International Journal of Human–Computer Interaction

ISSN: 1044-7318 (Print) 1532-7590 (Online) Journal homepage: www.tandfonline.com/journals/hihc20

Heuristics for AI-Driven Graphical Asset Generation Tools in Game Design and Development Pipelines: A User-Centered Approach

Kaisei Fukaya, Damon Daylamani-Zad & Harry Agius

To cite this article: Kaisei Fukaya, Damon Daylamani-Zad & Harry Agius (04 Mar 2026): Heuristics for AI-Driven Graphical Asset Generation Tools in Game Design and Development Pipelines: A User-Centered Approach, International Journal of Human–Computer Interaction, DOI: 10.1080/10447318.2026.2632170

To link to this article: https://doi.org/10.1080/10447318.2026.2632170

INTERNATIONAL JOURNAL OF HUMAN–COMPUTER INTERACTION
https://doi.org/10.1080/10447318.2026.2632170

Heuristics for AI-Driven Graphical Asset Generation Tools in Game Design and Development Pipelines: A User-Centered Approach

Kaisei Fukaya, Damon Daylamani-Zad and Harry Agius

College of Engineering, Design and Physical Sciences, Brunel University of London, Kingston Lane, Uxbridge, UK

ABSTRACT

Graphical assets play an important role in design and development of games. There is potential in the use of AI-driven generative tools to aid in creation of such assets, improving pipelines. However, there is little research to address how generative methods can fit into the wider pipeline, and no guidelines or heuristics for creating such tools. Hence, we conducted a user study with 16 game designers and developers to examine their behaviour and interaction with such tools. Findings highlight that early design stage is preferred by all participants. Designers and developers prioritise rapid variations over initial quality of assets. Results also strongly raised the need for better integration of such tools in existing design/development environments and pipelines, specifically regarding common data formats and output manipulability. Informed by these results, we provide a set of heuristics for creating tools that meet the expectations and needs of game designers and developers.

KEYWORDS

Graphical game assets; artificial intelligence PCG; user behavior; user interface

1. Introduction

While the usage of procedural content generation (PCG) in games is widespread, it is largely applied to certain forms of content, such as environments, levels and narratives. Games such as No Man's Sky (Hello Games, 2016), Minecraft (Mojang Studios, 2011) and Dwarf Fortress (Bay 12 Games, 2006) use PCG to create new content for players to see and explore. Togelius et al. (Togelius et al., 2011) define this as *online* PCG. In other cases PCG is used to streamline development, such as with Starfield (Skrebels, 2023) in which large quantities of content were generated then curated and refined by hand. This approach is referred to as *offline* PCG (Togelius et al., 2011), and can involve humans designers to varying degrees (Lai et al., 2020; Liapis et al., 2016).

Previous work has established frameworks that define tools for generating graphical assets in game creation pipelines (Fukaya et al., 2024, 2025). These graphical asset generators (GAGs) must meet the needs and expectations of game designers and developers, while also providing a high *quality of experience* (QoE). Existing research on the QoE of games has been conducted across many platforms, including PC, mobile (Laghari et al., 2020), cloud (Laghari et al., 2019) and VR (Jumani et al., 2024; Laghari et al., 2024). Yet no research has examined the needs of game designers and developers, nor established QoE for GAG tool interfaces so far.

It is apparent that different generative methods and techniques are designed with different use cases in mind, each attempting to solve a problem or streamline a process within a larger creative pipeline. Practitioners use many tools during game design and production, from game engine specific tools, to asset production software such as Photoshop and Blender. These tools each have their place within workflows and development pipelines. While workflows look different from company to company or from one individual to another, there exists three ubiquitous stages of development: prototyping/design, production and testing (Ramadan & Widyani, 2013). All graphical assets are formulated and produced

CONTACT Damon Daylamani-Zad damon.daylamani-zad@brunel.ac.uk College of Engineering, Design and Physical Sciences, Brunel University of London, Kingston Lane, Uxbridge, UB8 3PH, UK

Supplemental data for this article can be accessed online at https://doi.org/10.1080/10447318.2026.2632170.

during the prototyping, design and production stages. Within this, there are many ways that PCG can help, from early inspiration and creating rough placeholders to creating fully fledged assets or remixing existing assets for variety. Each of these use cases have different requirements for the complexity of input, method of interaction and quality of output. However, no research has yet explored the behavior and interaction patterns of the users, the purposes for which designers and developers use PCG tools, and how these tools fit within game pipelines.

While there is an extensive body of literature pertaining to GAGs there are limited case studies that examine their use and fit in design and development pipelines (Lai et al., 2020). While it is clear that GAGs have the potential to improve game development asset pipelines, the behavior, opinions and preferences of game designers and developers remain generally unheard. Though recent work examines usage applications for generative AI systems such as ChatGPT among general user bases (Yan et al., 2024), the specific needs and requirements within the video game industry have not yet been addressed. Individual GAG systems may be validated with user feedback (Li et al., 2021; Shen et al., 2021; Shen & Chen, 2022), but feedback is not typically contextualized within the larger scope of the development pipeline.

In this paper, we will investigate the preferences of game designers and developers regarding the practical application of graphical asset generators (GAGs). We will collect opinions from a diverse range of practitioners to identify where GAG tools can be effectively deployed within game production pipelines and establish design considerations that will maximize their utility. Our findings provide practical insights and expand existing approaches to game development pipelines.

2. Review of current research

The process of creating games is complex, requiring design decisions at every level from the overall concept and gameplay down to the characters and environments. Designers and developers make use of increasingly capable creative tools to streamline this task, which they often tailor to their own needs (Laghari et al., 2025; Newell et al., 2021; Seidel et al., 2019).

In a qualitative study examining expectations of Finnish game developers regarding their tools, Kasurinen et al. (2013) present some key insights. Across the seven organizations examined, developers were largely satisfied with the tools they have. When it comes to assets, many preferred to purchase rather than create them in-house. It is found that the companies "expect their tools to allow easy prototyping and the ability to design while implementing" (Kasurinen et al., 2013). Many of the companies relied on third-party game engines, and thus compatibility with these engines was a large part of the consideration when selecting new tools.

The vast amount of content required in video games requires an equally vast amount of time and resources to create. Generative approaches can be applied to reduce this work load (Seidel et al., 2019). It is important for designers and developers to have control over the content that is created, thus generative tools typically involve mixed-initiative (MI) approaches. The user study of Walton et al. (Walton et al., 2022), examines user opinions of a PCG level creation tool using a mixed-methods approach. The findings suggest that the MI-PCG method stimulates user creativity, and helps to inspire new ideas. They discover that the inclusion of qualitative data from participants is important for gaining the full picture, as it provides important context to the quantitative results.

The concept of designer modeling extends MI-PCG by considering the adaptability of a system to user needs (Liapis et al., 2013). In such a system, the tool aligns itself to the designer's style choices, helps to break design fixation and provides inspiration.

Sketchar (Ling et al., 2024) is a mixed-initiative generative tool encompassing a full character design pipeline. In this method, users iteratively build on each aspect of a design, including character profiles, concept art, relationships with other characters and dialogue between them, using large language models (LLMs) such as ChatGPT and image generators such as DALL-E. In user testing, designers identify the benefits of this method in refining existing, early-stage ideas as well as visually communicating their design ideas to artists.

Colado et al. (2023), examine mixed-initiative generative approaches to serious game design. Much like Sketchar (Ling et al., 2024), this method implemented an iterative text-to-image generation workflow, this time for character concepts as well as scenarios. Results from initial testing suggests that this

method can save time in development and aid smaller teams with less resources. It is clear that generative methods have clear applications in graphical asset pipelines. According to Unity's game report in 2024, 62% of developers that have adopted AI tools use them for asset generation (Unity Technologies, 2024). However, the impact of generative, mixed-initiative tools for graphical game assets is yet to be examined.

3. Motivation & current development pipeline

Graphical Asset Generation/Transformation (GAGeTx) is a framework for designing graphical asset generators (GAGs) based on user needs and requirements (Fukaya et al., 2024, 2025). It tasks the user with making a series of decisions that feed into one-another, including the type of asset to be generated, the technique and approach to be used, and whether data format conversion is required as part of the pipeline. The high level summary of GAGeTx is presented in Figure 1.

This framework currently addresses the *how* of graphical asset generation. The user chooses an asset type and dimensionality, selects a "technique" to use and considers the input types required for the "technique." A relevant "approach" can then be selected based on whether it uses the chosen "technique" and input types, and generates the target asset type. Format conversion may then be applied if the chosen "approach" does not produce results in the format required by the user. This process begins with the assumption that the user knows what they need to generate, how they will incorporate it within their pipeline, and the quality to be expected.

However, the framework does not yet consider the issues of *where to generate* and *how to interact*. In a practical scenario, generative tools for creating assets will likely function within a larger pipeline. There should be a clear purpose for the generative tool. Consideration for the purpose of a generator also has implications on the standard to which it is implemented, the compromise between quality of output, and the volume or variety of results. For example, a generator used for early prototyping or ideation in the pipeline must meet different quality and speed standards to one used for producing "game ready" assets, which is a stage toward the end of the pipeline. This position in the development pipeline may also dictate the input types available, their quality, and the output format needed. Which also has an affect on the choice of "approach." Furthermore, the interface and presentation of features should be considered. This ties in with the purpose, as for example, fine-grained control can be a requirement for "game ready" asset creation, but a hindrance in the ideation phase where maximal speed and minimal interaction is required.

Through the analysis of *existing game development pipelines*, Ramadan and Widyani (2013), introduced a comprehensive game development life cycle (GDLC). In this framework, practitioners begin with a rough concept of the game then begin a design and development loop starting with *pre-production*, in which more detailed concepts are established, then followed by *production*, in which the concepts are implemented, and *testing* in which these implementations are appraised. Insights from testing then feed into further pre-production and the cycle continues, iterating on the design until the practitioners are satisfied and they move on to refinement in the beta phase, before finally releasing the product.

