Intelligent Generation of Graphical Game Assets: A Conceptual Framework and Systematic Review of the State of the Art

KAISEI FUKAYA, Brunel University of London, London, United Kingdom of Great Britain and Northern Ireland
DAMON DAYLAMANI-ZAD, Brunel University of London, London, United Kingdom of Great Britain and Northern Ireland
HARRY AGIUS, Brunel University of London, London, United Kingdom of Great Britain and Northern Ireland

Procedural content generation (PCG) can be applied to a wide variety of tasks in games, from narratives, levels, and sounds to trees and weapons. A large amount of game content is composed of *graphical assets*, such as clouds, buildings, or vegetation, that do not require gameplay function considerations. There is also a breadth of literature examining the procedural generation of such elements for purposes outside of games. The body of research, focused on specific methods for generating specific assets, provides a narrow view of the available possibilities. Hence, it is difficult to have a clear picture of all approaches and possibilities, with no guide for interested parties to discover possible methods and approaches for their needs and no facility to guide them through each technique or approach to map out the process of using them. Therefore, a systematic literature review has been conducted, yielding 239 accepted papers. This article explores state-of-the-art approaches to *graphical asset* generation, examining research from a wide range of applications, inside and outside of games. Informed by the literature, a conceptual framework has been derived to address the aforementioned gaps.

CCS Concepts: • **Software and its engineering** → **Interactive games**; • **Applied computing** → **Computer games**; • **Computing methodologies** → **Artificial intelligence**; *Machine learning;*

Additional Key Words and Phrases: Graphical asset generation, procedural generation, games, artificial intelligence, deep learning

ACM Reference Format:
Kaisei Fukaya, Damon Daylamani-Zad, and Harry Agius. 2025. Intelligent Generation of Graphical Game Assets: A Conceptual Framework and Systematic Review of the State of the Art. *ACM Comput. Surv.* 57, 5, Article 118 (January 2025), 38 pages. https://doi.org/10.1145/3708499

1 Introduction

Skilled artists and designers build digital content via a combination of technique and creativity. They use software such as the Autodesk products [6], Blender [12], Unity engine [251], and Unreal

Authors' Contact Information: Kaisei Fukaya, Brunel University of London, London, United Kingdom of Great Britain and Northern Ireland; e-mail: kaisei.fukaya@brunel.ac.uk; Damon Daylamani-Zad, Brunel University of London, London, United Kingdom of Great Britain and Northern Ireland; e-mail: damon.daylamani-zad@brunel.ac.uk; Harry Agius, Brunel University of London, London, United Kingdom of Great Britain and Northern Ireland; e-mail: harry.agius@brunel.ac.uk.

ACM 0360-0300/2025/01-ART118
https://doi.org/10.1145/3708499

engine [43]. These tools help to streamline the technical aspects of content creation, though humans still hold full creative responsibility. **Procedural content generation (PCG)** involves the algorithmic production of digital content. When specific content is desired, PCG applications can further reduce the effort and time required while retaining varying degrees of creative control. Algorithms can be applied at runtime to produce unique experiences, e.g., in Minecraft [176], or during production to streamline design and development pipelines, e.g., with SpeedTree [91]. In the context of development tools, the latter form holds more relevance, though many methods can be applied in both contexts. Regardless, PCG tools can both optimise and democratise the content creation process. Search-based PCG methods seek out quality content by searching a content space and evaluating the results [249], while some systems learn from existing content via **machine learning (ML)** [242]. For creative tasks, mixed-initiative approaches may be applied, allowing machine and user to co-create content [148]. These methods are popular for generating content for games, such as levels, loot [13], puzzles, or stories [9].

This work will focus on purely visual forms of content that will be referred to as *graphical assets*, which we define as static, standalone pieces of digital visual content in formats that can be displayed by common graphics engines, including mesh, voxel, point-cloud, bitmap, and vector graphics. Content such as animation, visual effects, and text are excluded by this definition, albeit their visual elements are included, such as riggable characters, shapes, and fonts. This defines the scope of the research to cross-domain transferable, non-gameplay affecting forms of content. Graphical assets have practical uses in numerous fields, and in many cases it is necessary to use generative methods to automate all, or part of, their production. In particular, graphical assets are used in areas such as games [103, 123], medicine [250, 272], architectural visualisation [183, 184], product design [2], 3D printing [124], and training computer-vision algorithms [100]. While these areas use graphical assets for vastly different purposes, the methods and results need not be tied to a single application. An asset, whether it is a 3D mesh or 2D graphic, can be applied in any domain as long as it meets the requirements. Hence, a core contribution of this research is the inclusion and consideration of generative methods for graphical assets used outside of the games domain.

Furthermore, we focus on generative methods that employ a level of "intelligence," which we define as those that apply logic, reasoning, or a degree of initiative in the production of assets, whether through rules that are built-in, such as in a grammar, or learned 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. We base this on the usage of the term in existing literature [4, 165, 169].

While existing surveys broadly examine PCG for game content [79, 242, 249], virtual environments [233], buildings [128], deep learning [156], and reinforcement-learning-based PCG [67, 112, 178], none focus on the full range of purely graphical content. For example, Hendrikx et al. [79] provide an overview of PCG methods for games, which includes all forms of content from "Game Bits" to "Game Systems" and "Game Design." Here, *graphical assets* comprise a smaller portion of the analysed literature, considerations are naturally game-focused, and thus all forms of graphical assets are not generally considered. Kuzias and von Mammen [128] comprehensively focus on PCG methods for generating buildings only but do not other types of graphical assets. Likewise, Smelik et al. [233] survey methods of generating content for virtual worlds, covering terrain, buildings, vegetation, and the like. The scope does not include props, characters, or 2D elements. Many advancements have been made in generative approaches and in particular generative deep learning in the subsequent decade since these surveys were published. Hughes et al. [88] present a systematic review of **generative adversarial networks (GANs)** in creative and design applications, discussing the applications of GAN-based methods, and recent advances in deep learning have presented impressive results in text-to-image generation [207, 211, 213] and

3D shape learning using differentiable rendering [78, 106, 278]. The remaining body of research focuses on specific methods for generating specific assets, such as making a cloud or a chair.

Consequently, readers are required to know what they are looking for, providing a narrow view of available approaches and possibilities and making it difficult to have a clear overview picture. Those already looking into generative deep learning or GANs may seek out Reference [88], those interested in reinforcement learning will find References [67, 112], and those interested in digital environments may find Reference [233]. There are great resources within individual research fields. Yet, this is not so useful in a practical sense when the end goal is simply: "find the best and most appropriate method for generating x." Hence, there is no guide to help interested parties discover the possible methods and approaches for their needs, and no facility to guide them through each technique or approach, mapping out the process of using them. Furthermore, extending the scope beyond methods used in games would provide a more complete view of what is possible and encourage cross-pollination of ideas.

This work aims to address this gap by aggregating state-of-the-art graphical asset generation methods with the goal of answering the following research questions:

RQ1 What *"intelligent"* generative methods are in use for producing graphical assets?
RQ2 What are the forms of graphical asset that have been produced through *"intelligent"* generative methods?
RQ3 Can methods for graphical asset generation map onto the requirements and purposes of the end-user and how?

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, we aim to elucidate the relationship between the requirements of the user and the multitude of methods available (RQ3). We present the results of a systematic literature review and a devised framework named ***Graphical Asset Generation/Transformation* (GAGeTx)** for navigating these existing approaches and how to use each one.

2 The Approach to Literature Search

To obtain relevant literature, a systematic literature search inspired by PRISMA and the approach of Reference [88] has been employed, incorporating a series of screenings as shown in Figure 1. 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 1(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 groups 2 and 3.

Using the the inclusion and exclusion criteria seen in Figure 1(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 1(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. In the case of the larger, broad search queries, it was necessary to deem a search exhausted once a full page of results had not passed the criteria due to

the large number of results and the impracticality of assessing every page. However, the smaller targeted queries aimed to fill any gaps from broad searches. Furthermore, as searches took place on multiple different web databases, the ordering of each set of results was subject to the default relevance ordering. The number of results per page was also variable and thus recorded. Some queries were too long for the ScienceDirect database, which limits the number of Boolean operators and does not accept wildcard operators. These queries had to be decomposed into smaller strings.

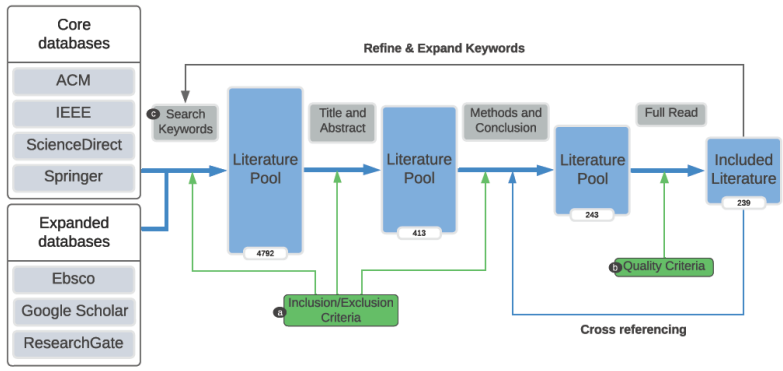
The pool of accepted literature was refined by evaluating the methods and conclusions against the inclusion and exclusion criteria, then further refined by evaluating the full text against the quality criteria. At this stage, patterns emerged from the accepted literature, specifically, the common classes of graphical asset. These key graphical asset types 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 frequency tables or Figure 1.

3 GAGeTx: A Framework for Graphical Asset Generation/Transformation

The conceptual framework, GAGeTx, was developed through an **inductive content analysis (ICA)** [42, 253] of the accepted pool of literature, detailed in Section 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. ICA 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 the second round of searches, which subsequently expanded the literature pool. At this stage, the final refinement of the category arrangement and naming took place as a collaboration between researchers, resulting in the categories and sub-categories of GAGeTx.

Five key aspects emerged as differentiators among 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, 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



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.	— *Non-"intelligent"* methods.
— 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)

Fig. 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.

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 or *externally* by transforming human creative input, e.g., a text prompt to an image. *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. 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, then the *approach* can be seen as the solution, i.e., the way in which the *technique* is achieved. Different graphics formats, such

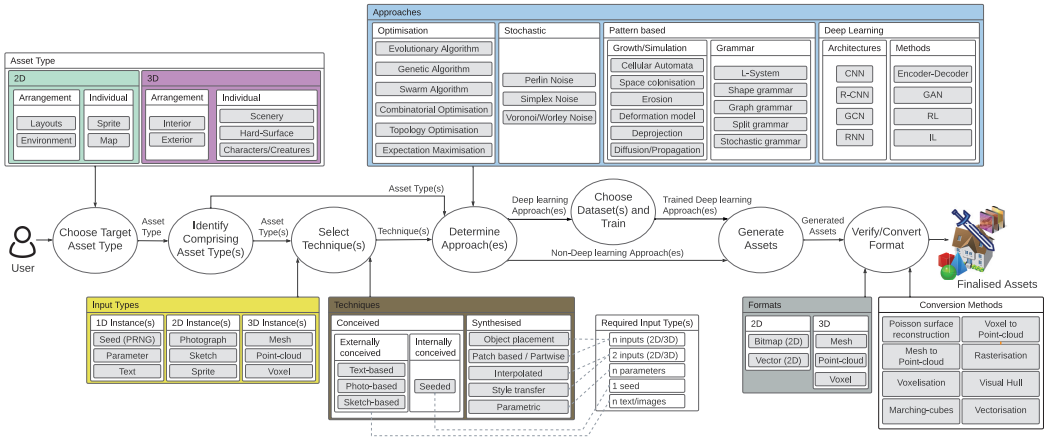


Fig. 2. GAGeTx framework proposed for graphical asset generators.

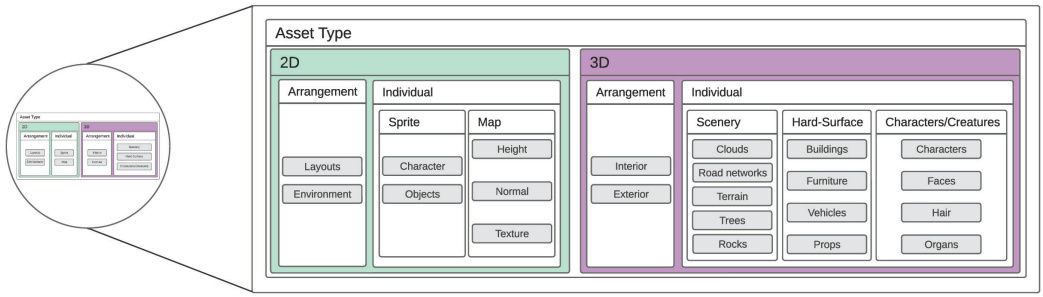


Fig. 3. Asset types categorisations, populated with types observed in the literature.

