [475] is currently one of the most popular game engines in use, employed in the creation of many games each year, there is a multitude of other engines in use throughout the industry. Though many of these are proprietary, others such as Unreal Engine and Godot are openly accessible, providing APIs and frameworks for plugin and tool development. While the Unity engine does have a broad user-base, other engines are popular for different types of product, with which come different workflows and pipelines as well as user needs and preferences.

Due to the small sample sizes of the two user studies, findings cannot be generalised to a user-base at large. Furthermore, some insights can only be derived by corroboration between the quantitative and qualitative data. The findings, however, demonstrate validity and significance, with parallel insights in larger-scale studies. For instance, there is a high amount of generative tool usage for prototyping and concept work among users of generative AI in Unity game development [471]. Participants for both user studies were recruited via a convenience sampling approach. This was achieved through contacting gatekeepers of various organisations such as LUUG, BCS Animation and Games specialist group and game design students. 16 and 18 participants were recruited for user studies 1 and 2 respectively. With the number of creative games industry staff in the UK alone being approximately 24,000 in 2023 [446], larger scale testing would be imperative for supporting further insights. A more polished version of the tool with improved output quality would help to generate interest in participation. Additionally, in-person monitoring of participant's usage of the tool would provide further insights into the usability, controllability and effectiveness of the tool. Furthermore, the order in which mock-ups (study 1) and techniques (study 2) were presented to participants was intended to incrementally introduce the tool's functionality. However, it is acknowledged that using a pre-defined order may have introduced order effects, which could have influenced participant preferences. Future studies should address this by randomising or counterbalancing [202] the order of repeated measures.

During the interviews conducted in both studies, note-taking was used to record participant responses. This had the potential to introduce interviewer bias with regard the points recorded. Future work should make use of audio recording and transcription to ensure that the full nuance of participant responses can be analysed.

7.1.3 Future Research and Development

This section will present the avenues for research in the immediate, short-term and long-term future, as summarised in table 7.2. These are presented under three domains: *Graphical asset generation*, *Evaluation* and *UX for generative tools*. Further research into *graphical asset generation* entails efforts to build on the GAGeTx framework and provide accessible, reliable generative tooling based on user needs. Research into *evaluation* will seek to improve and standardise metrics and measures surrounding GAG tools, thus aiding in selecting appropriate methods based on user needs, while advancing GAGeTx. Further research in UX for generative tools will help to ensure that GAG tools can be affectively and seamlessly deployed in design and production pipelines.



Domain	Timeframe		
	Immediate future	Short-term future	Long-term future
Graphical asset generation	Web-based automated GAGeTx	Library of integratable generative methods.	Engine independent, integratable graphical asset generation framework
Evaluation	Empirical studies on user preferences and requirements for controllability in GAGs.	A framework for measuring controllability in GAG tools.	Standardised automated evaluation metrics
UX for generative tools	Empirical studies on user preferences and requirements for UX with respect to GAG techniques.	UX guidelines for graphical asset generation techniques.	Natural user interface (NUI)

TABLE 7.2: Summary of future research and development.

Immediate future

In the immediate pursuit of research in the domain of graphical asset generation, GAGeTx should be implemented as a web-based interactive tool, serving to improve the applicability and dissemination of this research. Thus far, the GAGeTx framework facilitates the choice of generation method, based on user needs. An interactive web-based tool would guide a user through the steps and selection process, retrieving appropriate matches from a growing database of state-of-the-art methods, and facilitating the ranking of relevant matches based on metric preference.

With regard to *evaluation*, controllability continues to be a challenging aspect to meaningfully measure. As is apparent from this research and others [464, 251], controllability and the balance of initiative between user and algorithm are pertinent when it comes to the usefulness of a given generative system. In order to support further understanding of the relationship between controllability and the usefulness of GAG tools, further empirical user-centred studies should be conducted. Such studies should examine the balance of initiative between user and GAG, how this affects the preferred usage of GAGs, and how much control each interaction type provides.

In the immediate future regarding *UX for generative tools*, further investigation into user preferences and requirements with respect to GAG techniques is necessary. According to Unity's game report 62% of developers that have begun adopting AI tools use them for asset generation [471]. As more game designers and developers integrate generative systems into their workflows, the demand for useful and user-friendly tools will only grow, highlighting a need for sustained research into generative tool integration. Further user-centred empirical studies should seek to investigate questions such as "what interface features are most effective for each technique?" and "what features are required, based on the type of design and production usage?". This would lead to identifying intuitive interfaces for GAGs.

Short-term future

In the pursuit of advancing the domain of *graphical asset generation*, a library of integratable GAG methods would serve to greatly improve the accessibility and applicability of GAG. Throughout this research, generative methods for graphical assets have been documented comprehensively. The many methods introduced and applied in the literature each independently solve problems across a range of fields. Furthermore, there are many methods that serve the same purpose, with varying benefits and drawbacks. Although it is valuable to develop and maintain a list of these methods, as shown in appendix A1, a library of these methods including ready-made implementations, would greatly enhance the accessibility and dissemination of GAG methods. Achieving this would open up the possibility for plug and play GAG systems, configured based on user need.

In the short-term regarding *evaluation*, building on the findings of empirical studies in the immediate future, focus should be placed on expanding controllability findings into a framework for mapping usage needs to interaction types and input complexity. By establishing controllability factors, such a framework would provide guidance on selecting the correct GAG techniques, based on the amount of control needed for a task. This would enable the comparison of GAG methods that share similar levels of controllability, and in combination with UX preferences, allow for the selection of GAG methods that match the balance of initiative users need.

In the short-term, the domain of *UX for generative tools* would benefit from a set of guidelines to help creators of GAG tools to effectively integrate their solutions. While the current work demystifies how each type of generative method can be used, it does not provide guidance for how to present these methods as tools for end-users such as game designers and developers. Building on experimentation in the short-term, the development of user experience guidelines relating to technique processes and interaction types would help to address this gap. Such guidelines should aim to provide best practices to ensure the efficient usage of graphical asset generation methods, aiding in their adoptability.

Long-term future

Graphics have and will continue to have a large impact on how we communicate, plan and entertain. Recent adoption of generative tools marks a huge turning point in how video games are designed and developed, and more broadly in how digital graphics are created. When used well, GAGs can serve as tools for streamlining design processes while preserving the human essence of creativity, and provide those with non-graphical skill sets a method of expressing their ideas visually. GAGeTx provides a starting point for future tools to incorporate the correct state-of-the-art methods according to the needs and requirements of users.

In the long-term, development of a modular, cross-engine software framework for graphical asset generation would be highly beneficial. Such a system could provide users with a range of methods to select and combine based on their needs. This would directly lead on from the curation of generative methods achieved in the short-term. A tool of this kind would be highly impactful, raising the ceiling for quality, variety and scale in game development, with additional potential use cases outside of games.

In the long-term with regard to *evaluation*, efforts should be made to standardise and automate the evaluation of GAGs. So far, the framework for metrics provides a descriptive presentation of metrics that are used in evaluating and comparing generative methods. However, with the wide variation in metrics used, comparisons are difficult to achieve unless all counterparts use the same metrics. Standardised metrics for generative techniques will ensure that new methods can easily be compared with existing alternatives. Achieving this will first require an in-depth investigation into which metrics best assess the performance of generative methods and the quality of their outputs. This will inform a prescriptive model of evaluation metrics. Following this, existing approaches may be assessed using these metrics, such that new and old methods can be ranked. This would also serve to strengthen the effectiveness of a GAGeTx web-tool, or a game engine integrated software framework, as it would facilitate the ranking of approaches for a given user goal.

With regard to *UX for generative tools*, research in the long-term should seek to examine the benefits of moving beyond user interaction via GUI. Natural user interfaces (NUIs) allow users to interact with computers in a manner that feels more intuitive and natural, by use of gestures, voice commands, and other forms of natural input. NUIs enhance UX by making software interactions more seamless and engaging [258]. Generative tools and the game design and production process at large may benefit greatly from more natural interaction methods, reducing the time and effort in bringing ideas to fruition and reducing the time cost of learning new tools. Utilising multimodal inputs, such as gestures and voice, allows for a richer interaction experience. This is evident in applications like gesture recognition systems, which have been successfully implemented in various fields, including medical applications and robotics [14, 313]. NUIs may also address the balance of initiative in PCG by providing un-restrained controllability, allowing users to intuitively provide as much input as necessary for a task, according to their capabilities.

References

[1] Mohamed Abdelaziz et al. "Generating 3D Model for Human Body Shapes from 2D images using Deep Learning". In: *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference, MIUCC* (2021), pp. 291–295.

[2] Zeina Abu-Aisheh et al. "An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems". In: *Proceedings of the International Conference on Pattern Recognition Applications and Methods - Volume 1*. ICPRAM 2015. Lisbon, Portugal: SCITEPRESS - Science and Technology Publications, Lda, 2015, 271–278. ISBN: 9789897580765. DOI: 10.5220/0005209202710278.

[3] Panos Achlioptas et al. "Learning representations and generative models for 3d point clouds". In: *International conference on machine learning*. PMLR. 2018, pp. 40–49.

[4] Panos Achlioptas et al. "ShapeGlot: Learning language for shape differentiation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8938–8947.

[5] Adobe. *Adobe Firefly*. 2024. URL: https://www.adobe.com/products/firefly.html.

[6] Adobe. *Substance Source*. 2023. URL: https://www.substance3d.com/.

[7] Adobe Inc. *Adobe Photoshop*. Version 2024. URL: https://www.adobe.com/products/photoshop.html.

[8] Allen Institute for AI. *RoboTHOR ObjectNav Challenge*. 2021. URL: https://github.com/allenai/robothor-challenge.

[9] Samar Al-Saqqa, Samer Sawalha, and Hiba AbdelNabi. "Agile software development: Methodologies and trends." In: *International Journal of Interactive Mobile Technologies* 14.11 (2020).

[10] Jorge Alcaide-Marzal, Jose Antonio Diego-Mas, and Gonzalo Acosta-Zazueta. "A 3D shape generative method for aesthetic product design". In: *Design Studies* 66 (2020), pp. 144–176. ISSN: 0142694X.

[11] Grégoire Allaire, François Jouve, and Anca-Maria Toader. "A level-set method for shape optimization". In: *Comptes Rendus Mathematique* 334.12 (2002), pp. 1125–1130. ISSN: 1631-073X.

[12] Dragomir Anguelov et al. *SCAPE Dataset*. 2005. URL: http://ai.stanford.edu/$\sim$drago/Projects/scape/scape.html.

[13] Izabella Antoniuk and Przemyslaw Rokita. "Generation of complex underground systems for application in computer games with schematic maps and L-systems". In: *Lecture Notes in Computer Science* 9972 LNCS (2016), pp. 3–16. ISSN: 16113349.

[14] F. Argelaguet et al. "Spatial and rotation invariant 3d gesture recognition based on sparse representation". In: (2017). DOI: `10.1109/3dui.2017.7893333`.

[15] Iro Armeni et al. *S3DIS Dataset*. 2016. URL: `http://buildingparser.stanford.edu/dataset.html`.

[16] Autodesk. *Autodesk Media & Entertainment Collection*. 2024. URL: `https://www.autodesk.co.uk/collections/media-entertainment/`.

[17] Abdulrahman Ayman, Yasser Mansour, and Hazem Eldaly. "Applying machine learning algorithms to architectural parameters for form generation". In: *Automation in Construction* 166 (2024), p. 105624. ISSN: 0926-5805. DOI: `https://doi.org/10.1016/j.autcon.2024.105624`.

[18] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. "Multimodal machine learning: A survey and taxonomy". In: *IEEE transactions on pattern analysis and machine intelligence* 41.2 (2018), pp. 423–443.

[19] Seonghoon Ban and Kyung Hoon Hyun. "3D Computational Sketch Synthesis Framework: Assisting Design Exploration Through Generating Variations of User Input Sketch and Interactive 3D Model Reconstruction". In: *CAD Computer Aided Design* 120 (2020), p. 102789. ISSN: 00104485.

[20] Shaun Bangay. "Deterministic procedural generation of mesh detail through gradient tiling". In: *ACM International Conference Proceeding Series* (2017).

[21] Aaron Bangor, Philip Kortum, and James Miller. "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale". In: *J. Usability Studies* 4.3 (2009), 114–123.

[22] Noa Barzilay, Tal Berkovitz Shalev, and Raja Giryes. "MISS GAN: A Multi-IlluStrator style generative adversarial network for image to illustration translation". In: *Pattern Recognition Letters* 151 (2021), pp. 140–147. ISSN: 01678655. arXiv: `2108.05693`.

[23] Bay 12 Games. *Dwarf Fortress*. 2006. URL: `http://www.bay12games.com/dwarves/`.

[24] Michael Becher et al. "Feature-based volumetric terrain generation". In: *Proceedings - I3D 2017: 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2017), pp. 1–9.

[25] Tony Beltramelli. *Pix2Code Dataset*. 2017. URL: `https://github.com/tonybeltramelli/pix2code`.

[26] Heli Ben-Hamu et al. "Multi-chart generative surface modeling". In: *ACM Trans. Graph.* 37.6 (2018), pp. 1–15. ISSN: 15577368. arXiv: `1806.02143`.

[27] Hilary Berger and Paul Beynon-Davies. "The utility of rapid application development in large-scale, complex projects". In: *Information Systems Journal* 19.6 (2009), pp. 549–570.

[28] Manush Bhatt et al. "Design and Deployment of Photo2Building: A Cloud-based Procedural Modeling Tool as a Service". In: *PEARC20: Practice & Experience in Advanced Research Computing*. 2020, pp. 132–138. ISBN: 9781450366892.

[29] Mikołaj Bińkowski et al. "Demystifying mmd gans". In: *arXiv preprint arXiv:1801.01401* (2018).

[30] Daniel T. Bishop et al. “A brief gamified immersive intervention to improve 11–14-year-olds’ cycling-related looking behaviour and situation awareness: A school-based pilot study”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 97 (2023), pp. 17–30. ISSN: 1369-8478. DOI: `https://doi.org/10.1016/j.trf.2023.06.019`.

[31] Volker Blanz and Thomas Vetter. “A Morphable Model For The Synthesis Of 3D Faces”. In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400708978.

[32] Blender Foundation. *Blender*. 2024. URL: `https://www.blender.org/`.

[33] Blizzard North. *Diablo 2*. 2000.

[34] Federica Bogo et al. *DFAUST Dataset*. 2014. URL: `http://faust.is.tue.mpg.de/`.

[35] Federica Bogo et al. *FAUST Dataset*. 2014. URL: `http://faust.is.tue.mpg.de/`.

[36] Sanja Bonic, Janos Bonic, and Stefan Schmid. “Broomrocket: Open Source Text-to-3D Algorithm for 3D Object Placement”. In: *ACM Games* 2.3 (Aug. 2024). DOI: `10.1145/3648233`.

[37] Giuseppe Bono. “Text-to-building: experiments with AI-generated 3D geometry for building design and structure generation”. In: *Architectural Intelligence* 3.1 (2024), p. 24. ISSN: 2731-6726. DOI: `10.1007/s44223-024-00060-5`.

[38] Alexei Botchkarev. “Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology”. In: *arXiv preprint arXiv:1809.03006* (2018).

[39] John Brooke. “SUS: A quick and dirty usability scale”. In: *Usability evaluation in industry* 189 (Nov. 1996), pp. 189–194.

[40] Adrian Bulat and Georgios Tzimiropoulos. “How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks)”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1021–1030.

[41] Michael Burri et al. “EuRoC MAV Dataset”. In: (2016). URL: `https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets`.

[42] Hongrui Cai et al. *CaricatureFace Dataset*. 2021. URL: `https://github.com/Juyong/CaricatureFace`.

[43] Hongrui Cai et al. *CaricatureFace Dataset*. 2021. URL: `https://github.com/Juyong/CaricatureFace`.

[44] Hongrui Cai et al. “Landmark Detection and 3D Face Reconstruction for Caricature using a Nonlinear Parametric Model”. In: *Graphical Models* 115 (2021), p. 101103. ISSN: 15240703. arXiv: `2004.09190`.

[45] Zehranaz Canfes et al. “Text and Image Guided 3D Avatar Generation and Manipulation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, pp. 4421–4431.

[46] Jun Cao et al. "Facade geometry generation from low-resolution aerial photographs for building energy modeling". In: *Building and Environment* 123 (2017), pp. 601–624. ISSN: 03601323.

[47] Qiong Cao et al. "Vggface2: A dataset for recognising faces across pose and age". In: *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*. IEEE. 2018, pp. 67–74.

[48] Jaime R Carbonell. "Mixed-Initiative Man-Computer Instructional Dialogues. Final Report." In: (1970).

[49] Blanca Ceballos, María Teresa Lamata, and David A. Pelta. "A comparative analysis of multi-criteria decision-making methods". In: *Progress in Artificial Intelligence* 5.4 (2016), pp. 315–322. ISSN: 2192-6360. DOI: `10.1007/s13748-016-0093-1`.

[50] Menglei Chai et al. *3DHW Dataset*. 2016. URL: `http://kunzhou.net/zjugaps/autohair/`.

[51] Angel Chang et al. "Matterport3d: Learning from rgb-d data in indoor environments". In: *arXiv preprint arXiv:1709.06158* (2017).

[52] Angel X. Chang et al. *ShapeNet Dataset*. 2015. URL: `https://shapenet.org/`.

[53] Anpei Chen et al. "SofGAN: A Portrait Image Generator with Dynamic Styling". In: *ACM Trans. Graph.* 41.1 (Feb. 2022). ISSN: 0730-0301. DOI: `10.1145/3470848`.

[54] Chih Fan Chen and Evan Suma Rosenberg. "Dynamic Omnidirectional Texture Synthesis for Photorealistic Virtual Content Creation". In: *Adjunct Proceedings - 2018 IEEE International Symposium on Mixed and Augmented Reality, ISMAR-Adjunct 2018* (2018), pp. 85–90.

[55] Jiacheng Chen et al. "Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2661–2670.

[56] Kevin Chen et al. "Text2shape: Generating shapes from natural language by learning joint embeddings". In: *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*. Springer. 2019, pp. 100–116.

[57] Rui Chen et al. "Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 22246–22256.

[58] Tianrun Chen et al. "Deep3DSketch-im: rapid high-fidelity AI 3D model generation by single freehand sketches". In: *Frontiers of Information Technology & Electronic Engineering* 25.1 (2024), pp. 149–159. ISSN: 2095-9230. DOI: `10.1631/FITEE.2300314`.

[59] Wenhu Chen et al. "Subject-driven Text-to-Image Generation via Apprenticeship Learning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 30286–30305.

[60] Zhiqin Chen and Hao Zhang. "Learning Implicit Fields for Generative Shape Modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[61] An-Chieh Cheng et al. "Autoregressive 3D Shape Generation via Canonical Mapping". In: *Computer Vision – ECCV*. Ed. by Shai Avidan et al. Springer, 2022, pp. 89–104.

[62] Yen-Chi Cheng et al. "SDFusion: Multimodal 3D Shape Completion, Reconstruction, and Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 4456–4465.

[63] Kyunghyun Cho et al. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* (2014).

[64] Sungjoon Choi et al. *redwood-3dscan Dataset*. 2016. URL: `http://redwood-data.org/3dscan/`.

[65] Yunjey Choi et al. "Stargan v2: Diverse image synthesis for multiple domains". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8188–8197.

[66] Noam Chomsky. *Aspects of the Theory of Syntax*. 1965, p. 251.

[67] Tarin Clanuwat et al. *K-MNIST Dataset*. 2018. URL: `https://github.com/rois-codh/kmnist`.

[68] CloudCompare. *Mesh-sample points*. 2015. URL: `https://www.cloudcompare.org/doc/wiki/index.php/Mesh\%5CSample_points`.

[69] Jasmine Collins et al. "Abo: Dataset and benchmarks for real-world 3d object understanding". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 21126–21136.

[70] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[71] IBM Corporation. "Correlations". In: 2021. URL: `https://www.ibm.com/docs/en/spss-statistics/beta?topic=features-correlations`.

[72] IBM Corporation. "Independent-samples t test". In: 2021. URL: `https://www.ibm.com/docs/en/spss-statistics/beta?topic=tests-independent-samples-t-test`.

[73] IBM Corporation. "K-S Subcommand (One-Sample) (NPAR TESTS command)". In: 2021. URL: `https://www.ibm.com/docs/en/spss-statistics/beta?topic=nt-k-s-subcommand-one-sample-npar-tests-command`.

[74] IBM Corporation. "MODELSUMMARY Subcommand (TSMODEL command)". In: 2021. URL: `https://www.ibm.com/docs/en/spss-statistics/beta?topic=tsmodel-modelsummary-subcommand-command`.

[75] Angela Dai et al. *ScanNet Dataset*. 2018. URL: `http://www.scan-net.org/`.

[76] Ayan Das et al. “Cloud2Curve: Generation and Vectorization of Parametric Sketches”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. ii. IEEE, 2021, pp. 7084–7093. ISBN: 978-1-6654-4509-2. arXiv: `2103.15536`.

[77] Josh Urban Davis et al. “Designing Co-Creative AI for Virtual Environments”. In: *C&C '21: Creativity and Cognition* (2021), pp. 1–11.

[78] Damon Daylamani-Zad, Fotios Spyridonis, and Kamal Al-Khafaaji. “A framework and serious game for decision making in stressful situations; a fire evacuation scenario”. In: *International Journal of Human-Computer Studies* 162 (2022), p. 102790. ISSN: 1071-5819. DOI: `https://doi.org/10.1016/j.ijhcs.2022.102790`.

[79] Matt Deitke et al. “Objaverse: A universe of annotated 3d objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13142–13153.

[80] Matt Deitke et al. “ProcTHOR: Large-Scale Embodied AI Using Procedural Generation”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 5982–5994.

[81] Biplab Deka et al. *Rico Dataset*. 2018. URL: `https://interactionmining.org/rico`.

[82] Johanna Delanoy et al. “3D Sketching using Multi-View Deep Volumetric Prediction”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–22. ISSN: 2577-6193.

[83] Johanna Delanoy et al. “Combining voxel and normal predictions for multi-view 3D sketching”. In: *Computers and Graphics (Pergamon)* 82 (2019), pp. 65–72. ISSN: 00978493.

[84] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes. “Proceduralization for editing 3D architectural models”. In: *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016* (2016), pp. 194–202.

[85] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes. “Proceduralization of urban models”. In: *2017 25th Signal Processing and Communications Applications Conference, SIU 2017* (2017), pp. 1–4.

[86] Jiankang Deng et al. *WildUV Dataset*. 2018. URL: `https://jiankangdeng.github.io/`.

[87] Yu Deng et al. “Accurate 3D Face Reconstruction With Weakly-Supervised Learning: From Single Image to Image Set”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 285–295.

[88] Jacob Devlin. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[89] Rahul Dey, Jason G. Doig, and Christos Gatzidis. “Procedural feature generation for volumetric terrains using voxel grammars”. In: *Entertainment Computing* 27.December 2017 (2018), pp. 128–136. ISSN: 18759521.

[90] Gabriel Dias Fernandes and António Ramires Fernandes. “Space Colonisation for Procedural Road Generation”. In: *Proceedings - ICGI 2018: International Conference on Graphics and Interaction* (2018).

[91] Ming Ding et al. "CogView: Mastering Text-to-Image Generation via Transformers". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 19822–19835. arXiv: 2105.13290.

[92] Yang Dongsheng, Kuang Ping, and Xiaofeng Gu. "3D Reconstruction based on GAT from a Single Image". In: *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2020* (2020), pp. 122–125.