To build on this, we present an expanded version of the GDLC, incorporating the graphical asset creation pipeline as it relates to general development, shown in Figure 2. Here we identify points in the pipeline at which practitioners utilize creative tools, through observing existing generative methods, tools and pipelines in the literature. These include building inspiration in the interim between initiation

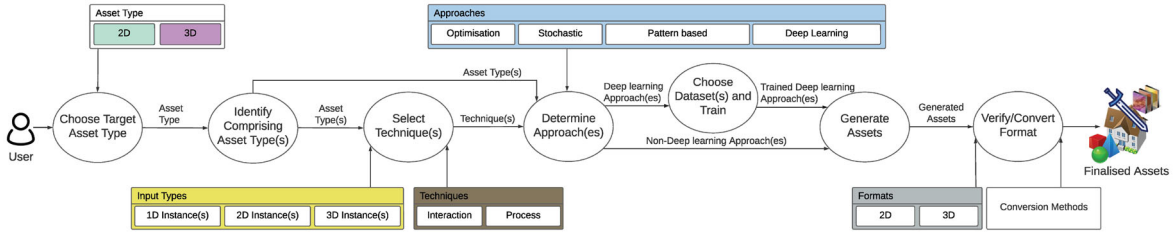

Figure 1. The *graphical asset Generation/Transformation* (GAGeTx) framework. A framework for designing graphical asset generators based on user needs and requirements.

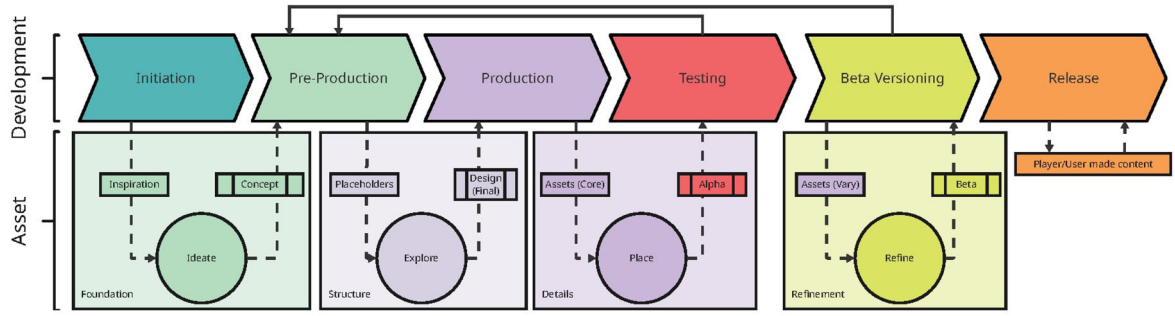

Figure 2. Current game design and development pipeline, adapted from (Ramadan & Widyani, 2013).

and pre-production; generating placeholder assets, and exploring designs during pre-production and early production; implementing these designs as full (Core) assets in production; creating variations of these assets in the refinement stage; and supporting player/user made content.

3.1. Generating inspiration

Creative block and design fixation (Jansson & Smith, 1991) are widely experienced in creative fields, including game production. Generative methods for graphical assets can be used to help circumvent this issue by providing unexpected or varied outputs that can serve as inspiration. In this context, the quality of the generated content is less important than its ability to spark the user's creativity, serving as a starting point or trigger for further design iterations.

3.2. Exploring ideas

During early stages of design, iteration is key for expanding on initial inspirations and producing finalized designs. During this process, designs are flexible, thus ideas can be quickly tested and evaluated before committing extensive resources and time. The speed and configurability of some generative methods may help with exploring ideas and potential design directions for this use case (Saharia et al., 2022).

3.3. Creating placeholder assets

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

3.4. Creating variations of existing (complete) assets

In games, asset variation can help to immerse players and reduce the repetitiveness of content throughout a digital environment. Many generative methods facilitate the ability to create similar variations of assets. This is most commonly seen with methods that rely on parametric modeling (Getto et al., 2020; Jones et al., 2021), or with the usage of VAEs (Tan et al., 2018). Therefore, there is potential in the usage of generative tools for increasing the diversity and richness of content available, making the digital environment more engaging and varied for players.

3.5. Creating assets from existing (complete) designs

During the design phase, the appearance of an asset may be determined and planned through the use of concept art. This concept art is later used as reference for creating the final asset during the production phase. A graphical asset generator may be used to achieve the latter task, by converting initial designs and plans into finalized assets. For example (Silva Junior et al., 2018), use orthographic concept art from multiple views to generate 3D assets, even utilizing the initial art for texturing.

3.6. Creating assets from scratch

Some generative tools, such as SpeedTree (Interactive Data Visualization Inc, 2022), encompass a full pipeline for designing, configuring then producing assets. In these cases, the process covers the creation of assets from scratch.

3.7. Player (or user) made content

A common application of PCG, beyond its use in generating graphical assets, is for *online* content generation. This typically refers to the generation of game elements such as levels and loot to produce unexpected and re-playable experiences. With regard to graphical assets however, some games such as Spore and Dreams provide tools that allow players to create their own designs within a constrained creation framework. In a sense, many of these tools are constrained versions of development tools, there is therefore potential for the application of generative tooling within similar systems. Generative systems can also be used for avatar personalization, such as through the reconstruction of faces from photographs (Lin et al., 2021).

Figure 3 presents the key characteristics of each game centered use case. As shown, graphical asset generators may be applied *offline* during the production of a game product, or *online* during run-time. Uses may also necessitate high quality outputs, such as in *creating variations of existing designs* and *creating assets from scratch*, or low quality outputs with an emphasis on speed, such as in the case of *generating inspiration*. These uses may also aid in streamlining part of a development process, such as with *exploring ideas*, or replace an entire process, as with *creating assets from scratch*.

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

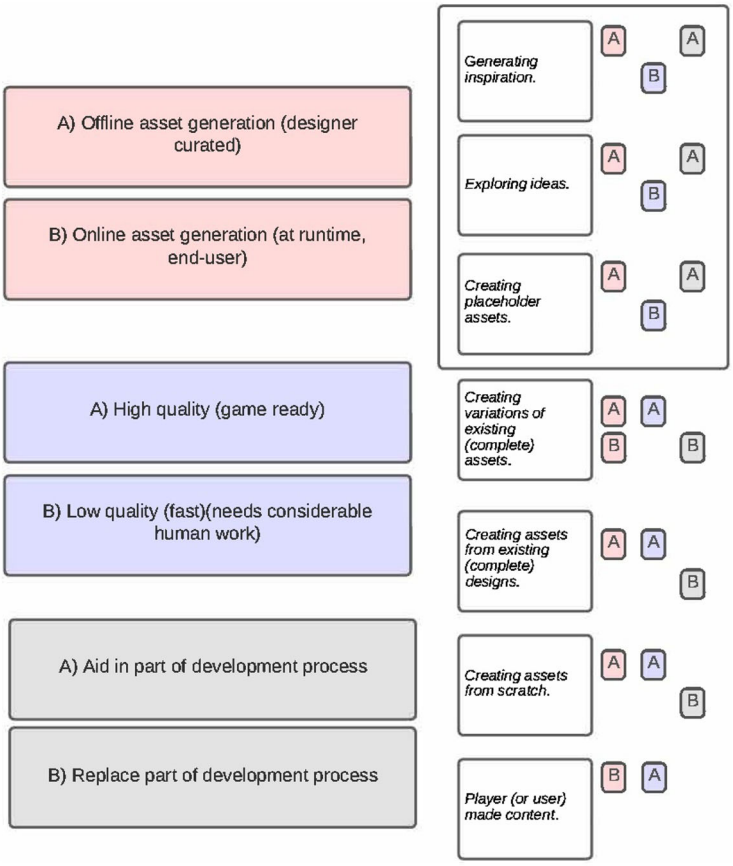

Figure 3. The key characteristics of design and production uses. *Generating inspiration*, *exploring ideas* and *creating placeholder assets* are grouped together as they share the same characteristics.

4. Research questions

The aim of this study is two-fold: (a) to establish if GAG tools would be found useful by game designers and developers, and what their requirements are for interacting with such tools, and (b) to discover where in the pipeline such tools would best fit, and how the chosen generative method may line up with this. As a result, expanding and providing nuance to the GAGeTx framework.

This study will obtain game designer and developer feedback with regard to generating assets for game projects through interaction with UI mockups. To ensure that opinions are not shaped by the type of asset, technique or approach to generation; the UI mockups have been designed to present a fully customizable generative system, where the user decides these specifics.

There are five core questions this study seeks to answer, shown below in order of importance.

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

We take a mixed methods approach, collecting quantitative and qualitative data from participants. Our methodology attempts to answer a series of research questions, which expand the above; presented in Table 1 as RQ1-RQ9, assessing a relative set of hypotheses, listed H1-H7 in Table 2 and demographical impact hypotheses RH1-RH3 listed below. To achieve this, three prototypes have been devised, as shown in Figure 4.

RH1.1 Size of team impacts preference.
RH2.1 Experience impacts preference.

Table 1. List of research questions and associated questionnaire questions.