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 4 will discuss asset types, then Section 6 will examine techniques. Section 7 will examine approaches while discussing the bulk of the literature. Generating and converting assets will be examined in Sections 8 and 9, respectively.

4 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 assets 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 3, and frequency Tables 1 and 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.

Table 1. Frequency and Breakdown of 2D Asset Types within Categories

Asset Category	Asset Type	Freq.
Arrangement	Layout	4
	Environment	2
Individual (Sprite)	Character	5
	Objects	25
Individual (Map)	Height	8
	Normal	3
	Texture	8

Table 2. Frequency and Breakdown of 3D Asset Types within Categories

Asset Category	Asset Type	Freq.
Arrangement	Interior	15
	Exterior	4
Individual (Hard-Surface)	Buildings	21
	Furniture	51
	Vehicles	39
	Props	40
Individual (Scenery)	Cloud	3
	Road network	6
	Terrain	17
	Tree	6
Individual (Characters or Creatures)	Character	18
	Hair	4
	Face	12
	Organ	3

4.1 2D Assets

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 [92] 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.

In print, web, and UI design the arrangement of graphical elements is key to the presentation of information. Unlike sprites or 3D environments, these arrangements must be precise to capture the attention of an audience or user and convey information succinctly. 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. 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.

Individual 2D assets take two primary forms, sprites and maps. Sprites are standalone 2D assets, which for the purpose of this review includes general bitmap images that mimic photographs [202], artwork [61], represent 2D characters [209], or scenery [123]. Approaches to rendering 3D meshes make use of various 2D maps that define how a shader renders the surface of a 3D object. In modern rendering approaches, many types of map may be employed, including height maps [93, 238], normal maps [240], and texture maps [44, 59].

4.2 3D Assets

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 subcategories as given in Table 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 [280] and games [72, 286], while in some cases entire cities are generated [116, 258]. Having a simple structure, consisting of walls, doors, windows, and roofs, buildings are suited to approaches that work by combining simple elementary or parametric components, such as grammars or procedural growth-based algorithms. Some approaches specifically focus on building façades where these same techniques are applied.

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.

Vehicles: 3D digital environments that resemble the modern world are likely to require 3D assets that represent vehicles. The ShapeNet dataset [18] contains many categories of 3D shape, including airplanes, buses, and cars; due to the popularity of the dataset, there are many generative approaches validated on such vehicle models, including mesh [154, 162], voxel [119, 281], and point-cloud [152] 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 [265], guitars, lamps, or bottles [140]. As a popular dataset for generative approaches, ShapeNet [18] contains many types of prop.

Clouds: 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.

Roads: Roads, or road networks, are usually designed with functionality in mind. 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.

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. Alternatively, other tasks may require character mesh generation, such as checking clothing fit in online shops [1].

Faces: A key element of character identity, thus high-quality representations encompassing mesh and texture are required. Furthermore, where customisation is allowed, the ability to adjust and customise player-character appearance is key.

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.

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 and the use of volumetric representations.

Trees: Many 3D digital environments make use of tree models, as shown by the popularity of SpeedTree [91]. 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.

Organs: Due to the need for accurate imaging and visualisation, medical fields benefit from reconstructive visualisation/modelling approaches, particularly organs. 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 that domain.

4.3 Choose Target Asset Type and Identify Their Comprising Asset Types

The target asset type is that which the user intends to generate and is dependant on the user's particular use case or intended product. Within the literature, many graphical asset types have been observed, though it is acknowledged that these examples are not exhaustive and thus techniques and approaches applied to the generation of these asset types may be applicable to further, similar asset types. As such, a generic categorisation of graphical asset types is provided in Figure 2.

A graphical asset may be composed of multiple sub-elements, for example, a 3D furniture asset may include the model, as well as texture maps, and a character may be composed of face, hair, and body models. Producing such assets may require an approach that considers and generates all elements or multiple approaches that handle them separately. When the target asset is an arrangement, the user may also want to generate the objects that will be arranged; in this case, the objects can also be considered sub-elements for which an appropriate generation approach must be selected.

5 Input Types

Within the literature, there is 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) 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 [44, 103, 159, 268].

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 [65], inferred using deep learning [87], or optimised via evolutionary algorithm [179].

Text as an input has seen recent usage in deep-learning-based text-to-image transformation [206, 211, 213], 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 [157].

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 [28, 127, 266] or in combination with procedural modelling [87, 183].

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 to be scanned or photographed. Photographs are largely interpreted via deep learning [31, 131, 151, 180].

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 [29, 30, 65, 74].

6 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 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, then 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 3 presents the frequency 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.

6.1 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 licence 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 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 [206, 211, 213], though text-based generation is achieved in 3D applications, such as with texture and displacement maps [53] or full model generation [160]. Alternatively, photo-based 3D asset generation has seen considerable exploration, with single-view [131, 151, 180, 197, 210], multi-view [131], and scan-based generation of 3D assets [62, 133], and many utilising depth data from RGB-D images [295]. 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 [28, 283] or 2D [55, 209] 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 to produce novel outputs [59, 214, 246]. 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 [48, 230].

6.2 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 [136, 137], 3D interiors [139, 199], or 3D exteriors [239, 294]. *Patch-based/Partwise* asset

generation involves the piecing together of existing components into novel configurations. For instance, Reference [248] builds road networks out of pre-defined patches, and References [74, 124] 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. [260] 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 [172], where successful examples show consistency in their mappings [60, 146]. 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 [64], many works have followed [8, 61, 108], including approaches to caricature generation [84] and photo cartoonisation [228]. Furthermore, the concept of style transfer is extended into three dimensions with mesh texturisation [80] and functionality preserving stylisation [164].

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 [225]. 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 that directly impact the results of the noise generation [48, 216]. Some methods aim at converting existing meshes into *procedural* models [29, 30, 65], while others use these models as mediums for photo-based reconstruction [1, 225].

6.3 Select Techniques

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. If the technique is not the priority, then 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 time required.

There are three methods for obtaining input data: sourcing existing data, creating data, or automating the creation of data. As shown in Figure 4, when data instances already exist, such as 3D meshes within ShapeNet [18] or photographs found on the internet, and they are of acceptable quality and relevance, they may be used as inputs. If not, then 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 5 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 to create, automate, or source the inputs also grows.

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

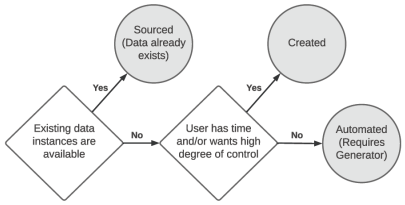


Fig. 4. Decision process for obtaining inputs.

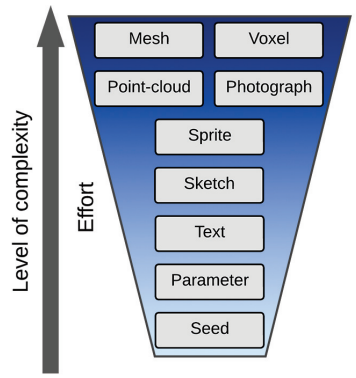


Fig. 5. Input complexity and effort.

Table 3. Frequency and Breakdown of Techniques within Categories

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

a target *asset type* using particular *inputs*. For improved presentation within the framework (Figure 2), approaches have been grouped and taxonomised under four main headings: *optimisation*, *stochastic*, *pattern-based*, and *deep learning*. Table 4 presents the frequency of each approach within the literature (many papers feature multiple times, as they demonstrate multiple techniques). Though in many cases a combination of approaches is 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.

7.1 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 fitness of candidates will improve, resulting in stronger (high fitness) candidates. The **Procedural Iterative Constrained Optimiser (PICO)** framework [124] is centred around a graph that

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
    end while
  end if
  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
      $input\_type_s \leftarrow IN[i]$
    end while
  end if
  return $technique_s, input\_type_s$ ▷ Pass selected technique and its input type to next step
end procedure

Table 4. Frequency and Breakdown of Approaches within Categories

Approach category	Approach	Freq.	Description
Optimisation	Evolutionary Algorithm	2	Iterative optimisation via mutation across generations.
	Genetic Algorithm	6	Evolutionary algorithm utilising crossover in population.
	Swarm Algorithm	1	Optimisation through swarm modelling (many agent).
	Combinatorial Optimisation	1	Optmising a series of decisions for given constraints.
	Topology Optimisation	1	Constraint-based optimisation of topology & volume.
	Expectation Maximisation	1	Iterative optimisation via estimation and adjustment.
Stochastic	Perlin Noise	4	Noise with a smooth gradient.
	Simplex Noise	1	Improved Perlin, faster and better quality.
	Voronoi/Worley Noise	1	From distances between cell-patterned random points.
Pattern-based (Growth/Simulation)	Cellular Automata	2	Iteratively applied grid adjacency and occupancy rules.
	Space colonisation	4	Iterative growth toward attraction points in a volume.
	Erosion	2	Simulation of terrain formation through natural forces.
	Deformation model	3	Manipulating existing shapes/designs with constraints.
	Deprojection	8	Non-DL reconstruction of 3D shape from 2D data.
	Diffusion/Propagation	7	Progressive manipulation of shape within a volume.
Pattern-based (Grammar)	L-System	5	Rule-based recursive command re-writing.
	Shape grammar	13	Grammars for combining/manipulating shapes.
	Graph grammar	1	Generative grammars for formulating graph data.
	Split grammar	1	Subdividing shapes into smaller elements via split rules.
	Stochastic grammar	2	Generative grammar applying probabilistic rules.
Deep Learning (Architectures)	CNN	37	Learning image features via convolutional filtering.
	R-CNN	4	Regional feature extraction via convolutional filtering.
	GCN	10	Learning graph features via convolutional filtering.
	RNN	1	Learning sequential data features.
Deep Learning (Methods)	Encoder-Decoder	42	Transformation via learning to encode then decode data.
	GAN	72	Unsupervised learning to fool a discriminator.
	RL	2	Reward/penalty-based learning in an environment.
	IL	1	Learning to mimic human choices or behaviour.

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)** [74] 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 (GAs)*** are a popular class of evolutionary algorithm, employed in the generation of buildings [170, 286], vehicles [7], props [117, 170], and clouds [179]. A CNN is used to learn a fitness function for the optimisation of cloud shapes, scoring generated clouds based on how real they appear [179]. Alternatively, *GAs* are applied in object placement [239]. In this method, an optimal scene layout is generated based on fitness to a set of positional rules defined by the authors. GAs have also been used for camera parameter estimation [170].

Interactive genetic algorithms (IGAs) integrate user input as part of the fitness calculation, which allows a user to influence the development of the asset. The 3DCSS framework [7], for example, successfully integrates *IGAs* into the car-design process. Alternatively, Reference [286] attempts 3D building generation with textures using *IGAs*. Though, in a user study, their *IGA* approach to texture generation was rated poorly, the generation of building models was effective. In another method [117], *IGAs* are 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 [111]. 3D asteroid meshes have been generated based on real data for the purpose of simulating traversal of such terrains [144]. 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 Reference [164] 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 Reference [110], wherein 3D solutions are generated based on user-provided sketches and constraints using the level-set method of Reference [3]. 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 Reference [129], is applied in 3D cloud generation using photographs [94].

7.2 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* [195], *Simplex noise* [196], and *Voronoi/Worley noise* [276]. 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 [167], 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 that can be applied via mesh surface

displacement. For example, Reference [48] employs *Simplex noise* in height map terrain generation as part of a multi-step pipeline for 3D environment generation, while Reference [144] uses *Simplex noise* to create surface variation on asteroid models, and Reference [216] uses *Simplex noise* for surface variation when generating volumetric caves. Alternatively, Reference [230] uses 3D *Perlin noise* to generate consistent height mapping around a spherical surface, and Reference [32] initialises 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, Reference [10] introduces 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 [83]. Extending to volumetric space allows for overhangs and tunnels in terrain. Montenegro et al. [177] propose a method, based on Reference [155], that 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 [50] and sprites for 2D game environments with *Perlin noise* [123].

7.3 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 [257] and Conway's Game of Life [63]. 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. [72] 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 [5]. 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, Reference [208] uses space colonisation for the real-time generation of trees, and Reference [75] combines 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 [297], and space colonisation is used in the generation of road networks [33], 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 [48] 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. The climate simulation models temperature, wind, and precipitation. Franke and Müller [50] 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 Reference [40], 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 Reference [260], 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. [46] 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, as demonstrated in Reference [62]. Effective RGB-D scanning pipelines are also suggested by References [182, 232], performing on par with other state-of-the-art photogrammetry approaches, while Reference [115] introduces a framework for simultaneously texturing and reconstructing scenes using RGB-D data. Alternatively, Reference [19] suggests 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 [229], refining and converting the result to a mesh via marching-cubes [161].