[93] D. Dori and Wenyin Liu. "Sparse pixel vectorization: an algorithm and its performance evaluation". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 21.3 (1999), pp. 202–215.

[94] Wallas H.S. Dos Santos, Paulo Ivson, and Alberto Barbosa Raposo. "CAD Shape Grammar: Procedural Generation for Massive CAD Model". In: *Proceedings - 30th Conference on Graphics, Patterns and Images, SIBGRAPI 2017* (2017), pp. 31–38.

[95] Scott C. Douglas and Jiutian Yu. *Why ReLU Units Sometimes Die: Analysis of Single-Unit Error Backpropagation in Neural Networks*. 2018. arXiv: 1812.05981 [cs.LG].

[96] Laura Downs et al. "Google scanned objects: A high-quality dataset of 3d scanned household items". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2553–2560.

[97] Xue Mei Du et al. "Terrain Edge Stitching Based on Least Squares Generative Adversarial Networks". In: *2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2019* (2019), pp. 157–161.

[98] Zhenlong Du et al. "3D building fabrication with geometry and texture coordination via hybrid GAN". In: *Journal of Ambient Intelligence and Humanized Computing* (2020). ISSN: 18685145.

[99] Noah Duncan, Lap-Fai Yu, and Sai-Kit Yeung. "Interchangeable components for hands-on assembly based modelling". In: *ACM Trans. Graph.* 35.6 (2016). ISSN: 0730-0301. DOI: 10.1145/2980179.2982402.

[100] Marek Dvorožňák et al. "Monster Mash: A Single-View Approach to Casual 3D Modeling and Animation". In: *ACM Trans. Graph.* 39.6 (2020). ISSN: 0730-0301.

[101] Johannes Edelsbrunner et al. "Procedural Modeling of Round Building Geometry". In: *Proceedings - 2016 International Conference on Cyberworlds, CW 2016* (2016), pp. 81–88.

[102] Encord. "Mean Square Error (MSE) | Machine Learning Glossary | Encord | Encord — encord.com". In: [Accessed 06-01-2025]. 2025. URL: https://encord.com/glossary/mean-square-error-mse.

[103] Blizzard Entertainment. *World of Warcraft Frostmourne Letter Opener in Shape of Sword with Base, 22.5 cm, Home, Office Accessory, Gift and Merchandise for Fans & Collectors*. URL: https://www.amazon.co.uk/Warcraft-Frostmourne-Accessory-Merchandise-Collectors/dp/B08RZ99XYX?th=1.

[104] Environment Agency. *UK Environment Agency Data*. 2022. URL: https://data.gov.uk/dataset/f0db0249-f17b-4036-9e65-309148c97ce4/national-lidar-programme.

[105] Epic Games. *Unreal Engine*. 2024. URL: `https://www.unrealengine.com/en-US`.

[106] Mark Everingham et al. *PASCAL VOC*. 2012. URL: `http://host.robots.ox.ac.uk/pascal/VOC/`.

[107] Amin Fadaeddini, Babak Majidi, and Mohammad Eshghi. "A Case Study of Generative Adversarial Networks for Procedural Synthesis of Original Textures in Video Games". In: *2nd National and 1st International Digital Games Research Conference: Trends, Technologies, and Applications (DGRC)*. IEEE, 2018, pp. 118–122. ISBN: 9781728111148.

[108] Yangyu Fan et al. "Full Face-and-Head 3D Model with Photorealistic Texture". In: *IEEE Access* 8.4 (2020), pp. 210709–210721. ISSN: 21693536.

[109] Farbrausch. *.kkrieger*. 2004.

[110] Andrey Fedorov et al. "Interactive reconstruction of the 3D-models using single-view images and user markup". In: *Proceedings of the 2nd International Conference on Image and Graphics Processing* (2019), pp. 73–77. ISSN: 21531633.

[111] Li Fei-Fei et al. *ImageNet Dataset*. 2021. URL: `https://image-net.org/`.

[112] Roland Fischer et al. "AutoBiomes: procedural generation of multi-biome landscapes". In: *Visual Computer* 36.10-12 (2020), pp. 2263–2272. ISSN: 01782789.

[113] Marek Fiser et al. "Learning geometric graph grammars". In: *Proceedings - SCCG 2016: 32nd Spring Conference on Computer Graphics* (2016), pp. 7–15.

[114] Cristopher Flagg and Ophir Frieder. "Reconstruction of Artifacts from Digital Image Repositories". In: *J. Comput. Cult. Herit.* 16.1 (Dec. 2022). ISSN: 1556-4673. DOI: `10.1145/3552298`.

[115] Kai Franke and Heinrich Müller. "Procedural generation of 3D karst caves with speleothems". In: *Computers and Graphics (Pergamon)* (2021). ISSN: 00978493.

[116] Kevin Frans, Lisa Soros, and Olaf Witkowski. "CLIPDraw: Exploring Text-to-Drawing Synthesis through Language-Image Encoders". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 5207–5218.

[117] FreeImages. URL: `https://www.freeimages.com/`.

[118] Jonas Freiknecht and Wolfgang Effelsberg. "Procedural Generation of Multistory Buildings with Interior". In: *IEEE Trans. Games* 12.3 (2020), pp. 323–336. ISSN: 24751510.

[119] Timo Friedrich, Barbara Hammer, and Stefan Menzel. "Voxel-Based Three-Dimensional Neural Style Transfer". In: *Lecture Notes in Computer Science* 12861 (2021), pp. 334–346.

[120] Huan Fu et al. "3d-front: 3d furnished rooms with layouts and semantics". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10933–10942.

[121] Huan Fu et al. *3D-FUTURE Dataset*. 2021. URL: `https://tianchi.aliyun.com/specials/promotion/alibaba-3d-future`.

[122] Rao Fu et al. *ShapeCrafter: A Recursive Text-Conditioned 3D Shape Generation Model*. 2022.

[123] Yuta Fukatsu and Masaki Aono. “3D Mesh Generation by Introducing Extended Attentive Normalization”. In: *8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)* (2021), pp. 1–6.

[124] Yuuya Fukumoto, Daiki Shimizu, and Chihiro Shibata. “Generation of Character Illustrations from Stick Figures Using a Modification of Generative Adversarial Network”. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* 2 (2018), pp. 183–186. ISSN: 07303157.

[125] Yusuke Funabiki, Yuta Muraki, and Ken-ichi Kobori. “Automatic Generation of Background Computer Graphics by Deep Learning According to User’s Preference”. In: *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*. 2020, pp. 867–869. DOI: `10.1109/GCCE50665.2020.9291765`.

[126] Jun Gao et al. “GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images”. In: *Advances In Neural Information Processing Systems*. 2022.

[127] Lin Gao et al. “Efficient and Flexible Deformation Representation for Data-Driven Surface Modeling”. In: *ACM Trans. Graph.* 35.5 (2016). ISSN: 0730-0301.

[128] Lin Gao et al. “SceneHGN: Hierarchical Graph Networks for 3D Indoor Scene Generation With Fine-Grained Geometry”. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 45.7 (2023), pp. 8902–8919.

[129] Lin Gao et al. “SDM-NET: Deep generative network for structured deformable mesh”. In: *ACM Trans. Graph.* 38.6 (2019), pp. 1–15. ISSN: 15577368. arXiv: `1908.04520`.

[130] Lin Gao et al. “TM-NET: Deep Generative Networks for Textured Meshes”. In: *ACM Trans. Graph.* 40.6 (2021), pp. 1–15. arXiv: `2010.06217`.

[131] Xiang Gao, Yingjie Tian, and Zhiquan Qi. “RPD-GAN: Learning to draw realistic paintings with generative adversarial network”. In: *IEEE Trans. Image Processing* 29 (2020), pp. 8706–8720. ISSN: 19410042.

[132] Yang Gao, Yuan Yao, and Yunliang Jiang. “Multi-target 3D Reconstruction from RGB-D Data”. In: *CSSE 2019: Proceedings of the 2nd International Conference on Computer Science and Software Engineering* 2 (2019), pp. 184–191.

[133] Martin Gardner. “The Fantastic Combinations of John Conway’s New Solitaire Game 'Life'”. In: *Scientific American* 223 (1970), pp. 120–123.

[134] Leon Gatys, Alexander Ecker, and Matthias Bethge. “A Neural Algorithm of Artistic Style”. In: *Journal of Vision* 16.12 (2016), pp. 326–326. ISSN: 1534-7362.

[135] Geospatial Information Authority of Japan. *Fundamental geospatial data (Japan)*. 2022. URL: `https://www.gsi.go.jp/ENGLISH/index.html`.

[136] Roman Getto, Arjan Kuijper, and Dieter W. Fellner. “Automatic procedural model generation for 3D object variation”. In: *Visual Computer* 36.1 (2020), pp. 53–70. ISSN: 01782789.

[137] Theo Gevers, Hamdi Dibeklioglu, and Albert Ali Salah. *UvA-NEMO*. 2022. URL: `http://www.uva-nemo.org/index.html`.

[138] Sarah F. F. Gibson. "Constrained elastic surface nets: Generating smooth surfaces from binary segmented data". In: *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*. Ed. by William M. Wells, Alan Colchester, and Scott Delp. Springer, 1998, pp. 888–898. ISBN: 978-3-540-49563-5.

[139] Georgia Gkioxari, Justin Johnson, and Jitendra Malik. "Mesh R-CNN". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 9784–9794. ISBN: 978-1-7281-4803-8.

[140] Kirill Golubev, Aleksander Zagarskikh, and Andrey Karsakov. "Dijkstra-based Terrain Generation Using Advanced Weight Functions". In: *Procedia Computer Science* 101 (2016), pp. 152–160. ISSN: 18770509.

[141] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[142] Google. *Gemini*. 2024. URL: `https://gemini.google.com/?hl=en`.

[143] Google. *Google Earth Engine*. 2022. URL: `https://developers.google.com/earth-engine`.

[144] Google. *Quick, Draw! Dataset*. 2024. URL: `https://github.com/googlecreativelab/quickdraw-dataset`.

[145] Stephen Gould, Richard Fulton, and Daphne Koller. *Stanford Background Dataset*. 2009. URL: `http://dags.stanford.edu/projects/scenedataset.html`.

[146] Daniele Gravina et al. "Procedural Content Generation through Quality Diversity". In: *2019 IEEE Conference on Games (CoG)*. London, United Kingdom: IEEE Press, 2019, 1–8.

[147] Michael Cerny Green et al. "Organic Building Generation in Minecraft". In: *Proceedings of the 14th International Conference on the Foundations of Digital Games* (2019), pp. 1–7.

[148] Grinding Gear Games. *Path of Exile*. 2013.

[149] Ralph Gross et al. *Multi-Pie Dataset*. 2009. URL: `http://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html`.

[150] Nielsen Norman Group. *Wizards: Definition and Design Recommendations*. Accessed: 2024-09-13. 2024. URL: `https://www.nngroup.com/articles/wizards/`.

[151] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. "Geometry Images". In: 21.3 (2002), 355–361. ISSN: 0730-0301.

[152] Yanran Guan and Oliver van Kaick. "Diverse part synthesis for 3D shape creation". In: *Computer-Aided Design* 175 (2024), p. 103746. ISSN: 0010-4485. DOI: `https://doi.org/10.1016/j.cad.2024.103746`.

[153] Yanran Guan et al. "FAME: 3D Shape Generation via Functionality-Aware Model Evolution". In: *IEEE Trans. Visualization and Computer Graphics* 2626 (2020), pp. 1–1. ISSN: 1077-2626. arXiv: `2005.04464`.

[154] Greg Guest. *Applied thematic analysis*. Sage, 2012.

[155] Jianwei Guo et al. "Realistic Procedural Plant Modeling from Multiple View Images". In: *IEEE Trans. Visualization and Computer Graphics* 26.2 (2020), pp. 1372–1384.

[156] Agrim Gupta, Piotr Dollar, and Ross Girshick. *LVIS Dataset*. 2019. URL: `https://www.lvisdataset.org/`.

[157] Mohamed Hassan et al. *PROX Dataset*. 2019. URL: `https://prox.is.tue.mpg.de/`.

[158] Kaiming He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[159] Yuze He et al. "T3 Bench: Benchmarking Current Progress in Text-to-3D Generation". In: *arXiv preprint arXiv:2310.02977* (2023).

[160] Yuze He et al. "Text-image conditioned diffusion for consistent text-to-3D generation". In: *Computer Aided Geometric Design* 111 (2024), p. 102292. ISSN: 0167-8396. DOI: `https://doi.org/10.1016/j.cagd.2024.102292`.

[161] Eric Heim. "Constrained generative adversarial networks for interactive image generation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2019), pp. 10745–10753. ISSN: 10636919. arXiv: `1904.02526`.

[162] Hello Games. *No Man's Sky*. 2016. URL: `https://www.nomanssky.com/`.

[163] Paul Henderson, Vagia Tsiminaki, and Christoph H. Lampert. "Leveraging 2D Data to Learn Textured 3D Mesh Generation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2020), pp. 7495–7504. ISSN: 10636919. arXiv: `2004.04180`.

[164] Mark Hendrikx et al. "Procedural Content Generation for Games: A Survey". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 9.1 (2013). ISSN: 1551-6857.

[165] Amir Hertz et al. "Deep geometric texture synthesis". In: *ACM Trans. Graph.* 39.4 (2020), pp. 1–11. ISSN: 15577368. arXiv: `2007.00074`.

[166] Jack Hessel et al. "CLIPScore: A Reference-free Evaluation Metric for Image Captioning". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 7514–7528.

[167] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[168] Samet Hicsonmez et al. *GANILLA Datasets*. 2020. URL: `https://github.com/giddyyupp/ganilla/blob/master/docs/datasets.md`.

[169] Eka Wahyu Hidayat et al. "Visualization of A Three-Dimensional Tree Modeling using Fractal Based on L-System". In: *2020 IEEE International Conference on Sustainable Engineering and Creative Computing (ICSECC)*. 2020, pp. 418–422.

[170] Tommy Hinks et al. "Point Cloud Data Conversion into Solid Models via Point-Based Voxelization". In: *Journal of Surveying Engineering* 139.2 (2013), pp. 72–83.

[171] Houssam Hnaidi et al. "Feature based terrain generation using diffusion equation". In: *Computer Graphics Forum*. 7th ser. 29 (2010), pp. 2179–2186.

[172] Haodi Hou et al. "MW-GAN: Multi-Warping GAN for Caricature Generation with Multi-Style Geometric Exaggeration". In: *IEEE Trans. Image Processing* 30 (2021), pp. 8644–8657. ISSN: 19410042. arXiv: 2001.01870.

[173] Han Hu et al. "Semi-supervised adversarial recognition of refined window structures for inverse procedural façade modelling". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 192 (2022), pp. 215–231. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2022.08.014.

[174] Hao Hu, Chao Zhang, and Yanxue Liang. "A study on the automatic generation of banner layouts". In: *Computers & Electrical Engineering* 93 (2021), p. 107269. ISSN: 0045-7906.

[175] Liwen Hu et al. *USC-HairSalon Dataset*. 2015. URL: http://www-scf.usc.edu/$\sim$liwenhu/SHM/database.html.

[176] Ruizhen Hu et al. "Graph2Plan: learning floorplan generation from layout graphs". In: *ACM Trans. Graph.* 39.4 (2020). ISSN: 0730-0301.

[177] Ruizhen Hu et al. "Photo-to-shape material transfer for diverse structures." In: *ACM Trans. Graph.* 41.4 (2022), pp. 131–1.

[178] Yiwei Hu et al. "Generating Procedural Materials from Text or Image Prompts". In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH '23. Los Angeles, CA, USA: Association for Computing Machinery, 2023. ISBN: 9798400701597. DOI: 10.1145/3588432.3591520.

[179] Dongjin Huang et al. "Mesh-controllable multi-level-of-detail text-to-3D generation". In: *Computers & Graphics* 123 (2024), p. 104039. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2024.104039.

[180] Gary B. Huang et al. *LFW Dataset*. 2007. URL: http://vis-www.cs.umass.edu/lfw/.

[181] Haibin Huang et al. "Shape Synthesis from Sketches via Procedural Models and Convolutional Networks". In: *IEEE Trans. Visualization and Computer Graphics* 23.8 (2017), pp. 2003–2013. ISSN: 10772626.

[182] Jingwei Huang et al. "ArrangementNet: Learning Scene Arrangements for Vectorized Indoor Scene Modeling". In: *ACM Trans. Graph.* 42.4 (July 2023). ISSN: 0730-0301. DOI: 10.1145/3592122.

[183] Rowan T. Hughes, Liming Zhu, and Tomasz Bednarz. "Generative Adversarial Networks–Enabled Human–Artificial Intelligence Collaborative Applications for Creative and Design Industries: A Systematic Review of Current Approaches and Trends". In: *Frontiers in Artificial Intelligence* 4 (2021). ISSN: 2624-8212.

[184] Ka-Hei Hui et al. "Neural Template: Topology-Aware Reconstruction and Disentangled Generation of 3D Meshes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 18572–18582.

[185] Ka-Hei Hui et al. "Neural Wavelet-Domain Diffusion for 3D Shape Generation". In: *SIGGRAPH Asia Conference Papers*. SA '22. Daegu, Republic of Korea: ACM, 2022. ISBN: 9781450394703.

[186] Jing Huo et al. *WebCaricature Dataset*. 2018. URL: `https://cs.nju.edu.cn/rl/WebCaricature.htm`.

[187] IADF TC. *DFC2022*. 2022. URL: `https://www.grss-ieee.org/community/technical-committees/2022-ieee-grss-data-fusion-contest/#:$\sim$:text=Thesemi-supervisedlearningchallenge,researchinautomaticlandcover`.

[188] Didimo Inc. "Didimo". In: 2023.

[189] Smithsonian Institution. *Smithsonian Open Access*. URL: `https://www.si.edu/openaccess`.

[190] Interactive Data Visualization Inc. (IDV). *SpeedTree*. 2024. URL: `https://store.speedtree.com/`.

[191] Phillip Isola et al. *Edges2Shoes*. 2017. URL: `https://www.kaggle.com/datasets/balraj98/edges2shoes-dataset`.

[192] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 5967–5976. arXiv: `1611.07004`.

[193] Georgi Ivanov et al. "An Explorative Design Process for Game Map Generation Based on Satellite Images and Playability Factors". In: *International Conference on the Foundations of Digital Games* (2020).

[194] Kei Iwasaki, Yoshinori Dobashi, and Makoto Okabe. "Example-Based Synthesis of Three-Dimensional Clouds from Photographs". In: *Proceedings of the Computer Graphics International Conference*. CGI '17. Yokohama, Japan: ACM, 2017. ISBN: 9781450352284.

[195] Ajay Jain, Amber Xie, and Pieter Abbeel. "VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 1911–1920.

[196] Ajay Jain et al. "Zero-Shot Text-Guided Object Generation With Dream Fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 867–876.

[197] David G. Jansson and Steven M. Smith. "Design fixation". In: *Design Studies* 12.1 (1991), pp. 3–11. ISSN: 0142-694X.

[198] Japan Aerospace Exploration Agency. *AW3D30 Dataset*. 2022. URL: `https://www.eorc.jaxa.jp/ALOS/en/dataset/aw3d30/aw3d30_e.htm`.

[199] Junho Jeon et al. "Texture map generation for 3D reconstructed scenes". In: *Visual Computer* 32.6-8 (2016), pp. 955–965. ISSN: 01782789.

[200] Diego Jesus, António Coelho, and António Augusto Sousa. "Layered shape grammars for procedural modelling of buildings". In: *Visual Computer* 32.6-8 (2016), pp. 933–943. ISSN: 01782789.

[201] Bruno Ježek, Adam Ouhrabka, and Antonin Slabý. "Procedural Content Generation via Machine Learning in 2D Indoor Scene". In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis and Patrick Bourdot. Springer, 2020, pp. 34–49. ISBN: 978-3-030-58465-8.

[202] Rajiv S. Jhangiani et al. *Experimental Design*. [Accessed 2025-01-21]. 2022. URL: https://socialsci.libretexts.org/Bookshelves/Psychology/Research_Methods_and_Statistics/Research_Methods_in_Psychology_(Jhangiani_Chiang_Cuttler_and_Leighton)/05%3A_Experimental_Research/5.03%3A_Experimental_Design.

[203] Penglei Ji, Ming Zeng, and Xinguo Liu. "View Consistent 3D Face Reconstruction Using Siamese Encoder-Decoders". In: *Communications in Computer and Information Science* 1314 CCIS (2020), pp. 209–223. ISSN: 18650937.

[204] Chao Jia et al. "Scaling up visual and vision-language representation learning with noisy text supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 4904–4916.

[205] Chenfanfu Jiang et al. "Configurable 3D Scene Synthesis and 2D Image Rendering with Per-pixel Ground Truth Using Stochastic Grammars". In: *International Journal of Computer Vision* 126.9 (2018), pp. 920–941. ISSN: 1573-1405.

[206] R. Kenny Jones et al. "ShapeAssembly: Learning to Generate Programs for 3D Shape Structure Synthesis". In: *ACM Trans. Graph.* 39.6 (2020). ISSN: 15577368. arXiv: 2009.08026.

[207] R. Kenny Jones et al. "ShapeMOD: Macro Operation Discovery for 3D Shape Programs". In: *ACM Trans. Graph.* 40.4 (2021), pp. 1–16. ISSN: 15577368. arXiv: 2104.06392.

[208] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. *Labeled PSB*. 2010. URL: https://people.cs.umass.edu/$\sim$kalo/papers/LabelMeshes/.

[209] Amol Kapoor, Hunter Larco, and Raimondas Kiveris. "Nostalgin: Extracting 3D city models from historical image data". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019), pp. 2565–2575. arXiv: 1905.01772.

[210] Abhishek Kar, Christian Häne, and Jitendra Malik. "Learning a multi-view stereo machine". In: *Advances in neural information processing systems* 30 (2017).

[211] Rashed Karim et al. *SLAWT Dataset*. 2018. URL: https://www.doc.ic.ac.uk/$\sim$rkarim/la_lv_framework/wall/index.html.

[212] Rafal Karp and Zaneta Swiderska-Chadaj. "Automatic generation of graphical game assets using GAN". In: *ICCTA 2021: 2021 7th International Conference on Computer Technology Applications* (2021), pp. 7–12.

[213] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 43.12 (2021), 4217–4228. ISSN: 0162-8828.

[214] Tero Karras et al. "Progressive growing of GANs for improved quality, stability, and variation". In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* (2018), pp. 1–26. arXiv: 1710.10196.

[215] Jussi Kasurinen, Jukka-Pekka Strandén, and Kari Smolander. "What do game developers expect from development and design tools?" In: *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*. 2013, pp. 36–41.

[216] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3D Mesh Renderer". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), pp. 3907–3916. ISSN: 10636919. arXiv: 1711.07566.

[217] A. Kaufman, D. Cohen, and R. Yagel. "Volume graphics". In: *Computer* 26.7 (1993), pp. 51–64.

[218] Hadi Kazemi, Seyed Mehdi Iranmanesh, and Nasser M. Nasrabadi. "Style and content disentanglement in generative adversarial networks". In: *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019* (2019), pp. 848–856. arXiv: 1811.05621.

[219] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction". In: *Proceedings of the fourth Eurographics symposium on Geometry processing* 7 (2006), pp. 61–70. ISSN: 0730-0301.

[220] Rubaiat Habib Kazi et al. "DreamSketch: Early stage 3D design explorations with sketching and generative design". In: *UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 2017, pp. 401–414. ISBN: 9781450349819.