Research questions		Survey questions
RQ1	Which prototype/mockup (M_i) is deemed the most useful to game designers/developers?	"I would find this software tool that generates graphical assets useful in the projects I work on."
RQ2	Where in the pipeline would game designers/developers find value in M_i ?	"Where in your development pipeline would you find value in this software tool?"
RQ2.1	Do designers/developers find value in M_i for Generating inspiration.	
RQ2.2	Do designers/developers find value in M_i for Exploring ideas.	
RQ2.3	Do designers/developers find value in M_i for Creating placeholder assets.	
RQ2.4	Do designers/developers find value in M_i for Creating variations of existing (complete) assets.	
RQ2.5	Do designers/developers find value in M_i for Creating assets from existing (complete) designs.	
RQ2.6	Do designers/developers find value in M_i for Creating assets from scratch.	
RQ2.7	Do designers/developers find value in M_i for Player (or user) made content.	
RQ3	If M_i were to be used in a pipeline, would designers/developers prioritize volume of output or quality of output?	"Considering your answer to the previous questions, would you prefer if this tool generated a large variety of assets at a lower quality, or that it generated a small variety of high-quality assets?"
RQ4	How much time would designers/developers find acceptable for M_i to take in generating a single asset?	"Please indicate the largest timescale you would find acceptable for generating a single graphical asset, if you were to use this software tool in your projects."
RQ5	Do designers/developers prefer M_i as a stand-alone solution or integrated into a game-engine?	"Given the option, would you prefer such a system to exist as stand-alone software, or integrated into your game engine editor of choice?"
RQ6	How important is the ability to configure or modify M_i according to designers/developers?	"Please rate on a scale of 1 (not important) to 5 (very important), how important to you, is the ability to configure or modify such a tool. For example, importing your own bespoke algorithms or trained models?"
RQ7	Which M do designers/developers identify as more useable?	System Usability Scale (Brooke, 1996)
RQ8	What are the concerns and perceived benefits of this tool?	
RQ9	What are the desired asset types?	

Table 2. List of hypotheses and associated research questions.

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

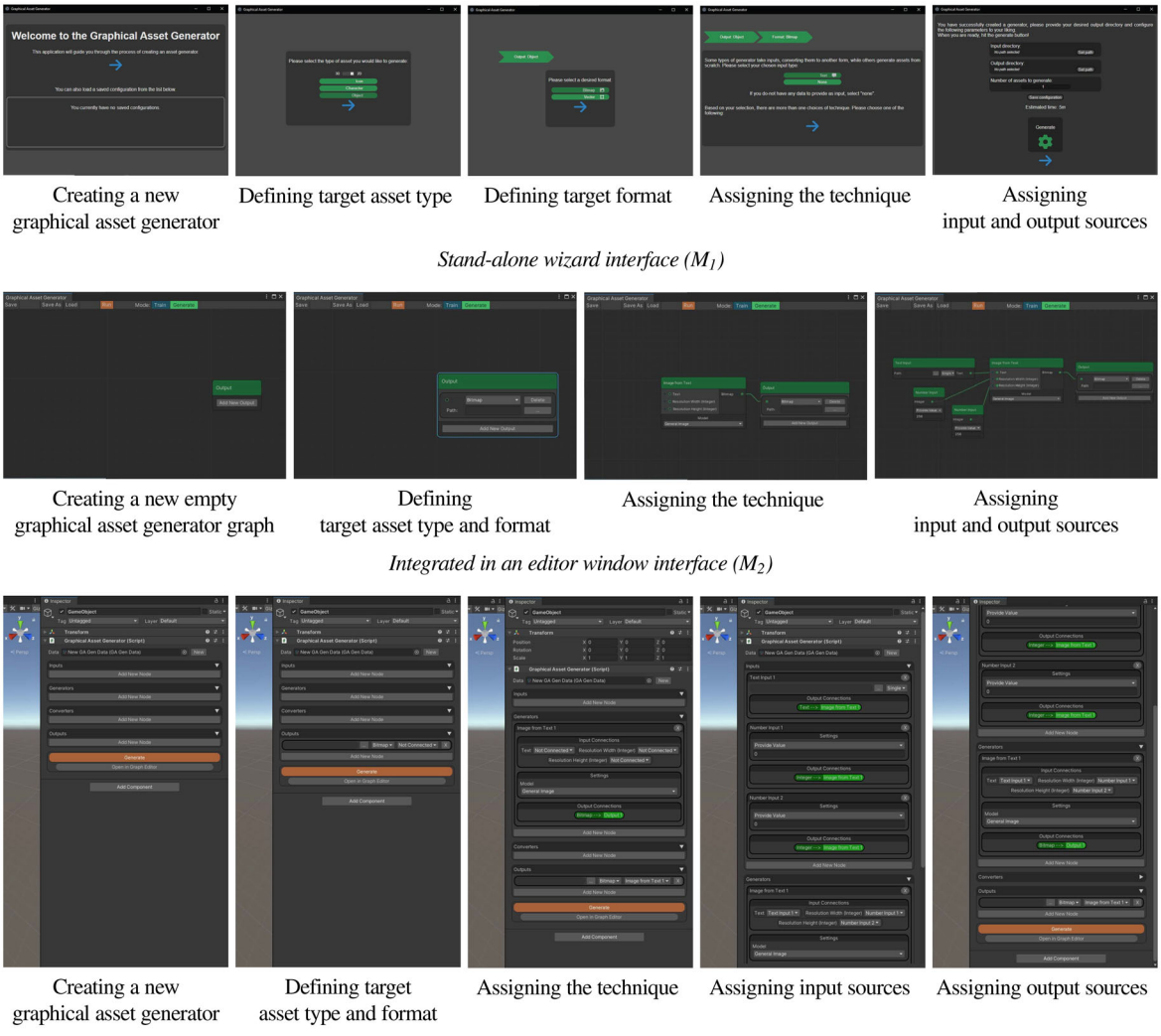

Figure 4. Equivalent step-by-step process for using each interface.

RH2.2 Age impacts preference.
RH3.1 Gender impacts preference.

5. Procedure

To obtain preferences of game developers and designers with regard to generative tools for graphical assets, we developed three mockup interfaces, designed to cover broad forms of interaction. Each participant was asked to complete a form consisting of demographic questions, followed by a section for each mockup. In each section, a mockup is introduced and instructions for installing and using the mockup are provided. The participant is asked to try the mockup at their own pace, then answer a set of questions.

After all three sections are completed, the participant is provided the opportunity to book a semi-structured interview to provide qualitative feedback. In analysis, the questionnaire responses are compared between the three mockups. The study was approved by University's Research Ethics Committee (37113-LR-Jun/2022-39654-1). The following subsections provide more detail on the participants, mockups, questionnaires and interviews.

5.1. Participants and data collection

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

The pool of participants was formed of 16 (13 male, 3 female) game designers and developers, aged 18–39 years old. As the exact age of users was not required, they were asked to choose their age range, 81% were in 18–25 category and 19% in 35–39 category. Ten participants had one year professional experience in the industry, three had between 1 and 4 years of professional experience and three had 5–9 years of experience. The minimum team size reported was 1 and the maximum current team size of the participants was 25, with a median of 11. Exactly half of the participants (50%) reported as artists and designers, and the other half (50%) reported working as developers which included technical artists and programmers.

5.2. Interface prototype mockups

In order to explore the preferences of participants, three prototype mockups have been devised, as presented in section 6. These mockups M_1, M_2 and M_3 represent three forms of tool implementation; stand-alone, integrated in an editor inspector and integrated in an editor window respectively. These mockups were designed as representations of the flow of interaction and were therefore not functional under the hood. M_2 and M_3 have been implemented in the Unity engine. Figure 4 shows a step-by-step process within each of the three mockups, achieving an equivalent result. As the figure shows, M_1 takes a direct approach by asking the user a series of questions that drill down on a configuration based on the user's need. M_2 and M_3 take a more free-form approach, giving the user a palette of options from which to build their generator. Users were provided walk-through videos to watch before using each interface, explaining their usage and the features available.

5.3. Questionnaires

Participants were given the opportunity to explore each mockup at their own discretion and complete a questionnaire for each one, as well as a fourth questionnaire for collecting data independent of the mockups, including demographic data. These questionnaires consist of 5-point and 7-point Likert scales, multiple-choice tick boxes and dichotomous questions. Demographic data, such as age, years of experience and team size, were recorded by range. These surveys were hosted on Microsoft Forms. The research questions and the corresponding survey questions are presented in Table 1.

To analyze questionnaire responses, a combination of descriptive analysis and hypothesis testing will be conducted. Differences in responses between mockups will be assessed through assumption testing, followed by omnibus tests across all mockups, and post-hoc pairwise comparisons with appropriate corrections for family-wise error rate. Furthermore, the relationship between respondent demographics and responses will be examined using correlation analysis.

5.4. Semi-structured interviews

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

An inductive thematic analysis of the interview notes will be conducted. Interview data will be systematically coded, with codes reviewed and grouped into themes. The analysis will aim to contextualize the quantitative findings.

6. Mockups

In order to obtain game designer and developer feedback, three UI mockups have been developed and compared. To ensure that opinions are not shaped by the type of asset, technique or approach to generation; the UI mockups are designed to present a fully customizable generative system, where the user decides these specifics.

Following the definitions of Fukaya et al. (2025), each graphical asset generator can consist of any number of "techniques," which are in turn comprised of an *interaction* type and a *process* type. Inputs are provided via the *interaction* type and outputs of the process can be fed as an input to another "technique," or as a final artifact. In addition, an user can choose to convert the format of input and output data as needed.

The three mockup interfaces represent *stand-alone wizard*, *editor window integrated* and *editor inspector integrated* implementations of this system. Figure 4 shows an example usage process implemented in each of the three mockups, involving the configuration of a generator that takes text as input, outputting bitmap images.

6.1. M_1: Stand-alone wizard

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

A software wizard is an application designed to guide an user through a series of steps (N. N. Group, 2024). These steps can be dependent on choices made in previous steps, thus allowing for branching behavior. This can be used to match the logical process for GaGeTx, with information from each procedure feeding into later procedures.

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

As shown in Figure 4, this mockup introduces the process, asks the user what type of asset they need to generate, the format, and then the input type. The choice of input then determines the possible techniques, in this example, there is only one choice so the user does not receive a selection. This is because there is only one technique for image generation via text. The user then arrives at a screen in which they choose input and output data paths, set the number of assets to generate, and begin the generation process.

