Context Aware Asset Placement Tool

GDEV60001 GAMES DEVELOPMENT PROJECT

Cayden Andrews
SUPERVISOR: PETER COOPER
SECOND SUPERVISOR: KIERAN OSBORNE

Contents

Abstract 3
Introduction 4
Aims and Objectives 5
Literature Review 6
Tooling 6
Human Computer Interaction 8
User Experience and User Interface 8
Procedural Content Generation 11
Noise Generation 13
Existing Tools 15
Scene Zoning 16
Summary 16
Research Methodologies 18
Results and Findings 22
Discussion and Analysis 28
Conclusion 33
Recommendations 35
References 36
Appendices 40
Appendix 1 – A 41
Appendix 2 – A 45
Appendix 2 – B 45
Appendix 2 – C 45
Appendix 2 – D 46
Appendix 2 – E 46
Appendix 2 – F 46
Appendix 2 – G 46
Appendix 2 – H 46
Appendix 2 – I 47
Appendix 2 – J 48
Appendix 2 – K 48
Appendix 2 - L 48

Abstract

Modern game releases are seeing an increase in scope and costs with an increase to the size of game worlds and the quality of their environments. Game tools are becoming increasingly important to ensure deadlines are met and production costs are lowered while ensuring a high-quality product.

This paper describes the key aspects involved within the creation of game development tools, including human computer interaction, user experience and user interfaces. Procedural content generation is also mentioned alongside existing tools as to determine whether the addition of contextual awareness to a tool through user defined zoning and subsequent procedural content generation would speed up the development of non-procedurally generated game scenes.

An object placement tool making use of user defined object palettes and zones has been created. The tool allows for the creation and saving of palettes for future use or modification, alongside the definition of zones within a scene using vertices to define the bounds. The user can then apply the palette to the zone to fill a region with objects according to the user specifications within the palette.

Unity has been used for the creation of the artefact as it is free and has a wide array of object placement tools available on the asset store for comparison with the developed tool.

The developed tool was compared against two other tools with different methods of object placement to determine the impacts of contextual awareness on a tool. All tools were tested using three different sized scenes and a specification on how they were to be filled. Participants were observed during the testing and had to fill out a questionnaire following the testing.

The addition of contextual awareness brought benefits on iteration to an existing scene, with a hindrance on the development speed of smaller game scenes. Users preferred the contextually aware tool for larger scenes and found it efficient once setup, believing it to be beneficial for the development of variations of a game scene.

Introduction

Game Development Tools consist of programs that aim to assist or simplify the completion of a specific task in developing a game. Tools development is a key part of the development pipeline, used to accelerate the creation and improve the quality of a product (Lightbown, 2021, 2015).

Tools are often game independent, used within subsequent projects and expanded upon to meet changing needs (Lightbown, 2021). Tools consist within the context of built-in, 1st party and 3rd party tools (Kovanen, 2018), where tools are made by an engine's creator, a game company or produced by independent developers.

As a result of the rising scopes and size of games, primarily in AAA productions, tools are having an increasing impact on games levels of polish, ability to meet deadlines and production costs (Thomas, 2025).

For example, The Witcher 3 (2015) had a total cost of $84 million (CD PROJEKT, 2015) whereas Cyberpunk 2077 (2020) cost $329 million as of (CD PROJEKT, 2021) financial report.

The primary focus of this paper is on the tools used within the development of game scenes / environments, specifically object placement tools due to the prevalence of non-procedural large game worlds in recent releases such as ARC Raiders (2025), Dune Awakening (2025) and Elden Ring (2022).

This study aims to determine whether the addition of contextual awareness to object placement tools will speed up the development of game scenes by investigating the time taken to detail multiple different sized scenes, recording time taken, usability and preferences from a variety of object placement tools.

To do this, existing asset brush tools and procedural tools have been identified, with research into game engine tools, procedural content generation, scene zoning and human computer interaction.

Various methods of user defined 'zoning' will be compared based on ease of use, using defined user experience needs outlined in the research, and the time taken to set up within a scene. Equally, using research into procedural generation, a method of 'zone' filling to enable the user to 'paint all' and regenerate specific 'zone' types on editing their used palettes, will be designed, taking in various user defined parameters.

Finally, a 'zoning' implementation will be selected, alongside a finalised method of 'zone' filling. These being used alongside any defined user experience needs, will then allow the assembly of a finalised context aware asset placement tool.

Users will then test the tool against other tools. Feedback from testing will be compared to identify any trends in preferred tools, an average of the time taken to complete each scene using each tool, and any other key information on useability. The artefact will then be critiqued based on this feedback to identify any strengths and weaknesses.

Lastly, a conclusion will be authored, stating whether the tool increased the speed of development for these scenes, and if the tool was easily understood and used.

Aims and Objectives

The key aim of the project is to determine whether the addition of context awareness to environmental tools in engine will speed up the development time of non-procedural game scenes. Based on this, the following objectives were identified:

- (O1) Define User Experience (UX) needs. This includes any User Interface (UI) accommodations such as feedback, hierarchy and simple naming schemes.
- (O2) Identify and compare methods of manual 'zoning'. That is, what functionality is given to the user to layout and edit regions within a game scene, and how does this compare to other methods.
- (O3) Research procedural content generation. This being what methods of procedural generation are appropriate for use in non-uniform 'zones' of a scene.
- (O4) Identify and compare methods of noise generation. Including their speed, complexity and how appropriate they are for the tool.
- (O5) Identify what Brush and Procedural tools currently exist. For instance, what functionality do they have and how do they work?
- (O6) Using the previous research, create a context aware asset placement tool.
- (O7) Get users to test the tool, comparing it against other tools. Use the feedback to identify any trends and other key information, to then critique the tool.

To ensure accurate user feedback and data for analysis, the testing will be done using a live viewing as well as a questionnaire given immediately after testing.

Literature Review

Tooling

(Cambridge University Press & Assessment, 2025b) define a tool as something that helps to do a particular activity. Within the context of Game Development, this can range from a program that converts a file between data types to a full terrain editor.

(Thorn, 2011) states that game development tools are a means to connect inactive game content, such as audio and graphics, to the game engine, allowing it to be acted upon. (Kovanen, 2018) builds upon this, stating that tools provide re-usable, customisable functionalities. With modern game engines being a collection of abstracted tools, used to implement specific game features (Kovanen, 2018).

Using these definitions, a tool within the context of game development can be defined as a program that provides re-usable functionality to help complete a particular task. An example may be a tool to import a 3D game model from a variety of file formats into a game engine for use within a game.

(Barczak & Woźniak, 2019) state that a game engine is built with components that provide the essential functionality to produce a game.

(Thorn, 2011) expands upon this, describing game engines as modular, where an engine is subdivided into various component such as a rendering component, where each component performs a unique task. None of the modules can be removed without taking away from the engine's effectiveness, and none can be replaced unless the replacement performs the same task (Thorn, 2011).

(Barczak & Woźniak, 2019) describes one of the components as being software tools, used to optimise and increase the efficiency of production.

Consequently, tools development is a key part of the game development pipeline, used to accelerate the creation process and ultimately improve the quality of the product (Lightbown, 2021, 2015). In a conversation, quoted at GDC2021, between (Lightbown, 2021) and John Romero – co-founder of id Software, John states "Tools live longer than games do". (Lightbown, 2021) expands upon this, stating that a tool made for a game is often used within subsequent projects.

(Lightbown, 2021) states that tools should be an accelerator, not a hindrance to project development, where developers should focus on these questions:

1. What is the problem the user wants to solve?
2. What can be learned from other tools that solve the same problem?
3. How do users use my tools, and how do those tools fit in with the user's other tools?

This importance is further emphasised when considering the iterative cycle of games development (Kultima, 2015; Eitan, 2012; Bernardi, 2025), where aspects of games are often tested and iterated upon continuously to produce a high-quality product.

It is worth noting that it is not on the user to tell the developer how the tool should work, they should define the problem they want to solve so the developer can then create an appropriate tool

(Lightbown, 2021). This is further expanded upon with reference to the User Centred Design process (Lightbown, 2021, 2015; Interaction Design Foundation, 2016b), this is discussed in Human Computer Interaction.

(Kovanen, 2018) states that there are three types of tools, these being:

- Built-in,
- 1st party
- 3rd party

Built-in tools are integrated into the engine, providing an easy-to-use base for the creation of a game or other tools.

1st party tools are made by the game creators. These are more specific in functionality, with a lower level of usability and polish as they are not made for public use. However, 1st party tools can become less extensible and maintainable if the creator leaves the team, this is due to the lesser documentation (Kovanen, 2018). Equally, the cost of development for these tools can lead to outsourcing for 3rd party tools.

3rd party tools are made by unrelated developers. These tools have varying quality and support, with some requiring purchased licenses to use (Kovanen, 2018). These tools sometimes have higher quality and customisability than 1st party tools, providing more creative freedom. The artefact created for the projects falls under the 3rd party tool category.

Tools are often developed to provide unique characteristics and feel to a game, providing a game studio with a unique style for their games (Kovanen, 2018; GDC Festival of Gaming, 2015).

(Kovanen, 2018) states that for a tool to be easily integrated into the development pipeline, it needs to have minimal setup and favour customisability, enhancing usability (Lightbown, 2015).

Tools within game engines should ideally avoid the prerequisite setup work to enable non-programmers to use the tool and avoid cognitive overload (Shaker et al., 2016). (Kovanen, 2018) suggests that tools should aim to work upon simply adding the tool to the engine. This often simply requires good encapsulation for the tool (Kovanen, 2018; Snyder, 1986).

(Cambridge University Press & Assessment, 2025a) define customizable as being able to change to suit a user's needs. Thus, for a tool to be customisable, the tool must have tweakable parameters for the user to alter the output. This is proportionate to the use cases of the tool and may equally enable the use of the tool in wider contexts, allowing for the creation of unique outcomes (Kovanen, 2018).

(Kovanen, 2018) cites the Unity Engine as a key example for the necessity of 3rd party tools, as many of the present built-in tool's fall far behind other engines such as Unreal Engine. A key example and reason for the development of the artefact and other similar tools is the Unity terrain tool. The tool has seen few updates since Unity 2.0 (Kovanen, 2018), allowing for the creation of hills, rocks, grass and trees. The key problem that the artefact and others within the Unity Asset Store attempt to solve is the limited customisability for the foliage and object placement, especially when creating large

urban environments or environments that contain a higher variety of foliage types and miscellaneous details.

Human Computer Interaction

(Interaction Design Foundation, 2016a; Carroll, 2014) Define Human Computer Interaction (HCI) as the interaction between users and the computer. HCI emerged in the early 1980s with the rise in home computers and the apparent lack of usability in these computers (Carroll, 2014).

This was soon alleviated with products such as the Apple Macintosh, where UNIX commands were swapped in favour of a drag and drop Graphical User Interface (GUI) (Carroll, 2014). (Interaction Design Foundation, 2016a) state that HCI should resemble and open-ended dialogue, this is expanded upon by (Cooper, 2004; Lightbown, 2015) within User Experience and User Interface (UX and UI). (Interaction Design Foundation, 2016a) state that most practical UX considerations come from the results of HCI research, with HCI being the academic, broader topic, with UX as a practical subset.

User Experience and User Interface

The International Organisation for Standardization (ISO) defines UX as: “a person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service” (ISO 9241-210, 2010). (Green & Labrecque, 2023) reiterate this, stating that UX refers to how people experience a product and form opinions on that product, both positive and negative. Thus, UX is focused on addressing user’s needs and grievances (Interaction Design Foundation, 2016b).

UX and the subsequent factors such as the need for a product, the identified users and why the product is needed, alongside UI design and accessibility began gaining focus in 2010 with the development of the iPhone, App Store and Android Operating system (OS) (Green & Labrecque, 2023).

(Lightbown, 2015) states that any problem with tools is often the UX and not the user themselves, and that the UX is the most vital aspect within a tool. This is due to the impact UX can have on the efficiency of tool use, resulting in significant savings of time (Frost, 2003) and money. It is stated that for a tool to have a good UX, the tool must be useful, usable and desirable. (Lightbown, 2015) compares this to Maslow’s hierarchy of human needs, where a tool must be useful before being usable, and usable before being desirable.