[221] Lin Ziwen Kelvin and Bhojan Anand. "Procedural Generation of Roads with Conditional Generative Adversarial Networks". In: *Proceedings - 2020 IEEE 6th International Conference on Multimedia Big Data, BigMM 2020* (2020), pp. 277–281.

[222] Lin Ziwen Kelvin and Anand Bhojan. "Procedural Generation of Roads with Conditional Generative Adversarial Networks". In: *ACM SIGGRAPH 2020 Posters* (2020), pp. 1–2. arXiv: 1707.03383.

[223] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4.

[224] Bernhard Kerbl et al. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. 2023. arXiv: 2308.04079 [cs.GR]. URL: https://arxiv.org/abs/2308.04079.

[225] Ahmed Khalifa et al. "PCGRL: Procedural Content Generation via Reinforcement Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 16.1 (2020), pp. 95–101.

[226] Asad Khan et al. "Learning-detailed 3D face reconstruction based on convolutional neural networks from a single image". In: *Neural Computing and Applications* 33.11 (2021), pp. 5951–5964. ISSN: 14333058.

[227] Samin Khan et al. "ProcSy: Procedural Synthetic Dataset Generation Towards Influence Factor Studies Of Semantic Segmentation Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2019.

[228] Jungeon Kim et al. "TextureMe : High-Quality Textured Scene Reconstruction in Real Time". In: *ACM Trans. Graph.* 41.3 (2022), pp. 1–18.

[229] Sangpil Kim, Hyung gun Chi, and Karthik Ramani. "Object Synthesis by Learning Part Geometry with Surface and Volumetric Representations". In: *CAD Computer Aided Design* 130 (2021), p. 102932. ISSN: 00104485.

[230] Suzi Kim, Dodam Kim, and Sunghee Choi. “CityCraft: 3D virtual city creation from a single image”. In: *Visual Computer* 36.5 (2020), pp. 911–924. ISSN: 01782789.

[231] Takumi Kimura, Takashi Matsubara, and Kuniaki Uehara. “ChartPointFlow for Topology-Aware 3D Point Cloud Generation”. In: *MM '21: Proceedings of the 29th ACM International Conference on Multimedia* (2021), pp. 1396–1404. arXiv: `2012.02346`.

[232] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: `1312.6114 [stat.ML]`. URL: `https://arxiv.org/abs/1312.6114`.

[233] Mariatul Kiptiah Binti Ariffin, Shiqah Hadi, and Somnuk Phon-Amnuaisuk. “Evolving 3D models using interactive genetic algorithms and L-systems”. In: *Lecture Notes in Computer Science* 10607 LNAI.i (2017), pp. 485–493. ISSN: 16113349.

[234] Brendan F Klare et al. “Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1931–1939.

[235] Roman Klokov, Edmond Boyer, and Jakob Verbeek. “Discrete Point Flow Networks for Efficient Point Cloud Generation”. In: *Lecture Notes in Computer Science* 12368 LNCS (2020), pp. 694–710. ISSN: 16113349. arXiv: `2007.10170`.

[236] Vladimir V. Kniaz et al. “Image-to-Voxel Model Translation for 3D Scene Reconstruction and Segmentation”. In: *Lecture Notes in Computer Science* 12352 LNCS (2020), pp. 105–124. ISSN: 16113349.

[237] Vladimir A. Knyaz, Vladimir V. Kniaz, and Fabio Remondino. “Image-to-voxel model translation with conditional adversarial networks”. In: *Lecture Notes in Computer Science* 11129 LNCS.1 (2019), pp. 601–618. ISSN: 16113349.

[238] Martin Koestinger et al. *AFLW Dataset*. 2011. URL: `https://www.tugraz.at/institute/icg/research/team-bischof/lrs/downloads/aflw/`.

[239] Eric Kolve et al. “Ai2-thor: An interactive 3d environment for visual ai”. In: *arXiv preprint arXiv:1712.05474* (2017).

[240] Richard Konečný, Stella Syllaiou, and Fotis Liarokapis. “Procedural modeling in archaeology: Approximating ionic style columns for games”. In: *2016 8th International Conference on Games and Virtual Worlds for Serious Applications, VS-Games 2016* (2016), pp. 1–8.

[241] Fanwei Kong, Nathan Wilson, and Shawn Shadden. “A deep-learning approach for direct whole-heart mesh reconstruction”. In: *Medical Image Analysis* 74 (2021), p. 102222. ISSN: 13618423. arXiv: `2102.07899`.

[242] Xiang Kong et al. “BLT: Bidirectional Layout Transformer for Controllable Layout Generation”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 474–490. ISBN: 978-3-031-19790-1.

[243] Oliver Korn et al. “Procedural Content Generation for Game Props? A Study on the Effects on User Experience”. In: *Computers in Entertainment* 15.2 (2017), pp. 1–15.

[244] Jon A Krosnick. “Questionnaire design”. In: *The Palgrave handbook of survey research* (2018), pp. 439–455.

[245] Vojtech Krs et al. “PICO: Procedural Iterative Constrained Optimizer for Geometric Modeling”. In: *IEEE Trans. Visualization and Computer Graphics* 27.10 (2021), pp. 3968–3981. ISSN: 19410506.

[246] Nina Krüger et al. “Deep Learning-Based Pulmonary Artery Surface Mesh Generation”. In: *Statistical Atlases and Computational Models of the Heart. Regular and CMRxRecon Challenge Papers*. Ed. by Oscar Camara et al. Cham: Springer Nature Switzerland, 2024, pp. 140–151. ISBN: 978-3-031-52448-6.

[247] Hailan Kuang et al. “3D face reconstruction with texture details from a single image based on GAN”. In: *Proceedings - 2019 11th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2019* (2019), pp. 385–388.

[248] Ping Kuang, Dingli Luo, and Haoshuang Wang. “Masked 3D conditional generative adversarial network for rock mesh generation”. In: *Cluster Computing* 22.s6 (2019), pp. 15471–15481. ISSN: 15737543.

[249] Bipin Kuriakose et al. “Synthesizing Images from Hand-Drawn Sketches using Conditional Generative Adversarial Networks”. In: *Proceedings of the International Conference on Electronics and Sustainable Communication Systems, ICESC 2020* (2020), pp. 774–778.

[250] Vivek Kwatra et al. “Texture Optimization for Example-Based Synthesis”. In: *ACM Trans. Graph.* 24.3 (2005), 795–802. ISSN: 0730-0301.

[251] Gorm Lai, William Latham, and Frederic Fol Leymarie. “Towards friendly mixed initiative procedural content generation: Three pillars of industry”. In: *Proceedings of the 15th International Conference on the Foundations of Digital Games*. 2020, pp. 1–4.

[252] Kevin Lai et al. *Washington RGB-D Dataset*. 2011. URL: `https://rgbd-dataset.cs.washington.edu/`.

[253] Jonathan Lampel. “The Art of Good Topology Guide”. In: 2024. URL: `https://cgcookie.com/posts/the-art-of-good-topology-blender`.

[254] Christoph Lassner et al. *UP-3D Dataset*. 2017. URL: `https://files.is.tuebingen.mpg.de/classner/up/`.

[255] A Laurentini. “The Visual Hull Concept for Silhouette-Based Image Understanding”. In: *IEEE Trans. Pattern Analysis & Machine Intelligence* 16.02 (1994), pp. 150–162. ISSN: 1939-3539.

[256] Felix Järemo Lawin, Per-Erik Forssén, and Hannes Ovrén. “Kinect V2 Dataset”. In: (2018). URL: `https://www.cvl.isy.liu.se/en/research/datasets/kinect2-dataset/`.

[257] Vuong Le et al. *Helen Dataset*. 2012. URL: `http://www.ifp.illinois.edu/$\sim$vuongle2/helen/`.

[258] Doyeob Lee et al. “Real-Time Recognition Method of Counting Fingers for Natural User Interface”. In: *KSII Transactions on Internet and Information Systems* 10.5 (2016), pp. 2363–2374. DOI: `10.3837/tiis.2016.05.022`.

[259] Hsin-Ying Lee et al. "Neural design network: Graphic layout generation with constraints". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer. 2020, pp. 491–506.

[260] Jae Joong Lee, Bosheng Li, and Bedrich Benes. "Latent L-systems: Transformer-based Tree Generator". In: *ACM Trans. Graph.* 43.1 (Nov. 2023). ISSN: 0730-0301. DOI: `10.1145/3627101`.

[261] Jiahui Lei et al. "Pix2Surf: Learning Parametric 3D Surface Models of Objects from Images". In: *Lecture Notes in Computer Science* 12363 LNCS (2020), pp. 121–138. ISSN: 16113349.

[262] Bowen Li et al. "Controllable Text-to-Image Generation". In: *Advances in Neural Information Processing Systems* 32 (2019).

[263] Chang Li et al. "Combining data-and-model-driven 3D modelling (CDMD3DM) for small indoor scenes using RGB-D data". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 180 (2021), pp. 1–13. ISSN: 09242716.

[264] Dan Li et al. "3D scene reconstruction using a texture probabilistic grammar". In: *Multimedia Tools and Applications* 77.21 (2018), pp. 28417–28440. ISSN: 15737721.

[265] Hai Li et al. "Saliency Guided Subdivision for Single-View Mesh Reconstruction". In: *Proceedings - 2020 International Conference on 3D Vision, 3DV 2020* (2020), pp. 1098–1107.

[266] Jianan Li et al. "Attribute-Conditioned Layout GAN for Automatic Graphic Design". In: *IEEE Trans. Visualization and Computer Graphics* 27.10 (2021), pp. 4039–4048. ISSN: 19410506. arXiv: `2009.05284`.

[267] Jianan Li et al. "LayoutGAN: Synthesizing Graphic Layouts with Vector-Wireframe Adversarial Networks". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 43.7 (2021), pp. 2388–2399. ISSN: 19393539.

[268] Manyi Li et al. "Grains: Generative recursive autoencoders for indoor scenes". In: *ACM Trans. Graph.* 38.2 (2019), pp. 1–16. ISSN: 15577368. arXiv: `1807.09193`.

[269] Minchen Li et al. "OptCuts: Joint Optimization of Surface Cuts and Parameterization". In: *ACM Trans. Graph.* 37.6 (2018). ISSN: 0730-0301.

[270] Ruihui Li et al. "SP-GAN: Sphere-guided 3D shape generation and manipulation". In: *ACM Trans. Graph.* 40.4 (2021), pp. 1–12. ISSN: 15577368. arXiv: `2108.04476`.

[271] Shidi Li, Miaomiao Liu, and Christian Walder. "EditVAE: Unsupervised Parts-Aware Controllable 3D Point Cloud Shape Generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.2 (2022), pp. 1386–1394.

[272] Song Li et al. "Deep 3D caricature face generation with identity and structure consistency". In: *Neurocomputing* 454 (2021), pp. 178–188. ISSN: 18728286.

[273] Tingting Li et al. "Pgan: Prediction generative adversarial nets for meshes". In: *2019 IEEE International Conference on Visual Communications and Image Processing, VCIP 2019* (2019), pp. 11–14.

[274] Xi Li et al. "3D Shape Reconstruction of Furniture Object from a Single Real Indoor Image". In: *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2020* (2020), pp. 101–104.

[275] Xi Zhi Li, Rene Weller, and Gabriel Zachmann. "AstroGen - Procedural Generation of Highly Detailed Asteroid Models". In: *2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018* (2018), pp. 1771–1778.

[276] Yi-Na Li, Kang Zhang, and Dong-Jin Li. "Rule-Based Automatic Generation of Logo Designs". In: *Leonardo* 50.2 (2017), pp. 177–181. ISSN: 0024-094X.

[277] Yushi Li and George Baciu. "HSGAN: Hierarchical Graph Learning for Point Cloud Generation". In: *IEEE Trans. Image Processing* 30 (2021), pp. 4540–4554. ISSN: 19410042.

[278] Yushi Li and George Baciu. "SG-GAN: Adversarial Self-Attention GCN for Point Cloud Topological Parts Generation". In: *IEEE Trans. Visualization and Computer Graphics* (2021). ISSN: 1077-2626.

[279] Zhihao Liao and Kai Xu. "Wavelet transform-assisted generative model for efficient 3d deep shape generation". In: *Multimedia Tools and Applications* (2024). ISSN: 1573-7721. DOI: 10.1007/s11042-024-18862-0.

[280] Antonios Liapis, Gillian Smith, and Noor Shaker. "Mixed-initiative content creation". In: *Procedural Content Generation in Games*. Springer, 2016. Chap. 11, pp. 195–214.

[281] Antonios Liapis, Georgios Yannakakis, and Julian Togelius. "Designer modeling for personalized game content creation tools". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 9. 2. 2013, pp. 11–16.

[282] LIFULL. "LIFULL HOME's Dataset". PhD thesis. 2020. URL: https://www.nii.ac.jp/dsc/idr/en/lifull/.

[283] Sohee Lim, Minwoo Shin, and Joonki Paik. "Point Cloud Generation Using Deep Adversarial Local Features for Augmented and Mixed Reality Contents". In: *IEEE Trans. Consumer Electronics* (2022). ISSN: 15584127.

[284] Chen-Hsuan Lin et al. "Magic3D: High-Resolution Text-to-3D Content Creation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 300–309.

[285] Cheng Lin et al. "Modeling 3D Shapes by Reinforcement Learning". In: *Lecture Notes in Computer Science* 12355 LNCS (2020), pp. 545–561. ISSN: 16113349. arXiv: 2003.12397.

[286] Jiangke Lin, Yi Yuan, and Zhengxia Zou. "MeInGame: Create a Game Character Face from a Single Portrait". In: (2021). arXiv: 2102.02371.

[287] Jiangke Lin, Yi Yuan, and Zhengxia Zou. *RGB 3D Face Dataset*. 2021. URL: https://github.com/FuxiCV/MeInGame.

[288] Tsung-Yi Lin et al. *COCO Dataset*. 2022. URL: https://cocodataset.org/.

[289] Yu Lin et al. "Generating Point Cloud from Single Image in the Few Shot Scenario". In: *Proceedings of the 29th ACM International Conference on Multimedia*. ACM, 2021, pp. 2834–2842.

[290] Yuanyuan lin et al. "Free editing of Shape and Texture with Deformable Net for 3D Caricature Generation". In: *The Visual Computer* 40.7 (2024), pp. 4675–4687. ISSN: 1432-2315. DOI: `10.1007/s00371-024-03461-9`.

[291] Aristid Lindenmayer. "Mathematical models for cellular interactions in development I. Filaments with one-sided inputs". In: *Journal of Theoretical Biology* 18.3 (1968), pp. 280–299. ISSN: 0022-5193.

[292] Wangxin Ling and Dongzhi He. "A Deep Learning Method Based on Structure Inference for Single-view 3D Reconstruction". In: *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. Vol. 3. 1. IEEE, 2021, pp. 2240–2244. ISBN: 978-1-7281-8028-1. arXiv: `arXiv:1011.1669v3`.

[293] Bogdan Lipuš and Nikola Guid. "A new implicit blending technique for volumetric modelling". In: *The Visual Computer* 21.1 (2005), pp. 83–91. ISSN: 1432-2315.

[294] Jialin Liu et al. "Deep learning for procedural content generation". In: *Neural Computing and Applications* 33.1 (2021), pp. 19–37. ISSN: 1433-3058.

[295] Vivian Liu and Lydia B. Chilton. "Design Guidelines for Prompt Engineering Text-to-Image Generative Models". In: *Conference on Human Factors in Computing Systems - Proceedings* 1.1 (2022), pp. 1–27. arXiv: `2109.06977`.

[296] Yan Liu et al. "Research on Lightweight 3D Reconstruction Techniques Based on Gaussian Splatting". In: *Proceedings of the 2023 International Conference on Advances in Artificial Intelligence and Applications*. AAIA '23. Association for Computing Machinery, 2024, 186–194. ISBN: 9798400708268. DOI: `10.1145/3603273.3634711`.

[297] Yuan Liu et al. *SyncDreamer: Generating Multiview-consistent Images from a Single-view Image*. 2024. arXiv: `2309.03453 [cs.CV]`.

[298] Zhengzhe Liu et al. *ISS: Image as Stepping Stone for Text-Guided 3D Shape Generation*. 2022.

[299] Zhengzhe Liu et al. "Towards Implicit Text-Guided 3D Shape Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 17896–17906.

[300] Zhibo Liu, Feng Gao, and Yizhou Wang. "A Generative Adversarial Network for AI-Aided Chair Design". In: *Proceedings - 2nd International Conference on Multimedia Information Processing and Retrieval, MIPR 2019* (2019), pp. 486–490. arXiv: `2001.11715`.

[301] Ziwei Liu et al. *CelebA Dataset*. 2015. URL: `https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`.

[302] Matthew Loper et al. *SMPL Dataset*. 2015. URL: `https://smpl.is.tue.mpg.de/`.

[303] William E. Lorensen and Harvey E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987* 21.4 (1987), pp. 163–169.

[304] Jonathan Lorraine et al. "ATT3D: Amortized Text-to-3D Object Synthesis". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 17946–17956.

[305] Yawen Lu, Yuxing Wang, and Guoyu Lu. “Single Image Shape-from-Silhouettes”. In: *MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia* (2020), pp. 3604–3613.

[306] Dylan Lukes et al. “Synthesis of Web Layouts from Examples”. In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. Athens, Greece: ACM, 2021, 651–663. ISBN: 9781450385626.

[307] Zhaoliang Lun et al. “Functionality preserving shape style transfer”. In: *ACM Trans. Graph.* 35.6 (2016), pp. 1–14. ISSN: 15577368.

[308] Naureen Mahmood et al. *AMASS Dataset*. 2019. URL: `https://amass.is.tue.mpg.de/`.

[309] Jameel Malik et al. “Handvoxnet: Deep voxel-based network for 3d hand shape and pose estimation from a single depth map”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7113–7122.

[310] Jameel Malik et al. *SynHand5M*. 2018. URL: `https://cloud.dfki.de/owncloud/index.php/s/iCMRF7a5FkXrdpn`.

[311] Benoit B. Mandelbrot and John W. Van Ness. “Fractional Brownian Motions, Fractional Noises and Applications”. In: *SIAM Review* 10.4 (1968). Full publication date: Oct., 1968, pp. 422–437. ISSN: 00361445.

[312] Gilda Manfredi et al. “TreeSketchNet: From Sketch to 3D Tree Parameters Generation”. In: *ACM Trans. Intell. Syst. Technol.* 14.3 (Mar. 2023). ISSN: 2157-6904. DOI: `10.1145/3579831`.

[313] A. Maqueda et al. “Human–computer interaction based on visual hand-gesture recognition using volumetric spatiograms of local binary patterns”. In: *Computer Vision and Image Understanding* 141 (2015), pp. 126–137. DOI: `10.1016/j.cviu.2015.07.009`.

[314] Aleix Martinez and Robert Benavente. “AR Face Dataset”. In: (1998). URL: `https://www2.ece.ohio-state.edu/$\sim$aleix/ARdatabase.html`.

[315] Andelo Martinovic and Luc Van Gool. “Bayesian grammar learning for inverse procedural modeling”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2013), pp. 201–208. ISSN: 10636919.

[316] Mansour Mehranfar et al. “Automated data-driven method for creating digital building models from dense point clouds and images through semantic segmentation and parametric model fitting”. In: *Advanced Engineering Informatics* 62 (2024), p. 102643. ISSN: 1474-0346. DOI: `https://doi.org/10.1016/j.aei.2024.102643`.

[317] Luke Melas-Kyriazi et al. “Realfusion: 360 deg reconstruction of any object from a single image”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 8446–8455.

[318] Mostafa Merras et al. “Multi-view 3D reconstruction and modeling of the unknown 3D scenes using genetic algorithms”. In: *Soft Computing* 22.19 (2018), pp. 6271–6289. ISSN: 14337479.

[319] Lars Mescheder et al. "Occupancy Networks: Learning 3D Reconstruction in Function Space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[320] Gal Metzer et al. "Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 12663–12673.

[321] Lu Mi et al. "Revisiting Latent-Space Interpolation via a Quantitative Evaluation Framework". In: (2021). arXiv: 2110.06421. URL: http://arxiv.org/abs/2110.06421.

[322] Microsoft Corporation. *Visual Studio*. Version 2022. URL: https://visualstudio.microsoft.com/.

[323] Midjourney inc. *Midjourney*. 2024. URL: https://www.midjourney.com/home.

[324] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 405–421. ISBN: 978-3-030-58452-8.

[325] Shervin Minaee et al. "Image segmentation using deep learning: A survey". In: *IEEE Trans. pattern analysis and machine intelligence* (2021).

[326] Anand Mishra, Karteek Alahari, and C.V. Jawahar. "IIIT 5K-word Dataset". In: (2012). URL: https://cvit.iiit.ac.in/research/projects/cvit-projects/the-iiit-5k-word-dataset.

[327] Paritosh Mittal et al. "AutoSDF: Shape Priors for 3D Completion, Reconstruction and Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 306–315.

[328] Kaichun Mo et al. "PartNet Dataset". In: (2019). URL: https://www.shapenet.org/download/parts.

[329] Kaichun Mo et al. "StructureNet: hierarchical graph networks for 3D shape generation". In: *ACM Trans. Graph.* 38.6 (2019), pp. 1–19. ISSN: 0730-0301.

[330] Anselmo Montenegro et al. "A New Method for Modeling Clouds Combining Procedural and Implicit Models". In: *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2017, pp. 173–182.

[331] Stylianos Moschoglou et al. *AgeDB Dataset*. 2017. URL: https://complexity.cecs.ucf.edu/agedb/.

[332] Carlos Eduardo Vaisman Muniz and Wagner Luiz Oliveira dos Santos. "Generative Design applied to Cloud Modeling". In: *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 2021, pp. 79–86. ISBN: 978-1-6654-0189-0.

[333] Hagley Museum. *Hagley Museum and Library*. 2024. URL: https://www.hagley.org/research.

[334] Nadermx. *Nadermx/backgroundremover: Background remover lets you remove background from images and video using AI with a simple command line interface that is free and open source.* URL: https://github.com/nadermx/backgroundremover.

[335] Nakendraprasath K. *Cloud image classification Dataset*. 2020. URL: https://www.kaggle.com/nakendraprasathk/cloud-image-classification-dataset.

[336] Shu Naritomi and Keiji Yanai. "3D Mesh Reconstruction of Foods from a Single Image". In: *AIxFood 2021 - Proceedings of the 3rd Workshop on AIxFood, co-located with ACM MM 2021* (2021), pp. 7–11.

[337] NASA. *NASA Visible Earth*. 2022. URL: https://visibleearth.nasa.gov/.

[338] Nelson Nauata et al. "House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation". In: *Lecture Notes in Computer Science* 12346 LNCS (2020), pp. 162–177. ISSN: 16113349. arXiv: 2003.06988.

[339] Teo T. Niemirepo, Marko Viitanen, and Jarno Vanne. "Open3DGen: Open-Source Software for Reconstructing Textured 3D Models from RGB-D Images". In: *Proceedings of the 12th ACM Multimedia Systems Conference*. MMSys '21. Istanbul, Turkey: ACM, 2021, 12–22. ISBN: 9781450384346.

[340] Maria-Elena Nilsback and Andrew Zisserman. *Oxford-102 flower Dataset*. 2008. URL: https://www.robots.ox.ac.uk/$\sim$vgg/data/flowers/102/.

[341] Gen Nishida, Adrien Bousseau, and Daniel G. Aliaga. "Procedural modeling of a building from a single image". In: *Computer Graphics Forum* 37.2 (2018), pp. 415–429. ISSN: 14678659.