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 [214, 224, 295, 296]. Gao et al. [62] 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 [69].

Patch-based generation is applied to road networks and web design. In the method of Reference [248], 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 [163] learns layout constraints for positioning web elements.

Deriving from the work of Chomsky [23], *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. [134] propose a **probabilistic context-free grammar (PCFG)** for reconstructing buildings from images, learning rules from existing models, and Reference [168] introduces an approach to grammar learning that focuses on façades. Alternatively, Reference [17] attempts to reconstruct façades from low-resolution images, while Reference [97] employs 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 [41]. Demir et al. [29] 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 [30]. Grammars have also been applied in the generation of ancient Roman and Greek style structures [121]. Nishida et al. [183] 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 [11] 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 [184].

Grammars have also been applied in interior layout generation. For example, a stochastic grammar with **Spatial And-Or Graph (S-AOG)** is introduced in Reference [100]. 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. [51] introduce an algorithm for generating full building assets including interiors and textures. Alternatively, a **Scene Grammar Variational Autoencoder (SGVAE)** approach is introduced [199], 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 et al. [147] 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)** [275] 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 Reference [2] uses a generative grammar system that produces variations of products by combining predefined 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 [37]. **Geometric graph grammars (*GGGs*)** are applied to the generation of road networks [49]. The *GGG* extends the concept of a graph grammar by encoding geometric data alongside topology.

Volumetric terrain generation can be achieved via voxel grammars [32]. As a form of shape grammar, voxel grammars define rules that are applied to a given starting point. Alternatively, Reference [204] combines grammars with a swarm algorithm to generate entire environments consisting of terrain, vegetation, and bodies of water. *Swarm grammars* [256] 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 [153]. L-systems have since been adapted to three dimensions [81] and applied to tree reconstruction from photos [75]. They have also been applied in road generation [221] and combined with IGA for novel 3D shape generation [117].

7.4 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 to derive novel content. GANs, first introduced by Reference [70], 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 [44, 103, 159], sketch-based techniques [127, 299], and text-to-image generation [193, 202, 206, 213, 217, 291].

Basic GAN-based implementations have been applied in the generation of spell icons [103] and textures for games [44], aiding the process of design ideation [159] and generating images of indoor scenes [268]. Such generation is *seeded*, as content is produced by randomly sampling the extracted feature space to find novel content. GANs have been successfully applied in style-transfer tasks [8, 61, 228, 290] as well as image generation from sketch-input [127], art colourisation [299], and

caricature generation [84]. An influential framework for such approaches, Pix2Pix [92], employs a **conditional GAN (cGAN)**. cGANs take additional inputs, allowing the end-user to specify the kind of result generated. For example, Reference [270] conditions a face image generator on high-level attributes, such as age, gender, and hair colour. CONGAN [77] 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 [105] introduces a style-based GAN architecture to great effect, producing high-quality images of various types.

LayoutGAN [136] 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 [137]. User-sketch-based cGAN approaches have also been applied to the generation of height map-based terrains [190, 238, 266]. For example, Sketch2Map [266] allows designers to draw simple maps that represent terrains, while a similar sketch-based approach [38] allows users to sketch sections of terrain that are seamlessly joined. Zhang et al. [301] 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 describes its features and colours in text. Existing GAN-based attempts at generating novel images using text inputs include [291] CAGAN [217], SAM-GAN [193], CycleGAN [304], MirrorGAN [202], LeicaGAN [201], and ControlGAN [132]. These approaches extract semantic meaning from input text and apply appropriate transformations to an image via GAN-based architectures.

DALL-E [207] and CogView [34] 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 [206] and Imagen [213], make use of diffusion and CLIP [203] mechanisms to generate high-fidelity images from text. Stable-diffusion [211] successfully applies a diffusion-based approach that 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 [119, 120, 162, 245, 303], generation [140, 146, 227], and interpolation [227, 293, 303] of new mesh, point-cloud, and voxel assets. Furthermore, diffusion models have been applied in 3D shape generation [90, 293].

The **Sphere as Prior GAN (SP-GAN)** [140] is capable of generating point-clouds in a structure-aware manner, while SG-GAN [146] and HSGAN [145] generate point-clouds in topologically and hierarchically aware manners, respectively. Voxel generation is also achieved via GANs [231, 237, 261] and cGANs [185]. cGAN has also been applied to generating varied voxel-based rock shapes with user-defined boundaries [126].

Hertz et al. [80] 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 [138], which instead makes use of 2D displacement maps.

3D model conception can be a combined effort between user and machine. Davis et al. [26] introduce a VR-based co-creative AI that 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 [78, 106, 192] and 2D-to-3D style-transfer [106], improved upon with use of normal maps [278], and applied in game character face generation from photographs [225]. With a similar approach, GET3D [56] 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 [181] and façade image generation [244]. These methods are centred around the 2D domain, however, may still be applicable in combination with other

methods for generating 3D buildings. Alternatively, Reference [39] presents 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 [22, 173]. 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 [255]. U-net is a popular form of encoder-decoder [212]. 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 [211, 213], image translation [92], and interpreting sketches [27, 127]. In general, encoder-decoder networks have been successfully applied in photo-based [106, 131, 180, 197, 265, 283], sketch-based [25, 27, 28, 222, 283], and text-based 3D shape-generation tasks [16, 52, 95, 157, 160, 174, 215, 289].

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 [21, 59, 60, 101, 102, 141, 284], textures [59], characters [246], and hair [214].

Jones et al. [101] 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 [102].

SDM-NET [60] employs VAEs in learning the structure and geometry of objects, producing high-quality generated and interpolated results. This is developed further with TM-NET [59], which produces textured 3D models. A VAE is applied in the generation of novel object and character poses [246]; this incorporates the **rotation-invariant mesh difference (RIMD)** data representation [57], 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 [252]. The self-attention mechanism of these models allows for more effective sequence-to-sequence learning. In addition, U-net-based diffusion models have been successfully employed in text-to-3D [198, 282] and image reconstruction tasks [158, 200] via multi-view image generation. Mescheder et al. [171] 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 [194]. Similar implicit representations have also been used, such as IM-NET's implicit fields [20], and DMTet [223], 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.

Deep learning has been applied in the estimation of hair flow fields. Single-image hair reconstruction is achieved in this way using VAE [214] or GAN [296]. DeepSketchHair [224] 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, Reference [295] generates 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 References [89, 118, 154, 180, 281].

Pixel2Mesh [263, 264] 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 [35] and for multi-view reconstruction [273].

There are many other approaches that incorporate the template deformation concept. For instance, Image2Mesh [197] encodes an input image using a CNN, finds the closest base model, then deforms it via free-form-deformation [219] to match the input. A similar approach uses free-form-deformation to generate lung models from single-view images [272], while template mesh deformation is applied to liver [250] and heart [122] reconstruction.

EasyMesh [245] also takes a deformation approach while processing input images into silhouettes to gain consistency in training. Similar approaches use deformation to generate meshes [135, 143, 189] and point-clouds [292]. These approaches are limited to deforming existing topologies. Instead, Mesh R-CNN [68] is capable of reconstructing varying topologies from single-view images. This approach expands on Mask R-CNN [76] 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. [188] circumvent this limitation by introducing topology modification modules that 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 [265].

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 Reference [162], which utilises generated multi-view silhouette data. Yang et al. [283] 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 [271]. Lin et al. [152] 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. [279] address this latter issue with a novel weighted voxel representation.

Deformation of template meshes is also applied to photo reconstruction for characters [254, 288, 298] and hands [166]. Additionally, human body meshes have been reconstructed from photographs for the purpose of testing clothing fit in e-commerce [1]; while RODIN [267] 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 [151, 225]. Furthermore, the approach of Reference [151] is capable of extracting texture from an input image and applying it to a reconstructed face model. Fan et al. [45] introduce a pipeline for full head and face reconstruction based on 3DMM, and Reference [99] uses 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 Reference [113], where meshes are refined via displacement, and Reference [125],

where a GAN produces a depth map for a given image. Face generation can be achieved via GAN, such as Reference [125], or PGAN [132], which generates faces with the use of geometry images [73]. The approach of Reference [220] employs geometry images and a GAN architecture based on progressive growing GANs [104]. StyleGAN [105] has also been expanded upon for learning 3D-aware face generation [187]. In the method of Reference [142], 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 Reference [31]. Facial landmarks may be used to improve the accuracy of facial reconstruction methods. Cai et al. [15] introduce a method for automatically detecting these landmarks for caricatures. Some approaches take sketches as input; for example, in the method of Reference [27]. This method struggles with reconstructing thin structures, due to the resolution of the voxel space. This is addressed by combining voxels with normal maps [28], where an additional normal prediction network, based on Pix2Pix [92], 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. [285] introduce a method for generating human body meshes from sketches, and a CNN is applied in converting sketch data to procedural model parameters [87].

Some research attempts to extract maximal information from a scene, attempting to either understand a scene holistically or identify and extract individual items. CDMD3DM [133], for example, reconstructs small-scale indoor scenes using RGB-D data as an input, and Reference [96] produces accurate texture maps for RGB-D-based scene generation. Full scenes have been reconstructed from single-view images using cGAN architectures [119, 120] and CNN-based approaches [274]. Some methods conceive scenes and objects using GAN-based approaches, including point-clouds of outdoor scenes [226], entire cities from single-view images [116], and voxel-based scenes that are segmented by object class [231]. Alternatively, full scenes and individual asset classes can be successfully generated by utilising **large language models (LLMs)**. 3D-GPT [243] achieves this by using LLMs as decision-making agents that select a sequence of functions and inputs for the procedural modelling framework Infinigen [205]. 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 [98, 139, 262, 300] and case-based reasoning [235, 236] to the task. SceneHGN takes a hierarchical approach to indoor scene generation, recursively breaking down the task into room, functional regions, objects, and object parts [58]. In addition, transformer models have been successfully applied to interior arrangement tasks, such as ATISS [191] and SceneFormer [269], 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 [247]. Floor plans can also be generated based on layout graphs via graph neural networks, as presented in Graph2Plan [86].

AAEs combine the encoder-decoder concept with the the adversarial mechanic of GANs. For example, Reference [302] applies AAE to style transfer, in which the latent representations of content and style images are learned within an encoder-decoder, evaluated by a discriminator. AAEs 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 [300]. 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 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 Reference [150] 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 [85]. In this approach, the network attempts to find an optimal layout for a randomised set of elements.