This is important for tooling where the UX is paramount to the tool’s success. While a tool may have a pleasant looking UI, if it is difficult to use and does not cater to the user’s needs, the UX will suffer, and the tool may be discarded or replaced (Lightbown, 2015).

(Lightbown, 2015) states that to make a tool useful, the needs of the end users and their goals must be identified, to provide a clear direction and set of tasks for the tool to accomplish. Reinforcing (Interaction Design Foundation, 2016b) focus on addressing user needs and affecting the UI design choices made, dependant on the tool’s goals.

Within usability, the two main measures are efficiency and learnability, where efficiency measures the speed for completion of a task and learnability measures the ease of use for a new or returning user (Lightbown, 2015). Within the artefact, the UI will be kept simplistic, with tooltips and simple names

for buttons and variables, to ensure the UI is easily understood by users with minimal Unity experience.

Learnability is important as documentation may be out of date and experts are not always available. A good measure of learnability is whether a user can complete a task on the first attempt.

(Cooper, 2004) suggests that software-based products should be more human during interaction. This is expanded upon by (Lightbown, 2015), suggesting that a good user experience resembles an interaction with another human and not a machine. This may be provided by using simple language, previews to show the outputs before confirmation and attempts to choose the best option for a task automatically, potentially resulting in better learnability.

Such interaction can be seen within (Gen90Software, 2023)'s tool. The tool makes use of a preview before allowing the user to place objects within a Unity Scene. Allowing the user to see the result and easily visualise the alterations different settings may have on the output.

Finally, a tool can be made desirable with an appealing design. This can present the tool as a high quality and professional product, contributing to user satisfaction. This may also give the user more confidence in the tool's abilities, because of the appearance of a finalised product.

(Lightbown, 2015) suggests that early feedback is vital to improving tool UX (Frost, 2003), (Eitan, 2012; Bernardi, 2025) expand upon this, citing the importance of an iterative design and testing process to ensure a high-quality product. Using pre-visualizations is one suggested way of doing this, alongside using analytics to identify key areas for improvement. (Green & Labrecque, 2023) build upon this, stating that communication between the different teams and research can prevent any miscommunications in the project direction and choices made. As a result, research and feedback can both steer the product in the right direction, while good communication can prevent any wrong changes in a projects direction.

Such a method is proposed by the User Centred Design Process, being a primary method of designing tools for human use by involving humans in the design process, either through feedback during development or placing user's needs at the centre of any decisions (Williams, 2009).

This involves many iterations where many small improvements are made and evaluated with the user. This lowers the potential for large time-consuming changes that are not in line with the user's goals. (Interaction Design Foundation, 2016b) expand upon this, stating that UX is a multidisciplinary field, encompassing visual design, psychology and interaction design. Where designing for user needs includes accommodations for various disabilities, thus the main approach to UX design is User Centred. (Green & Labrecque, 2023) support this, stating that a user centric approach and research into other similar solutions can both expose any designer or developer bias and workflow weaknesses, allowing the team to adjust early to better suit the user's needs. (Lightbown, 2015) simplifies the process into analysis, design and evaluation.

Analysis involves identifying problems, this can be done through metrics and live viewing. Metrics should be complimented by live viewings as what user do is often different from what they say they

do (Lightbown, 2015). Live viewings can reveal why users do certain things alongside any inefficiencies in the tool.

(Lightbown, 2015) Defines Design as the use of the analysis to decide what to improve. Constraints, natural mapping, hierarchy and feedback are four suggested ways of doing this. (Lightbown, 2015) States that constraints are limits on the user to prevent mistakes or errors, this is enhanced by feedback, where pop-ups, wait cursors and progress bars can be used to convey relevant info to the user and prevent any unwanted actions. (Lightbown, 2015) Defines natural mapping as the placement of UI elements to match the actions performed, this added context can increase a tools learnability. This is further supported by hierarchy, where thickness, size, contrast and position are used to highlight key info to the user. (Shaker et al., 2016) expands upon this, stating that users can experience cognitive overload if they are required to performs a large amount of content selection, feedback from the system is slow, they are presented with a large amount of content/information on screen or when the user is required to provide a very specific input.

Evaluation is stated as testing whether the changes made are an improvement. (Lightbown, 2015) Suggests the use of Pre-Visualization (PV) or code to determine the effectiveness of the changes made. (Lightbown, 2015) recommends the use of PV when the time taken to implement the changes in code, outweigh the time taken to create a PV. PVs such as sketches can reduce the space for misunderstanding and interpretation (Lightbown, 2015). If a test is for efficiency however, it is faster to go to code.

(Williams, 2009) proposes other design methods, including Activity Centred Design and Goal Directed Design. Activity centred Design equally involves HCI. However, with a shift from what tasks must be performed to what tasks must be enabled by the tool/system. In doing so, users are seen as participants in activities (Williams, 2009). However, this system is largely theoretical (Williams, 2009), without a solid process when applied to designing applications.

Goal Directed Design is equally not too different from User Centred Design (Williams, 2009), including Research, Analysis, identifying user needs and business requirements, Design, Refinement and Support. The primary difference being where User Centred Design focuses on the user and considers their goals and tasks, Goal Directed Design focuses on the goal of the user, with a lesser focus on UX.

Thus, the User Centred Design Process will be used for the artefact, due to its similarity to other methods, with a clear set of processes and its focus on the user, with consideration to the goals and tasks, rather than just the goals or the activity. (Williams, 2009) reinforces this, stating that User Centred Design inherently considers the activity and the goal as these cannot be separated from the user when developing a tool.

(Green & Labrecque, 2023) refer to the Minimum Viable Product (MVP). The MVP is a focused feature set that is highly useful for the user. Using the MVP as a target can prevent feature creep and ensure all contributors are on the same page. To determine the MVP, (Green & Labrecque, 2023) suggest identifying what is important for the user and what is important for the team.

Procedural Content Generation

(Green, 2016; Eliasz, 2024) define Procedural Content Generation (PCG) as the process of something being created by an algorithm or procedure, as opposed to already existing. This is supported by (Watkins, 2016), who states PCG is the concept by which content can be generated by code. This often involves the combination of manually created assets and algorithms using random numbers (Eliasz, 2024).

A PCG method is simply a method that will output something. With a PCG system being a system that incorporates a PCG method(s) as one of its parts, such as an AI-assisted game tool (Shaker et al., 2016).

It is worth noting that PCG itself has no randomness; identical inputs will result in identical outputs. This is because true randomness is impossible (Watkins, 2016). (Green, 2016) expands upon this, stating that computers are deterministic, being nothing more than highly sophisticated calculators. Randomness must be implemented through random inputs to produce a range of outputs. This is done through pseudorandom numbers (PRNs). Thus, most references to procedural generation refer to procedural generation utilizing randomness (Green, 2016).

(Green, 2016) states that methods of generating PRNs vary, within games and other simple applications, simple algorithms are used for the sake of speed. However, within cryptography, more complex algorithms are used to prevent estimation from previous outputs. This can be done due to the nature of PRNs and seeding. PRNs are cyclical (Watkins, 2016), eventually they will repeat. Thus, cryptography and other sensitive subjects make use of more advanced techniques, this is unneeded for games and tooling, however.

PRN generators take in a seed, this represents the starting position along the PRN sequence (Pitt, 2023). This is why using the same seed will result in the same output. (Shaker et al., 2016) refers to this as Deterministic PCG, where the same content may be produced given the same inputs.

PRNs are used, both due to speed and the ability to find a previous result through seeds (Watkins, 2016), either designated by the user, system PRN generation implementation.

The popular game Minecraft makes use of PCG for its world generation, taking either a user specified seed or using details like the date, time and the user's hardware and configuration (Pitt, 2023).

PCG can provide benefits to games and tools alike, these include:

- Increased replayability,
- Lowered budget, resulting in more risk taking, leading to wider game diversity (Shaker et al., 2016),
- Larger games,
    - As seen in 1980s Rogue - *Michael Toy* and *Glenn Wichman*
- Reduced file size (Eliasz, 2024),
- Faster production,

(Green, 2016)

The key driver behind PCG advancement was the storage limitation of computers. This inability to store large amounts of game content led to games such as Elite, storing a seed number to use for galaxy generation, where each galaxy has 256 planets with unique properties, or Rogue, using PCG to generate levels on starting the game (Shaker et al., 2016).

However, PCG can have notable drawbacks such as:

- Taxing on hardware (for live generation),
- Repetition,
- Lower quality control,

(Green, 2016)

The issue of repetition can be mitigated, (Watkins, 2016) suggest increasing the size and range of the PRN sequence, alongside increasing the possible values for the seed.

These issues will be mitigated in the artefact. Firstly, generation is done in development on placing objects, not on game loading and quality control will be mitigated by providing a large array of parameters for the user to tweak to produce an ideal environment. Should Noise generation be incorporated into the artefact, repetition will be mitigated with a large seed number that can be configured by the user.

(Shaker et al., 2016) lists some common desirable properties of PCG algorithms:

- Speed: this can vary between runtime PCG and PCG done during development of a project.
- Reliability: this is dependent on criteria, constraints and the content produced (Shaker et al., 2016), such as tree generation or dungeon generation. In the case of dungeon generation, produced dungeons require an accessible entrance and exit to be valid.
- Control: this is dependent on the content produced but allows a user to specify some aspect of the generated content such as room count for a level generator.
- Diversity: this is the generation of unique content without looking like minor variations upon previous outputs.
- Believability: content produced should not look like it has been produced by an algorithm.

Some notable PCG applications include Blizzards Diablo level layout, Gearbox Software's Borderlands Unique Weapons / Items (Watkins, 2016) and Diablo 2's mix of procedural and handcrafted locations, where the main acts are procedural while the safe area between acts is hand crafted (Pitt, 2023).

A notable PCG environment tool is Peril (James et al., 2025), a level creation framework using a procedural assistance system to automate aspects of the level, including walls and ceiling. On runtime, Peril uses the placed floor agents to then generate walls and ceilings (James et al., 2025). Allowing for rapid iteration and prototyping.

(Doran & Parberry, 2010) suggests that PCG tools for game development should generate content based on a set of designer-centric parameters. With the user not needing to understand the

underlying workings to use the tool effectively. Procedural tools should be intuitive, requiring a low degree of human input while allowing for a high degree of control (Doran & Parberry, 2010).

(Shaker et al., 2016) proposes that the development of PCG software for game development could help in understanding the process used to "manually" generate content, outlining the constraints and allowances of the design problem to be solved. (Shaker et al., 2016) builds upon this, stating that PCG is iterative, where the increased understanding of the design process can lead to better PCG algorithms, further increasing design process understanding.

Noise Generation

Noise Generation is often used in tandem with PCG to produce terrain, this is done through the production of an intensity map (Shaker et al., 2016). This is represented as a two-dimensional matrix, storing the intensity and an x and y position on the map. For terrain, the value can correspond to the height of the terrain at that position (Shaker et al., 2016), resulting in a height map.

Heightmaps are often interpolated to smooth out the noise to produce more believable terrain. (Shaker et al., 2016) states that methods such as bilinear and bicubic interpolation can look very artificial, suggesting the use of fractals to mitigate this.

(Shaker et al., 2016) proposes a simpler method to fractal mathematics, to produce a similar result. This is done through running the simple terrain generation methods multiple times at different scales, adding them together with a magnitude corresponding to the scale.

(Shaker et al., 2016) states that this can be done by scaling the different generated terrains by the inverse of the frequency, should the function for generation be noise(f), the generation can be defined as:

$$noise(f) + \frac{1}{2}noise(2f) + \frac{1}{4}noise(4f) + \cdots \quad (1)$$

(Shaker et al., 2016)

(Shaker et al., 2016) refers to most fractal terrain generation methods as implementations of the concept of fractional Brownian motion (fBm). (Shaker et al., 2016) simplifies fBm as the concept of terrain generation through starting at a random point and taking a random walk, following a set of properties. The result is often approximated due to the computational cost of taking millions of these walks (Shaker et al., 2016). This cost is important when considering the frequency of use and the use case itself within the artefact, being used not for terrain, but for determining where objects are likely to be placed.