[342] Gen Nishida et al. "Interactive sketching of urban procedural models". In: *ACM Trans. Graph.* 35.4 (2016), pp. 1–11. ISSN: 15577368.

[343] NIST. *FRGC Dataset*. 2021. URL: https://www.nist.gov/programs-projects/face-recognition-grand-challenge-frgc.

[344] NVIDIA Development Inc. *Conversions across various representations*. 2019. URL: https://kaolin.readthedocs.io/en/v0.1/notes/conversions_tutorial.html.

[345] Cihan Öngün and Alptekin Temizel. "LPMNet: Latent part modification and generation for 3D point clouds". In: *Computers and Graphics (Pergamon)* 96 (2021), pp. 1–13. ISSN: 00978493. arXiv: 2008.03560.

[346] Cihan Ongun and Alptekin Temizel. "Paired 3D model generation with conditional generative adversarial networks". In: *Lecture Notes in Computer Science* 11129 LNCS (2019), pp. 473–487. ISSN: 16113349. arXiv: 1808.03082.

[347] OpenAI. *GPT-4o*. 2024. URL: https://openai.com/index/hello-gpt-4o/.

[348] OpenStreetMap Foundation. *OpenStreetMap Dataset*. 2024. URL: https://planet.openstreetmap.org/.

[349] Roy Or-El et al. "StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 13503–13513.

[350] Matthew J Page et al. "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews". In: *BMJ* 372 (2021). DOI: 10.1136/bmj.n71. eprint: https://www.bmj.com/content/372/bmj.n71.full.pdf. URL: https://www.bmj.com/content/372/bmj.n71.

[351] Junyi Pan et al. "Deep mesh reconstruction from single rgb images via topology modification networks". In: *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 9963–9972. ISSN: 15505499. arXiv: `1909.00321`.

[352] Junyi Pan et al. "Residual MeshNet: Learning to deform meshes for single-view 3D reconstruction". In: *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018* (2018), pp. 719–727.

[353] Emmanouil Panagiotou and Eleni Charou. "Procedural 3D Terrain Generation using Generative Adversarial Networks". In: (2020). ISSN: 22195491. arXiv: `2010.06411`. URL: `http://arxiv.org/abs/2010.06411`.

[354] Dong Huk Park et al. "Benchmark for Compositional Text-to-Image Synthesis". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 2021.

[355] Taesung Park et al. "Semantic image synthesis with spatially-adaptive normalization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2337–2346.

[356] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. *VGG Face Dataset*. 2015. URL: `https://www.robots.ox.ac.uk/$\sim$vgg/data/vgg_face/`.

[357] Despoina Paschalidou et al. "ATISS: Autoregressive Transformers for Indoor Scene Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 12013–12026.

[358] Or Patashnik et al. "Localizing Object-Level Shape Variations with Text-to-Image Diffusion Models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 23051–23061.

[359] Or Patashnik et al. "StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 2065–2074. DOI: `10.1109/ICCV48922.2021.00209`.

[360] Dario Pavllo et al. "Convolutional Generation of Textured 3D Meshes". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 870–882.

[361] Dunlu Peng et al. "SAM-GAN: Self-Attention supporting Multi-stage Generative Adversarial Networks for text-to-image synthesis". In: *Neural Networks* 138 (2021), pp. 57–67. ISSN: 18792782.

[362] Songyou Peng et al. "Convolutional Occupancy Networks". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Springer, 2020, pp. 523–540. ISBN: 978-3-030-58580-8.

[363] Ken Perlin. "An Image Synthesizer". In: *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '85. ACM, 1985, 287–296. ISBN: 0897911660.

[364] Ken Perlin. "Noise Hardware". In: *Real-Time Shading*. Course Notes. SIGGRAPH, 2001.

[365] Tomas Polasek et al. "ICTree: automatic perceptual metrics for tree models". In: *ACM Trans. Graph.* 40.6 (Dec. 2021). ISSN: 0730-0301. DOI: 10.1145/3478513.3480519.

[366] Gerard Pons-Moll et al. *Dyna Dataset*. 2015. URL: http://dyna.is.tue.mpg.de/.

[367] Jhony K. Pontes et al. "Image2Mesh: A Learning Framework for Single Image 3D Reconstruction". In: *Lecture Notes in Computer Science* 11361 LNCS (2019), pp. 365–381. ISSN: 16113349. arXiv: 1711.10669.

[368] Ben Poole et al. *DreamFusion: Text-to-3D using 2D Diffusion*. 2022. arXiv: 2209.14988 [cs.CV].

[369] Pulak Purkait, Christopher Zach, and Ian Reid. "SG-VAE: Scene Grammar Variational Autoencoder to Generate New Indoor Scenes". In: *Lecture Notes in Computer Science* 12369 LNCS (2020), pp. 155–171. ISSN: 16113349. arXiv: 1912.04554.

[370] PyTorch Contributors. *torch.unsqueeze*. 2023. URL: https://pytorch.org/docs/stable/generated/torch.unsqueeze.html.

[371] Guocheng Qian et al. *Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors*. 2023. arXiv: 2306.17843 [cs.CV].

[372] Tingting Qiao et al. "Learn, imagine and create: Text-to-image generation from prior knowledge". In: *Advances in Neural Information Processing Systems* 32 (2019). ISSN: 10495258.

[373] Tingting Qiao et al. "MirrorGAN: Learning Text-To-Image Generation by Redescription". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 1505–1514. ISBN: 978-1-7281-3293-8. arXiv: 1903.05854.

[374] Yuda Qiu et al. "3dcaricshop: A dataset and a baseline method for single-view 3d caricature face reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10236–10245.

[375] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[376] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67.

[377] Yasin Raies and Sebastian Von Mammen. "A Swarm Grammar-Based Approach to Virtual World Generation". In: *LNCS* 12693 (2021), pp. 459–474.

[378] Alexander Raistrick et al. "Infinite Photorealistic Worlds Using Procedural Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12630–12641.

[379] Rido Ramadan and Yani Widyani. "Game development life cycle guidelines". In: *2013 International Conference on Advanced Computer Science and Information Systems, ICACSIS 2013* (2013), pp. 95–100. DOI: 10.1109/ICACSIS.2013.6761558.

[380] Santhosh K Ramakrishnan et al. "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai". In: *arXiv preprint arXiv:2109.08238* (2021).

[381] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125* (2022).

[382] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". In: *Proceedings of the 38th International Conference on Machine Learning* 139 (2021), pp. 8821–8831.

[383] Rafsan Ratul et al. "Applicability of Space Colonization Algorithm for Real Time Tree Generation". In: *2019 22nd International Conference on Computer and Information Technology (ICCIT)*. 2019, pp. 1–6.

[384] Ygor Reboucas Serpa and Maria Andreia Formico Rodrigues. "Towards Machine-Learning Assisted Asset Generation for Games: A Study on Pixel Art Sprite Sheets". In: *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)* (2019), pp. 182–191. ISSN: 21596662.

[385] Stephen Regelous. "MASSIVE". In: (2022). URL: `https://www.massivesoftware.com/`.

[386] Jeremy Reizenstein et al. "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10901–10911.

[387] Jing Ren et al. "Intuitive and Efficient Roof Modeling for Reconstruction and Synthesis". In: *ACM Trans. Graph.* 40.6 (2021), pp. 1–17. arXiv: `2109.07683`.

[388] Mitchel Resnick et al. "Design Principles for Tools to Support Creative Thinking". In: (June 2018). DOI: `10.1184/R1/6621917.v1`.

[389] Elad Richardson et al. "TEXTure: Text-Guided Texturing of 3D Shapes". In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH '23. Los Angeles, CA, USA: Association for Computing Machinery, 2023. ISBN: 9798400701597. DOI: `10.1145/3588432.3591503`.

[390] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: (2022), pp. 10674–10685. ISSN: 10636919. DOI: `10.1109/cvpr52688.2022.01042`. arXiv: `2112.10752`.

[391] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Springer, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

[392] Amanda Ross and Victor L Willson. "One-way anova". In: *Basic and advanced statistical tests*. Brill, 2017, pp. 21–24.

[393] Nataniel Ruiz et al. "DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 22500–22510.

[394] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: `10.1007/s11263-015-0816-y`.

[395] SAE International. *CAESAR Dataset*. 2002. URL: `https://www.sae.org/standardsdev/tsb/cooperative/caesar.htm`.

[396] Christos Sagonas et al. *300W Dataset*. 2013. URL: https://ibug.doc.ic.ac.uk/resources/300-W/.

[397] Chitwan Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: (2022). arXiv: 2205.11487. URL: http://arxiv.org/abs/2205.11487.

[398] Shunsuke Saito et al. "3D hair synthesis using volumetric variational autoencoders". In: *ACM Trans. Graph.* 37.6 (2018), pp. 1–12. ISSN: 15577368.

[399] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016).

[400] Aditya Sanghi et al. "CLIP-Forge: Towards Zero-Shot Text-To-Shape Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 18603–18613.

[401] Khalil Satyadama, Reza Fuad Rachmadi, and Supeno Mardi Susiki Nugroho. "Procedural Environment Generation for Cave 3D Model Using OpenSimplex Noise and Marching Cube". In: *CENIM 2020 - Proceeding: International Conference on Computer Engineering, Network, and Intelligent Multimedia* (2020), pp. 144–148.

[402] Manolis Savva et al. *PiGraphs Dataset*. 2016. URL: https://aspis.cmpt.sfu.ca/scene-toolkit/pigraphs/.

[403] Christoph Schuhmann et al. "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs". In: *arXiv preprint arXiv:2111.02114* (2021).

[404] Christoph Schuhmann et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25278–25294.

[405] Henning Schulze, Dogucan Yaman, and Alexander Waibel. "CAGAN: Text-To-Image Generation with Combined Attention Generative Adversarial Networks". In: *Lecture Notes in Computer Science* 13024 (2021), pp. 392–404. ISSN: 16113349.

[406] Scratchapixel 2.0. *Rasterization: a Practical Implementation*. 2022. URL: https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation.

[407] Thomas W Sederberg and Scott R Parry. "Free-form deformation of solid geometric models". In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 1986, pp. 151–160.

[408] Soumyadip Sengupta et al. *CFPW Dataset*. 2016. URL: http://www.cfpw.io/.

[409] Gil Shamai, Ron Slossberg, and Ron Kimmel. "Synthesizing facial photometries and corresponding geometries using generative adversarial networks". In: *ACM Trans. Multim. Comp., Commu. Apps.* 15.3s (2019), pp. 1–24. ISSN: 15516865. arXiv: 1901.06551.

[410] Rahul Sharma. "Procedural City Generator". In: *Int. Conf. System Modeling & Advancement in Research Trends (SMART)*. IEEE, 2016, pp. 213–217. ISBN: 978-1-5090-3543-4.

[411] Hamid R Sheikh and Alan C Bovik. “Image information and visual quality”. In: *IEEE Transactions on image processing* 15.2 (2006), pp. 430–444.

[412] I. Chao Shen and Bing-Yu Chen. “ClipGen: A Deep Generative Model for Clipart Vectorization and Synthesis”. In: *IEEE Trans. Visualization and Computer Graphics* 28.12 (2021), pp. 4211–4224. ISSN: 19410506. eprint: `2106.04912`.

[413] I-Chao Shen and Bing-Yu Chen. *ClipNet Dataset*. 2020. URL: `https://drive.google.com/file/d/1-0AM1_dLJ26MHIiD3VZFlxCm_1QN996z/view?usp=sharing`.

[414] Tianchang Shen et al. “Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 6087–6101.

[415] Yuefan Shen et al. “DeepSketchHair : Deep Sketch-Based 3D Hair Modeling”. In: *IEEE Trans. visualization and computer graphics* 27.7 (2021), pp. 3250–3263.

[416] Tianyang Shi et al. “Face-to-Parameter Translation for Game Character Auto-Creation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.02 (2019), pp. 1733–1740. ISSN: 2374-3468. arXiv: `1909.01064`.

[417] Tianyang Shi et al. “Neural Rendering for Game Character Auto-Creation”. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 44.3 (2022), pp. 1489–1502. ISSN: 19393539.

[418] Takayuki Shinohara, Haoyi Xiu, and Masashi Matsuoka. “3D Point Cloud Generation Using Adversarial Training for Large-Scale Outdoor Scene”. In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS* (2021), pp. 2935–2938.

[419] Dongwook Shu, Sung Woo Park, and Junseok Kwon. “3D point cloud generative adversarial network based on tree structured graph convolutions”. In: *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 3858–3867. ISSN: 15505499. arXiv: `1905.06292`.

[420] Yezhi Shu et al. “GAN-based Multi-Style Photo Cartoonization”. In: *IEEE Trans. Visualization and Computer Graphics* (2021), pp. 1–14. ISSN: 19410506.

[421] SideFX. *Houdini*. 2024. URL: `https://www.sidefx.com/`.

[422] Nathan Silberman et al. *NYU Depth Dataset V2*. 2012. URL: `https://cs.nyu.edu/$\sim$silberman/datasets/nyu_depth_v2.html`.

[423] Sergio N. Silva Junior et al. “A 3D modeling methodology based on a concavity-aware geometric test to create 3D textured coarse models from concept art and orthographic projections”. In: *Computers and Graphics (Pergamon)* 76 (2018), pp. 73–83. ISSN: 00978493.

[424] Zackary P T Sin and Peter H F Ng. “Planetary Marching Cubes: A Marching Cubes Algorithm for Spherical Space”. In: *Proceedings of the 2018 the 2nd International Conference on Video and Image Processing* (2018), pp. 89–94.

[425] Vedant Singh et al. “Auto-encoding progressive generative adversarial networks for 3D multi object scenes”. In: *Proceeding - 2019 International Conference of Artificial Intelligence and Information Technology, ICAIIT 2019* 1 (2019), pp. 481–485. arXiv: `1903.03477`.

[426] Sketchfab. *Sketchfab*. 2024. URL: `https://sketchfab.com/`.

[427] Miroslava Slavcheva et al. *3D-Printed RGB-D Object Dataset*. 2016.

[428] Miroslava Slavcheva et al. "SDF-2-SDF: Highly Accurate 3D Object Reconstruction". In: *Lecture Notes in Computer Science*. Ed. by Bastian Leibe et al. Vol. 9905. Lecture Notes in Computer Science. Springer, 2016, pp. 680–696. ISBN: 978-3-319-46447-3.

[429] Ruben M. Smelik et al. "A Survey on Procedural Modelling for Virtual Worlds". In: *Computer Graphics Forum* 33.6 (2014), pp. 31–50.

[430] Dmitriy Smirnov et al. "MarioNette: Self-Supervised Sprite Learning". In: *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)* 34 (2021), pp. 5494–5505. arXiv: `2104.14553`. URL: `http://arxiv.org/abs/2104.14553`.

[431] Jakub Sochor et al. *BrnoComp- Speed Dataset*. 2018. URL: `https://github.com/JakubSochor/BrnoCompSpeed`.

[432] Hyun Oh Song et al. *Online Products Dataset*. 2016. URL: `https://cvgl.stanford.edu/projects/lifted_struct/`.

[433] Jiqiang Song et al. "An object-oriented progressive-simplification-based vectorization system for engineering drawings: model, algorithm, and performance". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 24.8 (2002), pp. 1048–1060.

[434] Peihua Song, Youyi Zheng, and Jinyuan Jia. "Web3d Learning Platform of Furniture Layout Based on Case-Based Reasoning and Distance Field". In: *E-Learning and Games*. Ed. by Feng Tian et al. Springer, 2017, pp. 235–250. ISBN: 978-3-319-65849-0.

[435] Peihua Song et al. "Web3D-based automatic furniture layout system using recursive case-based reasoning and floor field". In: *Multimedia Tools and Applications* 78.4 (2019), pp. 5051–5079. ISSN: 1573-7721.

[436] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. *SUN RGB-D Dataset*. 2016. URL: `https://rgbd.cs.princeton.edu/`.

[437] Shuran Song et al. *SUNCG Dataset*. 2017. URL: `https://sscnet.cs.princeton.edu/`.

[438] Laura South et al. "Effective Use of Likert Scales in Visualization Evaluations: A Systematic Review". In: *Computer Graphics Forum* 41.3 (2022), pp. 43–55. DOI: `https://doi.org/10.1111/cgf.14521`.

[439] Ryan Spick, Simon Demediuk, and James Walker. "Naive Mesh-to-Mesh Coloured Model Generation using 3D GANs". In: *ACSW '20: Proceedings of the Australasian Computer Science Week Multiconference* (2020), pp. 1–6. ISSN: 21531633.

[440] Ryan Spick and James Alfred Walker. "Realistic and textured terrain generation using GANs". In: *Proceedings - CVMP 2019: 16th ACM SIGGRAPH European Conference on Visual Media Production* (2019), pp. 1–10.

[441] Fotios Spyridonis and Damon Daylamani-Zad. "A serious game for raising designer awareness of web accessibility guidelines". In: *Human-Computer Interaction–INTERACT 2019: 17th IFIP TC 13 International Conference, Paphos, Cyprus, September 2–6, 2019, Proceedings, Part I 17*. Springer. 2019, pp. 3–12.

[442] Misha Sra et al. "Procedurally Generated Virtual Reality from 3D Reconstructed Physical Space". In: *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology* (2016), pp. 191–200.

[443] Srinath Sridhar et al. *ShapeNetCOCO*. 2019. URL: `https://github.com/drsrinathsridhar/xnocs/tree/master/dataset`.

[444] Srinath Sridhar et al. *ShapeNetPlain*. 2019. URL: `https://github.com/drsrinathsridhar/xnocs/tree/master/dataset`.

[445] Statista. *Global revenue of selected entertainment industry sectors*. Accessed: 2024-09-11. 2024. URL: `https://www.statista.com/chart/22392/global-revenue-of-selected-entertainment-industry-sectors/`.

[446] Statista. *Number of creative staff in video game studios in the United Kingdom (UK)*. Accessed: 2024-09-11. 2024. URL: `https://www.statista.com/statistics/273433/number-of-creative-staff-in-video-game-studios-in-the-united-kingdom-uk/`.

[447] Statista. *Video Games - Worldwide*. `https://www.statista.com/outlook/dmo/digital-media/video-games/worldwide`. [Accessed 11-09-2024]. 2024.

[448] Margaret-Anne Storey et al. "Guidelines for Using Mixed and Multi Methods Research in Software Engineering". In: *arXiv preprint arXiv:2404.06011* (2024).

[449] Wanchao Su et al. "Interactive Sketch-Based Normal Map Generation with Deep Neural Networks". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–17.

[450] Adam Summerville et al. "Procedural content generation via machine learning (PCGML)". In: *IEEE Trans. Games* 10.3 (2018), pp. 257–270.

[451] Adam James Summerville et al. *VGLC Dataset*. 2016. URL: `https://github.com/TheVGLC/TheVGLC`.

[452] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *CoRR* abs/1707.02968 (2017).

[453] Cheng Sun, Yiran Zhou, and Yunsong Han. "Automatic generation of architecture facade for historical urban renovation using generative adversarial network". In: *Building and Environment* 212 (2022), p. 108781. ISSN: 03601323.

[454] Chunyi Sun et al. *3D-GPT: Procedural 3D Modeling with Large Language Models*. 2023. arXiv: `2310.12945 [cs.CV]`.

[455] Xiao Sun and Zhouhui Lian. "EasyMesh: An efficient method to reconstruct 3D mesh from a single image". In: *Computer Aided Geometric Design* 80 (2020), p. 101862. ISSN: 01678396.

[456] Xingyuan Sun et al. *Pix3D Dataset*. 2018. URL: `http://pix3d.csail.mit.edu/`.

[457] Vinitra Swamy et al. "Multimodn—multimodal, multi-task, interpretable modular networks". In: *Advances in Neural Information Processing Systems* 36 (2024).

[458] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[459] Qingyang Tan et al. “Variational autoencoders for deforming 3d mesh models”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5841–5850.

[460] Jiapeng Tang et al. *DiffuScene: Denoising Diffusion Models for Generative Indoor Scene Synthesis*. 2024. arXiv: `2303.14207 [cs.CV]`.

[461] Unity Technologies. URL: `https://assetstore.unity.com/`.

[462] Edward Teng and Rafael Bidarra. “A semantic approach to patch-based procedural generation of urban road networks”. In: *FDG '17: Proceedings of the 12th International Conference on the Foundations of Digital Games* (2017), pp. 1–10.

[463] Catalina Tobon-Gomez and Arjan Geers. *LASC*. 2013. URL: `https://github.com/catactg/lasc`.

[464] Julian Togelius, Noor Shaker, and Mark J. Nelson. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016. Chap. 1. DOI: `10.1007/978-3-319-42716-4`.

[465] Julian Togelius et al. “Search-Based Procedural Content Generation: A Taxonomy and Survey”. In: *IEEE Trans. Computational Intelligence and AI in Games* 3.3 (2011), pp. 172–186.

[466] Jonathan Tompson et al. *NYU Hand Pose*. 2014. URL: `https://jonathantompson.github.io/NYU_Hand_Pose_Dataset.htm#overview`.

[467] Fei Tong et al. “X-ray2Shape: Reconstruction of 3D Liver Shape from a Single 2D Projection Image”. In: *42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)* (2020), pp. 1608–1611. ISSN: 1557170X.

[468] Trimble. *SketchUp 3D Warehouse*. 2022. URL: `https://3dwarehouse.sketchup.com/`.

[469] Turbosquid. *Turbosquid*. 2024. URL: `https://www.turbosquid.com/`.

[470] Radim Tyleček and Radim Šára. *CMP Dataset*. 2015. URL: `https://cmp.felk.cvut.cz/$\sim$tylecr1/facade/`.

[471] Unity Technologies. *Gaming Report*. Accessed: 2024-09-11. 2024. URL: `https://unity.com/resources/gaming-report?isGated=false`.

[472] Unity Technologies. *Muse*. 2024. URL: `https://unity.com/products/muse`.

[473] Unity Technologies. *Signed Distance Fields in the Visual Effect Graph*. 2024. URL: `https://docs.unity3d.com/Packages/com.unity.visualeffectgraph\@12.0/manual/sdf-in-vfx-graph.html`.

[474] Unity Technologies. *Unity Barracuda*. 2021. URL: `https://github.com/Unity-Technologies/barracuda-release` (visited on 06/14/2021).

[475] Unity Technologies. *Unity Engine*. 2024. URL: `https://unity.com/` (visited on 07/17/2024).

[476] USGS EROS Center. “SRTM Dataset”. In: (2018). URL: `https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-shuttle-radar-topography-mission-srtm-1#overview`.

[477] Ziya Usta, Alper Tunga Akın, and Çetin Cömert. "Deep learning aided web-based procedural modelling of LOD2 city models". In: *Earth Science Informatics* 16.3 (2023), pp. 2559–2571. ISSN: 1865-0481. DOI: `10.1007/s12145-023-01053-0`.

[478] Gül Varol et al. *SURREAL Dataset*. 2017. URL: `https://www.di.ens.fr/willow/research/surreal/data/`.

[479] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[480] Abbhinav Venkat et al. "Humanmeshnet: Polygonal mesh recovery of humans". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[481] Pascal Vincent et al. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *J. Mach. Learn. Res.* 11 (2010), 3371–3408. ISSN: 1532-4435.

[482] Yael Vinker et al. "CLIPasso: semantically-aware object sketching". In: *ACM Trans. Graph.* 41.4 (July 2022). ISSN: 0730-0301. DOI: `10.1145/3528223.3530068`.