6.2. M_2: Integrated in an editor window

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

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

M_2 is shown in Figure 4. Here, the user creates a new bitmap path in the output node, creates an "image from text" node, and then creates an input path node while defining image size parameters.

6.3. M_3: Integrated in an editor inspector view

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

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

In Figure 4, M_3 presents a vertical list of categories, which the user can add to. The user adds an output node with the format of "bitmap," adds an "image from text" generator node, and creates the necessary input nodes. These nodes are then bound to each other using drop-down menus in the UI.

7. Experiment results

To analyze quantitative data, we employ between-group analysis to directly answer RQ1 and RQ7, and observe differences in answers relating to RQ2, RQ3, RQ4, RQ5 and RQ6. As Likert-scale responses did not meet the assumption of normality according to Shapiro–Wilk tests (Royston, 1982) (Table 3), we adopted non-parametric statistical methods. As such, differences in preference between the three mockups are assessed using Friedman tests (Hollander & Wolfe, 1973). For dichotomous results we instead conduct Cochran's Q tests (Cochran, 1950) to assess differences in response between the three mockups, and follow this up with *post-hoc* McNemar tests (Agresti, 1990) between each mockup pair where appropriate. For each repeated pairwise comparison we also apply Holm-Bonferroni correction (Holm, 1979) (reported as *Sig. Adj*), to control the Type I error rate.

To analyze the demographic impact on responses, Spearman correlations (Conover, 1999) between Likert scale demographic questions and responses for RQ1–RQ7 are taken. As gender had binary

responses, we instead conduct Wilcoxon rank sum (Bauer, 1972) tests between male and female respondents for each Likert scale question (RQ1, RQ4, RQ6, RQ7). We additionally conduct Fisher's exact analysis (Agresti, 2002) for gender with dichotomous question results (RQ2, RQ3 and RQ5). Holm-Bonferroni correction is applied across demographic analyses. All statistical tests were evaluated using an alpha level (α) of 0.05.

Likert scale questions RQ1 and RQ6 are rated on 5-point Likert scales ranging from 0 (lowest) to 4 (highest). RQ4 is rated on a 7-point Likert scale with values ranging from 0 (*under a second*) to 6 (*more than 2 hr*). RQ7 SUS scores are between values 0 and 100, calculated based on the answers to five 5-point Likert scale questions (Brooke, 1996).

As for the demographics of participants, 13 of the participants were male and 3 were female. 13 participants were under the age of 25, and 10 participants had under 1 year of experience in professional game development, while 3 had between 1 and 4 years, and 3 had between 5 and 9 years. 12 participants typically work solo, and 4 participants typically work in a team size of 11–25. The most common roles of participants were artist and designer (50%), followed by technical artist and programmer (31.25%).

The analysis of interview notes followed an inductive approach. As data were recorded via researcher notes rather than verbatim transcripts, each recorded point was treated as a single semantic unit. An initial coding pass was conducted over the notes directly, and related codes were merged into themes, which we discuss in section7.4. Given the opt-in nature of the data collection and resulting small sample size, this analysis is intended to provide contextual support for the quantitative findings rather than serve as a stand-alone contribution.

7.1. Descriptive statistics

Overall, responses to RQ1 (usefulness) were moderately positive ($\mu = 2.71, \sigma = 0.65$). Of the options in RQ2, the most popular choices ($\mu > 0.5$) were RQ2.1 (generating inspiration), RQ2.2 (exploring ideas) and RQ2.3 (creating placeholder assets). The number of options picked by participants was between 2 and 3 ($\mu = 2.67, \sigma = 1.48$). For RQ3, participants show a clear preference for a high volume of lower quality assets over a lower volume of high quality assets ($\mu = 0.12, \sigma = 0.34$). RQ4 responses show an acceptable generation time for a single asset to be between 1 and 10 min ($\mu = 1.67, \sigma = 0.69$). RQ5 responses point toward preference for a tool integrated within a game-engine or editor ($\mu = 0.78, \sigma = 0.428$ where 1 equals integrated). For RQ6, results show that participants considered the ability to configure or modify the tools important ($\mu = 3.46, \sigma = 0.99$). The overall usability (RQ7) across all mockups was considered good ($\mu = 79.17, \sigma = 13.39$ according to the adjective rating scale (Bangor et al., 2009)).

7.2. Tool preference

Before conducting comparative statistical tests, we assessed the normality of Likert scale responses using Shapiro-Wilk tests. Table 3 presents our results, which indicate that responses to RQ1, RQ2 (frequency of options), RQ4, RQ6 and RQ7 deviate significantly from normality ($p < 0.05$). As the assumption of normality for most questions is not met, we conducted Friedman tests for each Likert scale question to assess differences in preference across the three mockups. None of these Friedman tests indicated statistically significant differences between mockups, as shown in Table 4. For dichotomous questions (RQ2, RQ3 and RQ5), we instead conducted a Cochran's Q test (presented in Table 5), where we find statistically significant differences in choice distributions across mockups for RQ2.1, RQ2.4, RQ2.6 and RQ2.7 ($p < 0.05$). Accordingly, post-hoc pairwise McNemar tests were conducted for these questions. As shown in Table 6, no differences between specific pairs of mockups can be identified. This is limited by a lack of in-group variability and identical ratings between groups for some comparisons.

Table 3. Shapiro-Wilk (Royston, 1982) Test results for Likert scale questions (RQ1, RQ2* frequency of options picked, RQ4, RQ6 and RQ7), presenting the mean, standard deviation (SD), *W* statistic and significance scores.

Research question	Overall mean	Overall SD	W	Sig.
RQ1: (0–4)	2.71	0.65	0.774	<0.001
RQ2*: (0–7)	2.67	1.48	0.871	<0.001
RQ4: (0–6)	1.67	0.69	0.770	<0.001
RQ6: (0–4)	3.46	0.99	0.862	<0.001
RQ7: (0–100)	79.17	13.39	0.902	<0.001

Table 4. Friedman test (Hollander & Wolfe, 1973) significance scores for Likert scale questions (RQ1, RQ4, RQ6, RQ7) and RQ2* frequency of options picked, alongside means and standard deviations per mockup.

	M_1		M_2		M_3		Overall (Friedman)	
Research question	Mean	SD	Mean	SD	Mean	SD	Chi-squared	Sig.
RQ1: (0–4)	2.50	0.52	2.81	0.66	2.81	0.75	5.00	0.082
RQ2*: (0–7)	2.31	1.08	3.19	1.91	2.5	1.26	3.71	0.156
RQ4: (0–6)	1.69	0.79	1.69	0.79	1.62	0.50	0.25	0.883
RQ6: (0–4)	3.38	1.31	3.50	0.52	3.50	1.03	1.56	0.459
RQ7: (0–100)	80.94	11.03	79.22	10.44	77.34	18.02	2.00	0.368

Table 5. Cochran's Q (Cochran, 1950) significance scores for binary choice questions (RQ2, RQ3 and RQ5).

		M_1		M_2		M_3		Overall (Cochran's Q)	
Research question		Mean	SD	Mean	SD	Mean	SD	Q	Sig.
RQ2:	RQ2.1 (0–1)	0.56	0.51	0.81	0.41	0.69	0.48	6	0.049
	RQ2.2 (0–1)	0.69	0.48	0.81	0.40	0.69	0.48	4	0.135
	RQ2.3 (0–1)	0.50	0.52	0.62	0.50	0.69	0.48	2.8	0.247
	RQ2.4 (0–1)	0.12	0.34	0.25	0.45	0.00	0.00	6	0.049
	RQ2.5 (0–1)	0.00	0.00	0.12	0.34	0.00	0.00	4	0.135
	RQ2.6 (0–1)	0.31	0.48	0.12	0.34	0.00	0.00	7.6	0.022
	RQ2.7 (0–1)	0.12	0.34	0.44	0.51	0.44	0.51	10	0.007
RQ3: (0–1)		0.12	0.34	0.12	0.34	0.12	0.34	0	1.000
RQ5: (0–1)		0.69	0.48	0.81	0.40	0.88	0.34	2.8	0.247

Table 6. *Post-hoc* McNemar tests (Agresti, 1990) conducted pairwise for statistically significant dichotomous questions (RQ2.1, RQ2.4, RQ2.6, RQ2.7).

		M_1 with M_2			M_1 with M_3			M_2 with M_3		
		Chi-squared	Sig.	Sig. Adj	Chi-squared	Sig.	Sig. Adj	Chi-squared	Sig.	Sig. Adj
RQ2	RQ2.1 (0–1)	2.25	0.134	0.401	0.5	0.479	0.959	0.5	0.479	0.959
	RQ2.4 (0–1)	0.5	0.479	–	*v	*v	*v	*v	*v	*v
	RQ2.6 (0–1)	1.33	0.248	–	*v	*v	*v	*v	*v	*v
	RQ2.7	3.2	0.0736	.147	3.2	0.736	.147	*s	*s	*s

Holm-Bonferroni correction (Holm, 1979) (sig. Adj) was applied where relevant to control for family-wise error rate. (*) v indicates cases where the test was not possible due to no *variability* in the responses for one of the two tested mockups. (*) s indicates cases where the test was not possible due to *identical* responses between the two tested mockups for all participants.

7.3. Demographic impact

Given the small sample size and ordinal nature of the data, we assess the demographic impact through non-parametric tests. Table 7 presents Spearman correlations between Likert scale demographic responses and each questionnaire question (RQ1–RQ7). Each Likert scale question (RQ1, RQ4, RQ6 and RQ7) has a statistically significant ($p < 0.05$) correlation with respondent *age*, *experience* and *size of team*. For RQ1, RQ6 and RQ7, the age, experience and team size of respondents each show strong negative correlations ($\rho < -0.5$). For RQ4 the results are similar, with negative correlations for age ($\rho = -0.495$), experience ($\rho = -0.563$) and team size ($\rho = -0.545$). RQ5 responses have a significant negative correlation with respondent age ($\rho = -0.542$, $p < 0.05$), and a moderate negative correlation with experience ($\rho = -0.395$, $p < 0.05$), as well as a moderate negative correlation for team size ($\rho = -0.318$) that is not significant after correction (unadjusted $p = 0.028$, adjusted $p = 0.165$). For RQ3, correlation with age is not statistically significant, while experience and team size are, with

Table 7. Spearman correlations (Conover, 1999) and associated *p*-values for RQ1–RQ7 with each single-choice demographic question.