One method used in games due to its efficiency is the diamond-square algorithm (Shaker et al., 2016). Simply put, four corners of a heightmap are set to seed values, the centre of the map is set to the average of the corers, plus a random value. Then the four midpoints of each side of the square, at the far sides, are set to the average of the 2 neighbouring vertices and the centre value of the square, plus a random value. The result is that the original map has been divided into four sections. The previous steps are repeated using a lower magnitude for the random values, this is repeated n times to produce an approximation of fBm generated terrain (Shaker et al., 2016).

Another possible method is Perlin Noise. (Unity Technologies, n.d.) already includes a Mathf function for Perlin Noise, taking in the X and Y position of the desired sample point. Perlin Noise involves creating a grid of random unit-length vectors (Hyttinen, 2017; Wu & Liu, 2024; Perlin, 1985). The four closest vectors to a sample point are used, for each corner, the dot product of the unit vector and the vector from the corner to the sample point are calculated (Hyttinen, 2017; Wu & Liu, 2024), giving a value per corner. The values are clamped from -1 to 1. Then a weighted average is calculated from this, with values that are closer to 0 and 1 being moved closer, resulting in an 'ease curve' (Wu & Liu, 2024; Perlin, 1985). Perlin noise was improved upon in 2002, using a fixed array of vectors for the grid instead of random vectors, alongside a different equation for interpolation (Wu & Liu, 2024; Perlin, 2002).

$$\text{1985 ease curve: } s(t) = 3t^2 - 2t^3 \quad (2)$$

(Wu & Liu, 2024)

$$\text{2002 ease curve: } s(t) = 6t^5 - 15t^4 + 10t^3 \quad (3)$$

(Wu & Liu, 2024)

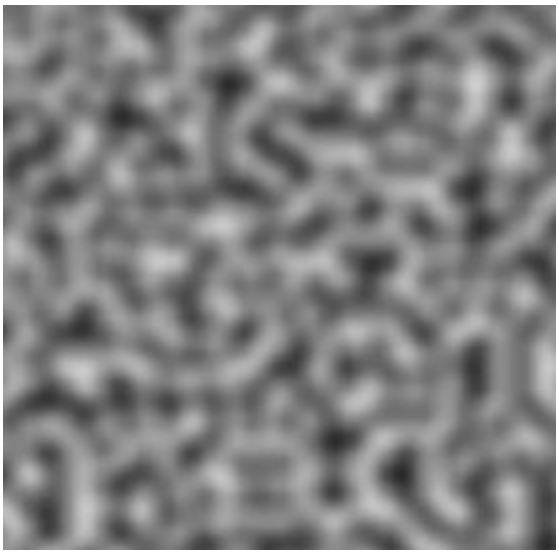

Figure 1. Perlin Noise (Hyttinen, 2017)

Figure 2. Perlin Noise (Unity Technologies, n.d.)

As seen in Figure 1 and Figure 2, Perlin Noise produces large regions of high and low intensity, simulating areas of higher and lower object density, providing more controlled variation for object placement. To include this within the artefact, a grouping slider can be applied to the tool, modifying the sample position scales to produce less or more noise. However, this function cannot be manually reseeded, meaning a different origin position must be chosen for addition to the X and Y coordinates entered into the function. This can be determined by the seed.

An advancement to Perlin Noise, being Simplex Noise exists (Gustavson, 2015) as seen in Figure 3. The key change within simplex noise is the use of simplices (Wu & Liu, 2024), using the vertices as the gradient points, these being a shape with N dimensions + 1 vertices (Gustavson, 2015). For a 2D space, this shape is an equilateral triangle, compared to Perlin's use of a simple lattice (Wu & Liu, 2024). (Gustavson, 2015) states that this makes it easier to interpolate values in the interior of the space when compared to base Perlin, though mostly beneficial at higher dimensions.

Simplex noise addresses some of Perlin Noises shortcomings (Gustavson, 2015). However, these shortcomings are only issues in higher dimensions (Hyttinen, 2017; Perlin, 2001). Equally, the projects focus is not on noise generation. Thus, Simplex Noise will not be implemented into the artefact if there is time available for implementing noise generation.

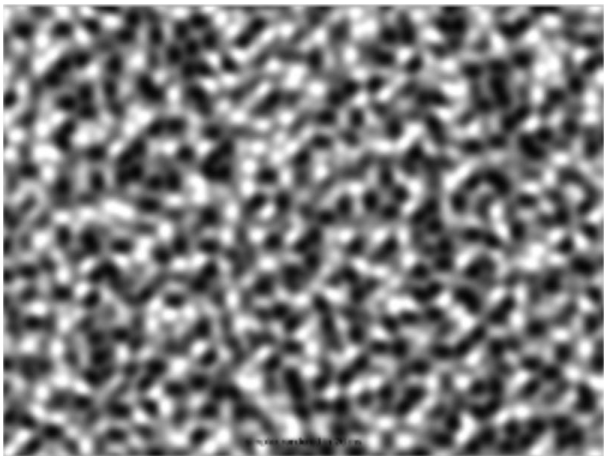

Figure 3. Simplex Noise (Wu & Liu, 2024)

As stated by (Shaker et al., 2016; Wu & Liu, 2024), generating multiple noise samples, each with increasing frequency and decreasing magnitude and layering them can generate fractal noise. This is often used in terrain generation, with each layer being used for different features such as hills or rocks (Wu & Liu, 2024). However, this would be unnecessary for the artefact as the tool is focused on details such as foliage, corresponding to only one layer within the fractal noise.

Other noise generation algorithms, such as Wavelet Noise, seek to address various Perlin Noise limitations such as trade-offs between loss of detail and aliasing (Hyttinen, 2017). However, this is not a problem for the artefact as the produced noise is not used for a texture and is only used to provide variance in object positions.

Such generation techniques could be incorporated into the object placement functionality of the artefact to produce smoother zone density changes. However, it must be noted that this is not necessary to solve the proposed research question and so is not a mandatory addition to the artefact. However, this may be implemented if time is available.

Existing Tools

There are many existing tools that solve similar problems or solve problems within the same space as the project artefact. A key tool being that made by (Gen90Software, 2023), an asset placement tool that allows the user to zone out an area using drag and drop points and seed-based randomisation.

The artefact will make use of a save and load system that will save object groups and their data to palette objects for assignment to zones and a similar zoning method will be implemented and extended to allow for multiple, permanent zones to be setup for use by the tool, these zones will also hold the instantiated objects from the associated palette. Should noise generation be incorporated, the seed-based randomisation will be used to provide retraceable results for future use.

Similar, non-zone-based tools have been made such as the object placement tools by (Biostart, 2024; NOT_Lonely, 2023; Funkybyte, 2024), these make use of: sequential and random placement, altering layers to paint on, directional placement for fences and editing prefab rotation and size.

While useful for the user, many of the features within these tools do not benefit the generation of environmental objects for a large scene or aid in answering the object question and so are not mandatory for inclusion or modification within the project artefact.

Scene Zoning

Unreal Engine 5 (UE5) makes use of a texture for biomes, outlining the positions in an identical manner to heightmaps, with the difference being that the biome map is in colour. Such a method could be used within the artefact, either removing the need for collision checks with the zones in the scene, instead only requiring a position check relative to the back corner of the scene, allowing for easy comparison with the biome map, or allowing for the creation of smaller colliders in the scene per zone, based on a grid system as seen in UE5, removing the comparison and just relying on collisions.

The method of the zoning itself will likely remain a drag and drop system for ease of use and intuitiveness, such as in (Gen90Software, 2023)'s tool, allowing designers to zone out the map far faster and easier than creating and exporting images from an image editor.

However, (Gen90Software, 2023)'s tool has a few limitations that need to be addressed. Firstly, the grid placement is limited to four vertices, this can be solved within the artefact with a circular list, where the last vertex links back to the first placed vertex, allowing for the user to create more complex zone shapes. Equally, using a list format would allow users to add extra vertices after initial zone creation, providing greater usability. Secondly, objects placed using the tool are sunken into the terrain, this is because the tool only accounts for prefabs with pivots at the bottom of the object. The artefact will solve this by moving the placed objects by half its size on the Y axis along the surface normal. On top of this, a Y offset will be available, allowing per-object alterations on the Y axis, allowing the user to move objects into or out of terrain.

Lastly, it is worth mentioning that (Gen90Software, 2023)'s tool does not make use of an in-engine tutorial to guide the user through the basics, using a PDF detailing each UI elements functionality. While this is useful as a reminder for a user coming back to a tool after a long period of not using it, this can affect the tools learnability, as documentation may be out of date or not cover the full range of features (Lightbown, 2015).

The artefact will make use of popup windows to guide the user through each element of the tool in a sequential manner to ensure the user is aware of each UI elements functionality without needing to leave the engine.

Summary

Tooling within games development seeks to accelerate the development process of games while retaining a high-quality product (Lightbown, 2015, 2021; Thomas, 2025). Tools must be easy to use, with a focus on the user through all aspects of development, including the UX, UI, its usefulness, usability and desirability.

A tool should be intuitive to use, resembling human interaction in order to accelerate the development process while also remaining focused on the goal(s) of the user, as to not cause cognitive overload (Shaker et al., 2016).

Procedural Content Generation has reduced the development cost and time for many games, either as a pre-release tool or through live generation during gameplay (Shaker et al., 2016; Green, 2016). However, within the research, PCG has primarily been applied to game scenes as a whole or smaller aspects of a game such as weapon generation (Pitt, 2023; Green, 2016; Shaker et al., 2016; Watkins, 2016).

Some existing tools contain aspects of this, with the implementation of single region painting and random object placement within a region influenced by user defined parameters (Doran & Parberry, 2010; Gen90Software, 2023). However, these tools lack the functionality to define multiple zones across larger game scenes or define parameters that would affect all game scenes as PCG does.

The aim of this study is to determine whether the addition of contextual awareness to an object placement tool, through user defined regions and object palettes resulting in procedural generation of content based on user defined parameters, will accelerate the development process of non-procedural game scenes.

A tool will be produced, allowing users to define multiple regions within a game scene, with the ability to create and assign object palettes to regions, making use of user defined parameters. This tool will then be tested against other available tools to determine if the addition of contextual awareness made an improvement.

Research Methodologies

To determine if the addition of context awareness to environmental tools will speed up the development time of non-procedural game scenes, secondary research was carried out. This research focused on Game Engine Tools, Procedural Generation, HCI and existing tools similar to the artefact. This is in line with Design Science research, where this information was used to inform development decisions for the artefact, and once the artefact was developed, testing was carried out to determine the effectiveness of the tool and any shortcomings.

Secondary research was gathered from sites such as Google Scholar and Oreilly, ensuring the research used has been through a review process. The documents gathered were chosen based on the relevancy of their abstracts to the key topics chosen to both answer the research question and develop the artefact.

Primary research has been conducted following the development of the tool to answer the study question. This data allowed for the comparison of tools with different features that are used to complete the same goal, and the identification of any benefits and drawbacks to different functionality dependant on scene size.

A mixed-method approach was taken to data gathering during testing, this is due to the various aspects of the tool, including analysis of the time taken to complete a task, and the quality of the UX.

Both questionnaires and live viewings have been used to test the artefact against other tools. Each tool was contained within its own project with multiple, different sized scenes. A level composition specification to create within the scenes was given to the users. For each tool, the users were timed for each scene individually. Following completion, the users filled out a questionnaire.

Questionnaires were used to gather data that is required for every tester, and to prevent any loss of information that could be caused by a delay in recording the information. Questionnaires are an effective data collection method for opinions and preferences (George, 2023) allowing for data collection on the UX as well as the time taken. Equally, the Quantitative data can easily be averaged and compared to come to a conclusion.