[483] Sebastian Von Mammen and Christian Jacob. "The evolution of swarm grammars- growing trees, crafting art, and bottom-up design". In: *IEEE Computational Intelligence Magazine* 4.3 (2009), pp. 10–19.

[484] John Von Neumann, Arthur W Burks, et al. "Theory of self-reproducing automata". In: *IEEE Trans. Neural Networks* 5.1 (1966), pp. 3–14.

[485] Paul Waddell. "UrbanSim: Modeling urban development for land use, transportation, and environmental planning". In: *Journal of the American planning association* 68.3 (2002), pp. 297–314.

[486] Catherine Wah et al. "The Caltech-UCSD Birds-200-2011 Dataset". In: *California Institute of Technology* (2011). URL: `http://www.vision.caltech.edu/datasets/cub_200_2011/`.

[487] Sean P Walton, Alma AM Rahat, and James Stovold. "Evaluating mixed-initiative procedural level design tools using a triple-blind mixed-method user study". In: *IEEE Transactions on Games* 14.3 (2021), pp. 413–422.

[488] Guan Wang et al. "The shape space of 3D botanical tree models". In: *ACM Trans. on Graphics* 37.1 (2018), pp. 1–18. ISSN: 15577368.

[489] Hao Wang et al. "Global-to-local generative model for 3D shapes". In: *ACM Trans. Graph.* 37.6 (2018), pp. 1–10. ISSN: 0730-0301.

[490] Kai Wang et al. "PlanIT: planning and instantiating indoor scenes with relation graph and spatial prior networks". In: *ACM Trans. Graph.* 38.4 (2019). ISSN: 0730-0301.

[491] Meng Wang and Hongkai Wang. "Facial Photo-Guided Head Anatomy Modeling Based on Deep Learning and 2D/3D Shape Prior Model Registration". In: *The World Conference on Intelligent and 3D Technologies*. Springer. 2023, pp. 247–257.

[492] Nanyang Wang et al. “Pixel2Mesh: 3D Mesh Model Generation via Image Guided Deformation”. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 43.10 (2021), pp. 3600–3613. ISSN: 19393539.

[493] Nanyang Wang et al. “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”. In: *Lecture Notes in Computer Science* 11215 LNCS (2018), pp. 55–71. ISSN: 16113349. arXiv: 1804.01654.

[494] Peng-Shuai Wang et al. “Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes”. In: *ACM Trans. Graph.* 37.6 (2018), pp. 1–11. ISSN: 0730-0301.

[495] Tengfei Wang et al. “RODIN: A Generative Model for Sculpting 3D Digital Avatars Using Diffusion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 4563–4573.

[496] Tong Wang and Shuichi Kurabayashi. “Sketch2Map: A Game Map Design Support System Allowing Quick Hand Sketch Prototyping”. In: *IEEE Conference on Games (CoG)* (2020), pp. 596–599. ISSN: 23254289.

[497] Xiaogang Wang and Xiaoou Tang. “CUHK Face Sketch”. In: (2009). URL: http://mmlab.ie.cuhk.edu.hk/archive/facesketch.html.

[498] Xiaolong Wang and Abhinav Gupta. “Generative Image Modeling Using Style and Structure Adversarial Networks”. In: *LNCS*. Lecture Notes in Computer Science 9908 (2016). Ed. by Bastian Leibe et al., pp. 318–335. ISSN: 16113349.

[499] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. “SceneFormer: Indoor Scene Generation with Transformers”. In: *2021 International Conference on 3D Vision (3DV)*. 2021, pp. 106–115.

[500] Yaohui Wang, Antitza Dantcheva, and Francois Bremond. “From Attribute-Labels to Faces: Face Generation Using a Conditional Generative Adversarial Network”. In: *Computer Vision – ECCV 2018 Workshops*. Ed. by Laura Leal-Taixé and Stefan Roth. Springer, 2019, pp. 692–698. ISBN: 978-3-030-11018-5.

[501] Yaqing Wang et al. “Generalizing from a Few Examples: A Survey on Few-Shot Learning”. In: *ACM Comput. Surv.* 53.3 (2020). ISSN: 0360-0300.

[502] Yifan Wang, Zichun Zhong, and Jing Hua. “DeepOrganNet: On-the-Fly Reconstruction and Visualization of 3D / 4D Lung Models from Single-View Projections by Deep Deformation Network”. In: *IEEE Trans. Visualization and Computer Graphics* 26.1 (2020), pp. 960–970. ISSN: 19410506. arXiv: 1907.09375.

[503] Yunhai Wang et al. *ShapeCOSEG Dataset*. 2012. URL: http://irc.cs.sdu.edu.cn/$\sim$yunhai/public_html/ssl/ssd.htm%0A.

[504] Zhongling Wang, Zhenzhong Chen, and Feng Wu. “Thermal to visible facial image translation using generative adversarial networks”. In: *IEEE Signal Processing Letters* 25.8 (2018), pp. 1161–1165.

[505] Zhou Wang. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[506] Webzen. *Metin2*. 2004.

[507] Webzen. *MU Online*. 2003.

[508] Chao Wen et al. *Pixel2Mesh++: 3D Mesh Generation and Refinement from Multi-View Images*. 2022.

[509] Zhenzhen Weng and Serena Yeung. "Holistic 3D Human and Scene Mesh Estimation from Single View Images". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2021), pp. 334–343. ISSN: 10636919. arXiv: 2012.01591.

[510] Contributors to Diablo Wiki. *Diablo II Sword Images*. URL: https://diablo-archive.fandom.com/wiki/Category:Diablo_II_Sword_Images.

[511] Contributors to Diablo Wiki. *Diablo III Sword icons*. URL: https://diablo-archive.fandom.com/wiki/Category:Diablo_III_sword_icons.

[512] Contributors to Final Fantasy Wiki. *Final Fantasy IX Weapons*. URL: https://finalfantasy.fandom.com/wiki/Final_Fantasy_IX_weapons.

[513] Frank Wilcoxon. "Individual comparisons by ranking methods". In: *Breakthroughs in statistics*. New York, NY: Springer, 1992, pp. 196–202.

[514] Andrew R. Willis et al. "Volumetric procedural models for shape representation". In: *Graphics and Visual Computing* 4 (2021), p. 200018. ISSN: 26666294.

[515] Jelmer M. Wolterink et al. *orCaScore Dataset*. 2016. URL: https://orcascore.grand-challenge.org/Data/.

[516] Steven Worley. "A Cellular Texture Basis Function". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. ACM, 1996, 291–294. ISBN: 0897917464.

[517] Wenming Wu et al. "Data-driven interior plan generation for residential buildings". In: *ACM Trans. Graph.* 38.6 (Nov. 2019). ISSN: 0730-0301. DOI: 10.1145/3355089.3356556.

[518] Zhennan Wu et al. "BlockFusion: Expandable 3D Scene Generation using Latent Tri-plane Extrapolation". In: *ACM Trans. Graph.* 43.4 (July 2024). ISSN: 0730-0301. DOI: 10.1145/3658188.

[519] Zhirong Wu et al. *ModelNet Dataset*. 2015. URL: https://modelnet.cs.princeton.edu/.

[520] Weihao Xia, Yujiu Yang, and Jing-Hao Xue. "Cali-sketch: Stroke calibration and completion for high-quality face image generation from human-like sketches". In: *Neurocomputing* 460 (2021), pp. 256–265. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.07.029.

[521] Nan Xiang et al. "Single-image Mesh Reconstruction and Pose Estimation via Generative Normal Map". In: *Proceedings of the 32nd International Conference on Computer Animation and Social Agents* (2019), pp. 79–84.

[522] Jianxiong Xiao et al. "Sun database: Large-scale scene recognition from abbey to zoo". In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 3485–3492.

[523] Haozhe Xie et al. “Weighted Voxel: a novel voxel representation for 3D reconstruction”. In: *Proceedings of the 10th International Conference on Internet Multimedia Computing and Service - ICIMCS '18* (2018), pp. 1–4.

[524] Bolai Xin and Mark Whitty. “A 3D grape bunch reconstruction pipeline based on constraint-based optimisation and restricted reconstruction grammar”. In: *Computers and Electronics in Agriculture* 196 (2022), p. 106840. ISSN: 0168-1699. DOI: `https://doi.org/10.1016/j.compag.2022.106840`.

[525] Weidan Xiong et al. “Shape-Inspired Architectural Design”. In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '18. Montreal, Quebec, Canada: ACM, 2018. ISBN: 9781450357050.

[526] Hao Xu and Jing Bai. “ARShape-Net: Single-View Image Oriented 3D Shape Reconstruction with an Adversarial Refiner”. In: *Lecture Notes in Computer Science*. Vol. 13069 LNAI. Springer, 2021, pp. 638–649. ISBN: 9783030930455.

[527] Jiale Xu et al. “Dream3D: Zero-Shot Text-to-3D Synthesis Using 3D Shape Prior and Text-to-Image Diffusion Models”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 20908–20918.

[528] Nand Kumar Yadav, Satish Kumar Singh, and Shiv Ram Dubey. “CSA-GAN: Cyclic synthesized attention guided generative adversarial network for face synthesis”. In: *Applied Intelligence* (2022). ISSN: 0924-669X. DOI: `10.1007/s10489-021-03064-0`.

[529] Guandao Yang et al. “Pointflow: 3d point cloud generation with continuous normalizing flows”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 4541–4550.

[530] Jie Yang et al. “DSG-Net: Learning Disentangled Structure and Geometry for 3D Shape Generation”. In: *ACM Trans. Graph.* 42.1 (2022). ISSN: 0730-0301.

[531] Jinglun Yang, Youhua Li, and Lu Yang. “Shape transformer nets: Generating viewpoint-invariant 3D shapes from a single image”. In: *Journal of Visual Communication and Image Representation* 81 (2021), p. 103345. ISSN: 10959076.

[532] Kaizhi Yang et al. “Deep 3D Modeling of Human Bodies from Freehand Sketching”. In: *MultiMedia Modeling*. Ed. by Jakub Lokoč et al. Springer, 2021, pp. 36–48. ISBN: 978-3-030-67835-7.

[533] Yipin Yang et al. *SPRING Dataset*. 2014. URL: `https://graphics.soe.ucsc.edu/data/BodyModels/index.html`.

[534] Li Yi et al. *ShapeNetPart Dataset*. 2016. URL: `https://cs.stanford.edu/$\sim$ericyi/project_page/part_annotation/`.

[535] Dumim Yoon and Kyung Joong Kim. “3D game model and texture generation using interactive genetic algorithm”. In: *Computers in Entertainment* 14.1 (2017), pp. 1–16. ISSN: 15443981.

[536] Aron Yu and Kristen Grauman. *UT Zappos50K Dataset*. 2017. URL: `https://vision.cs.utexas.edu/projects/finegrained/utzap50k/`.

[537] Fisher Yu et al. *LSUN Dataset*. 2015. URL: https://www.yf.io/p/lsun.

[538] Hang Yu et al. "Multi-View Shape Generation for a 3D Human-like Body". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 19.1 (2023). ISSN: 1551-6857.

[539] Jiahui Yu et al. *Scaling Autoregressive Models for Content-Rich Text-to-Image Generation*. 2022.

[540] Shanxin Yuan et al. *BigHand2.2M Dataset*. 2017. URL: https://sites.google.com/site/qiyeincv/home/bibtex_cvpr2017.

[541] Ye Yuan, Yasuaki Ito, and Koji Nakano. "Art Font Image Generation with Conditional Generative Adversarial Networks". In: *Proceedings - 2020 8th International Symposium on Computing and Networking Workshops, CANDARW 2020* (2020), pp. 151–156.

[542] Ke Yue, Yidong Li, and Huifang Li. "Progressive semantic image synthesis via generative adversarial network". In: *2019 IEEE International Conference on Visual Communications and Image Processing, VCIP 2019* (2019), pp. 1–4.

[543] Anny Yuniarti, Agus Zainal Arifin, and Nanik Suciati. "A 3D template-based point generation network for 3D reconstruction from single images". In: *Applied Soft Computing* 111 (2021), p. 107749. ISSN: 15684946.

[544] Huayi Zeng, Jiaye Wu, and Yasutaka Furukawa. "Neural Procedural Reconstruction for Residential Buildings". In: *Lecture Notes in Computer Science* 11207 (2018), pp. 759–775. ISSN: 16113349.

[545] Xiaohui Zeng et al. "LION: Latent Point Diffusion Models for 3D Shape Generation". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[546] Xiaohua Zhai et al. "Scaling vision transformers". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 12104–12113.

[547] Fan Zhang et al. "Mesh deformation-based single-view 3D reconstruction of thin eyeglasses frames with differentiable rendering". In: *Graphical Models* 135 (2024), p. 101225. ISSN: 1524-0703. DOI: https://doi.org/10.1016/j.gmod.2024.101225.

[548] Hongkun Zhang et al. "TexPainter: Generative Mesh Texturing with Multi-view Consistency". In: *ACM SIGGRAPH 2024 Conference Papers*. SIGGRAPH '24. Denver, CO, USA: Association for Computing Machinery, 2024. ISBN: 9798400705250. DOI: 10.1145/3641519.3657494.

[549] Jian Zhang et al. "Example-based rapid generation of vegetation on terrain via CNN-based distribution learning". In: *Visual Computer* 35.6-8 (2019), pp. 1181–1191. ISSN: 01782789.

[550] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. "Adding Conditional Control to Text-to-Image Diffusion Models". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 3813–3824. DOI: 10.1109/ICCV51070.2023.00355.

[551] Meng Zhang and Y. Zheng. "Hair-GAN: Recovering 3D hair structure from a single image using generative adversarial networks". In: *Visual Informatics* 3.2 (2019), pp. 102–112. ISSN: 2468502X.

[552] Meng Zhang et al. "Modeling hair from an RGB-D camera". In: *ACM Trans. Graph.* 37.6 (2018), pp. 1–10. ISSN: 0730-0301.

[553] Qimeng Zhang et al. "Shrubbery-shell inspired 3D model stylization". In: *Computers and Graphics (Pergamon)* 82 (2019), pp. 13–21. ISSN: 00978493.

[554] Richard Zhang, Phillip Isola, and Alexei A. Efros. "Colorful Image Colorization". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 649–666. ISBN: 978-3-319-46487-9.

[555] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[556] Shangzhan Zhang et al. "MaPa: Text-driven Photorealistic Material Painting for 3D Shapes". In: *ACM SIGGRAPH 2024 Conference Papers*. SIGGRAPH '24. Denver, CO, USA: Association for Computing Machinery, 2024. ISBN: 9798400705250. DOI: `10.1145/3641519.3657504`.

[557] Sizhuo Zhang and Nanfeng Xiao. "Detailed 3D Human Body Reconstruction From a Single Image Based on Mesh Deformation". In: *IEEE Access* 9 (2021), pp. 8595–8603.

[558] Song-Hai Zhang, Yuan-Chen Guo, and Qing-Wen Gu. "Sketch2model: View-aware 3d modeling from single free-hand sketches". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6012–6021.

[559] Wenbo Zhang. "Sketch-To-Color Image with GANs". In: *Proc. 2nd Int. Conf. on Information Technology and Computer Application (ITCA)*. 2020, pp. 322–325. ISBN: 9780738111414.

[560] Zaiwei Zhang et al. "Deep Generative Modeling for Scene Synthesis via Hybrid Representations". In: *ACM Trans. Graph.* 39.2 (2020). ISSN: 0730-0301.

[561] Zhiqiang Zhang et al. "Customizable GAN : Customizable Image Synthesis Based on Adversarial Learning". In: *Communications in Computer and Information Science* 1332 (2020), pp. 336–344.

[562] Meihua Zhao et al. "3D-RVP: A method for 3D object reconstruction from a single depth view using voxel and point". In: *Neurocomputing* 430 (2021), pp. 94–103. ISSN: 18728286.

[563] Yiru Zhao et al. "Stylized adversarial autoencoder for image generation". In: *MM 2017 - Proceedings of the 2017 ACM Multimedia Conference* (2017), pp. 244–251.

[564] Jia Zheng et al. "Structured3d: A large photo-realistic dataset for structured 3d modeling". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*. Springer. 2020, pp. 519–535.

[565] Tianyue Zheng and Weihong Deng. "Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments". In: *Beijing University of Posts and Telecommunications, Tech. Rep* 5.7 (2018), p. 5.

[566] Tianyue Zheng, Weihong Deng, and Jiani Hu. *CALFW Dataset*. 2018. URL: `http://whdeng.cn/CALFW/index.html?reload=true`.

[567] X. Zheng et al. "SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation". In: *Computer Graphics Forum* 41.5 (2022), pp. 52–63.

[568] Xinru Zheng et al. "Content-aware generative modeling of graphic design layouts". In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–15.

[569] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. "Publaynet: largest dataset ever for document layout analysis". In: *2019 International conference on document analysis and recognition (ICDAR)*. IEEE. 2019, pp. 1015–1022.

[570] Bolei Zhou et al. *Places2 Dataset*. 2017. URL: `http://places2.csail.mit.edu/`.

[571] Qian-Yi Zhou and Vladlen Koltun. *3D Scene Dataset*. 2013. URL: `https://qianyi.info/scenedata.html`.

[572] Qingnan Zhou and Alec Jacobson. *Thingi10K*. 2016. URL: `https://ten-thousand-models.appspot.com/`.

[573] Hao Zhu et al. *RESY Dataset*. 2019. URL: `https://github.com/zhuhao-nju/hmd/blob/master/datasets/data.md`.

[574] Jun-Yan Zhu et al. *CycleGAN Datasets*. 2017. URL: `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/blob/master/docs/datasets.md`.

[575] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[576] Peihao Zhu et al. "Large-scale architectural asset extraction from panoramic imagery". In: *IEEE Transactions on Visualization and Computer Graphics* 28.2 (2020), pp. 1301–1316.

[577] Xiangyu Zhu et al. *300W-LP Dataset*. 2016. URL: `http://www.cbsr.ia.ac.cn/users/xiangyuzhu/projects/3DDFA/main.htm`.

[578] Xiangyu Zhu et al. "AFLW2000-3D Dataset". In: (2016). URL: `http://www.cbsr.ia.ac.cn/users/xiangyuzhu/projects/3DDFA/main.htm`.

[579] Xiahai Zhuang and Juan Shen. *MMWHS Dataset*. 2016. URL: `http://www.sdspeople.fudan.edu.cn/zhuangxiahai/0/mmwhs/data.html`.

[580] Christian Zimmermann et al. *FreiHAND Dataset*. 2019. URL: `https://lmb.informatik.uni-freiburg.de/projects/freihand/`.

[581] Silvia Zuffi et al. *SMAL Dataset*. 2017. URL: `https://smal.is.tue.mpg.de/`.

[582] J Ďurech, V Sidorin, and M Kaasalainen. *DAMIT Dataset*. 2022. URL: `https://astro.troja.mff.cuni.cz/projects/damit/`.

Appendix A: Literature review data

TABLE A1: All accepted GAG literature

Title	Asset type	Approach	Technique interaction	Technique process	Testing Type	Evaluation Process	Metrics	Datasets
[563] Stylized Adversarial AutoEncoder for Image Generation	2D General	AAE, CNN	Photo-wise	Style transferred	Comparative assessment	Metric analysis (Quantitative), Visual quality comparison, Ablation study	Log-Likelihood	LFW, IIIT 5K-word
[560] Deep Generative Modeling for Scene Synthesis via Hybrid Representations	OP Interior	AAE, CNN	Asset-wise	Arranged	Assessment	Questionnaire (Quantitative)	N/A	SUNCG
[418] 3D Point Cloud Generation Using Adversarial Training for Large-Scale Outdoor Scene	3D General	Autoencoder	Random seed	Random	Assessment	Qualitative observations	N/A	DFC2022
[345] LPMNet: Latent part modification and generation for 3D point clouds	3D General, Vehicles, Furniture	Autoencoder	Asset-wise	Interpolated, Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Minimum matching distance, Coverage, JSD, Chamfer distance, EMD	ShapeNetPart

[229] Object Synthesis by Learning Part Geometry with Surface and Volumetric Representations	3D General, Vehicles, Furniture, Props	Autoencoder	Asset-wise	Interpolated	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	Loss	ModelNet
[329] StructureNet: hierarchical graph networks for 3D shape generation	Furniture	Autoencoder	Asset-wise, Random seed	Random, Interpolated	Comparative assessment	Metric analysis (Quantitative), Qualitative Observations	Error	PartNet
[412] ClipGen: A Deep Generative Model for Clipart Vectorization and Synthesis	2D Icons	Autoencoder	Sketch-wise	Reconstructed	Assessment	Ablation study, Metric analysis (Quantitative)	Image distance, Chamfer distance, Earth Mover's loss	ClipNet
[289] Generating Point Cloud from Single Image in The Few Shot Scenario	3D General, Furniture, Vehicles	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Ablation study (Quantitative)	Chamfer distance, EMD	ModelNet, ShapeNet
[292] A Deep Learning Method Based on Structure Inference for Single-view 3D Reconstruction	3D General, Vehicles, Furniture	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	IoU, Chamfer distance	ShapeNet

[367] Image2Mesh: A Learning Framework for Single Image 3D Reconstruction	3D General, Vehicles, Furniture	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Accuracy, Precision, Recall, MSE, IoU, Surface distance	ShapeNet
[261] Pix2Surf: Learning Parametric 3D Surface Models of Objects from Images	3D General, Vehicles, Furniture	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitataive observations	Reconstruction error, Correspondence error, Consistency error, Discontinuity score	ShapeNetPlain, ShapeNet-COCO
[543] A 3D template-based point generation network for 3D reconstruction from single images	3D General, Vehicles, Furniture, Props	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	Classification accuracy, Chamfer distance	ShapeNet, ImageNet
[235] Discrete Point Flow Networks for Efficient Point Cloud Generation	3D General, Vehicles, Furniture, Props	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Speed test, Metric analysis (Quantitative), Qualitative observations	JSD, Minimum matching distance, Coverage, 1-nearest neighbour accuracy	ShapeNet

[531] Shape transformer nets: Generating viewpoint-invariant 3D shapes from a single image	3D General, Vehicles, Props, Furniture	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Chamfer distance, EMD	ShapeNet
[562] 3D-RVP: A method for 3D object reconstruction from a single depth view using voxel and point	Furniture	Autoencoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	IoU, Cross-entropy loss	ShapeNet
[268] GRAINS: Generative Recursive Autoencoders for INdoor Scenes	OP Interior	Autoencoder	Asset-wise	Arranged	Comparative assessment	Speed test, Qualitative observations, Questionnaire (Quantitative)	Speed	SUNCG
[369] SG-VAE: Scene Grammar Variational Autoencoder to Generate New Indoor Scenes	OP Interior	Autoencoder	Photo-wise, Random seed	Arranged	Comparative assessment	Ablation study, Metric analysis (Quntitative), Qualitative observations	IoU, Angular error, Displacement error	SUN RGB-D
[435] Web3D-based automatic furniture layout system using recursive case-based reasoning and floor field	OP Interior	Case-based reasoning	Asset-wise	Arranged	Comparative assessment	Metric analysis (Quantitative), Questionnaire (Quantitative)	Speed, Layout accuracy	N/A
[434] Web3d Learning Platform of Furniture Layout Based on Case-Based Reasoning and Distance Field	OP Interior	Case-based reasoning	Asset-wise	Arranged	Comparative assessment	Speed test, Qualitative observations	Speed	N/A