7.5 Determine Approaches

To determine an approach for a given task, the pool of existing generative approaches is 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 = [∀ app ∈ A | app.input_type = in ∧ app.technique = t ∧ 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| ∧ A[i] ≠ u.approach do                ▷ Allow user to choose approach
            approach_s ← A[i]
            i = i + 1
        end while
    end if
    return approach_s                                    ▷ Pass selected approach to next step
end procedure
```

7.6 Choose Datasets and Train

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 labelled or unlabelled datasets, of which many exist publicly. For 3D shape generation, ShapeNet [18], ModelNet [277], or PartNet [175] may be used; while, for 2D tasks, datasets such as ImageNet [47] are a common choice. There are many datasets for specific asset types, such as shoes [287], birds [259], and human poses [14], as discussed in Reference [54]. 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, such as in the case of References [100, 114]. 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. [242] 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. In the process of aggregating methods for graphical asset generation broadly, we presented a collated list of datasets that can be applied to specific asset types from across the range of application domains [54]. These datasets have relevance in training and assessing deep-learning-based generators that utilise the same input/output scheme as similar approaches.

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. [150] 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 6 presents the proposed process for choosing or creating a dataset 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, then 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, then 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.

8 Generate Assets Process

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, then 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.

9 Output Data Formats

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) 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

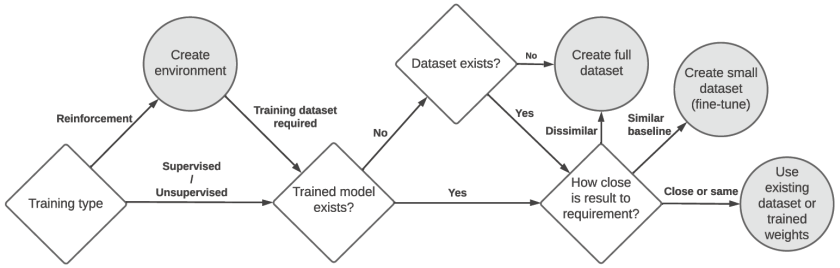


Fig. 6. Approach to choosing or creating a dataset.

presenting 3D shapes by volume, allowing for overhangs and tunnels in terrains [32, 230]. It is also common to convert voxel data to mesh data using marching cubes [161] or surface nets [66]. 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, such as in References [26, 109]. This is often a requirement for 3D content in real-time rendering and 3D printing. Alternatively, a polygon soup lacks full connectivity, seen in Reference [74]. 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 [186]. 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 [146, 149, 186, 227, 292].

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 [89, 90] or **signed distance fields (SDF)** [56, 187, 232, 303], these forms are not used to represent final *useable* assets and are therefore not deemed graphical asset formats.

9.1 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) 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 [161]. The inverse conversion, mesh to voxel, is achieved through voxelisation [107]. Similarly, point-cloud voxelisation is achievable using the method of Reference [82]. Conversion from mesh to point-cloud can be achieved through random point sampling [24], and conversion of point-cloud data to mesh data can be achieved using Poisson surface reconstruction [109].

For 2D formats, the process of converting bitmap data to vector data, vectorisation [36, 234], and the inverse, rasterisation [218], can be applied. Conversion from 3D to 2D can also be achieved

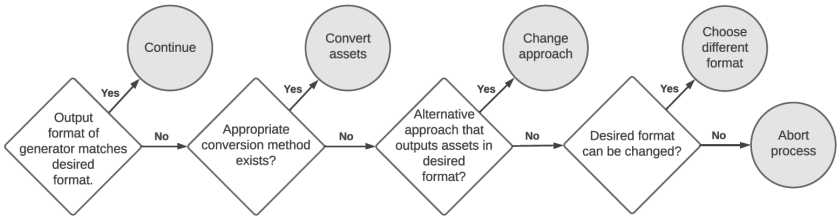


Fig. 7. Process for selecting a conversion method.

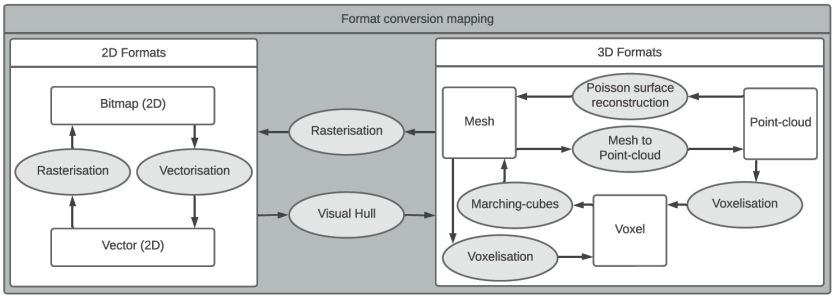


Fig. 8. The mapping between conversion methods and formats.

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 [130], which requires multiple images at different viewing angles. Alternatively, deep-learning generative approaches, such as Reference [222], 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.

9.2 Verifying and Converting the Data Format

Figure 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 reconsider the desired format.

Figure 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 it may be rasterised. Likewise, when converting from 2D to 3D using visual hull [130], any vector graphics must be converted to bitmap.

10 Evaluating Generative Methods

It is important to consider the efficacy of any resulting generator to ensure the quality of generated artefacts. This requires standardised evaluation metrics for benchmarking and comparison with

existing or future methods. Generally, PCG methods can be evaluated along the lines suggested by References [71, 241]. Furthermore, for graphical assets specifically, Reference [54] examines the relevant approach to evaluation based on three aspects: *Artefact Validation*, the importance of matching the intended asset type, e.g., "does the car generator produce assets that look like cars?"; *Artefact Quality*, "does the generator produce quality assets?"; and *Performance*, "is the generator fast and resource-efficient?"

11 Concluding Discussion

Current methods for generating and transforming graphical assets are dispersed across a wide array of applications. While this suggests that graphical asset generation is valuable in many contexts, there is no centralised way of finding the available approaches and techniques, and furthermore, no standardised process for selecting or devising a method for a particular task. To address this gap, a systematic literature review was conducted, bringing together the state-of-the-art in generative methods for graphical asset creation. We reviewed 239 papers, in which methods for generating 21 types of asset were explored, identifying 9 forms of technique and 28 types of approach. The GAGeTx framework assembles the findings with a process for building graphical asset generators or transformers based on the needs of the user or practitioner. The goal is to make the breadth of current methods accessible to interested parties, encourage cross-over of techniques and approaches, and ensure the visibility of state-of-the-art capabilities across disciplines. Though generative methods are rapidly evolving in the direction of deep learning, there is still much potential in applying existing methods cross-discipline. Areas where large quantities of graphical assets are required, such as game development, may benefit from such generative tools and automation. In this case, efforts must be made to establish the needs for such specialised users.

As for GAGeTx, more work is required to validate and expand the framework. In its current iteration, GAGeTx does not provide a way for selecting the best approach for a task, rather, it leaves this decision up to the user. To address this, future work will include the formation of a method for comparing each generative approach. Furthermore, the framework should be validated through usage. The breadth of techniques and approaches will undoubtedly continue to grow and progress with future advancements and with it the framework would evolve. Finally, implementation of the framework as a tool may maximise the impact and effectiveness of the framework.

References

[1] Mohamed Abdelaziz, Mohamed Ayman, Mohamed Osama, Tarek Medhat, Hager Sobeah, Maha Sayed, and Khaled Hussein. 2021. Generating 3D model for human body shapes from 2D images using deep learning. In *International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC'21)*. 291–295.

[2] Jorge Alcaide-Marzal, Jose Antonio Diego-Mas, and Gonzalo Acosta-Zazueta. 2020. A 3D shape generative method for aesthetic product design. *Des. Stud.* 66 (2020), 144–176.

[3] Grégoire Allaire, François Jouve, and Anca-Maria Toader. 2002. A level-set method for shape optimization. *Comptes Rendus Mathematique* 334, 12 (2002), 1125–1130.

[4] Nantheera Anantrasirichai and David Bull. 2022. Artificial intelligence in the creative industries: A review. *Artif. Intell. Rev.* 55, 1 (2022), 589–656.

[5] Izabella Antoniuk and Przemyslaw Rokita. 2016. Generation of complex underground systems for application in computer games with schematic maps and L-systems. *Lect. Notes Comput. Sci.* 9972 LNCS (2016), 3–16.

[6] Autodesk. 2024. Autodesk Media & Entertainment Collection. Retrieved from https://www.autodesk.co.uk/collections/media-entertainment/

[7] Seonghoon Ban and Kyung Hoon Hyun. 2020. 3D computational sketch synthesis framework: Assisting design exploration through generating variations of user input sketch and interactive 3D model reconstruction. *CAD Comput. Aid. Des.* 120 (2020), 102789.

[8] Noa Barzilay, Tal Berkovitz Shalev, and Raja Giryes. 2021. MISS GAN: A multi-IlluStrator style generative adversarial network for image to illustration translation. *Pattern Recog. Lett.* 151 (2021), 140–147. arXiv:2108.05693

[9] Bay 12 Games. 2006. Dwarf Fortress. Retrieved from http://www.bay12games.com/dwarves/

[10] Michael Becher, Michael Krone, Guido Reina, and Thomas Ertl. 2017. Feature-based volumetric terrain generation. *21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D'17)*. 1–9.
[11] Manush Bhatt, Rajesh Kalyanam, Gen Nishida, Liu He, Christopher May, Dev Niyogi, and Daniel Aliaga. 2020. Design and deployment of Photo2Building: A cloud-based procedural modeling tool as a service. In *Practice & Experience in Advanced Research Computing (PEARC'20)*. 132–138.
[12] Blender Foundation. 2024. Blender. Retrieved from https://www.blender.org/
[13] Blizzard North. 2000. Diablo 2. Blizzard Entertainment.
[14] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014. FAUST Dataset. Retrieved from http://faust.is.tue.mpg.de/
[15] Hongrui Cai, Yudong Guo, Zhuang Peng, and Juyong Zhang. 2021. Landmark detection and 3D face reconstruction for caricature using a nonlinear parametric model. *Graphic. Mod.* 115 (2021), 101103. arXiv:2004.09190
[16] Zehranaz Canfes, M. Furkan Atasoy, Alara Dirik, and Pinar Yanardag. 2023. Text and image guided 3D avatar generation and manipulation. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV'23)*. 4421–4431.
[17] Jun Cao, Henning Metzmacher, James O'Donnell, Jérôme Frisch, Vladimir Bazjanac, Leif Kobbelt, and Christoph van Treeck. 2017. Facade geometry generation from low-resolution aerial photographs for building energy modeling. *Build. Environ.* 123 (2017), 601–624.
[18] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet Dataset. Retrieved from https://shapenet.org/
[19] Chih Fan Chen and Evan Suma Rosenberg. 2018. Dynamic omnidirectional texture synthesis for photorealistic virtual content creation. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct'18)*. 85–90.
[20] Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*.
[21] An-Chieh Cheng, Xueting Li, Sifei Liu, Min Sun, and Ming-Hsuan Yang. 2022. Autoregressive 3D shape generation via canonical mapping. In *Computer Vision – ECCV*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 89–104.
[22] Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
[23] Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press. 251 pages.
[24] CloudCompare. 2015. Mesh-sample Points. Retrieved from https://www.cloudcompare.org/doc/wiki/index.php/Mesh%5CSample_points
[25] Ayan Das, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Yi-Zhe Song. 2021. Cloud2Curve: Generation and vectorization of parametric sketches. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'21)*. IEEE, 7084–7093. arXiv:2103.15536
[26] Josh Urban Davis, Fraser Anderson, Merten Stroetzel, Tovi Grossman, and George Fitzmaurice. 2021. Designing co-creative AI for virtual environments. In *Conference on Creativity and Cognition (C&C'21)*. 1–11.
[27] Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei A. Efros, and Adrien Bousseau. 2018. 3D sketching using multi-view deep volumetric prediction. *Proc. ACM Comput. Graph. Interact. Techniq.* 1, 1 (2018), 1–22.
[28] Johanna Delanoy, David Coeurjolly, Jacques Olivier Lachaud, and Adrien Bousseau. 2019. Combining voxel and normal predictions for multi-view 3D sketching. *Comput. Graph.* 82 (2019), 65–72.
[29] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes. 2016. Proceduralization for editing 3D architectural models.In *4th International Conference on 3D Vision (3DV'16)*. 194–202.
[30] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes. 2017. Proceduralization of urban models. In *25th Signal Processing and Communications Applications Conference (SIU'17)*. 1–4.
[31] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. 2019. Accurate 3D face reconstruction with weakly-supervised learning: From single image to image set. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'19)*. 285–295.