		Correlation		
Research question		Correlation (ρ)	Sig.	Sig. Adj
RQ1	Age	−0.558	<0.001	<0.001
	Experience*	−0.646	<0.001	<0.001
	Size of team	−0.628	<0.001	<0.001
RQ2.1	Age	−0.713	<0.001	<0.001
	Experience*	−0.828	<0.001	<0.001
	Size of team	−0.806	<0.001	<0.001
RQ2.2	Age	−0.788	<0.001	<0.001
	Experience*	−0.926	<0.001	<0.001
	Size of team	−0.904	<0.001	<0.001
RQ2.3	Age	0.389	0.006	0.038
	Experience*	0.243	0.096	0.481
	Size of team	0.178	0.226	0.678
RQ2.4	Age	−0.182	0.217	0.705
	Experience*	−0.033	0.822	0.822
	Size of team	0.017	0.909	1
RQ2.5	Age	−0.100	0.498	0.705
	Experience*	0.221	0.131	0.523
	Size of team	0.309	0.032	0.165
RQ2.6	Age	−0.198	0.176	0.705
	Experience*	0.133	0.367	0.795
	Size of team	0.231	0.011	0.458
RQ2.7	Age	−0.339	0.018	0.091
	Experience*	−0.164	0.265	0.795
	Size of team	−0.095	0.519	1
RQ3	Age	−0.182	0.217	0.705
	Experience*	0.401	0.005	0.033
	Size of team	0.561	<0.001	<0.001
RQ4	Age	−0.495	<0.001	0.003
	Experience*	−0.563	<0.001	<0.001
	Size of team	−0.545	<0.001	<0.001
RQ5	Age	−0.542	<0.001	<0.001
	Experience*	−0.395	0.006	0.033
	Size of team	−0.318	0.028	0.165
RQ6	Age	−0.641	<0.001	<0.001
	Experience*	−0.589	<0.001	<0.001
	Size of team	−0.529	<0.001	<0.001
RQ7	Age	−0.587	<0.001	<0.001
	Experience*	−0.798	<0.001	<0.001
	Size of team	−0.808	<0.001	<0.001

(*) Experience stands for years of professional experience.

moderate and strong positive correlation respectively (experience: $\rho = 0.401$, team size: $\rho = 0.561$). Of the selected pipeline applications for RQ2, *generating inspiration* (RQ2.1) and *exploring ideas* (RQ2.2) have significant and highly positive correlation with age, experience and team size ($\rho > 0.7$, $p < 0.05$), while *creating placeholder assets* (RQ2.3) has a moderate positive correlation with age ($\rho = 0.389$, $p < 0.05$).

While there is a disproportionate frequency of male respondents (13 male, 3 female), for completeness we conduct additional comparative tests. Wilcoxon rank-sum tests for each Likert scale question (RQ1, RQ4, RQ6 and RQ7) between male and female respondents are presented in Table 8. Responses for SUS scores (RQ7) are found to be significantly lower from female respondents than male respondents ($p < 0.05$). Fisher's exact tests between dichotomous results and gender groups are presented in Table 9. Here we find a significantly higher ($p < 0.05$) choice of *creating assets from scratch* (RQ2.6) and *quality over volume* (RQ3).

7.4. Interview results

During the interviews, the participants generally confirmed the results of the questionnaires. There is no clear preference between the three mockups but they emphasized compatibility with their existing tools. One participant mentioned that each mockup was fine, as long as they didn't need a converter or

Table 8. Wilcoxon rank-sum (Bauer, 1972) *p*-values for RQ1, RQ4, RQ6 and RQ7 with gender, and median values per gender.

Research question		Correlation			
		Male median	Female median	Sig.	Sig. Adj
RQ1	Gender	3	2	0.132	0.397
RQ4	Gender	2	1	0.224	0.449
RQ6	Gender	4	3	0.934	0.934
RQ7	Gender	85	70	0.003	0.011

(*) Experience stands for years of professional experience.

Table 9. Fisher's exact (Agresti, 2002) test *p*-values for RQ2, RQ3 and RQ5 with gender.

Research question		Correlation			
		Male picked (%)	Female picked (%)	Sig.	Sig. Adj
RQ2.1	Gender	74	33	0.067	0.336
RQ2.2	Gender	79	33	0.038	0.231
RQ2.3	Gender	64	33	0.197	0.469
RQ2.4	Gender	10	33	0.157	0.469
RQ2.5	Gender	0	33	0.013	0.093
RQ2.6	Gender	7	66	0.002	0.019
RQ2.7	Gender	29	66	0.086	0.342
RQ3	Gender	0	100	<0.001	<0.001
RQ5	Gender	76	100	0.320	0.469

(*) Experience stands for years of professional experience.

third tool to get the assets into their game. Most participants confirmed that they would prefer to use the tool for ideation and therefore prefer speed and volume over quality.

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

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

7.5. Findings

As there is no significant difference in preference between the three mockups, we primarily rely on descriptive statistics to observe patterns in responses supported by the interview findings. For H1, given answers to RQ5, we find a preference for integrated solutions over stand-alone implementations (M_1: $\mu = 0.69$, $\sigma = 0.48$, M_2: $\mu = 0.81$, $\sigma = 0.40$, M_3: $\mu = 0.88$, $\sigma = 0.34$ where 1 represents an "integrated solution"). This is supported by slight but non-significantly greater ratings of *usefulness* for the integrated solutions (RQ1 $M_2 > M_1$ and $M_3 > M_1$), with interviewees emphasizing a need for such tools to be compatible with their existing software pipeline. Regarding H2, results of RQ1 suggest no significant preference between window and inspector integrated interfaces. For H3, we find that respondents prefer early pipeline applications, with greater rates of selection for *inspiration*, *exploration* and *prototyping* across the board, which is corroborated in the interviews by an interest in *ideation*. With H4, we find a preference for volume over quality across the board ($\mu = 0.12$, $\sigma = 0.34$ where 0 represents

"volume over quality"). Regarding H5 (RQ4), a mean selection of 1.67 ($\sigma = 0.69$) shows a preference for a maximum generation time between 1 and 10 min for a single asset. For H6, a mean score of 3.46 ($\sigma = 0.99$) on a 5-point Likert scale (RQ6), suggests a moderate preference for the ability to configure or modify the tool. Answers to RQ7 present an overall mean SUS score of 79.17 ($\sigma = 13.39$). Following the adjective rating system provided by Bangor et al. (Bangor et al., 2009), the mean overall usability score can be classed as "Good" (78.75). The individual mean SUS scores for M_1, M_2 and M_3 were 80.94 ($\sigma = 11.03$), 79.22 ($\sigma = 10.44$), and 77.34 ($\sigma = 18.02$), respectively.

For the impact of demographics on ratings, RH1.1, RH2.1 and RH2.2, we find that *team size*, *experience* and *age* each negatively correlate with *usefulness* (RQ1), *acceptable generation speed* (RQ4), *need for configurability* (RQ6) and *useability* (RQ7). *Experience* and *team size* have strong positive correlations with a preference for low volumes of high quality assets (RQ3), while *age*, *experience* and *team size* all have positive correlations with the early pipeline applications: *generating inspiration* (RQ2.1), *exploring ideas* (RQ2.2) and *creating placeholder assets* (RQ2.3). Collectively, these findings indicate that age, experience and team size are associated with higher quality standards, greater appreciation for early-stage applications, and a more measured, selective approach toward adopting such tools. For gender (RH3.1), we find the statistical power to be too low for any conclusive findings, as the gender distribution of our respondents is heavily skewed.

8. Discussion and comparison of results

This study has found that there is a preference for all UIs to be integrated into game-engines/editors, and for the system to generate larger amounts of lower quality assets as opposed to smaller amounts of higher quality assets. This is also corroborated by a preference for using such a system in early stages of development to generate inspiration, explore ideas and create placeholders. The ability to configure or modify such a tool is also considered important. The maximum acceptable time scale for generating a single asset is between 1 and 10 min. This is not surprising, as existing tools such as Didimo (D. Inc, 2023) can take 5–10 min to generate assets, especially at higher qualities. The other findings suggest that a much faster generation speed would be ideal, in order to provide larger volumes of assets for inspiration and ideation. There is however, no statistically significant difference in preference between a stand-alone multi-choice based UI, integrated graph-view UI and integrated inspector UI for generating graphical assets.

We acknowledge the limitations of our sample size and the lack of granularity in interface configurations. Furthermore, the absence of functionality means that results reflect participant assumptions about tool capability, rather than direct experiences with fully featured systems. Given the rate of improvement in generative tools, preferences could shift as the competency of tools advances. Despite these limitations, our findings largely corroborate existing research, highlighting the importance of easy prototyping, the ability to design while implementing, compatibility with other tools and the ability to stimulate inspiration (Kasurinen et al., 2013; Walton et al., 2022).

Our results align with and expand upon existing research findings regarding game design and development tools (Colado et al., 2023; Kasurinen et al., 2013; Lai et al., 2020; Ling et al., 2024; Togelius et al., 2011; Walton et al., 2022), addressing the gaps in previous research. Namely, previous work has suggested that designers and developers value variety in outputs (Walton et al., 2022), which corroborates with the usage of generative methods as explorative tools (Colado et al., 2023; Kasurinen et al., 2013; Ling et al., 2024). Our results not only reflect this need and preference, but identify the exact use cases designers and developers target within the pipeline, providing further context to this requirement and allowing tools to cater to those specific needs.