[80] ProcTHOR: Large-Scale Embodied AI Using Procedural Generation	OP Interior	Case-based reasoning	Random seed	Arranged	Comparative assessment	Metric analysis (Quantiative), Ablation study	Speed	PROCTHOR, ARCHITECTHOR, RoboTHOR, HM3D, AI2-iTHOR
[119] Voxel-Based Three-Dimensional Neural Style Transfer	3D General, Props, Vehicles	CNN	Asset-wise	Style transferred	Assessment	Qualitative observations, Metric analysis (Quantitative)	Distance matrix, EMD	ModelNet
[416] Face-to-Parameter Translation for Game Character Auto-Creation	Face	CNN	Photo-wise	Guided	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	Mode score, FID, Speed	Helen
[87] Accurate 3D Face Reconstruction With Weakly-Supervised Learning: From Single Image to Image Set	Face	CNN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Loss, RMSE	CelebA, 300W-LP, I-JBA, LFW, LS3D
[491] Facial Photo-Guided Head Anatomy Modeling Based on Deep Learning and 2D/3D Shape Prior Model Registration	Face	CNN	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	RMSE	N/A
[312] TreeSketchNet: From Sketch to 3D Tree Parameters Generation	Trees	CNN	Sketch-wise	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	Hausdorf distance, RMSE	N/A

[82] 3D Sketching using Multi-View Deep Volumetric Prediction	3D General	CNN, Encoder-Decoder	Sketch-wise	Reconstructed	Assessment	Tutorial, Recorded interaction, Questionnaire (Mixed)	IoU, Speed	ShapeCOSEG
[351] Deep Mesh Reconstruction From Single RGB Images via Topology Modification Networks	3D General, Furniture, Vehicles	CNN, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Chamfer distance, EMD	ShapeNet
[309] HandVoxNet: Deep Voxel-Based Network for 3D Hand Shape and Pose Estimation From a Single Depth Map	Characters	CNN, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Joint error, Vertex error, Voxelized shape error	SynHand5M, NYU Hand Pose, BigHand2.2M
[521] Single-image Mesh Reconstruction and Pose Estimation via Generative Normal Map	3D General, Vehicles, Furniture, Normal map	CNN, Encoder-Decoder, GAN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative)	IoU, Normal distance	ShapeNet
[467] X-ray2Shape: Reconstruction of 3D Liver Shape from a Single 2D Projection Imag	Medical	CNN, GCN	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	Mean distance, Euclidean distance	N/A

[265] Saliency Guided Subdivision for Single-View Mesh Reconstruction	3D General, Furniture, Vehicles, Props	CNN, GCN, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	F-score, Mesh intersection ratio	ShapeNet
[182] ArrangementNet: Learning Scene Arrangements for Vectorized Indoor Scene Modeling	Buildings	CNN, MLP	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Precision, Recall, RMSE, CD, IOU, Coverage	Structure3D, Floor-SP
[352] Residual MeshNet: Learning to Deform Meshes for Single-View 3D Reconstruction	3D General, Vehicles, Props, Furniture	CNN, MLP, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Ablation study	Chamfer distance	ShapeNet
[341] Procedural modeling of a building from a single image	Buildings	CNN, Shape Grammar	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Speed, Distance metric, Accuracy, RMSE	CMP, ImageNet, SUN, ECP
[181] Shape Synthesis from Sketches via Procedural Models and Convolutional Networks	3D General, Props, Trees	CNNs	Sketch-wise	Guided	Comparative assessment	Recorded Interaction, Metric analysis (Quantitative), Qualitative observation	Accuracy, Error	ImageNet

[417] Neural Rendering for Game Character Auto-Creation	Face	CNNs	Photo-wise	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	Speed, Accuracy	CelebA, LFW, CFPW, AgeDB, CALFW, CPLFW, VGGFace2
[139] Mesh R-CNN	Furniture	CNNs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Chamfer distance, Normal consistency, Precision, Recall	ShapeNet, Pix3D
[83] Combining voxel and normal predictions for multi-view 3D sketching	3D General, Furniture, Props, Vehicles	CNNs, Encoder-Decoder	Sketch-wise	Reconstructed	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	Speed	3D Sketching using Multi-View Deep Volumetric Prediction
[494] Adaptive O-CNN: a patch-based deep representation of 3D shapes	3D General, Props, Vehicles	CNNs, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Ablation study (Quantitative)	Memory usage, Speed, Chamfer distance	ModelNet, ShapeNet

[480] HumanMeshNet: Polygonal Mesh Recovery of Humans	Characters	CNNs, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Surface error, Joint error, Procrustes analysis	SURREAL, UP-3D, Human3.6M
[226] Learning-detailed 3D face reconstruction based on convolutional neural networks from a single image.	Face	CNNs, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	Mean absolute error, RMSE	AFLW, 300W, Multi-pie, VGG Face, FRGC
[502] DeepOrganNet: On-the-Fly Reconstruction and Visualization of 3D / 4D Lung Models from Single-View Projections by Deep Deformation Network	Medical	CNNs, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Qualitative feedback	Chamfer distance, EMD, Hausdorff distance, F-score, IoU	N/A
[549] Example-based rapid generation of vegetation on terrain via CNN-based distribution learning	OP Exterior	CNNs, Encoder-Decoder	Random seed	Arranged	Assessment	Qualitative observations, Metric analysis (Quantitative)	Mean absolute error, Memory usage	NASA SRTM
[557] Detailed 3D Human Body Reconstruction From a Single Image Based on Mesh Deformation	Characters	CNNs, Encoder-Decoder, GCN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Mean per joint position error, Quaternion distance error, Joint error	Human3.6M, UP-3D, RESY

[123] 3D Mesh Generation by Introducing Extended Attentive Normalization	3D General	CNNs, GAN	Textual	Guided	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	FID	CUB-200-2011
[92] 3D Reconstruction based on GAT from a Single Image	3D General, Vehicles, Furniture	CNNs, GCN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative)	IoU, Chamfer distance	ShapeNet
[492] Pixel2Mesh: 3D Mesh Model Generation via Image Guided Deformation	3D General, Vehicles, Furniture, Props, Face	CNNs, GCN	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	F-score, Chamfer distance, EMD	ShapeNet, Online products, 300W-LP
[332] Generative Design applied to Cloud Modeling	Clouds	CNNs, Genetic Algorithm	Random seed	Random	Assessment	Metric analysis (Quantitative)	K-fold cross-validation, confusion matrix	Cloud image classification
[509] Holistic 3D Human and Scene Mesh Estimation from Single View Images	3D General, Characters, Furniture	Combination, CNN, R-CNN, Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	IoU, Domain specific	Pigraphs, PROX, LVIS, SUN RGB-D, Pix3D
[307] Functionality preserving shape style transfer	Furniture	Combinatorial optimisation	Asset-wise	Style transferred	Assessment	Questionnaire (Quantitative)	N/A	SketchUp 3D Warehouse

[199] Texture map generation for 3D reconstructed scenes	3D General, Textures	Deformation model	Photo-wise	Reconstructed	Comparative assessment	Speed test, Qualitative observations	Speed	3D Scene
[100] Monster mash: a single-view approach to casual 3D modeling and animation	Characters	Deformation model	Sketch-wise	Guided	Explorative	Tutorial, Interaction, Qualitative Feedback	N/A	N/A
[488] The Shape Space of 3D Botanical Tree Models	Trees	Deformation model	Parametric	Interpolated	Comparative	Questionnaire (Quantitative)	N/A	N/A
[110] Interactive reconstruction of the 3D-models using single-view images and user markup	3D General	Deprojection	Photo-wise	Reconstructed	N/A	N/A	N/A	N/A
[339] Open3DGen: open-source software for reconstructing textured 3D models from RGB-D images	3D General	Deprojection	Photo-wise	Reconstructed	Comparative assessment	Speed test, Metric analysis (Quantitative)	Speed, Camera pose accuracy	EuRoC MAV
[114] Reconstruction of Artifacts from Digital Image Repositories	3D General	Deprojection	Photo-wise	Reconstructed	Assessment	Qualitative observations	N/A	Hagley Museum model data

[428] SDF-2-SDF: Highly Accurate 3D Object Reconstruction	3D General, Props	Deprojection	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Speed, Reconstruction error	3D-Printed RGB-D Object
[423] A 3D modeling methodology based on a concavity-aware geometric test to create 3D textured coarse models from concept art and orthographic projections	3D General, Props, Vehicles, Furniture, Characters	Deprojection	Sketch-wise	Reconstructed	Assessment	Metric analysis (Quantitative), Qualitative observations	Speed, Surface distance	N/A
[31] A morphable model for the synthesis of 3D faces	Face	Deprojection	Photo-wise	Reconstructed	Assessment	Qualitative observations	N/A	N/A
[228] TextureMe: High-Quality Textured Scene Reconstruction in Real Time	Textures	Deprojection	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	PSNR, SSIM, Domain specific	3D Scene, ScanNet
[54] Dynamic Omnidirectional Texture Synthesis for Photorealistic Virtual Content Creation	Textures, 3D General, Furniture, Props	Deprojection	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations	N/A	redwood-3dscan
[524] A 3D grape bunch reconstruction pipeline based on constraint-based optimisation and restricted reconstruction grammar	3D General	Deprojection, Grammar	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative)	Precision, Recall, F-Score	Grape bunches

[155] Realistic Procedural Plant Modeling from Multiple View Images	Trees	Deprojection, L-System, Space Colonisation	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	Domain specific	N/A
[548] TexPainter: Generative Mesh Texturing with Multi-view Consistency	Textures	Diffusion	Textual, Asset-wise	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Speed, FID	N/A
[556] MaPa: Text-driven Photorealistic Material Painting for 3D Shapes	Textures	Diffusion	Textual, Asset-wise	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID, KID	ABO, TMT
[389] Texture: Text-guided texturing of 3d shapes	Textures	Diffusion	Textual, Asset-wise	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	User score, speed	N/A
[518] Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation	OP Exterior	Diffusion, Autoencoder	Asset-wise	Arranged	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CD, Surface normal error, MMD, Coverage, 1-NNA, EMD	3D-FRONT, 3D-FUTURE
[59] Subject-driven text-to-image generation via apprenticeship learning	2D General	Diffusion/Propagation	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	User score, speed, running cost, CLIP Score	DreamBench

[358] Localizing object-level shape variations with text-to-image diffusion models	2D General	Diffusion/Propagation	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	LPIPS, CLIP Score, IOU	N/A
[195] Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models	2D General	Diffusion/Propagation	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	CLIP-Similarity, R-precision	N/A
[284] Magic3d: High-resolution text-to-3d content creation	3D General	Diffusion/Propagation	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Speed, human ranking	N/A
[132] Multi-target 3D Reconstruction from RGB-D Data	3D General, Props	Diffusion/Propagation	Photo-wise	Reconstructed	Assessment	Metric analysis (Quantitative), Qualitative Observations	Speed, Accuracy	N/A
[552] Modeling hair from an RGB-D camera	Hair	Diffusion/Propagation	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations	N/A	N/A
[140] Dijkstra-based Terrain Generation Using Advanced Weight Functions	Terrain	Diffusion/Propagation	Parametric	Guided	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Speed	N/A
[24] Feature-based volumetric terrain generation	Terrain	Diffusion/Propagation, Perlin Noise	Sketch-wise	Guided	Assessment	Qualitative observations, Speed test	Speed	N/A

[390] High-Resolution Image Synthesis with Latent Diffusion Models	2D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	FID, Inception Score, Precision, Recall, PSNR, SSIM, LPIPS	LAION-400M
[375] Learning Transferable Visual Models From Natural Language Supervision	2D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Accuracy	N/A
[397] Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding	2D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID, User score, CLIP Score	COCO, DrawBench
[575] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks	2D General	Encoder-Decoder	Photo-wise	Style transferred	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	User score, FCN, Accuracy, IOU	Cityscapes
[539] Scaling Autoregressive Models for Content-Rich Text-to-Image Generation	2D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID, Automated captioning evaluation, User score	C4, BERT, LAION-400M, FIT400M, JFT-4B

[550] Adding Conditional Control to Text-to-Image Diffusion Models	2D General	Encoder-Decoder	Textual, Sketch-wise	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Human rank, FID, CLIP Score	LAION-5B
[304] Att3d: Amortized text-to-3d object synthesis	2D General	Encoder-Decoder	Textual	Guided	Assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	R-precision, running cost, speed	DF27
[482] Clipasso: Semantically-aware object sketching	2D General	Encoder-Decoder	Photo-wise	Style transferred	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	confusion matrix	N/A
[116] Clipdraw: Exploring text-to-drawing synthesis through language-image encoders	2D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	CLIP-Similarity	N/A
[319] Occupancy Networks	3D General	Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	IoU, CD, Normal Consistency	ShapeNet

[184] Neural Template: Topology-Aware Reconstruction and Disentangled Generation of 3D Meshes	Furniture, Vehicles, Props, 3D General	Encoder-Decoder	Photo-wise, Random seed, Asset-wise	Reconstructed, Interpolated, Style transferred, Random	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Point-to-surface-distance, CD, Light Field Distance	ShapeNet
[368] DreamFusion: Text-to-3D using 2D Diffusion	3D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	R-precision	MS-COCO
[297] SyncDreamer: Generating Multiview-consistent Images from a Single-view Image	3D General	Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	PSNR, SSIM, LPIPS, CD, IoU	Objaverse, Google Scanned Object
[371] Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors	3D General	Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	PSNR, LPIPS, CLIP-similarity	NeRF4, RealFusion15
[527] Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models	3D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID, CLIP precision	LAION-5B

[179] Mesh-controllable multi-level-of-detail text-to-3D generation	3D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CLIP-Similarity, CD, speed, memory usage, user score	N/A
[160] Text-image conditioned diffusion for consistent text-to-3D generation	3D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CLIP-Similarity	T3Bench
[152] Diverse part synthesis for 3D shape creation	3D General	Encoder-Decoder	Asset-wise	Guided	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Euclidean distance, CD, EMD	ShapeNet
[58] Deep3DSketch-im: rapid high-fidelity AI 3D model generation by single freehand sketches	3D General	Encoder-Decoder	Sketch-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	CD, user score	ShapeNet-Synthetic, ShapeNet-Sketch
[62] SDFusion: Multimodal 3D Shape Completion, Reconstruction, and Generation	3D General	Encoder-Decoder	Textual, Photo-wise, Asset-wise	Guided, Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Uni-directional Hausdorff Distance, Mutual difference, CD, F-score	Pix3D, ShapeGlot, Text2Shape

[216] Neural 3D Mesh Renderer	3D General, Props, Furniture, Vehicles	Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	IoU	ShapeNet
[57] Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation	3D General, Textures	Encoder-Decoder	Textual	Guided	Comparative assessment	Qualitative observations, Ablation study	N/A	LAION-5B
[320] Latent-nerf for shape-guided generation of 3d shapes and textures	3D General, Textures	Encoder-Decoder	Sketch-wise, Textual	Guided	Comparative assessment	Qualitative observations	N/A	N/A
[37] Text-to-building: experiments with AI-generated 3D geometry for building design and structure generation	Buildings	Encoder-Decoder	Textual	Guided	N/A	N/A	N/A	N/A
[495] Rodin: A generative model for sculpting 3d digital avatars using diffusion	Characters	Encoder-Decoder	Photo-wise, Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID	LAION-400M

[44] Landmark Detection and 3D Face Reconstruction for Caricature using a Nonlinear Parametric Model	Face	Encoder-Decoder	Photo-wise	Guided	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	Speed, Inter-pupil distance, Inter-ocular distance, Mean error	CaricatureFace
[203] View Consistent 3D Face Reconstruction Using Siamese Encoder-Decoders	Face	Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	Mean alignment error	300W-LP, AFLW2000-3D
[286] MeInGame: Create a Game Character Face from a Single Portrait	Face	Encoder-Decoder	Photo-wise	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Questionnaire	PSNR, SSIM, User score	WildUV, RGB 3D Face
[290] Free editing of Shape and Texture with Deformable Net for 3D Caricature Generation	Face	Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	User score	3DCaricShop
[299] Towards Implicit Text-Guided 3D Shape Generation	Furniture, 3D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	IOU, EMD, Inception score, Accuracy, PS, Frechet Point Cloud Distance, R-Precision	ShapeNet

[122] ShapeCrafter: A Recursive Text-Conditioned 3D Shape Generation Model	Furniture, 3D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CLIP-Similarity, ShapeGlot-Confidence, FID	Text2Shape++
[327] AutoSDF: Shape Priors for 3D Completion, Reconstruction and Generation	Furniture, Vehicles, 3D General	Encoder-Decoder	Photo-wise, Textual	Reconstructed, Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Uni-directional Hausdorff Distance, Total mutual distance, IOU, CD, F-score	ShapeNet
[400] CLIP-Forge: Towards Zero-Shot Text-To-Shape Generation	Furniture, Vehicles, Props, 3D General	Encoder-Decoder	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	MSE, IOU, FID, MMD, User score, MSE, Accuracy	ShapeNet
[246] Deep Learning-Based Pulmonary Artery Surface Mesh Generation	Medical	Encoder-Decoder	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative)	IOU, MD	N/A

[460] Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis	OP Interior	Encoder-Decoder	Asset-wise, Textual	Arranged	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID, KID, Classification accuracy, KL Divergence	3D-FRONT
[336] 3D Mesh Reconstruction of Foods from a Single Image	Props	Encoder-Decoder	Photo-wise	Reconstructed	Assessment	Metric analysis (Quantitative), Qualitative observations	Iou, Chamfer distance, Approach specific	N/A
[196] Zero-shot text-guided object generation with dream fields	Props, 3D General	Encoder-Decoder	Textual	Guided	Assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	R-Precision	COCO
[430] Marionette: Self-supervised sprite learning	2D Environment	Encoder-Decoder, CNN	Asset-wise	Arranged	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	PSNR, IoU	N/A
[163] Leveraging 2D Data to Learn Textured 3D Mesh Generation	3D General, Furniture, Vehicles, Characters	Encoder-Decoder, CNN	Random seed, Photo-wise	Random, Reconstructed	Assessment	Metric analysis (Quantitative), Qualitative observations	Inception score, FID, Kernel inception distance	CUB-200-2011, BrnoCompSpeed, ShapeNet
[185] Neural Wavelet-domain Diffusion for 3D Shape Generation	Furniture, Vehicles, 3D General	Encoder-Decoder, CNN	Random seed	Random	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	1-NNA, Coverage, MMD, CD, EMD	ShapeNet

[362] Convolutional Occupancy Networks	3D General, OP Interior	Encoder-Decoder, CNN	Sketch-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	IoU, CD, Normal Consistency, F-Score	ShapeNet, Synthetic Indoor Scene Dataset, ScanNet, Matterport3D
[298] ISS: Image as Stepping Stone for Text-Guided 3D Shape Generation	Furniture, Vehicles, Props	Encoder-Decoder, CNN	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Frechet Point Cloud Distance, FID	ShapeNet, CO3D
[241] A deep-learning approach for direct whole-heart mesh reconstruction	Medical	Encoder-Decoder, CNN	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	Dice, Jaccard, ASSD, Hausdorff distance	MMWHS, orCalScore, SLAWT, LASC
[508] Pixel2Mesh++: 3D Mesh Generation and Refinement from Multi-View Images	Furniture, Vehicles, 3D General	Encoder-Decoder, CNN, GCN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CD, F-score, IOU	ShapeNet
[387] Intuitive and efficient roof modeling for reconstruction and synthesis	Buildings	Encoder-Decoder, GCN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative)	Speed, Approach specific	N/A

[538] Multi-view Shape Generation for a 3D Human-like Body	Characters	Encoder-Decoder, GCN, CNN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Average Euclidean Distance, CD, Mean Per Joint Position Error, IOU	MPI-FAUST, UP-3D
[76] Cloud2curve: Generation and vectorization of parametric sketches	2D Icons	Encoder-Decoder, RNN	Sketch-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	FID	Quick, Draw!, K-MNIST
[245] PICO: Procedural Iterative Constrained Optimizer for Geometric Modeling	3D General, Props, Trees, Terrain	Evolutionary-Algorithm	Parametric, Asset-wise	Guided	Comparative assessment	Metric analysis (Quantitative), Interaction, Questionnaire (Quantitative)	Domain specific, Speed	N/A
[153] FAME: 3D Shape Generation via Functionality-Aware Model Evolution	Furniture	Evolutionary-Algorithm	Asset-wise, Parametric	Guided	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Speed, Accuracy	N/A
[194] Example-based synthesis of three-dimensional clouds from photographs	Clouds	Expectation Maximisation	Photo-wise	Guided	Comparative assessment	Qualitative observations	N/A	N/A
[231] ChartPointFlow for Topology-Aware 3D Point Cloud Generation	3D General	Flow-Based	Random seed	Random	Comparative assessment	Various domain tests (Mixed)	EMD	ShapeNet

[262] Controllable text-to-image generation	2D General	GAN	Textual	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	Inception Score, Error, Accuracy	CUB-200-2011, COCO
[373] Mirrorgan: Learning text-to-image generation by redescription	2D General	GAN	Textual	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Questionnaire (Quantitative), Ablation study	Inception Score, R-precision, User score	CUB-200-2011, COCO
[372] Learn, imagine and create: Text-to-image generation from prior knowledge	2D General	GAN	Textual	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Questionnaire (Quantitative)	Inception Score, R-precision, User score	CUB-200-2011, Oxford-102 flower
[213] A Style-Based Generator Architecture for Generative Adversarial Networks	2D General	GAN	Photo-wise	Interpolated, Random, Guided	Assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID, Path length, Separability	FFHQ
[212] Automatic generation of graphical game assets using GAN	2D Icons	GAN	Random seed	Random	Assessment	Metric analysis (Quantitative), Questionnaire (Quantitative)	FID	N/A
[414] Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis	3D General	GAN	Sketch-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CD, Normal Consistency, LFD	TurboSquid

[439] Naive Mesh-to-Mesh Coloured Model Generation using 3D GANs	3D General	GAN	Random seed	Random	N/A	N/A	N/A	N/A
[126] GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images	3D General, Furniture, Vehicles, Buildings, Characters, Textures	GAN	Random seed, Textual	Random, Interpolated, Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CD, Light Field Distance, Coverage, MMD, FID	TurboSquid, ShapeNet
[349] StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation	Face	GAN	Random seed	Random	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	FID, Kernel Inception Distance	FFHQ, AFHQ
[193] An Explorative Design Process for Game Map Generation Based on Satellite Images and Playability Factors	Terrain, Height map	GAN	Random seed	Random	Assessment	Metric analysis (Quantitative)	Kolmogorov-Smirnov test, Shapiro-Wilk test	NASA Visible Earth
[440] Realistic and Textured Terrain Generation using GANs	Terrain, Textures, Height map	GAN	Random seed	Random	Comparative	Visual quality comparison (Qualitative)	SSIM, MSE	SRTM

[453] Automatic generation of architecture facade for historical urban renovation using generative adversarial network	Buildings	GAN	Sketch-wise	Guided	Assessment	Metric analysis (Quantitative), Qualitative observations	FID	N/A
[45] Text and Image Guided 3D Avatar Generation and Manipulation	Face	GAN	Textual, Photo-wise	Guided	Comparative assessment	Qualitative observations, Ablation study	User score	N/A
[567] SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation	Furniture, Vehicles, Props, 3D General	GAN	Photo-wise, Parametric	Reconstructed, Interpolated	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Frechet Point Cloud Distance, FID, ECD, 1-NNA, MMD, Coverage	ShapeNet
[60] IM-NET: Learning implicit fields for generative shape modeling	3D General	GAN, AE	Photo-wise, Random seed	Reconstructed, Random, Interpolated	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	LFD	ShapeNet
[551] Hair-GAN: Recovering 3D hair structure from a single image using generative adversarial networks	Hair	GAN, Diffusion/Propagation	Photo-wise	Reconstructed	Assessment	Qualitative observations	N/A	N/A