Likert Scales have been used to analyse subjective data such as user opinions on the best feeling tool for scene sizes. Equally, these scales have been used alongside non-leading questions to ensure the data is accurate and could then be compared and averaged out to find general trends in opinion. (Joshi et al., 2015) define the Likert scale as a set of statements offered for a situation, where participants are asked to show their level of agreement with each individual statement on a metric scale. Likert scales offer both a symmetric and an asymmetric scale, with a symmetric Likert scale offering the ability for a participant to remain neutral / indifferent. Whereas an asymmetric scale offers less or no choices towards neutrality, resulting in a forced choice of slight agreement or disagreement (Joshi et al., 2015). Symmetric scales have been used to prevent influencing the participants responses.

Within these scales, seven points were used to provide a balance between allowing more precise responses while not overloading the participant with choice (Joshi et al., 2015).

Live viewings were also used to allow for identification of any inefficiencies with the tool, alongside displaying how participants used the tool, providing valuable information on top of the questionnaire, as often, participants can forget or omit information, or state something that is different from what they actually did (Lightbown, 2015).

The testing was a cross-sectional study, where data was collected from many individuals within a small timeframe, for a variety of tools. Two non-contextually aware tools were selected for comparison with the artefact to answer the study question. (Thomas, 2022b) states that this method involves observing variables without influencing them.

It is worth noting that while this study method is less time-consuming and allow for a large participant pool (Thomas, 2022b), it can be difficult to analyse cause-and-effect such as why a tool is better in a specific size scene compared to another. The inclusion of questionnaires with relevant questions aimed to negate this issue.

Equally, Questionnaires were used over interviews due to the similarities in the questions that would be posed, with Questionnaires providing time for the participant to answer in a less stressful environment compared to an interview.

Whereas interviews can be time-consuming and susceptible to bias from the participants (Doody & Noonan, 2013). Participants may seek to please the researcher with their answers or try to answer a question rather than remain silent if they have nothing to say on the topic. Equally, participant responses may be influenced from the researcher expressing surprise or disapproval (Doody & Noonan, 2013).

Equally, the use of a questionnaire fits with the questions posed using Likert scales, alongside the submission of a unique word, allowing users results to be removed from the study if requested.

Longitudinal and cohort studies were not done as they both use data gathered over time, making them more useful for the development process (Thomas, 2022b; George, 2023). As the study conducted required all participants to use every tool and is comparing the data gathered from each tool, these methods were not applicable.

To prevent any inaccuracies in data, confounding variables and observer bias were considered. Confounding variables were considered and negated through restriction (Thomas, 2022a), where participants were limited to individuals with a background in computer games related fields, ensuring that participants reflect the target users in a real-world scenario.

Observer bias has been considered are negated through the use of multiple data collection methods and a standardised procedure (Bhandari, 2022), this being what data is recorded during the observation. The data recorded was limited to moments where the user used workarounds to achieve a goal via unintended or inefficient means and where users got stuck, alongside comments on the tools. Observer expectancy is equally negated through limiting the communication between the researcher and participant, limited to only an explanation of the tasks that must be observed and answering any questions from the participant before or after the data collection. Equally, the

Hawthorne effect (Bhandari, 2022; Mahtani & Spencer, 2017) was considered, with rapport being built with participants to ensure they complete their tasks at a natural pace.

The Hawthorne effect is the alteration of a participant's behaviour when they are aware they are being observed (Mahtani & Spencer, 2017).

The main metric for the project is time taken to successfully paint a scene; this is recorded individually for scenes of varying size, for each tool. The lower the time taken to decorate a game scene, the more efficient the tool and the greater the time that can be spent working on other aspects of the game. The User Experience could be considered another metric, although this can be subjective, with more Qualitative data, making comparison harder. This includes details such as whether the button or field names made sense, whether the tool UI layout felt good to use, and how quickly it took the user to figure out how the tool functioned.

The sampling method used was non-probability (convenience) sampling, using randomly selected individuals from the University of Staffordshire Games courses. The only criteria being that they are an individual on a game related course, with no upper limit to the sample size. This was done to both provide the largest possible data set, nullifying any potential outliers in data, and to provide data for users with a range of disciplinary focus. Equally, convenience sampling was used over random sampling as it ensures participants have at minimum, a small amount of experience with game engine software.

The artefact was created and tested using Unity 6000.1.17f1 and the Visual Studio 2022 IDE. However, the tool uses no Unity 6 specific code and thus should be compatible with any Unity version containing the UI Toolkit, and an IDE capable of editing C# code.

To ensure accurate data on testing, the participants used the same Unity version and projects and were given the same questionnaire to fill out. Between tests, the project changes were discarded using the GitHub discard changes feature to remove any persistent or leftover data between participants. Equally, the timing data analysed was averaged out and then used to calculate percentage differences between the tools within the specific focus e.g. time taken to paint the small scene, ensuring any variation from hardware differences was accounted for.

To protect the rights and confidentiality of the participants, all data gathered contained no identifiable information and contained a memorable word, ensuring participant data could be removed from the study if requested. Equally, all data was stored on the University OneDrive.

Following data collection, the timings were averaged out per scene for each individual tool, allowing for identification of the fastest tool for each scene size. This data was used in conjunction with any qualitative data from the questionnaires such as preferred tools and any grievances with tools to determine whether the addition of contextual awareness to environmental tools in engine sped up the development time of non-procedural scenes environments.

Equally, most of the data regarding UX and tool preference used selection choices or Likert scales, ensuring easier comparison. As for unique data such as user grievances, this data was compiled to identify any key trends or common statements using thematic analysis.

Potential limitations to the research included not having a large enough sample size and time constraints. Participation was offered to society members and other level six students, with the option to be a participant also available to level five and four students. Time issues were resolved by ensuring that the minimum viable product was ready before the second half of the study duration, ensuring a large timeframe to conduct testing and analysis.

Results and Findings

The aims of the research were to determine whether the addition of contextual awareness to an object placement tool would speed up the development of non-procedural game scenes. For this reason, data was gathered to outline the average time taken for each tool to paint scenes of increasing size alongside repainting an existing region within a scene.

Additionally, data regarding the users rating of the tools based on usability, preference for scene sizes, UI and control scheme were gathered. These were used to supplement any conclusions drawn from the data, alongside identifying any issues or future changes that could be made to produce a higher quality tool to prove the hypothesis that the addition of contextual awareness will speed up the development of non-procedural game scenes.

This chapter sets out the results of the questionnaire and live viewings, separated into sections corresponding with the questionnaire in Appendix 1 – A, alongside supplementary data gathered from the live viewings, as seen in Appendix 2 – K.

Firstly, all respondents were individuals enrolled in games related courses, with 50% not having used object placement tools before, as seen in Appendix 2 – A.

Questions 4 through to 18 were answered during the testing, recording the times taken to fill out each scene using the current tool. This data was gathered to determine the efficiency of the tools within different sized scenes.

As shown in Figure 4 below, Tool 1 as a standard brush tool took an average of 3.56 and 3.65 minutes to paint the small and medium scenes respectively. Tool 1 saw a large increase for painting the larger scene, taking 6.29 minutes while taking 2.81 minutes to remove and repaint an already existing region within the large scene. Tool 1 is the fastest tool on average when used for smaller scenes, taking the longest when used to repaint existing regions.

Tool 2, making use of limited zoning capabilities, takes a slower time frame to tool 1, 4.22 minutes for the small scene and 4.51 minutes for the medium scene. Tool 2 takes 6.75 minutes on average when used to fill the large scene, being the slowest tool on average. When used to repaint an existing region, tool 2 takes a moderate amount of time, at an average of 1.66 minutes.

The context aware tool took an average of 6.91 minutes for the small scene. The time taken proceeds to lower for the medium and large scenes, taking 4.88 and 6.31 minutes respectively.

When used to remove and repaint an existing region, the context aware tool takes an average of 0.48 minutes.

The context aware tool takes the longest on small and medium scenes, while taking the same time as tool 1 on large scenes and taking the least time out of the tools when modifying existing regions.

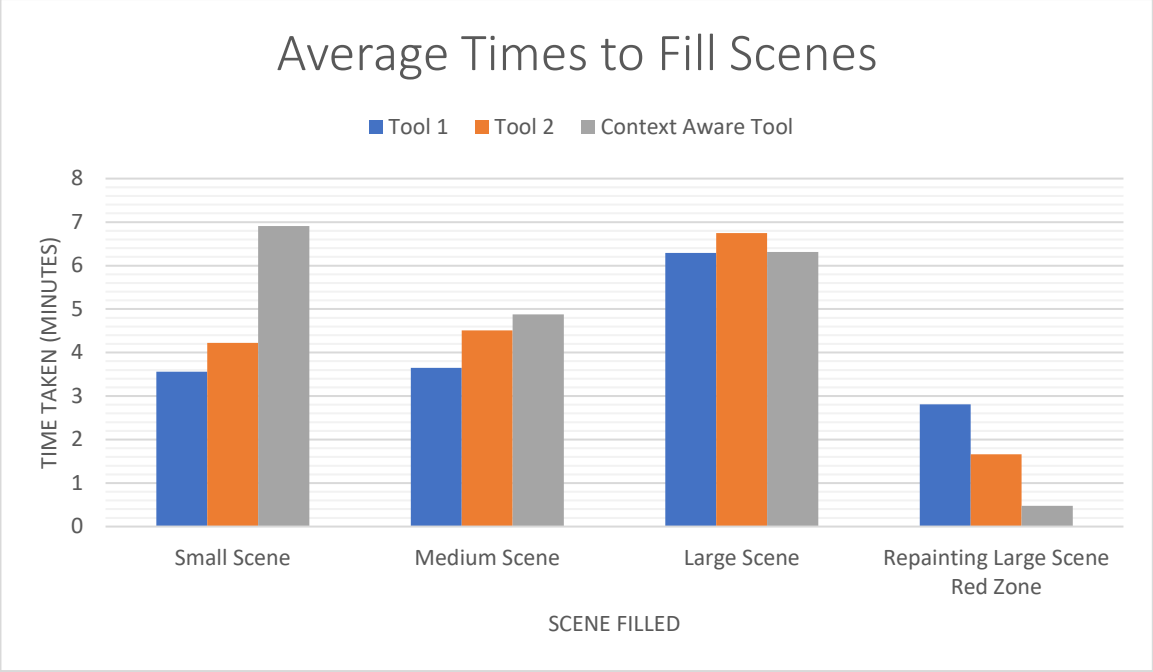

Figure 4. Average times to fill in scenes

It can be seen that the times taken for the small scene are higher than, or equal to, the times for the medium scene.

Some of the timing data was excluded from the average as some users did not adhere to the palette specification for tools 1 and 2 when painting the scenes. Equally, some individuals remade the entire red zone within the final tool, thus were also not included within the average. However, this data was important in analysing the tools usability. See Appendix 2 - L.

Following this, data was gathered using a Likert scale, where users were asked to rate each tools usability, being how easy the tool was to use and understand. As the data in Figure 5 demonstrates, 83% of participants reviewed the context aware tools usability as very good and above, with the final 16% rating the usability as good. Tool 1 was stated to be good by 33% of participants, with another 50% rating the tool as very good and above, with the final 16% rating it as fair.

Tool 2 received a usability rating of poor from 50% of the participants, with 33% rating it as fair and 16% rating it as very good.