Although previous research demonstrates an interest in using these tools in pipelines (Colado et al., 2023; Ling et al., 2024), they do not clarify how and where in the pipeline GAG tools would be suitable. Our findings confirm that they are preferred in the early stages. Furthermore, we expand on research that suggests designers and developers prefer well integrated configurable tools (Kasurinen et al., 2013) through direct quantitative results and interview feedback. Our results show that the style of interface

only matters as far as it is functional and integrates well with the rest of the creative pipeline. For integration with existing software the interface must be achieved through APIs, which also inform the design language of the tool.

Previous works focus on high-level tool implications (Togelius et al., 2011; Walton et al., 2022), but do not consider the practical aspects that can make or break the applicability of a tool. On the contrary, Lai et al. (2020) identify a need to respect existing work processes. We elaborate on this by pinpointing the use of standardized data formats as an important requirement for generative tools in practice, as well as the need for design features that align with the host environment and thus meet user expectations.

Users were interested in using the tool for inspiration and prototyping, and favored integration, and thus compatibility with their chosen game engine. Additionally, when observing the characteristics of the three preferred usages according to the characteristics shown in Figure 3, in general, users prefer generators for work in an *offline* context, with an expectation for lower quality and to augment, rather than replace an existing process in their pipeline.

Given the findings regarding pipeline preferences, we refine the graphical asset GDLC to include the *offline* generative tool applications, shown in Figure 5. Here, the pipeline applications *creating assets from existing designs* and *creating assets from scratch* both apply under the *Assets (core)* step and have thus been combined. We also show the human-computer interaction for each of these steps with early pipeline tasks: inspiration, placeholders and explore, employing an iterative shared initiative approach, where both human and AI provide equal input; and late pipeline tasks: Assets (core) and Assets (vary), taking a human-driven approach, in which designs are already well defined and the goal is to accurately implement them.

While existing work acknowledges that different balances of user-machine initiative can be useful in different situations (Lai et al., 2020; Liapis et al., 2016), research has not yet ascertained how this balance maps onto the stages of a design and development pipeline. Through our findings we identify the appropriate flow of initiative at different stages of a graphical asset pipeline as presented in Figure 5. At earlier stages of the pipeline less user control is required, thus the tool can have a larger share of the initiative. Once development shifts into a production phase, the design choices have already been made, and thus the user takes a more prescriptive role over the tool. This helps to inform the design and integration of GAG tools, based on their intended stage in the pipeline.

The flow of graphical asset data in both stand-alone and integrated scenarios is presented in Figure 6. As shown, creative tools can either be integrated within the main development environment (game engine) or implemented externally as stand-alone programs. Users interact with these tools via user-interfaces, and in the case of integrated tools, the implementation itself can rely on the native features of the environment through application programming interfaces (API). Stand-alone external generative tools, as with other creation tools such as Photoshop (Adobe Inc, 2005), Blender (Blender Foundation, 2024) and Visual Studio (Microsoft Corporation, 2022) instead rely on outputting artifacts in standardized formats that most game engines support, such as FBX, OBJ, PNG and JPEG. Integrated tools may implement GAG technique *interactions* and *processes* using game engine UI APIs and native engine features via back end APIs respectively.

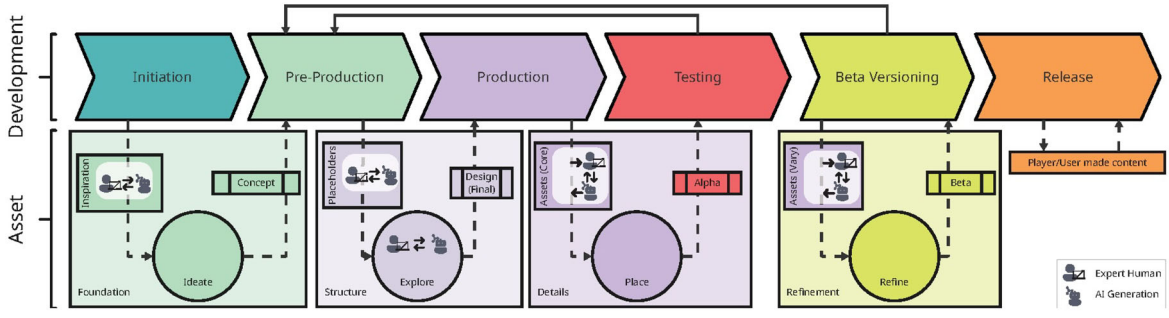

Figure 5. The game design and development pipeline based on the GDLC of Ramadan and Widyani (2013) and expanded to include graphical asset findings.

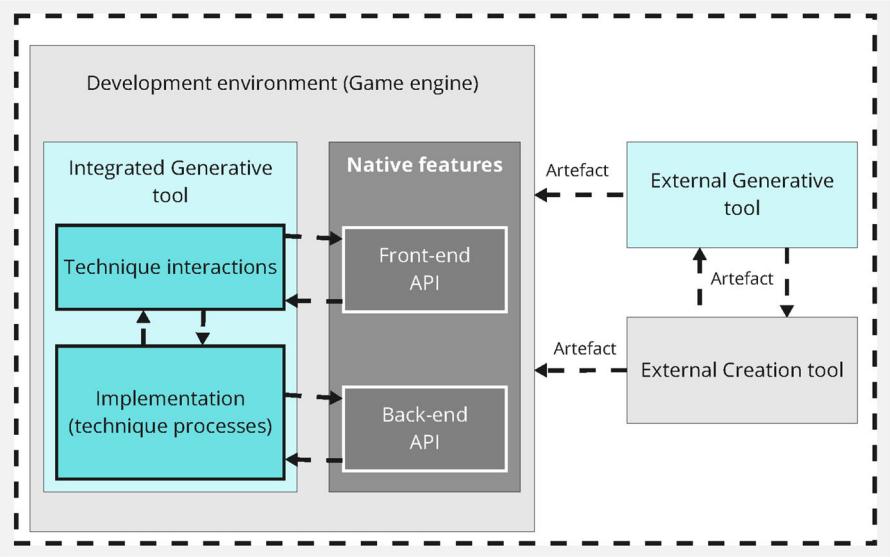

Figure 6. Graphical asset creation within the game production framework.

9. Heuristics for generative tools

To improve the applicability of our findings to the design and integration of GAG tools, we have derived a set of actionable heuristics, shown in Figure 7, spanning both the user intent as well as the implementation and integration of GAGs in design/development pipelines. Table 10 presents the mapping of these heuristics to the research questions.

The first consideration is user intent with regard to the pipeline application, what *technique* is used and what the desired outcome is. For user intent, we present three heuristics based on the findings:

- *H-PIP: Cater to early pipeline usage.* Users favor early pipeline GAGs for purposes of generating inspiration, exploring ideas, and creating placeholders, as presented in Figure 7. Furthermore, the system should aim for variety and volume of output rather than quality. Users prefer not to use the results in a finished product directly, but would rather use them as ideas to build on and refine.
- *H-SYN: Synergise generative and human-driven design and development.* The preferred use cases are all supportive of a human-driven design and development process, and do not directly replace the final creation of assets. Rather, they give designers and developers ideas to expand on, break creative block or help to streamline more mundane or inconsequential tasks. This entails the majority of applications, excluding player made content as shown in Figure 7.
- *H-CON: Facilitate tool configurability.* Game design and development is a creative endeavor, thus, unique and varied outputs are a necessity. A graphical asset generator that has a pre-defined style or rigid way of working becomes a tool with limited applicability. Configurability should be considered, regardless of the chosen application.

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

- *H-LAN: Match the design language of the environment.* The less the user has to learn on top of what they are accustomed to within their pipeline, the easier it is to incorporate the tool. If the interface of the tool provides functionality that other native features of the environment have, such as the ability to undo and redo changes, frustration or confusion can be avoided. This can be achieved through appropriate use of front-end UI APIs within the host software, as illustrated in Figure 6.
- *H-DAT: Use common and expected data formats.* To smoothly integrate the tool, whether it is integrated or stand-alone, the output artifacts must be in a format that the development environment can utilize. Typically these are ubiquitous data formats, such as OBJ or common proprietary formats