[279] Wavelet transform-assisted generative model for efficient 3d deep shape generation	3D General	GAN, Encoder-Decoder	Random seed	Random	Assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CD, EMD, MMD, Coverage	ShapeNet
[270] SP-GAN: sphere-guided 3D shape generation and manipulation	3D General	GAN, GCN	Random seed	Random	Comparative assessment	Metric analysis (Quantitative), Visual quality comparison, Ablation study	Minimum matching distance, Coverage, Frechet Point Cloud Distance	ShapeNet, SMPL, SMAL
[384] Towards Machine-Learning Assisted Asset Generation for Games: A Study on Pixel Art Sprite Sheets	2D Characters	GANs	Sketch-wise	Style transferred	Explorative	Qualitative observations, Metric analysis (Quantitative) Qualitative feedback	RMSE, MAE, SSIM	N/A
[124] Generation of Character Illustrations from Stick Figures Using a Modification of Generative Adversarial Network	2D Characters	GANs	Sketch-wise	Guided	Assessment	Metric analysis (Quantitative), Qualitative observations	Loss	N/A
[520] Cali-sketch: Stroke calibration and completion for high-quality face image generation from human-like sketches	2D Characters	GANs	Sketch-wise	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation Study	PSNR, SSIM, FID, Precision, Recall	CUHK Face Sketch

[528] CSA-GAN: Cyclic synthesized attention guided generative adversarial network for face synthesis	2D Characters	GANs	Sketch-wise	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	SSIM, PSNR, Visual information fidelity, LPIPS	CUHK Face Sketch, AR Face, WHU-IIP
[500] From Attribute-Labels to Faces: Face Generation Using a Conditional Generative Adversarial Network	2D Characters	GANs	Parametric	Guided	Assessment	Metric analysis (Quantiative)	Inception score, FID	CelebA, UvA-NEMO
[172] MW-GAN: Multi-Warping GAN for Caricature Generation With Multi-Style Geometric Exaggeration	2D General	GANs	Photo-wise	Guided	Comparative assessment	Ablation study, Metric analysis (Quantitative), Questionnare (Quantitative)	FID, Accuracy	WebCaricature
[218] Style and Content Disentanglement in Generative Adversarial Networks	2D General	GANs	Photo-wise	Style transferred	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	FID, LPIPS	CelebA, LSUN
[420] GAN-based Multi-Style Photo Cartoonization	2D General	GANs	Photo-wise	Style transferred	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations, Questionnaire (Quantitative)	FID	N/A

[131] RPD-GAN: Learning to Draw Realistic Paintings With Generative Adversarial Network	2D General	GANs	Photo-wise	Style transferred	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	FID	Places2, Stanford Background, CUHK Face Sketch, CelebA
[542] Progressive Semantic Image Synthesis via Generative Adversarial Network	2D General	GANs	Photo-wise, Textual	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	SSIM, Inception score	CUB-200-2011
[249] Synthesizing Images from Hand-Drawn Sketches using Conditional Generative Adversarial Networks	2D General	GANs	Sketch-wise	Guided	Assessment	Qualitative observations	N/A	Edges2Shoes, Ut-Zappos50K
[559] Sketch-to-Color Image with GANs	2D General	GANs	Sketch-wise	Guided	Assessment	Qualitative observations	N/A	N/A
[22] MISS GAN: A Multi-IlluStrator style generative adversarial network for image to illustration translation	2D General	GANs	Photo-wise	Style transferred	Comparative assessment	Ablation study, Qualitative observations	N/A	GANILLA, CycleGAN

[361] SAM-GAN: Self-Attention supporting Multi-stage Generative Adversarial Networks for text-to-image synthesis	2D General	GANs	Textual	Guided	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Inception score, FID, R-precision	CUB-200-2011, COCO
[561] Customizable GAN: Customizable Image Synthesis Based on Adversarial Learning	2D General	GANs	Sketch-wise, Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Human rank	CUB-200-2011, Oxford-102 flower
[405] CAGAN: Text-To-Image Generation with Combined Attention Generative Adversarial Networks	2D General	GANs	Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Inception score, FID	CUB-200-2011, COCO
[192] Image-to-image translation with conditional adversarial networks	2D General	GANs	Photo-wise	Style transferred	Assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	User score, FCN	Cityscapes
[359] StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery	2D General	GANs	Textual, Photo-wise	Style transferred	Comparative assessment	Qualitative observations	N/A	N/A
[541] Art Font Image Generation with Conditional Generative Adversarial Networks	2D Icons	GANs	Photo-wise	Style transferred	Comparative assessment	Qualitative observations	N/A	N/A

[266] Attribute-Conditioned Layout GAN for Automatic Graphic Design	2D Layout	GANs	Asset-wise	Arranged	Assessment	Ablation study, Metric analysis (Quantitative), Questionnare (Quantitative)	Speed, Overlap, Alignment	N/A
[267] LayoutGAN: Synthesizing Graphic Layouts With Vector-Wireframe Adversarial Networks	2D Layout	GANs	Asset-wise, Sketch-wise	Arranged	Assessment	Metric analysis (Quantitative), Questionnaire	Overlap, Alignment, IoU	RICO, Pix2Code
[165] Deep geometric texture synthesis	3D General	GANs	Asset-wise	Style transferred	Comparative assessment	Qualitative observations	N/A	Thingi10K
[248] Masked 3D conditional generative adversarial network for rock mesh generation	3D General	GANs	Parametric	Guided	Comparative assessment	Qualitative observations	N/A	N/A
[230] CityCraft: 3D virtual city creation from a single image	3D General, Buildings, Roads	GANs	Photo-wise	Guided	Assessment	Qualitative observations, Metric analysis (Quantitative)	SSE loss, RMSE	OpenStreetMap
[346] Paired 3D Model Generation with Conditional Generative Adversarial Networks	3D General, Furniture	GANs	Random seed, Parametric	Random	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Average absolute difference, Average voxel agreement ratio	ModelNet

[300] A Generative Adversarial Network for AI-Aided Chair Design	2D General	GANs	Random seed	Random	N/A	N/A	N/A	N/A
[161] Constrained Generative Adversarial Networks for Interactive Image Generation	2D General	GANs	Random seed, Parametric	Guided	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Error	Ut-Zappos50K, CelebA
[498] Generative Image Modeling Using Style and Structure Adversarial Networks	2D General	GANs	Random seed, Photo-wise	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Questionnaire	Classification score	NYU Depth v2
[523] Weighted voxel: a novel voxel representation for 3D reconstruction	3D General, Vehicles, Furniture	GANs	Photo-wise	Reconstructed	Assessment	Metric analysis (Quantitative)	IoU	ShapeNet
[26] Multi-chart generative surface modeling	3D General, Characters	GANs	Random seed	Random, Interpolated	Comparative assessment	Qualitative observations	N/A	DFAUST, CAESAR
[489] Global-to-local generative model for 3D shapes	3D General, Furniture, Vehicles	GANs	Random seed	Random	Comparative assessment	Metric analysis (Quantitative), Ablation Study (Quantitative)	3D Inception score, Symmetry score, Distribution distance	ShapeNet, ImageNet

[278] SG-GAN: Adversarial Self-Attention GCN for Point Cloud Topological Parts Generation	3D General, Props, Furniture, Vehicles, Characters	GANs	Random seed	Random, Interpolated	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	JSD, Coverage, Minimum matching distance, Chamfer distance, EMD	ShapeNet, DFAUST
[77] Designing Co-Creative AI for Virtual Environments	3D General, Props, Vehicles, Furniture	GANs	Random seed, Parametric	Guided	Explorative	Co-design	Speed	N/A
[305] Single Image Shape-from-Silhouettes	3D General, Vehicles, Furniture, Characters	GANs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative Observations, Ablation study	IoU, F-score	ShapeNet, FreiHAND
[237] Image-to-Voxel Model Translation with Conditional Adversarial Networks	3D General, Vehicles, Furniture, Characters	GANs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	IoU, Surface distance	VoxelCity, VoxelHome
[455] EasyMesh: An efficient method to reconstruct 3D mesh from a single image	3D General, Vehicles, Furniture, Props	GANs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	Average euclidean distance	PASCAL VOC, ShapeNet

[283] Point Cloud Generation Using Deep Adversarial Local Features for Augmented and Mixed Reality Contents	3D General, Vehicles, Furniture	GANs	Random seed	Random, Interpolated	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	JSD, Coverage, Minimum matching distance	ShapeNet
[526] ARShape-Net: Single-View Image Oriented 3D Shape Reconstruction with an Adversarial Refiner	3D General, Vehicles, Furniture, Props	GANs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	IoU	ShapeNet, Pix3D
[236] Image-to-Voxel Model Translation for 3D Scene Reconstruction and Segmentation	3D General, Vehicles, Furniture, Props, Characters	GANs	Photo-wise	Reconstructed	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	IoU	SemanticVoxels
[277] HSGAN: Hierarchical Graph Learning for Point Cloud Generation	3D General, Vehicles, Furniture	GANs	Random seed	Random, Interpolated	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	JSD, Coverage, Minimum matching distance, Frechet point cloud distance, Chamfer distance, EMD	ShapeNet

[419] 3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions	3D General, Vehicles, Furniture	GANs	Random seed	Random, Interpolated	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	Frechet point cloud distance, JSD, Coverage, EMD, Chamfer distance	ShapeNet
[98] 3D building fabrication with geometry and texture coordination via hybrid GAN	Buildings	GANs	Random seed	Random	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	JSD, MMD, F-score, Chamfer distance, EMD	CMP, ShapeNet
[338] House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation	Buildings	GANs	Random seed, Parametric	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Questionnaire, Ablation study	Realism(User score), FID, Graph edit distance	LIFULL HOME's
[173] Semi-supervised adversarial recognition of refined window structures for inverse procedural façade modelling	Buildings	GANs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations, Ablation study	MAE	CMP façade, LSAA
[273] PGAN: Prediction Generative Adversarial Nets for Meshes	Face, Normal map	GANs	Random seed	Random	Comparative assessment	Qualitative observations	N/A	BU-4DFE

[409] Synthesizing Facial Photometries and Corresponding Geometries Using Generative Adversarial Networks	Face, Textures	GANs	Random seed	Random	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Facial recognition distance, Sliced Wassertein distance	N/A
[353] Procedural 3D Terrain Generation using Generative Adversarial Networks	Terrain	GANs	Random seed	Random, Interpolated	Assessment	Qualitative observations	N/A	AW3D30, Google Earth Engine API
[108] Full Face-and-Head 3D Model With Photorealistic Texture	Face	GANs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	PSNR, SSIM	WildUV
[247] 3D Face Reconstruction with Texture Details from a Single Image Based on GAN	Face	GANs	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Relative error, RMSE	LFW
[272] Deep 3D caricature face generation with identity and structure consistency	Face	GANs	Photo-wise	Reconstructed	Comparative assessment	Ablation study, Qualitative observations, Metric analysis (Quantitative), Questionnaire	Speed, Inception score, Human score	WebCaricature, CelebA
[97] Terrain Edge Stitching Based On Least Squares Generative Adversarial Networks	Terrain, Height map	GANs	Random seed	Random	Assessment	Metric analysis (Quantitative)	FID	Geospatial data cloud

[107] A Case Study of Generative Adversarial Networks for Procedural Synthesis of Original Textures in Video Games	Textures	GANs	Random seed	Random	Assessment	Metric analysis (Quantitative)	Inception score	VGLC
[449] Interactive Sketch-Based Normal Map Generation with Deep Neural Networks	Normal map	GANs	Sketch-wise	Guided	Comparative assessment	Metric analysis (Quantitative), Qualitative Observations, Questionnaire (Quantitative)	Loss, Angular difference	Labeled PSB
[125] Automatic Generation of Background Computer Graphics by Deep Learning According to User's Preference	OP Exterior	GANs	Sketch-wise	Guided	Assessment	Qualitative observations	N/A	Fundamental geospatial data (Japan)
[222] Procedural Generation of Roads with Conditional Generative Adversarial Networks	Roads	GANs	Sketch-wise	Guided	Comparative assessment	Qualitative observations	N/A	OpenStreetMap
[496] Sketch2Map: A Game Map Design Support System Allowing Quick Hand Sketch Prototyping	Terrain, Height map	GANs	Sketch-wise	Guided	Comparative assessment	Qualitative observations	N/A	N/A
[425] Auto-Encoding Progressive Generative Adversarial Networks for 3D Multi Object Scenes	3D General, Furniture, Characters, OP Interior	GANs, AAE	Random seed	Random	Comparative assessment	Qualitative observations	N/A	SUNCG

[415] DeepSketchHair: Deep Sketch-Based 3D Hair Modeling	Hair	GANs, Diffusion/Propagation	Sketch-wise	Guided	Assessment	Qualitative observations, Tutorial, Interaction, Qualitative feedback, Speed test, Ablation study (Quantitative)	Speed, MSE	3DHW
[490] Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks	OP Interior	GCN	Asset-wise, Random seed	Arranged	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	Classification accuracy, speed, human ranking	SUNCG
[176] Graph2plan: Learning floorplan generation from layout graphs	OP Interior	GCN	Asset-wise	Arranged	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	IoU	RPLAN
[535] 3D Game Model and Texture Generation Using Interactive Genetic Algorithm	Buildings, Textures	Genetic Algorithm	Random seed, Parametric	Guided	Assessment	Questionnaire (Quantitative)	N/A	N/A
[442] Procedurally generated virtual reality from 3D reconstructed physical space	OP Exterior	Genetic Algorithm	Photo-wise	Arranged	Explorative	Interaction, Qualitative Feedback	N/A	N/A
[318] Multi-view 3D reconstruction and modeling of the unknown 3D scenes using genetic algorithms	3D General, Props, Buildings	Genetic Algorithms	Photo-wise	Reconstructed	Assessment	Qualitative observations	N/A	N/A

[233] Evolving 3D Models Using Interactive Genetic Algorithms and L-Systems	3D General, Props	Genetic Algorithms	Random seed, Parametric	Guided	Assessment	Qualitative observations	N/A	N/A
[19] 3D Computational Sketch Synthesis Framework: Assisting Design Exploration Through Generating Variations of User Input Sketch and Interactive 3D Model Reconstruction	Vehicles	Genetic Algorithms	Sketch-wise	Guided	Explorative	Interview, Interaction, Questionnaire (Quantitative), Qualitative feedback (interview)	N/A	N/A
[13] Generation of Complex Underground Systems for Application in Computer Games with Schematic Maps and L-Systems	Terrain, Height map	Grammars, Cellular Automata, L-System	Parametric	Guided	Assessment	Speed test, Qualitative observations	Speed	N/A
[410] Procedural City Generator	Roads	Grammars, L-System	Random seed, Parametric	Guided	N/A	N/A	N/A	N/A
[377] A Swarm Grammar-Based Approach to Virtual World Generation	Terrain	Grammars, L-System	Parametric	Guided	Assessment	Qualitative observations	N/A	N/A

[169] Visualization of A Three-Dimensional Tree Modeling using Fractal Based on L-System	Trees	Grammars, L-System	Parametric	Guided	Assessment	Qualitative observations	N/A	N/A
[94] CAD Shape Grammar: Procedural Generation for Massive CAD Model	3D General	Grammars, Shape Grammar	Parametric	Reconstructed	Assessment	Metric analysis (Quantitative), Qualitative observations	Memory usage, Speed	N/A
[514] Volumetric procedural models for shape representation	3D General, Buildings, Props, Furniture	Grammars, Shape Grammar	Parametric	Reconstructed	Assessment	Metric analysis (Quantitative), Qualitative observations	Speed, Lines of code	N/A
[10] A 3D shape generative method for aesthetic product design	3D General, Vehicles, Props, Furniture	Grammars, Shape Grammar	Parametric	Guided	Assessment	Application case studies	N/A	N/A
[85] Proceduralization of urban models	Buildings	Grammars, Shape Grammar	Parametric	Reconstructed	N/A	N/A	N/A	N/A

[101] Procedural Modeling of Round Building Geometry	Buildings	Grammars, Shape Grammar	Parametric	Reconstructed	Assessment	Qualitative observations	N/A	N/A
[240] Procedural Modeling in Archaeology: Approximating Ionic Style Columns for Games	Buildings	Grammars, Shape Grammar	Parametric	Reconstructed	Assessment	Metric analysis (Quantitative)	FID, Hausdorf distance	N/A
[200] Layered shape grammars for procedural modelling of buildings	Buildings	Grammars, Shape Grammar	Parametric	Reconstructed	Comparative assessment	Qualitative observations	N/A	N/A
[84] Proceduralization for Editing 3D Architectural Models	Buildings	Grammars, Split Grammar	Parametric	Reconstructed	Assessment	Qualitative observations	N/A	N/A
[264] 3D scene reconstruction using a texture probabilistic grammar	Buildings	Grammars, Stochastic Grammar	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Grammar precipitate, Speed	N/A

[205] Configurable 3D Scene Synthesis and 2D Image Rendering with Per-pixel Ground Truth Using Stochastic Grammars	OP Interior	Grammars, Stochastic Grammar	Asset-wise, Parametric	Arranged	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Absolute relative error, Square relative distance, Average log error, RMSE, Log RMSE, Threshold	SUNCG, ShapeNet
[89] Procedural feature generation for volumetric terrains using voxel grammars	Terrain	Grammars, Voxel Grammar	Parametric	Guided	Assessment	Speed test	Speed	N/A
[113] Learning geometric graph grammars	Roads	Graph Grammar	Random seed, Parametric	Guided	Assessment	Metric analysis (Quantitative)	Speed, encoded size	OpenStreetMap
[147] Organic building generation in minecraft	Buildings	Growth-Algorithm, Cellular Automata	Random seed, Parametric	Random, Guided	Assessment	Metric analysis (Quantitative), Qualitative Observations	Speed, Approach specific	N/A

[553] Shrubbery-shell inspired 3D model stylization	3D General, Props, Furniture	Growth-Algorithm, Space Colonisation	Asset-wise	Style transferred	Assessment	Qualitative observations	N/A	N/A
[260] Latent L-systems: Transformer-based Tree Generator	Trees	L-System, Transformer	Random seed	Random	Assessment	Metric analysis (Quantitative), Qualitative observations	Speed, ICTree	N/A
[17] Applying machine learning algorithms to architectural parameters for form generation	Buildings	MLP	Parametric	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	R-score, MSE, RMSE, MAE, Accuracy, Precision, Recall	N/A
[316] Automated data-driven method for creating digital building models from dense point clouds and images through semantic segmentation and parametric model fitting	Buildings	MLP	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative)	Precision, Recall, IOU, Accuracy	S3DIC
[477] Deep learning aided web-based procedural modelling of LOD2 city models	Buildings	MLP	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Precision, Recall	N/A
[201] Procedural Content Generation via Machine Learning in 2D Indoor Scene	OP Interior	MLP	Asset-wise	Arranged	Assessment	Metric analysis (Quantiative)	Domain specific	N/A

[547] Mesh deformation-based single-view 3D reconstruction of thin eyeglasses frames with differentiable rendering	Props	MLP	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Reconstruction error, IOU	N/A
[532] Deep 3D Modeling of Human Bodies from Freehand Sketching	Characters	MLP, CNN, Encoder-Decoder	Sketch-wise	Guided	Comparative assessment	Ablation study, Metric analysis (Quantitative), Qualitative observations	Reconstruction error, MPJPE	AMASS
[263] Combining data-and-model-driven 3D modelling (CDMD3DM) for small indoor scenes using RGB-D data	3D General, Furniture, Props	MLP, Deprojection	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Precision, Recall, RMSE	Kinect V2, Washington RGB-D, S3DIS
[53] Sofgan: A portrait image generator with dynamic styling	Face	MLP, GAN	Photo-wise, Parametric	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative), Ablation study	FID, LPIPS, IOU	CelebA, FFHQ
[178] Generating Procedural Materials from Text or Image Prompts	Textures	MLP, GCN	Photo-wise, Textual	Guided	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CLIP-Similarity, Sliced Wassersetein	Substance Source

[330] A New Method for Modeling Clouds Combining Procedural and Implicit Models	Clouds	Noise	Parametric, Random seed	Guided	Assessment	Qualitative observations	N/A	N/A
[401] Procedural Environment Generation for Cave 3D Model Using OpenSimplex Noise and Marching Cube	Terrain	Noise	Random seed, Parametric	Guided	Assessment	Metric analysis (Qualitative), Tutorial, Interaction, Qualitative Feedback	Domain specific	N/A
[424] Planetary Marching Cubes: A Marching Cubes Algorithm for Spherical Space	Terrain, Height map	Noise	Random seed	Random	N/A	N/A	N/A	N/A
[136] Automatic procedural model generation for 3D object variation	3D General, Props, Characters, Vehicles, Furniture	Parametrisation	Parametric	Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Domain specific	N/A
[306] Synthesis of web layouts from examples	2D Layout	Patch placement	Asset-wise	Arranged	Comparative assessment	Ablation study, Metric analysis (Quantitative)	RMSD, Accuracy, Speed	RICO
[462] A semantic approach to patch-based procedural generation of urban road networks	Roads	Patch placement	Asset-wise, Random seed	Random	Assessment	Metric analysis (Quantitative)	CNR(connected node ratio), Density	N/A

[243] Procedural Content Generation for Game Props? A Study on the Effects on User Experience	2D Environment	Perlin Noise	Random seed	Random	Comparative	Interaction, Questionnaire (Quantitative), T-test	N/A	N/A
[1] Generating 3D Model for Human Body Shapes from 2D images using Deep Learning	Characters	R-CNN	Photo-wise, Parametric	Guided	Assessment	Metric analysis (Quantitative)	Error	SPRING, CAESAR, COCO
[209] Nostalgin: Extracting 3D City Models from Historical Image Data	Buildings	R-CNN, GAN	Photo-wise	Reconstructed	Assessment	Qualitative and Quantitative observations by component	Precision, Recall, Loss	COCO, Places2
[274] 3D Shape Reconstruction of Furniture Object from a Single Real Indoor Image	Furniture	R-CNN, GCN	Photo-wise	Reconstructed	Assessment	Metric analysis (Quantitative)	Chamfer distance, F-score	3D-FUTURE
[174] A study on the automatic generation of banner layouts	2D Layout	RL	Asset-wise	Arranged	Comparative assessment	Metric analysis (Quantitative), Qualitative observations	Text alignment, Overlap, Visual balance	N/A
[285] Modeling 3D Shapes by Reinforcement Learning	3D General, Vehicles, Furniture, Props	RL, IL	Photo-wise	Reconstructed	Assessment	Qualitative observations, Metric analysis (Quantitative)	IoU, Chamfer distance	ShapeNet

[118] Procedural Generation of Multistory Buildings With Interior	Buildings	Shape Grammar	Parametric, Random seed	Guided	Assessment	Interaction, Questionnaire, Metric analysis (Quantitative)	Domain specific	N/A
[276] Rule-based automatic generation of logo designs	2D Icons	Shape Grammar	Random seed	Random	Assessment	Qualitative observations	N/A	N/A
[28] Design and Deployment of Photo2Building: A Cloud-based Procedural Modeling Tool as a Service	Buildings	Shape Grammar, CNN	Photo-wise	Reconstructed	Assessment	Speed test, Running cost analysis	Speed, Running cost	CMP
[342] Interactive sketching of urban procedural models	Buildings	Shape Grammar, CNN	Sketch-wise	Guided	Assessment	Tutorial, Recorded interaction, Questionnaire (Mixed)	Accuracy, RMSE, Speed	N/A
[544] Neural Procedural Reconstruction for Residential Buildings	Buildings	Shape Grammar, Encoder-Decoder, CNN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations	Accuracy, IoU, Precision, Recall	UK Environment agency
[46] Facade geometry generation from low-resolution aerial photographs for building energy modeling	Buildings	Shape Grammar, Random Forest	Photo-wise	Reconstructed	Assessment	Qualitative observations, Metric analysis (Quantitative)	RMSE	N/A