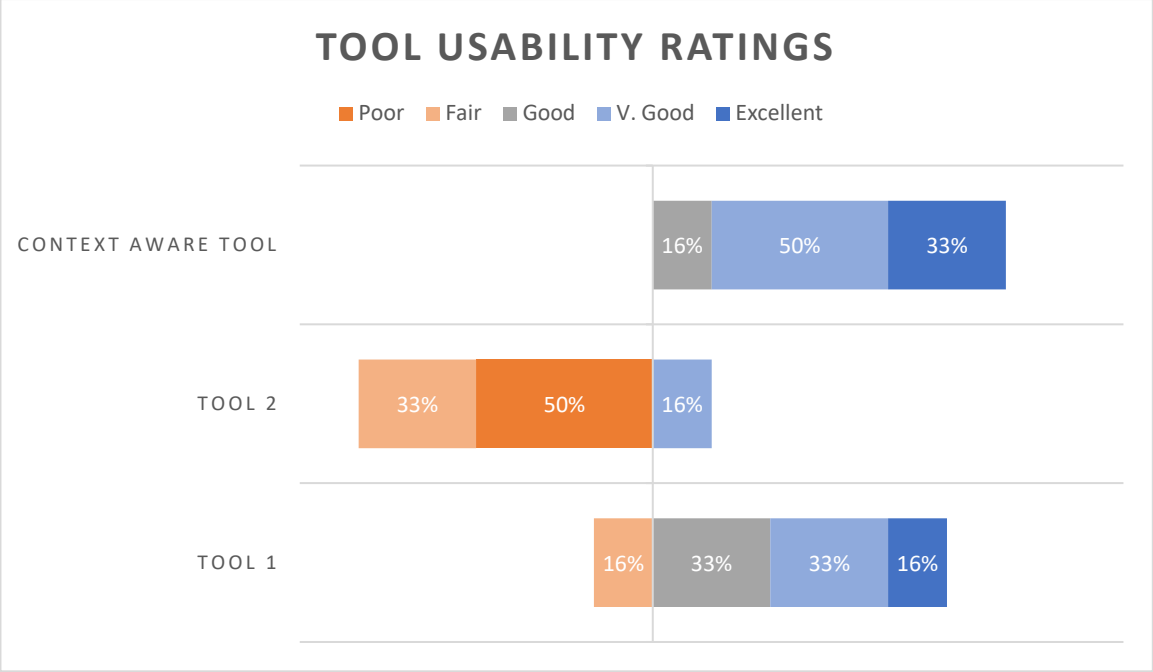

Figure 5. Participants Tool Usability Ratings

The next question asked participants what could be improved for the context aware tool to make it easier to use. There were three responses as seen in Appendix 2 – D, one participant suggested that the creation of parent and sub-zones be done automatically, both on creation and addition to the UI as the sub-zones did not go into the UI all the time. Another participant stated that there were performance problems on larger scenes with more objects.

Finally, the addition of tooltips were suggested to ease the user into the tool's workflow.

Participants were then asked to state what they disliked about the context aware tool as seen in Appendix 2 – E. One participant expressed that they felt the setup of the tool took too long, another participant expanded upon this, stating the workflow could be difficult to setup. Another participant reiterated the comment on sub-zone binding from the previous question, stating that this cost time.

However, one participant stated that they had no dislikes and had skipped the tutorial, finding the tool intuitive.

Following on from this, participants were asked to state what they liked about the context aware tool. The majority of the participants expressed an appreciation for the zoning through plotting points, stating that it was easy to use and allowed for the creation of more complex regions that could be left once created. One participant stated that the tool had a high learning curve but a large payoff once setup. See Appendix 2 – F.

The next question asked participants if they preferred another tool overall, and if so, what did that tool have that the other tools did not. This question received one response, with a participant preferring the prefab loading from tool 1. See Appendix 2 – G.

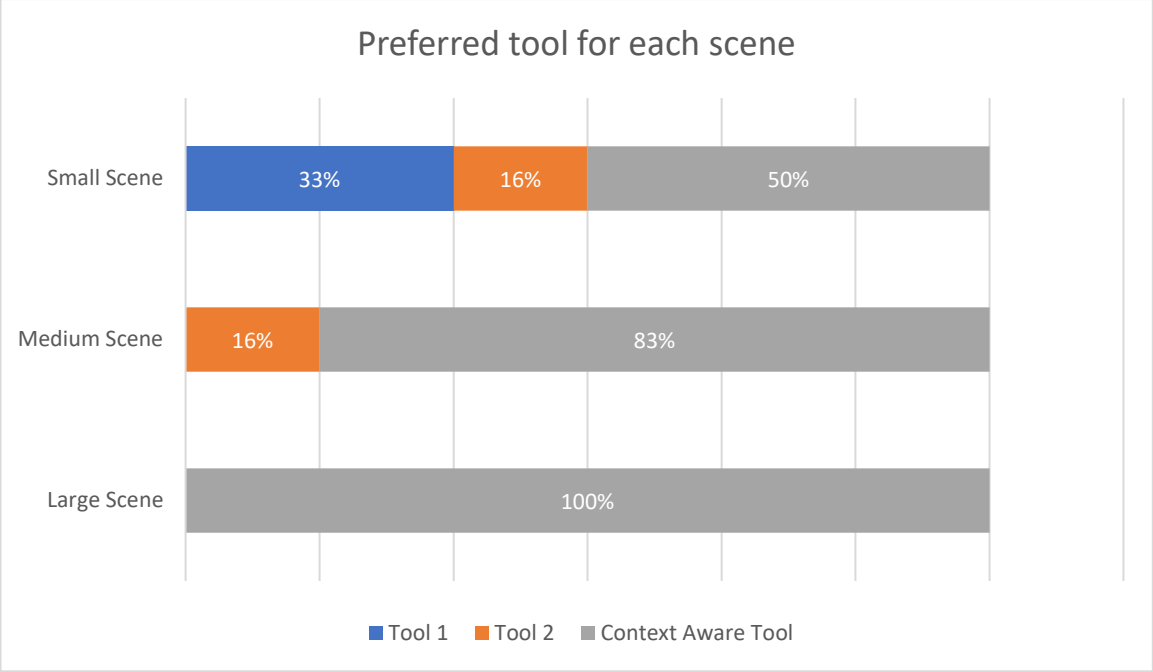

Figure 6. Participants tool preference for each scene size

As a continuation from the last question, participants were asked which tool they preferred for individual scenes. As shown in Figure 6, most of the participants preferred to use the context aware tool for all three scenes, having 50% prefer it for the small scene, 83% for the medium scene and 100% for the large scene. However, 50% of participants preferred the other tools for the small scene, with 33% of participants preferring tool 1 and 16% preferring tool 2. This extends to the medium scene where 16% of participant also prefer tool 2.

Participants were then asked why they preferred their chosen tools for the scenes. See Appendix 2 – I.

Participants who picked tool 1 for the small scene preferred the faster initial setup. Participants who chose tool 2 for the small and medium scenes preferred the trade-off between the time taken to setup and paint. Participants who chose the context aware tool found it easy to setup regions and fill them.

Participants who chose the context aware tool for the medium scene repeated the previous points, expanding upon them, stating the re-use of palettes made it faster and that it was easier to fill out the larger scenes.

All participants preferred the context aware tool for the large scene, repeating the previous points. With one participant stating they preferred the large fill functionality of the tool. One participant stated that the tool was slower for them. However, they preferred the control over the scene and felt the final result of the scene was the best out of the tools.

The final question asked participants which tool they preferred within various categories including UI, general usability and the Control Scheme.

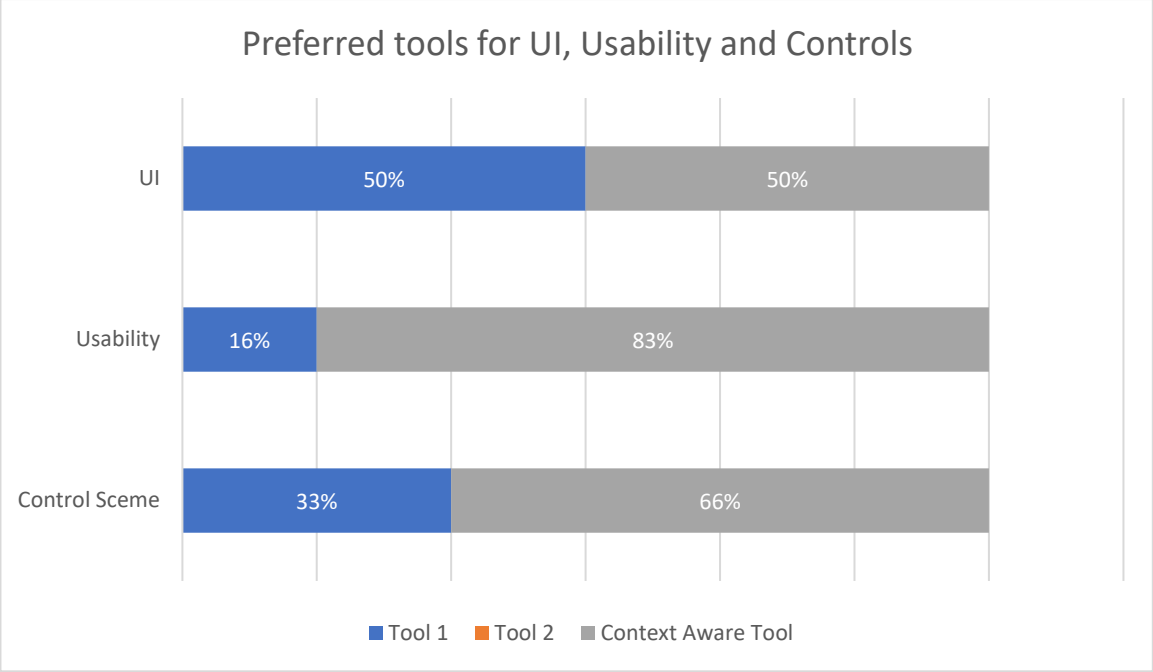

Figure 7. Preferred tools within set categories

As seen in Figure 7, over half of the participants preferred the context aware tool for both the usability and control scheme. However, for the UI, 50% of participants preferred the context aware tool while the other 50% preferred the UI of tool 1.

Lastly, all participants were observed during testing, having notes taken on any additional information spoken or observed. See Appendix 2 – K.

Multiple participants were observed to be more careful and deliberate with the context aware tool, increasing the time taken. Equally, some participants did not take time to experiment with the tools beforehand to get used to the functionality.

When using tool 2 and the context aware tool on larger scenes, the placement of objects caused long loading times within Unity due to the ray-casts for object placement, alongside performance issues with large amounts of objects within a scene.

Users felt the context aware tool would be good for larger maps where regions would need to be edited to produce variations for use in a changing game world, or upon iteration of game scenes where prototype models could be easily replaced, rather than for the creation of smaller scenes.

When using the context aware tool, for some users, the created sub-zones would not always enter into the UI automatically to allow for placement of the zone bounds. This resulted in users needing to drag and drop the sub-zone object from the game scene into the tool UI, increasing the time taken.

This was mostly prevalent when users forgot to assign palettes to zones, resulting in issues with the parent and sub-zones in the UI.

Lastly, when using the context aware tool, some participants placed the zone vertices to ensure the tools zone bounds were very accurate to the scene's regions, leading to an increase in the time taken per scene. This was not done with the other tools.

This chapter provided the results of the questionnaire with supporting notes from the live viewings. It has been shown that the context aware tool takes a low amount of time to repaint existing regions, while the standard brush tool takes the least time for painting smaller scenes.

Participants preferred the context aware tool for the larger scenes, with an 83% preference for medium scenes and a 100% preference for large scenes.

Users considered the context aware tool to have the best usability, with 83% rating its usability as very good and above and its control schemes as the best of the three tools.

However, issues with the context aware tool, such as the sub-zones not being automatically placed within the tool UI, and performance issues on larger scenes hindered their experience.

The following chapter will discuss these findings, linking them together alongside the information within the literature review and an evaluation of their significance.

Discussion and Analysis

This chapter provides critical discussion for the results in the previous chapter, alongside an evaluation of the investigation and the methodology.

The data suggests that the addition of contextual awareness to an object placement tool does not speed up the development time of non-procedural game scenes, being sub-optimal on smaller scenes. However, the tool shows a clear improvement over other tools when editing existing regions.

As seen in Figure 4, tool 1, being a standard brush tool, has the lowest time requirement to fill out the small and medium scenes. Whereas the context aware tool took the longest for the small scene and comes just behind the first two tools in the medium scene and providing an equal speed on average to tool 1 on the large scene. However, the context aware tool is the fastest when repainting an existing region. Tool 2, serving as a mid-way point between the two tools, making use of limited zoning without palettes, took the longest time for the large scene, with a moderate time for the small and medium scenes. Tool 2 equally took a moderate time for repainting a region.