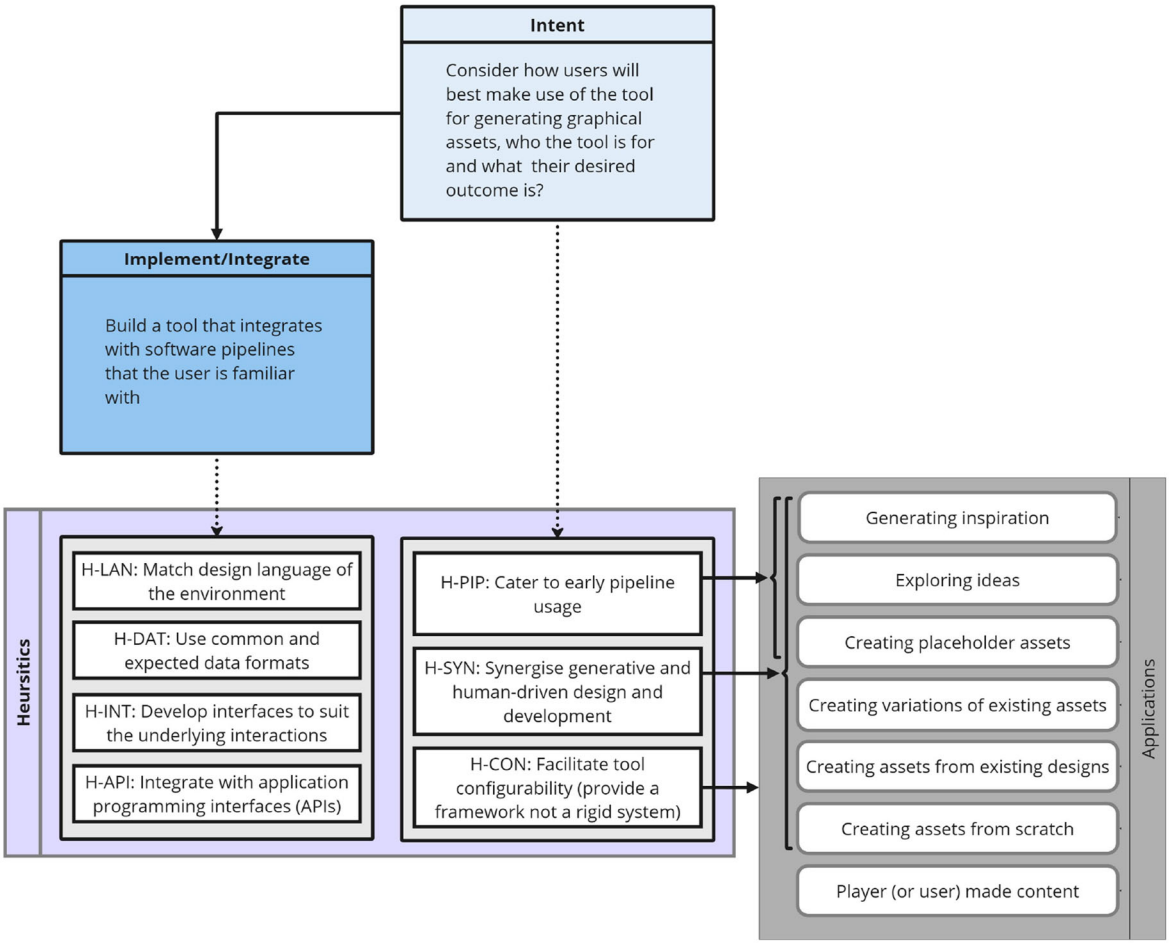

Figure 7. Mapping the user considerations, intent and implementation/integration to the heuristics.

Table 10. Mapping of heuristics to research questions.

Heuristics		Research question(s)
H-CON	Facilitate tool configurability	RQ6
H-PIP	Cater to early pipeline usage	RQ2, RQ3, RQ4
H-SYN	Synergise generative and human-driven design and development	RQ2
H-LAN	Match the design language of the environment	RQ8
H-DAT	Use common and expected data formats	RQ8, RQ9
H-INT	Develop interfaces to suit the underlying interactions	RQ1, RQ7
H-API	Integrate with application programming interfaces (APIs)	RQ5, RQ8

that are supported due to popularity, such as Autodesk FBX. This is particularly important in the transfer of artifacts from and between external generative tools, creation tools and the users development environment of choice, as shown in Figure 6.

- *H-INT: Develop interfaces to suit the underlying interactions.* Design an interface to best interact with the features of the tool, assuming that it provides good usability, there is no preference between step by step options, graph view windows and inspector editors.
- *H-API: Integrate with application programming interfaces (APIs).* The tool should be integrated as a part of an existing popular engine or editor, or at the very least should have full compatibility with said application. Technique interactions and processes must interact with each other while appropriately utilizing the host game engine's front-end and back-end APIs, as shown in Figure 6.

9.1. Theoretical and practical implications

As GAG methods advance, their applications in games will continue to become more and more prevalent. Given our findings and resulting heuristics, the role of GAG tools within game pipelines is clearer.

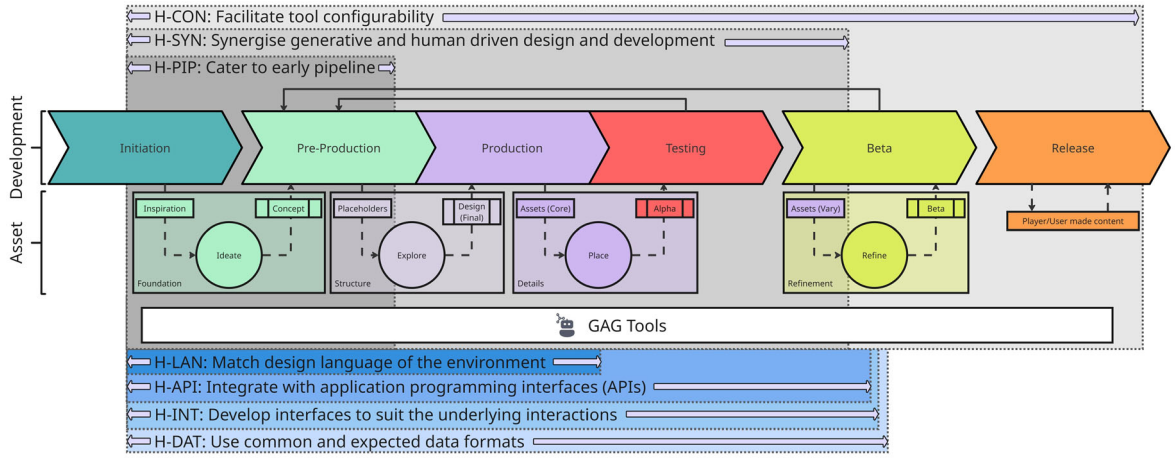

Figure 8. Refined game asset design and development pipeline with heuristics resulting from research findings.

With this, we adapt the existing GDLC framework (Ramadan & Widyani, 2013) to meet with the growing interest in GAG tools. We integrate these heuristics into the graphical asset GDLC, shown in Figure 8, illustrating their relevance at different stages of the pipeline.

As previously mentioned, whilst there is extensive work on the benefits and potentials of GAG tools in game design and development (Li et al., 2021; Shen et al., 2021; Shen & Chen, 2022), there are limited studies on their use and fit within a design and development pipeline (Colado et al., 2023; Ling et al., 2024). While the use of creative tools is well established (Kasurinen et al., 2013; Seidel et al., 2019), the behavior and preferences of game designers and developers with regard to GAG tools had not been explored. Hence, there are no preexisting heuristics or guidelines on how to effectively incorporate them into the design and development of a game.

This study has aimed to take the first steps in addressing this gap, and to contextualize the behavior and preference of designers and developers within the larger scope of the development pipeline. The results confirm that designers and developers are clearly interested in incorporating graphical asset generation tools, while many are already using them in some capacity (Unity Technologies, 2024).

There are a number of practical implications that arise from the study results. GAG tools should be developed for incorporation in the early stages of design and development, these tools should be integrated into existing design and development environment as much as possible, they should prioritize larger volumes of variations over quality, the users should be able to configure and have some control over the tools and the generated output artifacts, and finally the generated artifacts should comply with common data formats. These implications are derived from the behavior and preferences of users as there is a strong preference toward using GAG tools in the early design stages for generating inspiration, exploring ideas and creating placeholders. A large volume of variations that can be further refined by hand or used for ideation is preferred. There is also a strong interest in being able to configure GAG tools, and for outputs to comply with common data formats.

In order to efficiently take advantage of the opportunities provided by GAG tools, the practical needs and preferences of the game designers and developers are paramount. Whilst the current generic approach to AI driven content generation has provided powerful interesting tools, lack of empirical research in the behavior and preferences of game designers and developers, has led to obstacles for incorporating them well into the design and development pipelines.

10. Concluding discussions

This study has investigated the behavior and preferences of game designers and developers toward the use of graphical asset generation tools, e.g., PCG, and other intelligent generative approaches. The study considers how these users interact with generative tools within the creation pipeline and takes into account the different contributing factors such as their experience and the size of their respective teams. The results indicate that users prefer early design stage applications such as generating inspiration,

exploring ideas and creating placeholders. As such, users prefer tools that can output larger volumes of artifacts at the cost of quality. No preference is shown for any one form of interface, instead, users express a need for smooth integration of GAG tools within their existing development environments. This includes complying with the design language of the host software (when integrated), presenting the necessary controls for the tool to function and outputting artifacts via common data formats. Users express a need for tool configurability, and find generation times of up to 10 min to be acceptable. These findings have led to a set of heuristics for the use and integration of generative tools within game design and development pipelines, with considerations for the user's intended application.

Author contributions

CRediT: **Kaisei Fukaya**: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft; **Damon Daylamani-Zad**: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing; **Harry Agius**: Formal analysis, Methodology, Validation, Writing – original draft, Writing – review & editing.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Kaisei Fukaya http://orcid.org/0000-0001-9828-7641
Damon Daylamani-Zad http://orcid.org/0000-0001-7849-458X
Harry Agius http://orcid.org/0000-0002-8818-2683

References

Adobe Inc. (2005). *Adobe photoshop*. https://www.adobe.com/products/photoshop.html
Agresti, A. (1990). *Categorical data analysis*. Wiley.
Agresti, A. (2002). *Categorical data analysis*. Wiley.
Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *J. Usability Studies*, *4*(3), 114–123. https://dl.acm.org/doi/10.5555/2835587.2835589
Bauer, D. F. (1972). Constructing confidence sets using rank statistics. *Journal of the American Statistical Association*, *67*(339), 687–690. https://doi.org/10.1080/01621459.1972.10481279
Bay 12 Games. (2006). *Dwarf fortress.* http://www.bay12games.com/dwarves/
Blender Foundation. (2024). *Blender*. https://www.blender.org/
Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability Evaluation in Industry*, *189*(194), 4–7. https://doi.org/10.1201/9781498710411-35
Cochran, W. G. (1950). The comparison of percentages in matched samples. *Biometrika*, *37*(3–4), 256–266. https://doi.org/10.2307/2332378
Colado, I. J. P., Colado, V. M. P., Morata, A. C., Píriz, R. S. C., & Manjón, B. F. (2023). *Using new AI-driven techniques to ease serious games authoring* [Paper presentation]. 2023 IEEE Frontiers in Education Conference (FIE), College Station, TX (pp. 1–9). https://doi.org/10.1109/FIE58773.2023.10343021
Conover, W. (1999). *Practical nonparametric statistics* (3rd edition) Wiley.
D. Inc. (2023). Didimo Inc. https://www.didimo.co/
Fukaya, K., Daylamani-Zad, D., & Agius, H. (2024). Evaluation metrics for intelligent generation of graphical game assets: A systematic survey-based framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(12), 7998–8017. https://doi.org/10.1109/TPAMI.2024.3398998
Fukaya, K., Daylamani-Zad, D., & Agius, H. (2025). Intelligent generation of graphical game assets: A conceptual framework and systematic review of the state of the art. *ACM Computing Surveys*, *57*(5), 1–38. https://doi.org/10.1145/3708499
Getto, R., Kuijper, A., & Fellner, D. W. (2020). Automatic procedural model generation for 3D object variation. *The Visual Computer*, *36*(1), 53–70. https://doi.org/10.1007/s00371-018-1589-4
Hello Games. (2016). *No man's sky*. https://www.nomanssky.com/
Hollander, M., & Wolfe, D. A. (1973). *Nonparametric statistical methods*. John Wiley & Sons.
Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*(2), 65–70.