[112] AutoBiomes: procedural generation of multi-biome landscapes	Terrain, Height map	Simplex Noise, Erosion	Parametric, Asset-wise	Arranged	Assessment	Qualitative observations, Speed test	Speed	N/A
[115] Procedural generation of 3D karst caves with speleothems	Terrain	Simulation, Erosion, Perlin Noise, Voronoi Noise	Parametric	Guided	Assessment	Qualitative observations, Speed test	Speed	N/A
[90] Space Colonisation for Procedural Road Generation	Roads	Space Colonisation	Parametric	Guided	N/A	N/A	N/A	N/A
[383] Applicability of Space Colonization Algorithm for Real Time Tree Generation	Trees	Space Colonisation	Parametric	Guided	Assessment	Speed test, Metric analysis (Quantitative)	Speed, Domain specific	N/A
[275] AstroGen – Procedural Generation of Highly Detailed Asteroid Models	Terrain	Swarm Algorithm, Perlin Noise	Parametric	Guided	Assessment	Metric analysis (Quantitative), Speed test	Speed, Domain specific	DAMIT
[220] DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design	3D General	Topology optimisation	Sketch-wise	Guided	Explorative	Tutorial, Recorded Interaction, Questionnaire (Qualitative)	Speed	N/A

[242] Blt: Bidirectional layout transformer for controllable layout generation	2D Layout	Transformer	Asset-wise	Arranged	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	IOU, Overlap, Alignment, FID, human classification score	RICO, PubLayNet, Magazine, Image Ads, COCO, 3D-FRONT
[454] 3d-gpt: Procedural 3d modeling with large language models	OP Exterior, 3D General	Transformer	Textual	Guided	Assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CLIP score, failure rate, parameter diversity	N/A
[357] Atiss: Autoregressive transformers for indoor scene synthesis	OP Interior	Transformer	Asset-wise	Arranged	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	KL Divergence, FID	3D-FRONT
[499] Sceneformer: Indoor scene generation with transformers	OP Interior	Transformer	Asset-wise, Textual	Arranged	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	Accuracy, speed	SUNCG
[91] Cogview: Mastering text-to-image generation via transformers	2D General	VAE	Textual	Guided	Comparative assessment	Metric analysis (Quantitative), Questionnaire (Quantitative)	FID, Inception Score, Caption loss, User score	COCO
[382] Zero-shot text-to-image generation	2D General	VAE	Textual	Guided	Comparative assessment	Qualitative observations, Metric analysis (Quantitative)	FID, Inception Score	JFT-300M, CUB-200-2011, COCO

[128] SceneHGN: Hierarchical Graph Networks for 3D Indoor Scene Generation with Fine-Grained Geometry	OP Interior	VAE	Asset-wise	Arranged	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	FID, EMD	3D-FRONT
[129] SDM-NET: deep generative network for structured deformable mesh	3D General, Props, Furniture, Vehicles	VAE	Random seed, Asset-wise	Random, Reconstructed, Interpolated	Comparative assessment	Metric analysis (Quantitative), Qualitative Observations, Ablation study	JSD, Coverage, Minimum matching distance	ShapeNet, ModelNet
[130] TM-NET: deep generative networks for textured meshes	3D General, Vehicles, Furniture, Textures	VAE	Photo-wise, Random seed	Random, Interpolated, Reconstructed	Comparative assessment	Metric analysis (Quantitative), Qualitative Observations, Ablation study	LPIPS, SSIM, Fooling rate	ShapeNet
[459] Variational Autoencoders for Deforming 3D Mesh Models	Characters	VAE	Random seed, Asset-wise	Random, Interpolated	Comparative assessment	Metric analysis (Quantitative)	Vertex error, MSE	SCAPE, Dyna
[207] ShapeMOD: macro operation discovery for 3D shape programs	Furniture	VAE	Parametric, Asset-wise, Random seed	Random, Interpolated, Reconstructed	Comparative	Tutorial, Recorded interaction, Questionnaire (Mixed)	Approach Specific	N/A

[206] ShapeAssembly: learning to generate programs for 3D shape structure synthesis	Furniture	VAE	Parametric, Asset-wise, Random seed	Random, Interpolated, Reconstructed	Comparative assessment	Defined metric comparison (Quantitative), Visual qualities (Qualitative)	Rootedness, Stability, Realism(Classifier fool rate), FD	PartNet
[530] DSG-Net: Learning Disentangled Structure and Geometry for 3D Shape Generation	Furniture, 3D General	VAE	Random seed	Random, Interpolated	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CD, EMD, Coverage, Quality score*, Frechet Point Cloud Distance	PartNet
[271] EditVAE: Unsupervised Parts-Aware Controllable 3D Point Cloud Shape Generation	Furniture, Vehicles, 3D General	VAE	Random seed	Random	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	JSD, MMD, Coverage, CD, EMD	ShapeNet
[545] LION: Latent Point Diffusion Models for 3D Shape Generation	3D General	VAE, CNN	Asset-wise	Interpolated	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	1-NNA, CD, EMD, MMD, Coverage, IOU	ShapeNet
[61] Autoregressive 3D Shape Generation via Canonical Mapping	Furniture, Vehicles, 3D General	VAE, CNN	Photo-wise	Reconstructed	Comparative assessment	Metric analysis (Quantiative), Qualitative observations, Ablation study	CD, EMD, MMD, Coverage, 1-NNA, Total Mutual Distance	ShapeNet

[398] 3D hair synthesis using volumetric variational autoencoders	Hair	VAE, Diffusion/Propagation	Photo-wise	Reconstructed	Comparative assessment	Ablation study (Quantitative), Qualitative observations	IoU, Precision, Recall, Flow field loss	USC-HairSalon
[36] Broomrocket: Open Source Text-to-3D Algorithm for 3D Object Placement	OP Exterior		Textual	Arranged	Assessment	Metric analysis (Quantitative)	Lines of code	Sketchfab
[20] Deterministic procedural generation of mesh detail through gradient tiling	Terrain, Height map		Asset-wise, Random seed	Guided	Assessment	Metric analysis (Quantitative)	Flood extent, Speed	N/A

TABLE A2: Datasets used within GAG literature.

Datasets	
[180] LFW	[437] SUNCG
[187] DFC2022	[534] ShapeNetPart
[519] ModelNet	[328] PartNet
[413] ClipNet	[52] ShapeNet
[444] ShapeNetPlain	[436] SUN RGB-D
[80] PROCTHOR	[257] Helen
[301] CelebA	[503] ShapeCOSEG
[310] SynHand5M	[564] Structure3D
[470] CMP	[111] ImageNet
[82] 3D Sketching using Multi-View Deep Volumetric Prediction	[478] SURREAL
[238] AFLW	[476] Human3.6M
[486] CUB-200-2011	[335] Cloud image classification
[402] Pigraphs	[468] SketchUp 3D Warehouse

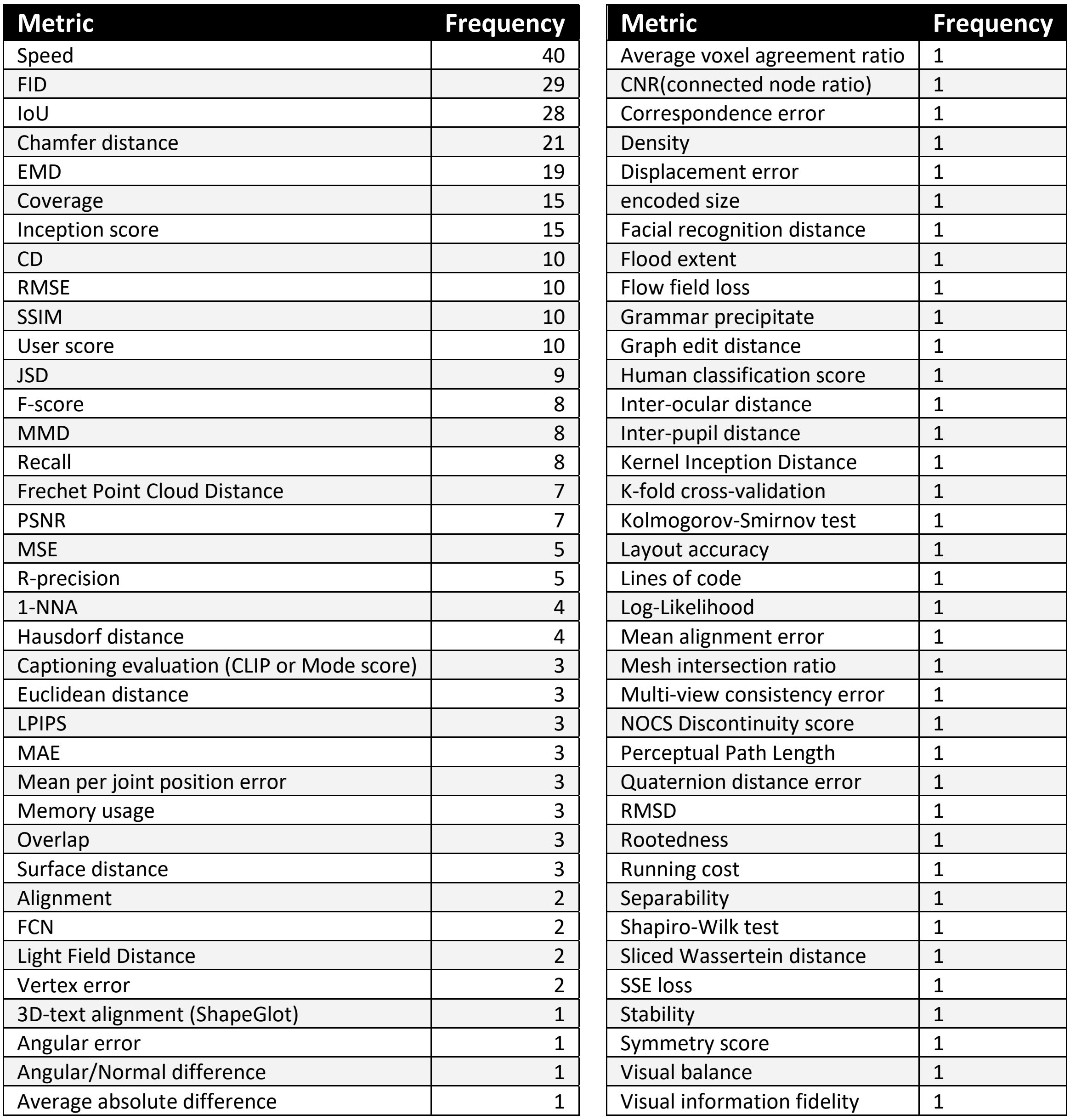
[571] 3D Scene	[41] EuRoC MAV
[333] Hagley Museum model data	[427] 3D-Printed RGB-D Object
[64] redwood-3dscan	[524] Grape bunches
[69] ABO	[120] 3D-FRONT
[393] DreamBench	[403] LAION-400M
[288] COCO	[70] Cityscapes
[376] C4	[404] LAION-5B
[304] DF27	[79] Objaverse
[371] NeRF4	[159] T3Bench
[210] ShapeNet-Synthetic	[456] Pix3D
[43] CaricatureFace	[577] 300W-LP
[86] WildUV	[374] 3DCaricShop
[122] Text2Shape++	[579] MMWHS
[35] MPI-FAUST	[213] FFHQ
[469] TurboSquid	[337] NASA Visible Earth

[476] SRTM	[476] CUHK Face Sketch
[186] WebCaricature	[570] Places2
[191] Edges2Shoes	[168] GANILLA
[81] RICO	[572] Thingi10K
[348] OpenStreetMap	[422] NYU Depth v2
[34] DFAUST	[237] VoxelCity
[106] PASCAL VOC	[236] SemanticVoxels
[282] LIFULL HOME's	[470] BU-4DFE
[198] AW3D30	[451] VGLC
[208] Labeled PSB	[135] Fundamental geospatial data (Japan)
[50] 3DHW	[517] RPLAN
[308] AMASS	[256] Kinect V2
[6] Substance Source	[533] SPRING
[121] 3D-FUTURE	[104] UK Environment agency
[582] DAMIT	[452] JFT-300M

[12] SCAPE	[175] USC-HairSalon
[426] Sketchfab	[326] IIIT 5K-word
[443] ShapeNetCOCO	[80] ARCHITECTHOR
[466] NYU Hand Pose	[55] Floor-SP
[180] Pix3D	[254] UP-3D
[396] 300W	[432] Online products
[157] PROX	[75] ScanNet
[177] TMT	[397] DrawBench
[88] BERT	[96] Google Scanned Objects
[317] RealFusion15	[558] ShapeNet-Sketch
[4] ShapeGlot	[578] AFLW2000-3D
[287] RGB 3D Face	[431] BrnoComp- Speed
[362] Synthetic Indoor Scene Dataset	[386] CO3D
[515] orCalScore	[144] Quick, Draw!
[340] Oxford-102 flower	[65] AFHQ

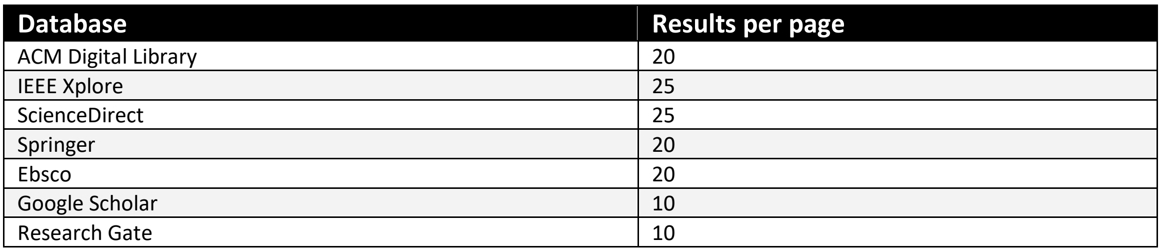
[302] SMPL	[314] AR Face
[137] UvA-NEMO	[537] LSUN
[145] Stanford Background	[536] Ut-Zappos50K
[574] CycleGAN	[25] Pix2Code
[395] CAESAR	[580] FreiHAND
[237] VoxelHome	[576] LSAA
[143] Google Earth Engine API	[252] Washington RGB-D
[569] PubLayNet	[366] Dyna
[8] RoboTHOR	[234] I-JBA
[540] BigHand2.2M	[522] SUN
[408] CFPW	[149] Multi-pie
[573] RESY	[156] LVIS
[56] Text2Shape	[211] SLAWT
[67] K-MNIST	[581] SMAL
[504] WHU-IIP	[497] CUHK Face Sketch

[15] S3DIS	[568] Magazine
[380] HM3D	[341] ECP
[331] AgeDB	[356] VGG Face
[204] FIT400M	[51] Matterport3D
[463] LASC	[259] Image Ads
[239] AI2-iTHOR	[40] LS3D
[566] CALFW	[343] FRGC
[546] JFT-4B	[565] CPLFW
[47] VGGFace2	



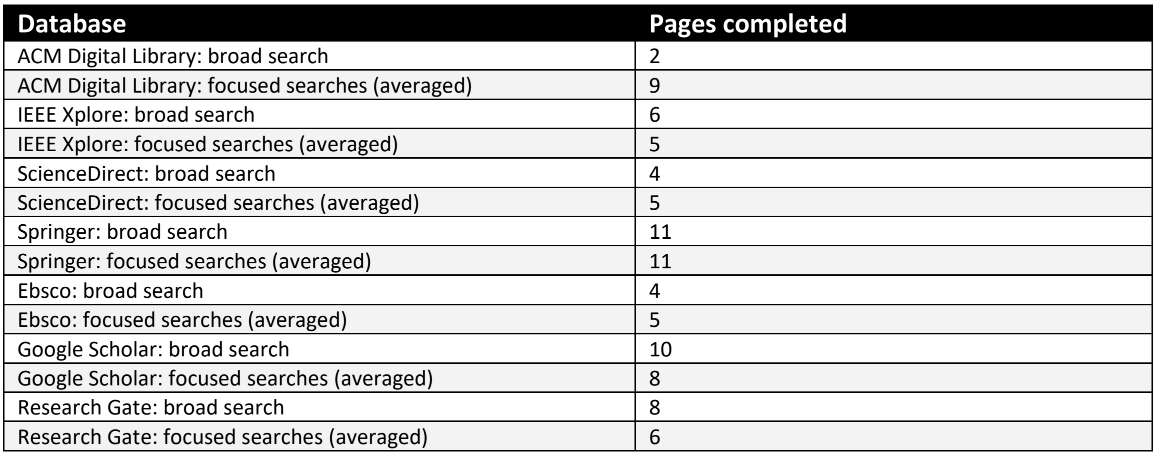
Metric	Frequency
Speed	40
FID	29
IoU	28
Chamfer distance	21
EMD	19
Coverage	15
Inception score	15
CD	10
RMSE	10
SSIM	10
User score	10
JSD	9
F-score	8
MMD	8
Recall	8
Frechet Point Cloud Distance	7
PSNR	7
MSE	5
R-precision	5
1-NNA	4
Hausdorf distance	4
Captioning evaluation (CLIP or Mode score)	3
Euclidean distance	3
LPIPS	3
MAE	3
Mean per joint position error	3
Memory usage	3
Overlap	3
Surface distance	3
Alignment	2
FCN	2
Light Field Distance	2
Vertex error	2
3D-text alignment (ShapeGlot)	1
Angular error	1
Angular/Normal difference	1
Average absolute difference	1
Average voxel agreement ratio	1
CNR(connected node ratio)	1
Correspondence error	1
Density	1
Displacement error	1
encoded size	1
Facial recognition distance	1
Flood extent	1
Flow field loss	1
Grammar precipitate	1
Graph edit distance	1
Human classification score	1
Inter-ocular distance	1
Inter-pupil distance	1
Kernel Inception Distance	1
K-fold cross-validation	1
Kolmogorov-Smirnov test	1
Layout accuracy	1
Lines of code	1
Log-Likelihood	1
Mean alignment error	1
Mesh intersection ratio	1
Multi-view consistency error	1
NOCS Discontinuity score	1
Perceptual Path Length	1
Quaternion distance error	1
RMSD	1
Rootedness	1
Running cost	1
Separability	1
Shapiro-Wilk test	1
Sliced Wassertein distance	1
SSE loss	1
Stability	1
Symmetry score	1
Visual balance	1
Visual information fidelity	1

TABLE A3: Metric frequency



Database	Results per page
ACM Digital Library	20
IEEE Xplore	25
ScienceDirect	25
Springer	20
Ebsco	20
Google Scholar	10
Research Gate	10

TABLE A4: Results per page for each database.



Database	Pages completed
ACM Digital Library: broad search	2
ACM Digital Library: focused searches (averaged)	9
IEEE Xplore: broad search	6
IEEE Xplore: focused searches (averaged)	5
ScienceDirect: broad search	4
ScienceDirect: focused searches (averaged)	5
Springer: broad search	11
Springer: focused searches (averaged)	11
Ebsco: broad search	4
Ebsco: focused searches (averaged)	5
Google Scholar: broad search	10
Google Scholar: focused searches (averaged)	8
Research Gate: broad search	8
Research Gate: focused searches (averaged)	6

TABLE A5: Pages of results completed for each database search.

Appendix
B: Participant recruitment material

Subject: Participation in a study looking at AI based asset generation tools in game design and development pipelines. Hello everyone! I am a postgraduate researcher at Brunel University London, interested in the integration of generative AI tools within game design and development workflows. Specifically, I am looking at how graphical assets such as 3D models can be procedurally generated and used beneficially in real-world game projects. Regarding this, I will be conducting a study to examine the needs and requirements of those experienced in game design and/or development. Imagine that there was a tool that could quickly generate inspiration in the early stages of design, perhaps you could sketch what you want and receive similarly shaped 3D models, or a tool that lets you use a combination of generation and hand-refinement. Would such a tool be useful in your projects? Obtaining your opinions and insights on this matter would be greatly appreciated. All this requires is that you take a look at a prototype tool for the Unity engine and complete some online questionnaires, this should take around 20-30 minutes. This whole process will be through a web form. In the form you will complete a series of tasks, answer some questions and upload what you have made throughout the process. After this, if you have the time, you may opt in for a quick chat (approximately 10 minutes), where you can share any ideas, questions or feedback on the prototype and concept as a whole. None of the data collected will be identifying, and thus your participation will be confidential and anonymous. If you like, you may also keep the tool for your own projects. If this sounds interesting to you, and you would like to participate in this study, please follow the link below: https://forms.office.com/e/yD17MDBKMw Thanks for reading! Kind Regards,

Kaisei Fukaya Kaisei.Fukaya@brunel.ac.uk

Appendix C: Interview questions

1. What do you think about the idea of a tool that generates assets for your projects?
2. Do you have any concerns about such a tool?
3. What benefits do you see for such a tool?
4. Are there particular types of asset that you would like to see a generative tool for?
5. Would you prefer a stand-alone tool or a tool integrated into your game engine of choice? and why?
6. Are there any thoughts you would like to share, that we haven't already covered?

Appendix
D: Sample notes from study 1

- Found mock-ups to be about as good as each other, as the process was the same. Preferred integrated mock-ups as they are more familiar with Unity interface and do not want to download a new software.
- Would use as long as they don't need a third software to get their assets into their game.
- Would use the software for helping when they are stuck with ideas, but does not trust it to create final production assets.
- Currently working on a game that requires a lot of clothing and cosmetic variety, generative AI would be useful for this and save a lot of time.
- Mostly use FBX for 3D content so ideally this should be an output option. While there are plenty of ways to convert formats, having options that cover their main choices will make things much easier.
- Stand-alone could be useful, but prefer not to download and open up more software than necessary.
- Was frustrated by lack of undo and redo controls, as this is how they are used to working within Unity. Did not like having to manually delete nodes and connections.

Appendix
E: Sample notes from study 2

- Terminology might not be understandable, needs tooltips.
- Ability to see the result in an end context, e.g. weapon in the hands of a character. Integration into an iterative design workflow.
- Participant not used 3d modelling tool before. this was rather accessible for them instead.
- Visual controls e.g. draggable handles rather than numbers.
- Manual creation could start with pre-sets as a starting point for design, rather than an empty canvas.
- Enjoyed randomisation, can play around a lot more.
- If they worked perfectly, then interpolate would be the best as you can also use it like the single reconstruction so it is more versatile.
- As they are now, 1 (manual) and 4 (random) are best.
- Would like the ability to edit images before generation, such as cropping etc.
- It is important to see the final design in context, this provides extra certainty and more fun which leads to more motivation to create designs.
- Interviewee suggested strong desire for viewing assets in final context during the design process, such as showing the sword on the character in real-time, to "envision the end goal". In addition, viewing the asset in a refined form while editing. It would be easier and more fun to create designs.
- Suggested starting with preset start points for manual editing.
- Suggested a preference for visual controls over numbers
- Enjoyed the randomisation technique as they could play around a lot more with little effort.

Appendix F: Sword generator tool

Link to Swordgen files on GitHub and animated GIFs of techniques: `https://github.com/Kaisei-Fukaya/SwordGeneratorTool`