[32] Rahul Dey, Jason G. Doig, and Christos Gatzidis. 2018. Procedural feature generation for volumetric terrains using voxel grammars. *Entert. Comput.* 27, Dec. 2017 (2018), 128–136.
[33] Gabriel Dias Fernandes and António Ramires Fernandes. 2018. Space colonisation for procedural road generation. In *International Conference on Graphics and Interaction (ICGI'18)*.
[34] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. 2021. CogView: Mastering text-to-image generation via transformers. *Advan. Neural Inf. Process. Syst.* 34 (2021), 19822–19835. arXiv:2105.13290
[35] Yang Dongsheng, Kuang Ping, and Xiaofeng Gu. 2020. 3D reconstruction based on GAT from a single image. In *17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP'20)*. 122–125.

[36] D. Dori and Wenyin Liu. 1999. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* 21, 3 (1999), 202–215.
[37] Wallas H.S. Dos Santos, Paulo Ivson, and Alberto Barbosa Raposo. 2017. CAD shape grammar: Procedural generation for massive CAD model. In *30th Conference on Graphics, Patterns and Images (SIBGRAPI'17)*. 31–38.
[38] Xue Mei Du, Fan Li, Hua Rui Yan, Rong Fu, and Yang Zhou. 2019. Terrain edge stitching based on least squares generative adversarial networks. In *16th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP'19)*. 157–161.
[39] Zhenlong Du, Haiyang Shen, Xiaoli Li, and Meng Wang. 2022. 3D building fabrication with geometry and texture coordination via hybrid GAN. *J. Amb. Intell. Human. Comput.* (2022), 1–12.
[40] Marek Dvorožňák, Daniel Sýkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. 2020. Monster mash: A single-view approach to casual 3D modeling and animation. *ACM Trans. Graph.* 39, 6 (2020), 1–12.
[41] Johannes Edelsbrunner, Sven Havemann, Alexei Sourin, and Dieter W. Fellner. 2016. Procedural modeling of round building geometry. In *International Conference on Cyberworlds (CW'16)*. 81–88.
[42] Satu Elo and Helvi Kyngäs. 2008. The qualitative content analysis process. *J. Advan. Nurs.* 62, 1 (2008), 107–115.
[43] Epic Games. 2024. Unreal Engine. Retrieved from https://www.unrealengine.com/en-US
[44] Amin Fadaeddini, Babak Majidi, and Mohammad Eshghi. 2018. 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'18)*. IEEE, 118–122.
[45] Yangyu Fan, Yang Liu, Guoyun Lv, Shiya Liu, Gen Li, and Yanhui Huang. 2020. Full face-and-head 3D model with photorealistic texture. *IEEE Access* 8, 4 (2020), 210709–210721.
[46] Andrey Fedorov, Ivan Ivashnev, Valery Afanasiev, Valery Krivtsov, Aleksandr Zatolokin, and Sergey Zyrin. 2019. Interactive reconstruction of the 3D-models using single-view images and user markup. In *2nd International Conference on Image and Graphics Processing*. 73–77.
[47] Li Fei-Fei, Jia Deng, Olga Russakovsky, Alex Berg, and Kai Li. 2021. ImageNet Dataset. Retrieved from https://image-net.org/
[48] Roland Fischer, Philipp Dittmann, René Weller, and Gabriel Zachmann. 2020. AutoBiomes: Procedural generation of multi-biome landscapes. *Vis. Comput.* 36, 10–12 (2020), 2263–2272.
[49] Marek Fiser, Bedrich Benes, Jorge Garcia Galicia, Michel Abdul-Massih, Daniel G. Aliaga, and Vojtech Krs. 2016. Learning geometric graph grammars. In *32nd Spring Conference on Computer Graphics (SCCG'16)*. 7–15.
[50] Kai Franke and Heinrich Müller. 2022. Procedural generation of 3D karst caves with speleothems. *Comput. Graph.* 102 (2022), 533–545.
[51] Jonas Freiknecht and Wolfgang Effelsberg. 2020. Procedural generation of multistory buildings with interior. *IEEE Trans. Games* 12, 3 (2020), 323–336.
[52] Rao Fu, Xiao Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. 2022. ShapeCrafter: A recursive text-conditioned 3d shape generation model. *Advances in Neural Information Processing Systems* 35 (2022), 8882–8895.
[53] Yuta Fukatsu and Masaki Aono. 2021. 3D mesh generation by introducing extended attentive normalization. In *8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA'21)*. 1–6.
[54] Kaisei Fukaya, Damon Daylamani-Zad, and Harry Agius. 2024. Evaluation metrics for intelligent generation of graphical game assets: A systematic survey-based framework. *IEEE Trans. Pattern Anal. Mach. Intell.* 46, 12 (2024), 7998–8017.
[55] Yuuya Fukumoto, Daiki Shimizu, and Chihiro Shibata. 2018. Generation of character illustrations from stick figures using a modification of generative adversarial network. In *IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC'18)*. 183–186.
[56] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. 2022. GET3D: A generative model of high quality 3D textured shapes learned from images. In *International Conference on Neural Information Processing Systems (NIPS'22)*.
[57] Lin Gao, Yu-Kun Lai, Dun Liang, Shu-Yu Chen, and Shihong Xia. 2016. Efficient and flexible deformation representation for data-driven surface modeling. *ACM Trans. Graph.* 35, 5 (2016), 1–17.
[58] Lin Gao, Jia-Mu Sun, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, and Jie Yang. 2023. SceneHGN: Hierarchical graph networks for 3D indoor scene generation with fine-grained geometry. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 7 (2023), 8902–8919.
[59] Lin Gao, Tong Wu, Yu-Jie Yuan, Ming-Xian Lin, Yu-Kun Lai, and Hao Zhang. 2021. TM-NET: Deep generative networks for textured meshes. *ACM Trans. Graph.* 40, 6 (2021), 1–15. arXiv:2010.06217
[60] Lin Gao, Jie Yang, Tong Wu, Yu Jie Yuan, Hongbo Fu, Yu Kun Lai, and Hao Zhang. 2019. SDM-NET: Deep generative network for structured deformable mesh. *ACM Trans. Graph.* 38, 6 (2019), 1–15. arXiv:1908.04520
[61] Xiang Gao, Yingjie Tian, and Zhiquan Qi. 2020. RPD-GAN: Learning to draw realistic paintings with generative adversarial network. *IEEE Trans. Image Process.* 29 (2020), 8706–8720.

[62] Yang Gao, Yuan Yao, and Yunliang Jiang. 2019. Multi-target 3D reconstruction from RGB-D data. In *2nd International Conference on Computer Science and Software Engineering (CSSE'19)*. 184–191.
[63] Martin Gardner. 1970. The fantastic combinations of John Conway's new solitaire game "Life." *Scient. Amer.* 223 (1970), 120–123.
[64] Leon Gatys, Alexander Ecker, and Matthias Bethge. 2016. A neural algorithm of artistic style. *J. Vis.* 16, 12 (2016), 326–326.
[65] Roman Getto, Arjan Kuijper, and Dieter W. Fellner. 2020. Automatic procedural model generation for 3D object variation. *Vis. Comput.* 36, 1 (2020), 53–70.
[66] Sarah F. F. Gibson. 1998. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'98)*, William M. Wells, Alan Colchester, and Scott Delp (Eds.). Springer, 888–898.
[67] Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. 2021. Adversarial reinforcement learning for procedural content generation. In *IEEE Conference on Games (CoG'21)*. IEEE Press, 1–8.
[68] Georgia Gkioxari, Justin Johnson, and Jitendra Malik. 2019. Mesh R-CNN. In *IEEE/CVF International Conference on Computer Vision (ICCV'19)*. IEEE, 9784–9794.
[69] Kirill Golubev, Aleksander Zagarskikh, and Andrey Karsakov. 2016. Dijkstra-based terrain generation using advanced weight functions. *Proced. Comput. Sci.* 101 (2016), 152–160.
[70] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *International Conference on Neural Information Processing Systems (NIPS'14)*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc.
[71] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. 2019. Procedural content generation through quality diversity. In *IEEE Conference on Games (CoG'19)*. IEEE Press, 1–8.
[72] Michael Cerny Green, Christoph Salge, Julian Togelius, and Michael Cerny Green. 2019. Organic building generation in Minecraft. *14th International Conference on the Foundations of Digital Games*. 1–7.
[73] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. 2002. Geometry images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. 355–361.
[74] Yanran Guan, Han Liu, Kun Liu, Kangxue Yin, Ruizhen Hu, Oliver van Kaick, Yan Zhang, Ersin Yumer, Nathan Carr, Radomir Mech, and Hao Zhang. 2020. FAME: 3D shape generation via functionality-aware model evolution. *IEEE Trans. Visualiz. Comput. Graph.* 2626 (2020), 1–1. arXiv:2005.04464
[75] Jianwei Guo, Shibiao Xu, Dong-Ming Yan, Zhanglin Cheng, Marc Jaeger, and Xiaopeng Zhang. 2020. Realistic procedural plant modeling from multiple view images. *IEEE Trans. Visualiz. Comput. Graph.* 26, 2 (2020), 1372–1384.
[76] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV'17)*. 2980–2988.
[77] Eric Heim. 2019. Constrained generative adversarial networks for interactive image generation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 10745–10753. arXiv:1904.02526
[78] Paul Henderson, Vagia Tsiminaki, and Christoph H. Lampert. 2020. Leveraging 2D data to learn textured 3D mesh generation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 7495–7504. arXiv:2004.04180
[79] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. *ACM Trans. Multim. Comput. Commun. Appl.* 9, 1 (2013), 1–22.
[80] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. Deep geometric texture synthesis. *ACM Trans. Graph.* 39, 4 (2020), 1–11. arXiv:2007.00074
[81] Eka Wahyu Hidayat, Ida Ayu Dwi Giriantari, Made Sudarma, and I. Made Oka Widyantara. 2020. Visualization of a three-dimensional tree modeling using fractal based on L-System. In *IEEE International Conference on Sustainable Engineering and Creative Computing (ICSECC'20)*. 418–422.
[82] Tommy Hinks, Hamish Carr, Linh Truong-Hong, and Debra F. Laefer. 2013. Point cloud data conversion into solid models via point-based voxelization. *J. Survey. Eng.* 139, 2 (2013), 72–83.
[83] Houssam Hnaidi, Eric Guérin, Samir Akkouche, Adrien Peytavie, and Eric Galin. 2010. Feature based terrain generation using diffusion equation. *Comput. Graph. Forum* 29 (2010), 2179–2186.
[84] Haodi Hou, Jing Huo, Jing Wu, Yu Kun Lai, and Yang Gao. 2021. MW-GAN: Multi-Warping GAN for caricature generation with multi-style geometric exaggeration. *IEEE Trans. Image Process.* 30 (2021), 8644–8657. arXiv:2001.01870
[85] Hao Hu, Chao Zhang, and Yanxue Liang. 2021. A study on the automatic generation of banner layouts. *Comput. Electric. Eng.* 93 (2021), 107269.
[86] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. 2020. Graph2Plan: Learning floorplan generation from layout graphs. *ACM Trans. Graph.* 39, 4 (2020), 118:1–118:14.

[87] Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. 2017. Shape synthesis from sketches via procedural models and convolutional networks. *IEEE Trans. Visualiz. Comput. Graph.* 23, 8 (2017), 2003–2013.
[88] Rowan T. Hughes, Liming Zhu, and Tomasz Bednarz. 2021. Generative adversarial networks–enabled human–artificial intelligence collaborative applications for creative and design industries: A systematic review of current approaches and trends. *Front. Artif. Intell.* 4 (2021), 1–17.
[89] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. 2022. Neural template: Topology-aware reconstruction and disentangled generation of 3D meshes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 18572–18582.
[90] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. 2022. Neural wavelet-domain diffusion for 3D shape generation. In *SIGGRAPH Asia Conference Papers (SA'22)*. ACM, Article 24, 9 pages.
[91] Interactive Data Visualization Inc. (IDV). 2024. SpeedTree. Retrieved from https://store.speedtree.com/
[92] Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 5967–5976. arXiv:1611.07004
[93] Georgi Ivanov, Magnus Håkon Petersen, Kristián Kovalský, Kristian Engberg, George Palamas, and Kristian Eng-Berg. 2020. An explorative design process for game map generation based on satellite images and playability factors. In *International Conference on the Foundations of Digital Games*.
[94] Kei Iwasaki, Yoshinori Dobashi, and Makoto Okabe. 2017. Example-based synthesis of three-dimensional clouds from photographs. In *Computer Graphics International Conference (CGI'17)*. ACM, Article 28, 6 pages.
[95] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. 2022. Zero-shot text-guided object generation with dream fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 867–876.
[96] Junho Jeon, Yeongyu Jung, Haejoon Kim, and Seungyong Lee. 2016. Texture map generation for 3D reconstructed scenes. *Vis. Comput.* 32, 6–8 (2016), 955–965.
[97] Diego Jesus, António Coelho, and António Augusto Sousa. 2016. Layered shape grammars for procedural modelling of buildings. *Vis. Comput.* 32, 6–8 (2016), 933–943.
[98] Bruno Ježek, Adam Ouhrabka, and Antonin Slabý. 2020. Procedural content generation via machine learning in 2D indoor scene. In *Augmented Reality, Virtual Reality, and Computer Graphics*, Lucio Tommaso De Paolis and Patrick Bourdot (Eds.). Springer, 34–49.
[99] Penglei Ji, Ming Zeng, and Xinguo Liu. 2020. View consistent 3D face reconstruction using siamese encoder-decoders. *Commun. Comput. Inf. Sci.* 1314 CCIS (2020), 209–223.