Interactive Data Visualization Inc. (2022). *(IDV), SpeedTree*. https://store.speedtree.com/
Jansson, D. G., & Smith, S. M. (1991). Design fixation. *Design Studies*, *12*(1), 3–11. https://doi.org/10.1016/0142-694X(91)90003-F
Jones, R. K., Charatan, D., Guerrero, P., Mitra, N. J., & Ritchie, D. (2021). Shapemod: Macro operation discovery for 3D shape programs. *ACM Transactions on Graphics*, *40*(4), 1–16. https://doi.org/10.1145/3450626.3459821
Jumani, A. K., Shi, J., Laghari, A. A., Estrela, V. V., Sampedro, G. A., Almadhor, A., Kryvinska, N., & Ul Nabi, A. (2024). Quality of experience that matters in gaming graphics: How to blend image processing and virtual reality. *Electronics*, *13*(15), 2998. https://doi.org/10.3390/electronics13152998
Kasurinen, J., Strandén, J.-P., & Smolander, K. (2013). *What do game developers expect from development and design tools?* [Paper presentation]. Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, EASE '13, Porto de Galinhas, Brazil (pp. 36–41). ACM. https://doi.org/10.1145/2460999.2461004
Laghari, A. A., Estrela, V. V., Li, H., Shoulin, Y., Khan, A. A., Anwar, M. S., Wahab, A., & Bouraqia, K. (2024). Quality of experience assessment in virtual/augmented reality serious games for healthcare: A systematic literature review. *Technology and Disability*, *36*(1–2), 17–28. https://doi.org/10.3233/TAD-230035
Laghari, A. A., He, H., Memon, K. A., Laghari, R. A., Halepoto, I. A., & Khan, A. (2019). Quality of experience (QOE) in cloud gaming models: A review. *Multiagent and Grid Systems*, *15*(3), 289–304. https://doi.org/10.3233/MGS-190313
Laghari, A. A., Laghari, K-u-R., Memon, K. A., Soomro, M. B., Laghari, R. A., & Kumar, V. (2020). Quality of experience (QOE) assessment of games on workstations and mobile. *Entertainment Computing*, *34*, 100362. https://doi.org/10.1016/j.entcom.2020.100362
Laghari, A. A., Li, H., Shoulin, Y., Jumani, A. K., Khan, A. A., & Dahri, F. H. (2025). Internet of things for gaming: A review. *Entertainment Computing*, *52*, 100910. https://doi.org/10.1016/j.entcom.2024.100910
Lai, G., Latham, W., & Leymarie, F. F. (2020). *Towards friendly mixed initiative procedural content generation: Three pillars of industry* [Paper presentation]. Proceedings of the 15th International Conference on the Foundations of Digital Games, FDG '20, New York, NYUSA (pp. 1–4). ACM. https://doi.org/10.1145/3402942.3402946
Li, J., Yang, J., Zhang, J., Liu, C., Wang, C., & Xu, T. (2021). Attribute-conditioned layout GAN for automatic graphic design. *IEEE Transactions on Visualization and Computer Graphics*, *27*(10), 4039–4048. https://doi.org/10.1109/TVCG.2020.2999335
Liapis, A., Smith, G., & Shaker, N. (2016). Mixed-initiative content creation. In S. Noor, J. Togelius, & N. Mark (Eds.), *Procedural content generation in games* (vol. 11, pp. 195–214). Springer International Publishing. https://doi.org/10.1007/978-3-319-42716-4_11
Liapis, A., Yannakakis, G., & Togelius, J. (2013). Designer modeling for personalized game content creation tools. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Boston, Massachusetts* (vol. 9, pp. 11–16). AAAI. https://doi.org/10.1609/aiide.v9i2.12587
Lin, J., Yuan, Y., & Zou, Z. (2021). Meingame: Create a game character face from a single portrait. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(1), 311–319. https://doi.org/10.1609/aaai.v35i1.16106
Ling, L., Chen, X., Wen, R., Li, T. J.-J., & Lc, R. (2024). Sketchar: Supporting character design and illustration prototyping using generative AI. *Proceedings of the ACM on Human-Computer Interaction*, *8*(CHI PLAY), 1–28. https://doi.org/10.1145/3677102
Microsoft Corporation. (2022). *Visual studio*. https://visualstudio.microsoft.com/
Mojang Studios. (2011). Minecraft. https://www.minecraft.net/
N. N. Group. (2024). *Wizards: Definition and design recommendations* (accessed: 2024 September 13). https://www.nngroup.com/articles/wizards/
Newell, K., Patel, K., Linden, M., Demetriou, M., Huk, M., & Mahmoud, M. (2021). *Evolution of software development in the video game industry* [Paper presentation]. 2021 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, Nevada (pp. 1989–1996). https://doi.org/10.1109/CSCI54926.2021.00367
Ramadan, R., & Widyani, Y. (2013). *Game development life cycle guidelines* [Paper presentation]. Proceedings of 2013 International Conference on Advanced Computer Science and Information Systems, ICACSIS 2013 (pp. 95–100). https://doi.org/10.1109/ICACSIS.2013.6761558
Royston, J. P. (1982). An extension of Shapiro and Wilk's *w* test for normality to large samples. *Journal of the Royal Statistical Society: Series C-Applied Statistics*, *31*(2), 115–124. https://doi.org/10.2307/2347973
Saharia, C., Chan, W., Saxena, S., Lit, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Karagol Ayan, B., Mahdavi, S. S., Gontijo-Lopes, R., Salimans, T., Ho, J., Fleet, D. J., & Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22)* (pp. 36479–36494). Curran Associates Inc.
Seidel, S., Berente, N., & Gibbs, J. (2019). Designing with autonomous tools: Video games, procedural generation, and creativity. In *ICIS 2019 Proceedings*. AIS.

Shen, I.-C., & Chen, B.-Y. (2022). Clipgen: A deep generative model for clipart vectorization and synthesis. *IEEE Transactions on Visualization and Computer Graphics*, *28*(12), 4211–4224. https://doi.org/10.1109/TVCG.2021.3084944

Shen, Y., Zhang, C., Fu, H., Zhou, K., & Zheng, Y. (2021). Deepsketchhair: Deep sketch-based 3D hair modeling. *IEEE Transactions on Visualization and Computer Graphics*, *27*(7), 3250–3263. https://doi.org/10.1109/TVCG.2020.2968433

Silva Junior, S. N., Chamone, F. C., Ferreira, R. C., & Nascimento, E. R. (2018). A 3D modeling methodology based on a concavity-aware geometric test to create 3D textured coarse models from concept art and orthographic projections. *Computers and Graphics*, *76*, 73–83. https://doi.org/10.1016/j.cag.2018.09.002

Skrebels, J. (2023). *Starfield includes more handcrafted content than any Bethesda game, alongside its procedural galaxy*. https://www.ign.com/articles/starfield-todd-howard-on-the-1000-procedurally-generated-planets-and-how-you-can-ignore-them

Tan, Q., Gao, L., Lai, Y.-K., & Xia, S. (2018). Variational autoencoders for deforming 3D mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5841–5850). IEEE.

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 172–186. https://doi.org/10.1109/TCIAIG.2011.2148116

Unity Technologies. (2024). *Gaming report* (accessed 2024 September 11). https://unity.com/resources/gaming-report?isGated=false

Walton, S. P., Rahat, A. A., & Stovold, J. (2022). Evaluating mixed-initiative procedural level design tools using a triple-blind mixed-method user study. *IEEE Transactions on Games*, *14*(3), 413–422. https://doi.org/10.1109/TG.2021.3086215

Yan, W., Hu, B., Li Liu, Y., Li, C., & Song, C. (2024). Does usage scenario matter? Investigating user perceptions, attitude and support for policies towards ChatGPT. *Information Processing & Management*, *61*(6), 103867. https://doi.org/10.1016/j.ipm.2024.103867

About the authors

Kaisei Fukaya is an AI researcher, and a PhD graduate from Brunel University of London. His research interests include Applications of AI, Machine Learning, Tools for Game development, Procedural Asset Generation and Serious Games.

Damon Daylamani-Zad is a Senior Lecturer in AI and Games at Brunel University of London. He is a Fellow of the BCS. Damon's research interests focus on applications of Artificial Intelligence and Machine Learning, Collaborative Games, Serious Gaming, and application of Evolutionary algorithms in Creative Computing.

Harry Agius is a Senior Lecturer in Computing at Brunel University of London and a Fellow of the BCS. His research focuses on digital media and AI based creative computing, notably games and personalization. He is Deputy Editor-in-Chief of Springer's Multimedia Tools and Applications journal.