Equally, the timings for the small scene were larger than or similar to the medium scene on average as a result of users being unused to and learning the tools. For the context aware tool, this timing included the time required for the creation of palettes and zones, increasing the time taken. Therefore, it could be suggested that the timing for the small scene without the inclusion of palette creation would be closer to the timing for tool 2 as seen within the medium scene.

As a result of the different hardware used to test the tool for different participants, tool 1 was used as a control as it resembles a standard brush tool. Following this, the timings were used to determine the average change in time as a percentage relative to tool 1, as seen in Figure 8.

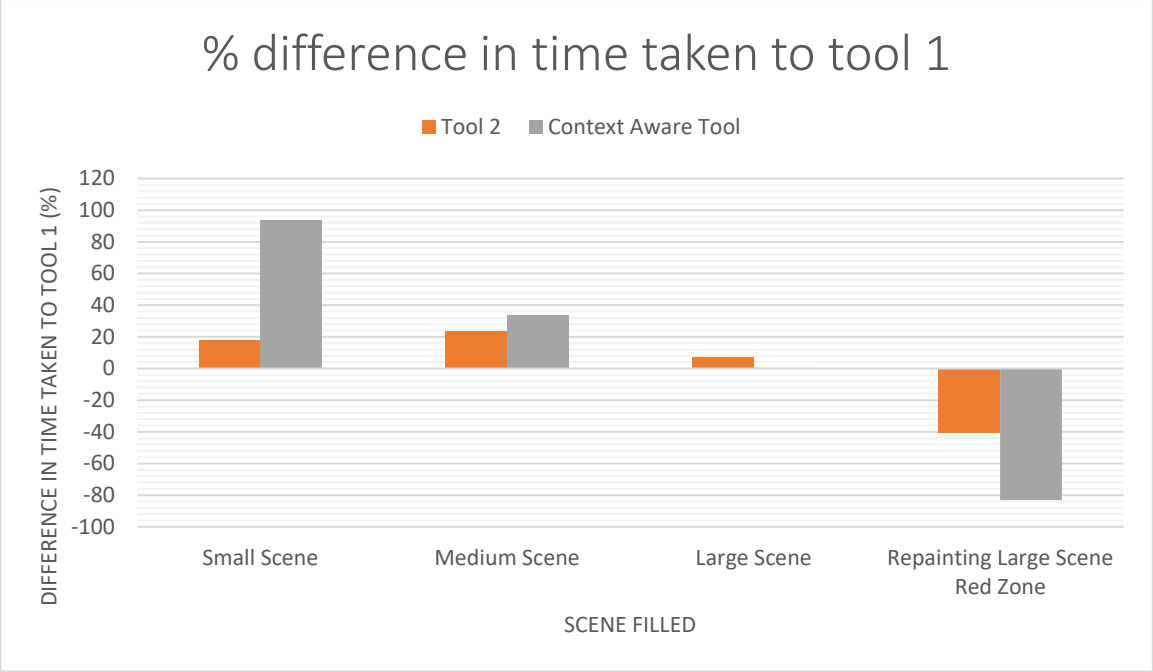

Figure 8. Difference in time to tool 1

The data suggests that the addition of limited zoning to a tool is detrimental to the efficiency when performing the initial painting of a region. Whereas the addition of saveable palettes alongside permanent zoning, allowing the tool to use user defined context for object placement, has no significant impact on the efficiency of a tool when initially painting larger regions. However, when repainting existing regions, this addition provides an 83% reduction in time taken.

This is important when considering the iterative cycle of game production resulting in frequent alterations to existing game scenes (Kultima, 2015; Eitan, 2012; Bernardi, 2025).

This is somewhat in line with the hypothesis, where the context aware tool does improve the efficiency. However, this efficiency is primarily seen following the initial setup, within the modification of existing game regions. Due to the rising scope and size of games (Thomas, 2025), the efficiency with the contextually aware tool is likely to be high within real projects, where a reduction in the time spent altering scenes could have a large impact on costs.

Within smaller projects where there is minimal iteration on game scenes, the addition of contextual awareness to a tool is likely to have a negative impact on efficiency during the initial development of scenes, with the only benefit being the persistence of palettes, removing the constant altering of tool settings to produce a desired output within subsequent scenes.

This is primarily a result of the tools time consuming initial setup, where time that would be spent filling the scene is spent modifying the palettes and setting up the initial zones within the tools UI.

As a result of the importance of usability on a tools efficiency (Lightbown, 2015), usability data was gathered as seen in Figure 5. The results show that the context aware tool had the highest usability rating of the three tools, with no ratings of Fair or below. Equally, tool 1 had high ratings, while tool 2

was rated as poor. This met expectations as tool 1 was a common brush-based tool, thus being intuitive to a wide range of users.

This data may also suggest that the implementation of an in-engine tutorial, alongside popup feedback and UI hierarchy were beneficial additions to improving the learnability of the tool. This can equally be seen by the lower timings for the small scene in Appendix 2 – B, where users who took more time to learn the tool before beginning the testing had lower times.

Furthermore, this high usability suggests that the tool would be unlikely to be replaced and an appropriate addition to larger projects.

As expected, the context aware tool was preferred for the larger scenes. This preference is in line with the reduction in UI alterations once the tool was setup. This was expanded upon within the questionnaire as seen in Appendix 2 – I, where users found the region definition fast once the palettes had been set up. Equally, users preferred the final output of the context aware tool when compared to the other two tools. This supports the PCG statements by (Doran & Parberry, 2010), where a PCG tool should require a low degree of human input using designer-centric parameters to allow for a high degree of control, with users not needing to understand the tools inner workings to use it effectively.

However, despite the larger time requirement, users preferred the context aware tool for the small scene, expanding upon this within the questionnaire as seen in Appendix 2 – F. Users preferred the one-time setup of the region palettes and precision compared to the other tools, citing the ease of painting the regions compared to the other tools.

It is worth noting that this preference for the context aware tool was only 50% for the small scene, as some users preferred the fast setup of tool 1.

This data is further reinforced by the overall usability data as seen in Figure 7. The majority of participants preferred the context aware tool for usability, the same result is shown for the control scheme. This suggests the simple, descriptive UI and tool process were beneficial for users. However, 50% of participants preferred tool 1s UI, favouring the graphical representation of assigned objects.

As evidenced by the results, the methodology undertaken was adequate for the project. The use of live viewings provided adequate supplementary data to support the results from the questionnaire. This allowed for the identification of any shortcomings of the developed tool that may have affected the outcome of the testing. Equally, the use of an anonymous questionnaire allowed users to share their opinions without any potential bias.

The use of Likert scales provided the appropriate amount of choice for users to express opinions on specific aspects of a tool, while providing the means to average this data to produce a generalised overview which could then be compared between tools and linked to other results.

The use of convenience sampling provided appropriate participants who had experience within games development. As a result, the data gathered was closer to an accurate representation of the tools being used within a real game development scenario.

Finally, the metrics gathered provided accurate data to determine the tools efficiency, highlighting key aspects that make a tool successful.

The generalisability of the results is limited by the small participant count. This has potentially resulted in inaccurate average results for timing data, where outlying data has a larger impact of the average result. This has potentially resulted in inaccurate analysis on the effectiveness of tools on different scene sizes.

The reliability of the data is impacted by the high object count of larger scenes, where high amounts of ray-casts resulted in frequent pauses where the user had to wait for unity to load. Equally, as observed, some users were quite imprecise when painting the regions within scenes with the first 2 tools, this was not drastic and so had only a minor impact on the timings. However, some participants were very accurate using the context aware tool due to the zoning implementation, placing zones to be accurate to the region definitions within the scenes. This included placing zones to accommodate small outcroppings from regions. As a result, some of the timings for the context aware tool were slightly inflated, increasing the average times to fill scenes. This had no impact on the repainting data.

Finally, users varied in their times taken to learn the tools completely prior to beginning the testing, resulting in slight disparity in timings between participants due to differing levels of competency with each tools workflow.

However, the results obtained are nonetheless valid for the purpose of answering the research question. Firstly, the variance in precision when using different tools was only limited to a few participants and may equally be present when used within a real game development scenario.

Secondly, linking the timing data and observational notes highlighted issues with the tools that affected the users speed when learning the tools, providing relevant feedback that may be used to improve upon the developed tool, as outlined in the conclusion, and highlighted potential reasons for the slower timings.

Finally, while the participant count was lower than desired, any outlying results have been excluded from the average, where all other data included had multiple similar recordings from other participants, suggesting their results were not outliers.

The performance impact on larger scenes caused wait times around the 1-minute mark for tools 2 and the context aware tool. However, the explicit times of these wait times alongside their frequency were not recorded. As a result, they were not excluded from the participants results for the larger scene timings and the average timing data was not reduced.

This may suggest that a more performant placement method not using ray-casts would result in the context aware tool being more efficient than tool 1 for the large scene, and potentially for the medium scene.

As a result, the data cannot confirm the exact impact on efficiency contextual awareness would have without these performance impacts.

The results gathered may have been different under a slightly different methodology. For example, more accurate timing data may have been gathered if participants had to fill two of each scene size. This may produce more accurate general data per participant, while also producing data for filling a small scene without the palette setup for the context aware tool.

On top of this, more accurate data may have been acquired if users were given an explicit testing scene for each tool where users were not timed as they completed the task, allowing users to learn the workflow of the tool in a controlled manner.

Conclusion

Through quantitative and qualitative analysis of various tools when used to paint multiple game scenes of differing size, it can be concluded that the addition of contextual awareness does not speed up development when used on smaller scenes. The developed tool has an equal speed to standard tools on large scenes in its current state. However, it does speed up the modification of existing regions.

The research approach as stated in Research Methodologies, including a questionnaire and live viewings, was used to provide easily comparable data that supplemented each data point to produce reliable and accurate results.

During the testing, users gave honest feedback, stating grievances with the tools when encountered and taking care to not rush the testing process. However, the tool had some usability issues that hindered the users, resulting in a potentially worse performance when using the context aware tool. These included an issue within the tools UI that was not encountered during development, alongside some difficulty in learning the tools workflow.

As stated previously, the secondary research served to influence the design of the produced tool to ensure it was of high quality, comparable to other tools and had appropriate functionality in terms of PCG, zoning and UI implementation. The research was from peer reviewed sources to ensure accurate research with verified impacts on the quality of tools.

The tool produced incorporated functionality that allowed users to create palettes of objects with groupings and weightings that could be saved for future use. These could then be applied to user specified zones created within the tools UI and defined through vertices within a game scene in editor, allowing for the automatic object placement using the associated palette.

Upon initial development, it was expected that the addition of contextual awareness would not speed up the development of smaller scenes due to the time required for the creation of palettes. This expectation was met. However, the longer average time for larger scenes did not match expectations, where it was believed the context aware tool would be the fastest. As stated in Discussion and Analysis, the question is raised of whether the longer timings were a result of performance issues brought about through the ray-cast based object placement, and whether the grievances with usability stated within the questionnaire also contributed to the longer timings.

On average, users showed a preference towards the context aware tool for larger scenes, stating that they found the vertex placement easy to use and convenient for filling regions with a more complex shape. Users did find the setup more time consuming, however.

The paper describes the key aspects of tools development with the addition of research into PCG for the creation of a contextually aware tool. Throughout the course of the work, the relevance of usability within tooling and PCG was emphasised and incorporated into the development of the produced tool, having an impact on a user's ability to learn, understand and use a tool effectively. Equally, HCI was highly important in producing a tool to undergo accurate comparison against other not contextually aware tools, ensuring the produced tool was at an appropriate quality to ensure fair testing.

Whilst the developed tool only represents the addition of contextual awareness to one type of game development tool, this study shows how contextual awareness may affect the game development process were it to be incorporated into other tools. This is most clearly exemplified by the timing data, displaying the average timings to fill different scenes. This is further reinforced by the users' comments on the tool, both within the questionnaire and live observations.