[100] Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. 2018. Configurable 3D scene synthesis and 2D image rendering with per-pixel ground truth using stochastic grammars. *Int. J. Comput. Vis.* 126, 9 (2018), 920–941.
[101] R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. 2020. ShapeAssembly: Learning to generate programs for 3D shape structure synthesis. *ACM Trans. Graph.* 39, 6 (2020), 1–20. arXiv:2009.08026
[102] R. Kenny Jones, David Charatan, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. 2021. ShapeMOD: Macro operation discovery for 3D shape programs. *ACM Trans. Graph.* 40, 4 (2021), 1–16. arXiv:2104.06392
[103] Rafal Karp and Zaneta Swiderska-Chadaj. 2021. Automatic generation of graphical game assets using GAN. In *7th International Conference on Computer Technology Applications (ICCTA'21)*. 7–12.
[104] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive growing of GANs for improved quality, stability, and variation. In *6th International Conference on Learning Representations (ICLR'18)*. 1–26. arXiv:1710.10196
[105] Tero Karras, Samuli Laine, and Timo Aila. 2021. A style-based generator architecture for generative adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 12 (2021), 4217–4228.
[106] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D mesh renderer. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 3907–3916. arXiv:1711.07566
[107] A. Kaufman, D. Cohen, and R. Yagel. 1993. Volume graphics. *Computer* 26, 7 (1993), 51–64.
[108] Hadi Kazemi, Seyed Mehdi Iranmanesh, and Nasser M. Nasrabadi. 2019. Style and content disentanglement in generative adversarial networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV'19)*. 848–856. arXiv:1811.05621
[109] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *4th Eurographics Symposium on Geometry Processing*. 61–70.
[110] Rubaiat Habib Kazi, Tovi Grossman, Hyunmin Cheong, Ali Hashemi, and George Fitzmaurice. 2017. DreamSketch: Early stage 3D design explorations with sketching and generative design. In *30th Annual ACM Symposium on User Interface Software and Technology (UIST'17)*. 401–414.

[111] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *International Conference on Neural Networks (ICNN'95)*. 1942–1948.
[112] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. PCGRL: Procedural content generation via reinforcement learning. *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entert.* 16, 1 (2020), 95–101.
[113] Asad Khan, Sakander Hayat, Muhammad Ahmad, Jinde Cao, Muhammad Faizan Tahir, Asad Ullah, and Muhammad Sufyan Javed. 2021. Learning-detailed 3D face reconstruction based on convolutional neural networks from a single image. *Neural Comput. Applic.* 33, 11 (2021), 5951–5964.
[114] Samin Khan, Buu Phan, Rick Salay, and Krzysztof Czarnecki. 2019. ProcSy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19) Workshops*.
[115] Jungeon Kim, Hyomin Kim, Hyeonseo Nam, Jaesik Park, and Seungyong Lee. 2022. TextureMe: High-quality textured scene reconstruction in real time. *ACM Trans. Graph.* 41, 3 (2022), 1–18.
[116] Suzi Kim, Dodam Kim, and Sunghee Choi. 2020. CityCraft: 3D virtual city creation from a single image. *Vis. Comput.* 36, 5 (2020), 911–924.
[117] Mariatul Kiptiah Binti Ariffin, Shiqah Hadi, and Somnuk Phon-Amnuaisuk. 2017. Evolving 3D models using interactive genetic algorithms and L-systems. *Lect. Notes Comput. Sci.* 10607, LNAI, i (2017), 485–493.
[118] Roman Klokov, Edmond Boyer, and Jakob Verbeek. 2020. Discrete point flow networks for efficient point cloud generation. *Lect. Notes Comput. Sci.* 12368, LNCS (2020), 694–710. arXiv:2007.10170
[119] Vladimir V. Kniaz, Vladimir A. Knyaz, Fabio Remondino, Artem Bordodymov, and Petr Moshkantsev. 2020. Image-to-voxel model translation for 3D scene reconstruction and segmentation. *Lect. Notes Comput. Sci.* 12352, LNCS (2020), 105–124.
[120] Vladimir A. Knyaz, Vladimir V. Kniaz, and Fabio Remondino. 2019. Image-to-voxel model translation with conditional adversarial networks. *Lect. Notes Comput. Sci.* 11129, LNCS, 1 (2019), 601–618.
[121] Richard Konečný, Stella Syllaiou, and Fotis Liarokapis. 2016. Procedural modeling in archaeology: Approximating ionic style columns for games. In *8th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games'16)*. 1–8.
[122] Fanwei Kong, Nathan Wilson, and Shawn Shadden. 2021. A deep-learning approach for direct whole-heart mesh reconstruction. *Med. Image Anal.* 74 (2021), 102222. arXiv:2102.07899
[123] Oliver Korn, Michael Blatz, Adrian Rees, Jakob Schaal, and Korion Gmbh. 2017. Procedural content generation for game props? A study on the effects on user experience. *Comput. Entert.* 15, 2 (2017), 1–15.
[124] Vojtech Krs, Radomir Mech, Mathieu Gaillard, Nathan Carr, and Bedrich Benes. 2021. PICO: Procedural iterative constrained optimizer for geometric modeling. *IEEE Trans. Visualiz. Comput. Graph.* 27, 10 (2021), 3968–3981.
[125] Hailan Kuang, Yiran Ding, Xiaolin Ma, and Xinhua Liu. 2019. 3D face reconstruction with texture details from a single image based on GAN. In *11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA'19)*. 385–388.
[126] Ping Kuang, Dingli Luo, and Haoshuang Wang. 2019. Masked 3D conditional generative adversarial network for rock mesh generation. *Cluster Comput.* 22, s6 (2019), 15471–15481.
[127] Bipin Kuriakose, Theres Thomas, Nikitha Elsa Thomas, Sharon John Varghese, and Veena A. Kumar. 2020. Synthesizing images from hand-drawn sketches using conditional generative adversarial networks. In *International Conference on Electronics and Sustainable Communication Systems (ICESC'20)*. 774–778.
[128] Damian Kutzias and Sebastian von Mammen. 2023. Recent advances in procedural generation of buildings: From diversity to integration. *IEEE Trans. Games* 16, 1 (2023), 16–35.
[129] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. 2005. Texture optimization for example-based synthesis. *ACM Trans. Graph.* 24, 3 (2005), 795–802.
[130] A. Laurentini. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.* 16, 02 (1994), 150–162.
[131] Jiahui Lei, Srinath Sridhar, Paul Guerrero, Minhyuk Sung, Niloy Mitra, and Leonidas J. Guibas. 2020. Pix2Surf: Learning parametric 3D surface models of objects from images. *Lect. Notes Comput. Sci.* 12363, LNCS (2020), 121–138.
[132] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. Torr. 2019. Controllable text-to-image generation. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2065–2075.
[133] Chang Li, Tianrong Guan, Meng Yang, and Ce Zhang. 2021. Combining data-and-model-driven 3D modelling (CDMD3DM) for small indoor scenes using RGB-D data. *ISPRS J. Photogram. Rem. Sens.* 180 (2021), 1–13.
[134] Dan Li, Disheng Hu, Yuke Sun, and Yingsong Hu. 2018. 3D scene reconstruction using a texture probabilistic grammar. *Multim. Tools Applic.* 77, 21 (2018), 28417–28440.
[135] Hai Li, Weicai Ye, Guofeng Zhang, Sanyuan Zhang, and Hujun Bao. 2020. Saliency guided subdivision for single-view mesh reconstruction. In *International Conference on 3D Vision (3DV'20)*. 1098–1107.

[136] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. 2021. LayoutGAN: Synthesizing graphic layouts with vector-wireframe adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 7 (2021), 2388–2399.
[137] Jianan Li, Jimei Yang, Jianming Zhang, Chang Liu, Christina Wang, and Tingfa Xu. 2021. Attribute-conditioned layout GAN for automatic graphic design. *IEEE Trans. Visualiz. Comput. Graph.* 27, 10 (2021), 4039–4048. arXiv:2009.05284
[138] Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. 2018. OptCuts: Joint optimization of surface cuts and parameterization. *ACM Trans. Graph.* 37, 6 (2018), 1–13.
[139] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. 2019. GRAINS: Generative recursive autoencoders for indoor scenes. *ACM Trans. Graph.* 38, 2 (2019), 1–16. arXiv:1807.09193
[140] Ruihui Li, Xianzhi Li, Ka Hei Hui, and Chi Wing Fu. 2021. SP-GAN: Sphere-guided 3D shape generation and manipulation. *ACM Trans. Graph.* 40, 4 (2021), 1–12. arXiv:2108.04476
[141] Shidi Li, Miaomiao Liu, and Christian Walder. 2022. EditVAE: Unsupervised parts-aware controllable 3D point cloud shape generation. *Proc. AAAI Conf. Artif. Intell.* 36, 2 (2022), 1386–1394.
[142] Song Li, Songzhi Su, Juncong Lin, Guorong Cai, and Li Sun. 2021. Deep 3D caricature face generation with identity and structure consistency. *Neurocomputing* 454 (2021), 178–188.
[143] Xi Li, Kuang Ping, Xiaofeng Gu, and Mingyun He. 2020. 3D shape reconstruction of furniture object from a single real indoor image. In *17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP'20)*. 101–104.
[144] Xi Zhi Li, Rene Weller, and Gabriel Zachmann. 2018. AstroGen—Procedural generation of highly detailed asteroid models. In *15th International Conference on Control, Automation, Robotics and Vision (ICARCV'18)*. 1771–1778.
[145] Yushi Li and George Baciu. 2021. HSGAN: Hierarchical graph learning for point cloud generation. *IEEE Trans. Image Process.* 30 (2021), 4540–4554.
[146] Yushi Li and George Baciu. 2021. SG-GAN: Adversarial self-attention GCN for point cloud topological parts generation. *IEEE Trans. Visualiz. Comput. Graph.* 28, 10 (2021), 3499–3512.
[147] Yi-Na Li, Kang Zhang, and Dong-Jin Li. 2017. Rule-based automatic generation of logo designs. *Leonardo* 50, 2 (2017), 177–181.
[148] Antonios Liapis, Gillian Smith, and Noor Shaker. 2016. Mixed-initiative content creation. In *Procedural Content Generation in Games*. Springer, 195–214.
[149] Sohee Lim, Minwoo Shin, and Joonki Paik. 2022. Point cloud generation using deep adversarial local features for augmented and mixed reality contents. *IEEE Trans. Consum. Electron.* 68, 1 (2022), 69–76.
[150] Cheng Lin, Tingxiang Fan, Wenping Wang, and Matthias Nießner. 2020. Modeling 3D shapes by reinforcement learning. *Lect. Notes Comput. Sci.* 12355, LNCS (2020), 545–561. arXiv:2003.12397
[151] Jiangke Lin, Yi Yuan, and Zhengxia Zou. 2021. MeInGame: Create a game character face from a single portrait. arXiv:2102.02371
[152] Yu Lin, Jinghui Guo, Yang Gao, Yi Fan Li, Zhuoyi Wang, and Latifur Khan. 2021. Generating point cloud from single image in the few shot scenario. In *29th ACM International Conference on Multimedia*. ACM, 2834–2842.
[153] Aristid Lindenmayer. 1968. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *J. Theoret. Biol.* 18, 3 (1968), 280–299.
[154] Wangxin Ling and Dongzhi He. 2021. A deep learning method based on structure inference for single-view 3D reconstruction. In *IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC'21)*. IEEE, 2240–2244. arXiv:1011.1669v3
[155] Bogdan Lipuš and Nikola Guid. 2005. A new implicit blending technique for volumetric modelling. *Vis. Comput.* 21, 1 (2005), 83–91.
[156] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N. Yannakakis, and Julian Togelius. 2021. Deep learning for procedural content generation. *Neural Comput. Applic.* 33, 1 (2021), 19–37.
[157] Vivian Liu and Lydia B. Chilton. 2022. Design guidelines for prompt engineering text-to-image generative models. In *Conference on Human Factors in Computing Systems*. 1–27. arXiv:2109.06977
[158] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. 2024. SyncDreamer: Generating Multiview-consistent Images from a Single-view Image. arXiv:2309.03453 [cs.CV]
[159] Zhibo Liu, Feng Gao, and Yizhou Wang. 2019. A generative adversarial network for AI-aided chair design. In *2nd International Conference on Multimedia Information Processing and Retrieval (MIPR'19)*. 486–490. arXiv:2001.11715
[160] Zhengzhe Liu, Yi Wang, Xiaojuan Qi, and Chi-Wing Fu. 2022. Towards implicit text-guided 3D shape generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 17896–17906.
[161] William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'87)*. 163–169.
[162] Yawen Lu, Yuxing Wang, and Guoyu Lu. 2020. Single image shape-from-silhouettes. In *28th ACM International Conference on Multimedia (MM'20)*. 3604–3613.

[163] Dylan Lukes, John Sarracino, Cora Coleman, Hila Peleg, Sorin Lerner, and Nadia Polikarpova. 2021. Synthesis of web layouts from examples. In *29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*. ACM, 651–663.

[164] Zhaoliang Lun, Evangelos Kalogerakis, Rui Wang, and Alla Sheffer. 2016. Functionality preserving shape style transfer. *ACM Trans. Graph.* 35, 6 (2016), 1–14.

[165] Pattie Maes. 1997. Intelligent software. In *2nd International Conference on Intelligent User Interfaces*. 41–43.

[166] Jameel Malik, Ibrahim Abdelaziz, Ahmed Elhayek, Soshi Shimada, Sk Aziz Ali, Vladislav Golyanik, Christian Theobalt, and Didier Stricker. 2020. HandVoxNet: Deep voxel-based network for 3D hand shape and pose estimation from a single depth map. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7113–7122.

[167] Benoit B. Mandelbrot and John W. Van Ness. 1968. Fractional Brownian motions, fractional noises and applications. *SIAM Rev.* 10, 4 (1968), 422–437.

[168] Andelo Martinovic and Luc Van Gool. 2013. Bayesian grammar learning for inverse procedural modeling. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 201–208.

[169] Marian Mazzone and Ahmed Elgammal. 2019. Art, creativity, and the potential of artificial intelligence. *Arts* 8, 1 (2019), 1–9.

[170] Mostafa Merras, Abderrahim Saaidi, Nabil El Akkad, and Khalid Satori. 2018. Multi-view 3D reconstruction and modeling of the unknown 3D scenes using genetic algorithms. *Soft Comput.* 22, 19 (2018), 6271–6289.

[171] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3D reconstruction in function space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*.

[172] Lu Mi, Tianxing He, Core Francisco Park, Hao Wang, Yue Wang, and Nir Shavit. 2021. Revisiting latent-space interpolation via a quantitative evaluation framework. Retrieved from http://arxiv.org/abs/2110.06421

[173] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. 2021. Image segmentation using deep learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 7 (2021), 3523–3542.

[174] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. 2022. AutoSDF: Shape priors for 3D completion, reconstruction and generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 306–315.

[175] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. 2019. StructureNet: Hierarchical graph networks for 3D shape generation. *ACM Trans. Graph.* 38, 6 (2019), 1–19.

[176] Mojang Studios. 2011. Minecraft. Mojang Studios, Xbox Game Studios.

[177] Anselmo Montenegro, Ícaro Baptista, Bruno Dembogurski, and Esteban Clua. 2017. A new method for modeling clouds combining procedural and implicit models. In *16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames'17)*. 173–182.

[178] Matthias Muller-Brockhausen, Mike Preuss, and Aske Plaat. 2021. Procedural content generation: Better benchmarks for transfer reinforcement learning. In *IEEE Conference on Games (CoG'21)*. IEEE, 01–08.

[179] Carlos Eduardo Vaisman Muniz and Wagner Luiz Oliveira dos Santos. 2021. Generative design applied to cloud modeling. In *20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames'21)*. IEEE, 79–86.

[180] Shu Naritomi and Keiji Yanai. 2021. 3D mesh reconstruction of foods from a single image. In *3rd Workshop on AIxFood (AIxFood'21), Co-located with ACM MM 2021*. 7–11.

[181] Nelson Nauata, Kai Hung Chang, Chin Yi Cheng, Greg Mori, and Yasutaka Furukawa. 2020. House-GAN: Relational generative adversarial networks for graph-constrained house layout generation. *Lect. Notes Comput. Sci.* 12346, LNCS (2020), 162–177. arXiv:2003.06988

[182] Teo T. Niemirepo, Marko Viitanen, and Jarno Vanne. 2021. Open3DGen: Open-source software for reconstructing textured 3D models from RGB-D images. In *12th ACM Multimedia Systems Conference (MMSys'21)*. ACM, 12–22.

[183] Gen Nishida, Adrien Bousseau, and Daniel G. Aliaga. 2018. Procedural modeling of a building from a single image. *Comput. Graph. Forum* 37, 2 (2018), 415–429.

[184] Gen Nishida, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Adrien Bousseau. 2016. Interactive sketching of urban procedural models. *ACM Trans. Graph.* 35, 4 (2016), 1–11.

[185] Cihan Ongun and Alptekin Temizel. 2019. Paired 3D model generation with conditional generative adversarial networks. *Lect. Notes Comput. Sci.* 11129, LNCS (2019), 473–487. arXiv:1808.03082

[186] Cihan Öngün and Alptekin Temizel. 2021. LPMNet: Latent part modification and generation for 3D point clouds. *Comput. Graph.* 96 (2021), 1–13. arXiv:2008.03560

[187] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. 2022. StyleSDF: High-resolution 3D-consistent image and geometry generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 13503–13513.

[188] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. 2019. Deep mesh reconstruction from single RGB images via topology modification networks. In *IEEE International Conference on Computer Vision*. 9963–9972. arXiv:1909.00321

[189] Junyi Pan, Jun Li, Xiaoguang Han, and Kui Jia. 2018. Residual MeshNet: Learning to deform meshes for single-view 3D reconstruction. In *International Conference on 3D Vision (3DV'18)*. 719–727.

[190] Emmanouil Panagiotou and Eleni Charou. 2020. Procedural 3D terrain generation using generative adversarial networks. Retrieved from http://arxiv.org/abs/2010.06411

[191] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. 2021. ATISS: Autoregressive transformers for indoor scene synthesis. In *International Conference on Neural Information Processing Systems (NIPS'21)*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 12013–12026.

[192] Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi. 2020. Convolutional generation of textured 3D meshes. In *International Conference on Neural Information Processing Systems (NIPS'20)*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 870–882.

[193] Dunlu Peng, Wuchen Yang, Cong Liu, and Shuairui Lü. 2021. SAM-GAN: Self-attention supporting multi-stage generative adversarial networks for text-to-image synthesis. *Neural Netw.* 138 (2021), 57–67.

[194] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional occupancy networks. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 523–540.

[195] Ken Perlin. 1985. An image synthesizer. In *12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'85)*. ACM, 287–296.

[196] Ken Perlin. 2001. *Noise Hardware*. SIGGRAPH.

[197] Jhony K. Pontes, Chen Kong, Sridha Sridharan, Simon Lucey, Anders Eriksson, and Clinton Fookes. 2019. Image2Mesh: A learning framework for single image 3D reconstruction. *Lect. Notes Comput. Sci.* 11361, LNCS (2019), 365–381. arXiv:1711.10669

[198] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. DreamFusion: Text-to-3D using 2D Diffusion. arXiv:2209.14988 [cs.CV]

[199] Pulak Purkait, Christopher Zach, and Ian Reid. 2020. SG-VAE: Scene grammar variational autoencoder to generate new indoor scenes. *Lect. Notes Comput. Sci.* 12369, LNCS (2020), 155–171. arXiv:1912.04554

[200] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. 2023. Magic123: One Image to High-quality 3D Object Generation Using Both 2D and 3D Diffusion Priors. arXiv:2306.17843 [cs.CV]

[201] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. 2019. Learn, imagine and create: Text-to-image generation from prior knowledge. *Advan. Neural Inf. Process. Syst.* 32 (2019).

[202] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. 2019. MirrorGAN: Learning text-to-image generation by redescription. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. IEEE, 1505–1514. arXiv:1903.05854

[203] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 8748–8763.

[204] Yasin Raies and Sebastian Von Mammen. 2021. A swarm grammar-based approach to virtual world generation. *Lect. Notes Comput. Sci.* 12693 (2021), 459–474.

[205] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. 2023. Infinite photorealistic worlds using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12630–12641.

[206] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022).

[207] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *38th International Conference on Machine Learning*. 8821–8831.

[208] Rafsan Ratul, Shaheena Sultana, Jarin Tasnim, and Arifinur Rahman. 2019. Applicability of space colonization algorithm for real time tree generation. In *22nd International Conference on Computer and Information Technology (ICCIT'19)*. 1–6.

[209] Ygor Reboucas Serpa and Maria Andreia Formico Rodrigues. 2019. Towards machine-learning assisted asset generation for games: A study on pixel art sprite sheets. In *18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames'19)*. 182–191.

[210] Jing Ren, Biao Zhang, Bojian Wu, Jianqiang Huang, Lubin Fan, Maks Ovsjanikov, and Peter Wonka. 2021. Intuitive and efficient roof modeling for reconstruction and synthesis. *ACM Trans. Graph.* 40, 6 (2021), 1–17. arXiv:2109.07683

[211] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10684–10695. arXiv:2112.10752

[212] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'15)*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer, 234–241.

[213] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. 2022. Photorealistic text-to-image diffusion models with deep language understanding. Retrieved from http://arxiv.org/abs/2205.11487

[214] Shunsuke Saito, Liwen Hu, Chongyang Ma, Hikaru Ibayashi, Linjie Luo, and Hao Li. 2018. 3D hair synthesis using volumetric variational autoencoders. *ACM Trans. Graph.* 37, 6 (2018), 1–12.

[215] Aditya Sanghi, Hang Chu, Joseph G. Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. 2022. CLIP-Forge: Towards zero-shot text-to-shape generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*. 18603–18613.

[216] Khalil Satyadama, Reza Fuad Rachmadi, and Supeno Mardi Susiki Nugroho. 2020. Procedural environment generation for cave 3D model using OpenSimplex noise and marching cube. In *International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM'20)*. 144–148.

[217] Henning Schulze, Dogucan Yaman, and Alexander Waibel. 2021. CAGAN: Text-to-image generation with combined attention generative adversarial networks. *Lect. Notes Comput. Sci.* 13024 (2021), 392–404.

[218] Scratchapixel 2.0. 2022. Rasterization: A Practical Implementation. Retrieved from https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation

[219] Thomas W. Sederberg and Scott R. Parry. 1986. Free-form deformation of solid geometric models. In *13th Annual Conference on Computer Graphics and Interactive Techniques*. 151–160.

[220] Gil Shamai, Ron Slossberg, and Ron Kimmel. 2019. Synthesizing facial photometries and corresponding geometries using generative adversarial networks. *ACM Trans. Multim. Comp., Commu. Apps.* 15, 3s (2019), 1–24. arXiv:1901.06551

[221] Rahul Sharma. 2016. Procedural city generator. In *International Conference on System Modeling & Advancement in Research Trends (SMART'16)*. IEEE, 213–217.

[222] I. Chao Shen and Bing-Yu Chen. 2021. ClipGen: A deep generative model for clipart vectorization and synthesis. *IEEE Trans. Visualiz. Comput. Graph.* 28, 12 (2021), 4211–4224. arXiv:2106.04912

[223] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep marching tetrahedra: A hybrid representation for high-resolution 3D shape synthesis. In *International Conference on Neural Information Processing Systems (NIPS'21)*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 6087–6101.

[224] Yuefan Shen, Changgeng Zhang, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2021. DeepSketchHair : Deep sketch-based 3D hair modeling. *IEEE Trans. Visualiz. Comput. Graph.* 27, 7 (2021), 3250–3263.

[225] Tianyang Shi, Zhengxia Zou, Zhenwei Shi, and Yi Yuan. 2022. Neural rendering for game character auto-creation. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 3 (2022), 1489–1502.

[226] Takayuki Shinohara, Haoyi Xiu, and Masashi Matsuoka. 2021. 3D point cloud generation using adversarial training for large-scale outdoor scene. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'21)*. 2935–2938.

[227] Dongwook Shu, Sung Woo Park, and Junseok Kwon. 2019. 3D point cloud generative adversarial network based on tree structured graph convolutions. In *IEEE International Conference on Computer Vision*. 3858–3867. arXiv:1905.06292

[228] Yezhi Shu, Ran Yi, Mengfei Xia, Zipeng Ye, Wang Zhao, Yang Chen, Yu Kun Lai, and Yong Jin Liu. 2021. GAN-based multi-style photo cartoonization. *IEEE Trans. Visualiz. Comput. Graph.* 28, 10 (2021), 3376–3390.

[229] Sergio N. Silva Junior, Felipe C. Chamone, Renato C. Ferreira, and Erickson R. Nascimento. 2018. A 3D modeling methodology based on a concavity-aware geometric test to create 3D textured coarse models from concept art and orthographic projections. *Comput. Graph.* 76 (2018), 73–83.

[230] Zackary P. T. Sin and Peter H. F. Ng. 2018. Planetary marching cubes: A marching cubes algorithm for spherical space. In *2nd International Conference on Video and Image Processing*. 89–94.

[231] Vedant Singh, Manan Oza, Himanshu Vaghela, and Pratik Kanani. 2019. Auto-encoding progressive generative adversarial networks for 3D multi object scenes. In *International Conference of Artificial Intelligence and Information Technology (ICAIIT'19)*. 481–485. arXiv:1903.03477

[232] Miroslava Slavcheva, Wadim Kehl, Nassir Navab, and Slobodan Ilic. 2016. SDF-2-SDF: Highly accurate 3D object reconstruction. In *Lecture Notes in Computer Science,* Vol. 9905, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer, 680–696.

[233] Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2014. A survey on procedural modelling for virtual worlds. *Comput. Graph. Forum* 33, 6 (2014), 31–50.

[234] Jiqiang Song, Feng Su, Chiew-Lan Tai, and Shijie Cai. 2002. An object-oriented progressive-simplification-based vectorization system for engineering drawings: Model, algorithm, and performance. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 8 (2002), 1048–1060.