The impacts include an increased initial setup time, with less time savings on smaller projects. However, the additions would have a benefit on larger games undergoing prototyping and iteration, where game scenes receive frequent alterations.

Recommendations

Future research should be undertaken to address some of the shortcomings of the study. One such alteration would be the improvement of the tool's performance, changing the ray-cast based placement for a position-based placement using a heightmap. Doing so would likely remove the delay on placing objects within larger scenes, providing a more accurate representation of the tools speed. Equally, the minor UI bugs have been resolved, ensuring a higher-quality workflow. However, this would require a new round of testing to determine if these issues were the cause of the slower times.

A potential addition for future research may be the incorporation of a brush interface for the context aware tool, creating a variation of the tool making use of the palette system without requiring the user make use of the zone functionality. This would allow users to make use of a brush style tool for smaller scenes while using the zone system for larger scenes or scenes that may undergo frequent alterations.

This could be expanded upon, where the brush placement could also define the zones while painting using the active palette, combining the speed of a brush tool for smaller scenes while also providing the zone functionality for future modification.

Further research would need to make use of a higher participant count to ensure more accurate data. This would provide more accurate trends in user preference and opinions, providing avenues for further research, while ensuring more generalizable average timing data.

Equally, future studies should make use of multiple scenes per size distinction, ensuring a more accurate timing per size while also providing data on smaller scenes without the addition of palette creation time.

Future studies should take into account users' ability to learn the tools, providing useful data on a tools usability and its potential incorporation into industry.

Avenues for future research include the addition of contextual awareness to other game development tools to determine if contextual awareness would provide benefits within different contexts.

References

Barczak, A. & Woźniak, H. (2019). Comparative study on game engines. *Studia Informatica. System and information technology.* [Online]. 23 (1–2). p.pp. 5–24. Available from: https://czasopisma.uph.edu.pl/index.php/studiainformatica/article/download/1864/1578. [Accessed: 8 November 2025].

Bernardi, H. (2025). *Insider Secrets: How Game Studios Quietly Out-Test the Competition (Presented by PickFu)*. [Online]. 2025. Available from: https://gdcvault.com/play/1035381/Insider-Secrets-How-Game-Studios. [Accessed: 8 February 2026].

Bhandari, P. (2022). Observer Bias | Definition, Examples, Prevention. *Scribbr*. [Online]. Available from: https://www.scribbr.co.uk/bias-in-research/observer-bias-explained/. [Accessed: 12 December 2025].

Biostart (2024). *Easy Placement Tool | Utilities Tools | Unity Asset Store*. [Online]. 25 March 2024. Unity Asset Store. Available from: https://assetstore.unity.com/packages/tools/utilities/easy-placement-tool-273061. [Accessed: 9 March 2025].

Cambridge University Press & Assessment (2025a). *customizable*. [Online]. 18 November 2025. Cambridge Dictionary. Available from: https://dictionary.cambridge.org/dictionary/english/customizable. [Accessed: 23 November 2025].

Cambridge University Press & Assessment (2025b). *tool*. [Online]. 12 November 2025. Cambridge Dictionary. Available from: https://dictionary.cambridge.org/dictionary/english/tool. [Accessed: 16 November 2025].

Carroll, J.M. (2014). *Human Computer Interaction - brief intro*. [Online]. Interaction Design Foundation - IxDF. Available from: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro. [Accessed: 23 October 2025].

CD PROJEKT (2021). *CD PROJEKT Group Presentation - FY 2020*. [Online]. 22 April 2021. Result Center. Available from: https://www.cdprojekt.com/en/accounting-year/2020/. [Accessed: 18 January 2026].

CD PROJEKT (2015). The CD PROJEKT Group summarizes the release of The Witcher 3. *The CD PROJEKT Group summarizes the release of The Witcher 3*. [Online]. Available from: https://www.cdprojekt.com/en/media/news/the-cd-projekt-group-summarizes-the-release-of-the-witcher-3/. [Accessed: 18 January 2026].

Cooper, A. (2004). *The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*. [Online]. Sams. Available from: https://learning.oreilly.com/library/view/inmates-are-running/0672326140/. [Accessed: 15 October 2025].

Doody, O. & Noonan, M. (2013). Preparing and conducting interviews to collect data. *Nurse Res*. [Online]. 20. p.pp. 28–32. Available from: https://www.researchgate.net/publication/236922140_Preparing_and_conducting_interviews_to_collect_data. [Accessed: 2 January 2026].

Doran, J. & Parberry, I. (2010). Controlled Procedural Terrain Generation Using Software Agents. *Computational Intelligence and AI in Games, IEEE Transactions on*. [Online]. 2. p.pp. 111–119. Available from: https://www.researchgate.net/publication/224133576_Controlled_Procedural_Terrain_Generation_Using_Software_Agents. [Accessed: 16 October 2025].

Eitan, G. (2012). *Rapid, Iterative Prototyping Best Practices*. [Online]. 2012. Available from: https://gdcvault.com/play/1016990/Rapid-Iterative-Prototyping-Best. [Accessed: 8 February 2026].

Eliasz, P.M. (2024). *Procedural Content Generation with Unreal Engine 5*. [Online]. Packt Publishing. Available from: https://learning.oreilly.com/library/view/procedural-content-generation/9781801074469/. [Accessed: 9 October 2025].

Frost, P. (2003). *The Tools Development of Turbine's Asheron's Call 2*. [Online]. 20 August 2003. Game Developer. Available from: https://www.gamedeveloper.com/programming/the-tools-development-of-turbine-s-i-asheron-s-call-2-i-. [Accessed: 23 November 2025].

Funkybyte (2024). *Prefab Painter 2 | Painting | Unity Asset Store*. [Online]. 29 March 2024. Available from: https://assetstore.unity.com/packages/tools/painting/prefab-painter-2-61331. [Accessed: 10 October 2025].

GDC Festival of Gaming (2015). *50 Game Camera Mistakes*. [Online]. 17 November 2015. Available from: https://www.youtube.com/watch?v=C7307qRmlMI. [Accessed: 23 November 2025].

Gen90Software (2023). *Props Placement Tool*. [Online]. 18 October 2023. Unity Asset Store. Available from: https://assetstore.unity.com/packages/tools/level-design/props-placement-tool-208217. [Accessed: 9 March 2025].

George, T. (2023). Primary Research | Definition, Types, & Examples. *Scribbr*. [Online]. Available from: https://www.scribbr.co.uk/research-methods/primary-research-explained/. [Accessed: 12 December 2025].

Green, D. (2016). *Procedural Content Generation for C++ Game Development*. [Online]. Packt Publishing. Available from: https://learning.oreilly.com/library/view/procedural-content-generation/9781785886713/. [Accessed: 9 October 2025].

Green, T. & Labrecque, J. (2023). *A Guide to UX Design and Development: Developer's Journey Through the UX Process*. Design Thinking. 1st Edn. [Online]. Berkeley, CA: Apress. Available from: https://learning.oreilly.com/library/view/a-guide-to/9781484295762/?ar/?email=%5Eu. [Accessed: 11 October 2025].

Gustavson, S. (2015). *Simplex Noise Demystified*. [Online]. Available from: https://www.researchgate.net/publication/281031452_Simplex_Noise_Demystified. [Accessed: 21 November 2025].

Hyttinen, T. (2017). *Terrain synthesis using noise*. [Online]. University of Tampere. Available from: https://trepo.tuni.fi/bitstream/handle/10024/101043/GRADU-1494236249.pdf. [Accessed: 20 November 2025].

Interaction Design Foundation (2016a). *What is Human-Computer Interaction (HCI)?* [Online]. 6 June 2016. The Interaction Design Foundation. Available from: https://www.interaction-design.org/literature/topics/human-computer-interaction. [Accessed: 22 October 2025].

Interaction Design Foundation (2016b). *What is User Experience (UX) Design? — updated 2025*. [Online]. 1 June 2016. The Interaction Design Foundation. Available from: https://www.interaction-design.org/literature/topics/ux-design. [Accessed: 5 October 2025].

ISO 9241-210 (2010). *ISO 9241-210:2010(en), Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. [Online]. March 2010. Available from: https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-1:v1:en. [Accessed: 29 October 2025].

James, D., Hicks, K. & Headleand, C. (2025). *Peril: A Level Design Tool for Games Education*. [Online]. p.p. 2. Available from: https://eprints.staffs.ac.uk/9254/1/Peril_Paper.pdf. [Accessed: 11 October 2025].

Joshi, A., Kale, S., Chandel, S. & Pal, D. (2015). Likert Scale: Explored and Explained. *British Journal of Applied Science & Technology*. [Online]. 7 (4). p.pp. 396–403. Available from: https://journalcjast.com/index.php/CJAST/article/view/381. [Accessed: 2 January 2026].

Kovanen, S. (2018). *Extending a game engine with custom tools*. [Online]. Aalto University. Available from: https://aaltodoc.aalto.fi/server/api/core/bitstreams/b857ce60-0d26-4037-8a1e-330b7e4279c1/content. [Accessed: 8 November 2025].

Kultima, A. (2015). Developers' perspectives on iteration in game development. In: *Proceedings of the 19th International Academic Mindtrek Conference*. [Online]. 22 September 2015, Tampere Finland: ACM, pp. 26–32. Available from: https://dl.acm.org/doi/10.1145/2818187.2818298. [Accessed: 8 February 2026].

Lightbown, D. (2015). *Designing the User Experience of Game Development Tools*. [Online]. A K Peters/CRC Press. Available from: https://learning.oreilly.com/library/view/designing-the-user/9781482240191/. [Accessed: 14 October 2025].

Lightbown, D. (2021). *Tools Live Longer Than Games Do: What I Learned About Tools Development From Games Industry Legends*. [Online]. 2021. Available from: https://gdcvault.com/play/1026964/Tools-Live-Longer-Than-Games. [Accessed: 8 October 2025].

Mahtani, K. & Spencer, E. (2017). *Hawthorne effect | Catalog of Bias*. [Online]. Available from: https://catalogofbias.org/biases/hawthorne-effect/. [Accessed: 2 January 2026].

NOT_Lonely (2023). *Object Placement Tool*. [Online]. 1 February 2023. Unity Asset Store. Available from: https://assetstore.unity.com/packages/tools/painting/object-placement-tool-99591. [Accessed: 9 March 2025].

Perlin, K. (1985). An image synthesizer. In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '85. [Online]. 1 July 1985, New York, NY, USA: Association for Computing Machinery, pp. 287–296. Available from: https://dl.acm.org/doi/10.1145/325334.325247. [Accessed: 20 November 2025].

Perlin, K. (2002). Improving noise. *ACM Trans. Graph.* [Online]. 21 (3). p.pp. 681–682. Available from: https://dl.acm.org/doi/10.1145/566654.566636. [Accessed: 23 November 2025].

Perlin, K. (2001). *Noise Hardware*. [Online]. 2001. ACM SIGGRAPH 2002 Course 36 Notes. Available from: https://userpages.cs.umbc.edu/olano/s2002c36/. [Accessed: 23 November 2025].

Pitt, C. (2023). *Procedural Generation in Godot: Learn to Generate Enjoyable Content for Your Games*. [Online]. Apress. Available from: https://learning.oreilly.com/library/view/procedural-generation-in/9781484287958/. [Accessed: 9 October 2025].

Shaker, N., Togelius, J. & Nelson, M.J. (2016). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research.* [Online]. p.p. 224. Available from: https://www.pcgbook.com/. [Accessed: 9 October 2025].

Snyder, A. (1986). Encapsulation and inheritance in object-oriented programming languages. *SIGPLAN Not.* [Online]. 21 (11). p.pp. 38–45. Available from: https://dl.acm.org/doi/10.1145/960112.28702. [Accessed: 23 November 2025].

Thomas, D. (2025). *Video games cost a whole day's wages says fan, as prices rise*. [Online]. 28 June 2025. BBC News. Available from: https://www.bbc.co.uk/news/articles/c2le0wpwe18o. [Accessed: 18 January 2026].