[235] Peihua Song, Youyi Zheng, Jinyuan Jia, and Yan Gao. 2019. Web3D-based automatic furniture layout system using recursive case-based reasoning and floor field. *Multim. Tools Applic.* 78, 4 (2019), 5051–5079.

[236] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. 2017. SUNCG Dataset. Retrieved from https://sscnet.cs.princeton.edu/

[237] Ryan Spick, Simon Demediuk, and James Walker. 2020. Naive mesh-to-mesh coloured model generation using 3D GANs. In *Australasian Computer Science Week Multiconference (ACSW'20)*. 1–6.

[238] Ryan Spick and James Alfred Walker. 2019. Realistic and textured terrain generation using GANs. In *16th ACM SIGGRAPH European Conference on Visual Media Production (CVMP'19)*. 1–10.

[239] Misha Sra, Sergio Garrido-Jurado, Chris Schmandt, and Pattie Maes. 2016. Procedurally generated virtual reality from 3D reconstructed physical space. In *22nd ACM Conference on Virtual Reality Software and Technology*. 191–200.

[240] Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. 2018. Interactive sketch-based normal map generation with deep neural networks. *Proc. ACM Comput.Graph. Interact. Techniq.* 1, 1 (2018), 1–17.

[241] Adam Summerville. 2018. Expanding expressive range: Evaluation methodologies for procedural content generation. *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entert.* 14, 1 (2018), 116–122.

[242] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (PCGML). *IEEE Trans. Games* 10, 3 (2018), 257–270.

[243] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 2023. 3D-GPT: Procedural 3D Modeling with Large Language Models. arXiv:2310.12945 [cs.CV]

[244] Cheng Sun, Yiran Zhou, and Yunsong Han. 2022. Automatic generation of architecture facade for historical urban renovation using generative adversarial network. *Build. Environ.* 212 (2022), 108781.

[245] Xiao Sun and Zhouhui Lian. 2020. EasyMesh: An efficient method to reconstruct 3D mesh from a single image. *Comput. Aid. Geom. Des.* 80 (2020), 101862.

[246] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. 2018. Variational autoencoders for deforming 3D mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5841–5850.

[247] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. 2024. DiffuScene: Denoising Diffusion Models for Generative Indoor Scene Synthesis. arXiv:2303.14207 [cs.CV]

[248] Edward Teng and Rafael Bidarra. 2017. A semantic approach to patch-based procedural generation of urban road networks. In *12th International Conference on the Foundations of Digital Games (FDG'17)*. 1–10.

[249] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Trans. Computat. Intell. AI Games* 3, 3 (2011), 172–186.

[250] Fei Tong, Megumi Nakao, Shuqiong Wu, Mitsuhiro Nakamura, and Tetsuya Matsuda. 2020. 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'20)*. 1608–1611.

[251] Unity Technologies. 2024. Unity Engine. Retrieved from https://unity.com/

[252] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

[253] Danya F. Vears and Lynn Gillam. 2022. Inductive content analysis: A guide for beginning qualitative researchers. *Focus Health Profess. Educ.: Multi-profess. J.* 23, 1 (2022), 111–127.

[254] Abbhinav Venkat, Chaitanya Patel, Yudhik Agrawal, and Avinash Sharma. 2019. HumanMeshNet: Polygonal mesh recovery of humans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*.

[255] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* 11 (2010), 3371–3408.

[256] Sebastian Von Mammen and Christian Jacob. 2009. The evolution of swarm grammars—Growing trees, crafting art, and bottom-up design. *IEEE Computat. Intell. Mag.* 4, 3 (2009), 10–19.

[257] John Von Neumann and Arthur W. Burks. 1966. *Theory of Self-reproducing Automata*. University of Illinois Press, USA.

[258] Paul Waddell. 2002. UrbanSim: Modeling urban development for land use, transportation, and environmental planning. *J. Amer. Plan. Assoc.* 68, 3 (2002), 297–314.

[259] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. 2011. The Caltech-UCSD birds-200-2011 dataset. California Institute of Technology. Retrieved from http://www.vision.caltech.edu/datasets/cub_200_2011/

[260] Guan Wang, Hamid Laga, Ning Xie, Jinyuan Jia, and Hedi Tabia. 2018. The shape space of 3D botanical tree models. *ACM Trans. Graph.* 37, 1 (2018), 1–18.

[261] Hao Wang, Nadav Schor, Ruizhen Hu, Haibin Huang, Daniel Cohen-Or, and Hui Huang. 2018. Global-to-local generative model for 3D shapes. *ACM Trans. Graph.* 37, 6 (2018), 1–10.

[262] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. 2019. PlanIT: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Trans. Graph.* 38, 4 (2019), 1–15.

[263] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu Gang Jiang. 2018. Pixel2Mesh: Generating 3D mesh models from single RGB images. *Lect. Notes Comput. Sci.* 11215, LNCS (2018), 55–71. arXiv:1804.01654

[264] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Hang Yu, Wei Liu, Xiangyang Xue, and Yu Gang Jiang. 2021. Pixel2Mesh: 3D mesh model generation via image guided deformation. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 10 (2021), 3600–3613.

[265] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018. Adaptive O-CNN: A patch-based deep representation of 3D shapes. *ACM Trans. Graph.* 37, 6 (2018), 1–11.

[266] Tong Wang and Shuichi Kurabayashi. 2020. Sketch2Map: A game map design support system allowing quick hand sketch prototyping. In *IEEE Conference on Games (CoG'20).* 596–599.

[267] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, and Baining Guo. 2023. RODIN: A generative model for sculpting 3D digital avatars using diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'23).* 4563–4573.

[268] Xiaolong Wang and Abhinav Gupta. 2016. Generative image modeling using style and structure adversarial networks. *Lect. Notes Comput Sci.* 9908 (2016), 318–335.

[269] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. 2021. SceneFormer: Indoor scene generation with transformers. In *International Conference on 3D Vision (3DV'21).* 106–115.

[270] Yaohui Wang, Antitza Dantcheva, and Francois Bremond. 2019. From attribute-labels to faces: Face generation using a conditional generative adversarial network. In *Computer Vision – ECCV 2018 Workshops*, Laura Leal-Taixé and Stefan Roth (Eds.). Springer, 692–698.

[271] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.* 53, 3 (2020), 1–34.

[272] Yifan Wang, Zichun Zhong, and Jing Hua. 2020. DeepOrganNet: On-the-fly reconstruction and visualization of 3D/4D lung models from single-view projections by deep deformation network. *IEEE Trans. Visualiz. Comput. Graph.* 26, 1 (2020), 960–970. arXiv:1907.09375

[273] Chao Wen, Yinda Zhang, Chenjie Cao, Zhuwen Li, Xiangyang Xue, and Yanwei Fu. 2022. Pixel2Mesh++: 3D mesh generation and refinement from multi-view images. *IEEE Trans. Patt. Anal. Mach. Intell.* 45, 2 (2022), 2166–2180.

[274] Zhenzhen Weng and Serena Yeung. 2021. Holistic 3D human and scene mesh estimation from single view images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 334–343. arXiv:2012.01591

[275] Andrew R. Willis, Prashant Ganesh, Kyle Volle, Jincheng Zhang, and Kevin Brink. 2021. Volumetric procedural models for shape representation. *Graph. Vis. Comput.* 4 (2021), 200018.

[276] Steven Worley. 1996. A cellular texture basis function. In *23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96).* ACM, 291–294.

[277] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. ModelNet Dataset. Retrieved from https://modelnet.cs.princeton.edu/

[278] Nan Xiang, Li Wang, Tao Jiang, Yanran Li, Xiaosong Yang, and Jianjun Zhang. 2019. Single-image mesh reconstruction and pose estimation via generative normal map. In *32nd International Conference on Computer Animation and Social Agents.* 79–84.

[279] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Xiaojun Tong. 2018. Weighted voxel: A novel voxel representation for 3D reconstruction. In *10th International Conference on Internet Multimedia Computing and Service (ICIMCS'18).* 1–4.

[280] Weidan Xiong, Pengbo Zhang, Pedro V. Sander, and Ajay Joneja. 2018. Shape-inspired architectural design. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D'18).* ACM, Article 12, 10 pages.

[281] Hao Xu and Jing Bai. 2021. In *ARShape-Net: Single-view Image Oriented 3D Shape Reconstruction with an Adversarial Refiner* (LNAI, Vol. 13069). Springer, 638–649.

[282] Jiale Xu, Xintao Wang, Weihao Cheng, Yan-Pei Cao, Ying Shan, Xiaohu Qie, and Shenghua Gao. 2023. Dream3D: Zero-shot text-to-3D synthesis using 3D shape prior and text-to-image diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'23).* 20908–20918.

[283] Jinglun Yang, Youhua Li, and Lu Yang. 2021. Shape transformer nets: Generating viewpoint-invariant 3D shapes from a single image. *J. Vis. Commun. Image Represent.* 81 (2021), 103345.

[284] Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, and Lin Gao. 2022. DSG-Net: Learning disentangled structure and geometry for 3D shape generation. *ACM Trans. Graph.* 42, 1 (2022), 1–17.

[285] Kaizhi Yang, Jintao Lu, Siyu Hu, and Xuejin Chen. 2021. Deep 3D modeling of human bodies from freehand sketching. In *MultiMedia Modeling*, Jakub Lokoč, Tomáš Skopal, Klaus Schoeffmann, Vasileios Mezaris, Xirong Li, Stefanos Vrochidis, and Ioannis Patras (Eds.). Springer, 36–48.

[286] Dumim Yoon and Kyung Joong Kim. 2017. 3D game model and texture generation using interactive genetic algorithm. *Comput. Entert.* 14, 1 (2017), 1–16.

[287] Aron Yu and Kristen Grauman. 2017. UT Zappos50K Dataset. Retrieved from https://vision.cs.utexas.edu/projects/finegrained/utzap50k/

[288] Hang Yu, Chilam Cheang, Yanwei Fu, and Xiangyang Xue. 2023. Multi-view shape generation for a 3D human-like body. *ACM Trans. Multim. Comput. Commun. Appl.* 19, 1 (2023), 1–22.

[289] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, and B. Hutchinson. 2022. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation 2, 3 (2022), p. 5. arXiv preprint arXiv:2206.10789

[290] Ye Yuan, Yasuaki Ito, and Koji Nakano. 2020. Art font image generation with conditional generative adversarial networks. In *8th International Symposium on Computing and Networking Workshops (CANDARW'20)*. 151–156.

[291] Ke Yue, Yidong Li, and Huifang Li. 2019. Progressive semantic image synthesis via generative adversarial network. In *IEEE International Conference on Visual Communications and Image Processing (VCIP'19)*. 1–4.

[292] Anny Yuniarti, Agus Zainal Arifin, and Nanik Suciati. 2021. A 3D template-based point generation network for 3D reconstruction from single images. *Appl. Soft Comput.* 111 (2021), 107749.

[293] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. 2022. LION: Latent point diffusion models for 3D shape generation. In *International Conference on Neural Information Processing Systems (NeurIPS'22)*.

[294] Jian Zhang, Changbo Wang, Chen Li, and Hong Qin. 2019. Example-based rapid generation of vegetation on terrain via CNN-based distribution learning. *Vis. Comput.* 35, 6–8 (2019), 1181–1191.

[295] Meng Zhang, Pan Wu, Hongzhi Wu, Yanlin Weng, Youyi Zheng, and Kun Zhou. 2018. Modeling hair from an RGB-D camera. *ACM Trans. Graph.* 37, 6 (2018), 1–10.

[296] Meng Zhang and Y. Zheng. 2019. Hair-GAN: Recovering 3D hair structure from a single image using generative adversarial networks. *Vis. Inform.* 3, 2 (2019), 102–112.

[297] Qimeng Zhang, Jaeho Im, Minju Park, Myung Jin Choi, Chang Hun Kim, and Yoonsik Shim. 2019. Shrubbery-shell inspired 3D model stylization. *Comput. Graph.* 82 (2019), 13–21.

[298] Sizhuo Zhang and Nanfeng Xiao. 2021. Detailed 3D human body reconstruction from a single image based on mesh deformation. *IEEE Access* 9 (2021), 8595–8603.

[299] Wenbo Zhang. 2020. Sketch-to-color image with GANs. In *International Conference on Information Technology and Computer Application (ITCA'20)*. 322–325.

[300] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. 2020. Deep generative modeling for scene synthesis via hybrid representations. *ACM Trans. Graph.* 39, 2 (2020), 1–21.

[301] Zhiqiang Zhang, Wenxin Yu, Jinjia Zhou, Xuewen Zhang, Jialiang Tang, Siyuan Li, Ning Jiang, Gang He, and Zhuo Yang. 2020. Customizable GAN: Customizable image synthesis based on adversarial learning. *Commun. Comput. Inf. Sci.* 1332 (2020), 336–344.

[302] Yiru Zhao, Bing Deng, Jianqiang Huang, Hongtao Lu, and Xian Sheng Hua. 2017. Stylized adversarial autoencoder for image generation. In *ACM Multimedia Conference (MM'17)*. 244–251.

[303] X. Zheng, Y. Liu, P. Wang, and X. Tong. 2022. SDF-StyleGAN: Implicit SDF-based StyleGAN for 3D shape generation. *Comput. Graph. Forum* 41, 5 (2022), 52–63.

[304] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. CycleGAN Datasets. Retrieved from https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/blob/master/docs/datasets.md

Received 17 February 2023; revised 5 November 2024; accepted 22 November 2024