Thomas, L. (2022a). Confounding Variables | Definition, Examples & Controls. *Scribbr*. [Online]. Available from: https://www.scribbr.co.uk/research-methods/confounding-variable/. [Accessed: 12 December 2025].

Thomas, L. (2022b). Cross-Sectional Study | Definitions, Uses & Examples. *Scribbr*. [Online]. Available from: https://www.scribbr.co.uk/research-methods/cross-sectional-design/. [Accessed: 12 December 2025].

Thorn, A. (2011). *Game Engine Design and Implementation*. [Online]. Jones & Bartlett Publishers. Available from: https://books.google.co.uk/books?hl=en&lr=&id=cFnPvWK9-akC&oi=fnd&pg=PR1&dq=game+engine+tools&ots=nmKFDwSAWZ&sig=u0fXzkf_T7w27Xf9bd1ZIk6zAQk&redir_esc=y#v=onepage&q=game%20engine%20tools&f=true. [Accessed: 25 November 2025].

Unity Technologies (n.d.). *Unity - Scripting API: Mathf.PerlinNoise*. [Online]. Available from: https://docs.unity3d.com/6000.2/Documentation/ScriptReference/Mathf.PerlinNoise.html. [Accessed: 16 November 2025].

Watkins, R. (2016). *Procedural Content Generation for Unity Game Development*. [Online]. Packt Publishing. Available from: https://learning.oreilly.com/library/view/procedural-content-generation/9781785287473/. [Accessed: 9 October 2025].

Williams, A. (2009). User-centered design, activity-centered design, and goal-directed design: a review of three methods for designing web applications. In: *Proceedings of the 27th ACM international conference on Design of communication*. SIGDOC '09. [Online]. 5 October 2009, New York, NY, USA: Association for Computing Machinery, pp. 1–8. Available from: https://doi.org/10.1145/1621995.1621997. [Accessed: 3 January 2026].

Wu, Z. & Liu, J. (2024). Perlin noise and its improvements: A literature review. *Applied and Computational Engineering*. [Online]. 77. p.pp. 278–287. Available from: https://www.researchgate.net/publication/393342810_Perlin_noise_and_its_improvements_A_literature_review. [Accessed: 20 November 2025].

Appendices

Appendices is information referred to in the main document. It is not included in the word count.

Do not put results here: only the raw data should be presented in an appendix. Other materials that may be included in an appendix includes, for example, blank questionnaires, copy of written tests used.

Remember do not include anything in an appendix that has not been referred to in the text.

Appendix 1 – A

Context Aware Asset Placement Tool review.

* Required

1. Input a Memorable Word (this will be used so your data can be removed if requested) *

2. Please Upload your consent form *

Upload file

File number limit: 1 Single file size limit: 100MB Allowed file types: Word, Excel, PPT, PDF, Image, Video, Audio

3. Have you used 3D Editor Asset Placement brush tools before?

4. Please Upload a Screen recording of you painting the 3 scenes with Tool 1.

Upload file

File number limit: 1 Single file size limit: 1GB Allowed file types: Word, Excel, PPT, PDF, Image, Video, Audio

5. How much time did it take to fill the small scene using tool 1? (Format: minutes:seconds)

6. How much time did it take to fill the medium scene using tool 1? (Format: minutes:seconds)

7. How much time did it take to fill the large scene using tool 1? (Format: minutes:seconds)

8. How much time did it take to repaint the large scene red biome using tool 1? (Format: minutes:seconds)

9. Please Upload a Screen recording of you painting the 3 scenes with Tool 2.

Upload file

File number limit: 1 Single file size limit: 1GB Allowed file types: Word, Excel, PPT, PDF, Image, Video, Audio

10. How much time did it take to fill the small scene using tool 2? (Format: minutes:seconds)

11. How much time did it take to fill the medium scene using tool 2? (Format: minutes:seconds)

12. How much time did it take to fill the large scene using tool 2? (Format: minutes:seconds)

13. How much time did it take to repaint the large scene red biome using tool 2? (Format: minutes:seconds)

14. Please Upload a Screen recording of you painting the 3 scenes with the Context Aware Tool.

Upload file

File number limit: 1 Single file size limit: 1GB Allowed file types: Word, Excel, PPT, PDF, Image, Video, Audio

15. How much time did it take to fill the small scene using the Context Aware Tool? (Format: minutes:seconds)

16. How much time did it take to fill the medium scene using the Context Aware Tool? (Format: minutes:seconds)

17. How much time did it take to fill the large scene using the Context Aware Tool? (Format: minutes:seconds)

18. How much time did it take to repaint the large scene red biome using the Context Aware Tool? (Format: minutes:seconds)

19. How Would you rate each tools Usability (how easy it was to use/understand)?

	Poor	Fair	Good	Very Good	Excell
Tool 1	○	○	○	○	○
Tool 2	○	○	○	○	○
Context Aware Tool	○	○	○	○	○

20. For the Context aware tool, if the usability was below good, what could be improved to make it easier to use?

21. What did you dislike about the context aware tool?

22. What did you like about the context aware tool?

23. If you preferred another tool, what did they have that the others did not?

24. Which tool did you prefer for each scene?

	Tool 1	Tool 2	Context Aware To
Small Scene	◯	◯	◯
Medium Scene	◯	◯	◯
Large Scene	◯	◯	◯

25. For the small scene, why did you prefer your chosen tool?

26. For the medium scene, why did you prefer your chosen tool?

27. For the large scene, why did you prefer your chosen tool?

28. For each Category, which tool did you prefer?

	Tool 1	Tool 2	Context Aware To
UI	◯	◯	◯
Usability	◯	◯	◯
Control Scheme	◯	◯	◯

Appendix 2 – A

Experience with brush tools:

Have you used 3D Editor Asset Placement brush tools before?	
ID: 4	No
ID: 5	Yes
ID: 6	No
ID: 7	Yes
ID: 8	No
ID: 9	Yes

Appendix 2 – B

Timing Data:

Tool 1	Small Scene	Medium Scene	Large Scene	Large Scene Red Biome
ID: 4	2:00	3:00	9:00	5:00
ID: 5	6:40	4:36	6:12	2:31
ID: 6	2:00	4:00	5:00	2:00
ID: 7	4:24	3:59	6:42	
ID: 9	2:45	2:41	4:33	1:44
Average:	213.8 sec 3:33.8 (3.56)	219.2 sec 3:39.2 (3.65)	377.4 sec 6:17.4 (6.29)	168.75 sec 2:48.75 (2.81)

Tool 2	Small Scene	Medium Scene	Large Scene	Large Scene Red Biome
ID: 5	4:37	4:19	5:37	1:50
ID: 7	5:33	6:36	11:29	2:41
ID: 9	2:29	2:37	3:09	0:28
Average:	253 sec 4:13 (4.22)	270.66 sec 4:30.66 (4.51)	405 sec 6:45 (6.75)	99.66 sec 1:39.66 (1.66)

CAAPT	Small Scene	Medium Scene	Large Scene	Large Scene Red Biome
ID: 4	10:00	3:30	6:30	0:23
ID: 5	12:13	6:26	7:14	0:30
ID: 6	1:30	2:00	2:00	0:25
ID: 7	7:41	7:04	6:35	
ID: 8	3:29	7:04	11:18	
ID: 9	6:33	3:16	4:16	0:36
Average:	414.33 sec 6:54.33 (6.91)	293.33 sec 4:53.33 (4.88)	378.83 sec 6:18.83 (6.31)	28.5 sec 0:28.5 (0.48)

Appendix 2 – C

Usability Data:

	Tool 1	Tool 2	CAAPT
ID: 4	Very Good - 4	Poor - 1	Excellent - 5

ID: 5	Good - 3	Poor - 1	Very Good - 4
ID: 6	Fair - 2	Poor - 1	Very Good - 4
ID: 7	Good - 3	Fair - 2	Excellent - 5
ID: 8	Very Good - 4	Very Good - 4	Good - 3
ID: 9	Excellent - 5	Fair - 2	Very good - 4
Average:	3.5	1.8	4.2

Appendix 2 – D

For the Context aware tool, if the usability was below good, what could be improved to make it easier to use?	
ID: 4	Create parent and sub-zones automatically. Sub-Zones sometimes don't go into Ui automatically.
ID: 5	Performance problems on larger scenes.
ID: 9	Tooltips - probably would've ben harder to understand workflow on my own. Plus automatic parenting and subzones.

Appendix 2 – E

What did you dislike about the context aware tool?	
ID: 4	Setup took too long.
ID: 6	nothing really, i skipped past the tutorial and found everything rather intuitive in regards to what had to be done.
ID: 8	A issue I found was that new sub zones wouldn't bind to the tool like a new zone would, costing a little time to rebind to a new subzone.
ID: 9	Workflow can be a bit complicated to set up

Appendix 2 – F

What did you like about the context aware tool?	
ID: 4	Easy to use
ID: 5	More percise than the other tools.
ID: 6	how simple the plotting points were and the ease of making different shapes
ID: 7	I love how you can just make whatever shape you need for placing all the objects. I also love how you can create your own presets and just set the weights on everything. It is very convenient to use.
ID: 8	Using the context aware tool allowed me to set an area and the assets that I plan to use and allow the engine to complete the rest of the task. It greatly reduces time spent painting instead that time is focused mainly on setting up the tool.
ID: 9	Very quick and easy to use once setup has been complete (high learning curve but large payoff)

Appendix 2 – G

If you preferred another tool, what did they have that the others did not?	
ID: 9	I prefered the prefab loading from tool 1

Appendix 2 – H

Preferred tool for each scene.

	Small Scene	Medium Scene	Large Scene
ID: 4	Tool 1	CAAPT	CAAPT
ID: 5	CAAPT	CAAPT	CAAPT
ID: 6	CAAPT	CAAPT	CAAPT
ID: 7	CAAPT	CAAPT	CAAPT
ID: 8	Tool 2	Tool 2	CAAPT
ID: 9	Tool 1	CAAPT	CAAPT

Appendix 2 – I

Why was the chosen tool preferred for the scene size?

	Small Scene	Medium Scene	Large Scene
Tool 1	• ID: 4 - The initial set up is quicker • ID: 9 - Easier to fill in the weird shapes with a mouse-driven brush		
Tool 2	• ID: 8 - It was the most efficient in terms of set up and time taken to paint	• ID: 8 - It was the most efficient in terms of set up and time taken to paint	
CAAPT	• ID: 6 - Easy to setup points to generate area and easy to fill in said area • ID: 7 - It did everything the other two tools did but better	• ID: 4 - After setting it up could re-use everything so it was quicker. It is also easier to fill out bigger scenes • ID: 6 - Easy to setup points to generate area and easy to fill in said area • ID: 7 - It did everything the other two tools did but better • ID: 9 - Biezer curve dots were quick to place with the large fill	• ID: 4 - After setting it up could re-use everything so it was quicker. It is also easier to fill out bigger scenes • ID: 6 - Easy to setup points to generate area and easy to fill in said area • ID: 7 - It did everything the other two tools did but better • ID: 8 - Despite being slower the control over the scene felt better and the final result was the best of the three. • ID: 9 - Biezer curve dots were quick to place with the large fill

Appendix 2 – J

For each category, which tool was preferred?

	UI	Usability	Control Scheme
ID: 4	CAAPT	CAAPT	CAAPT
ID: 5	Tool 1	Tool 1	Tool 1
ID: 6	Tool 1	CAAPT	CAAPT
ID: 7	CAAPT	CAAPT	CAAPT
ID: 8	CAAPT	CAAPT	CAAPT
ID: 9	Tool 1	CAAPT	Tool 1

Appendix 2 – K

Observational data:

Some participants took less time to learn the tools before starting, leading to time increases for the more complex tools 2 and the context aware tool.

Some users when using the context aware tool, placed zones very accurately to the boundaries, potentially increasing the time taken.

Some users were quite imprecise to the biome bounds when using tools 1 and 2, potentially reducing the time taken.

Users felt tool 2 was cumbersome for filling out large areas with complex region shapes.

Users felt the context aware tool would be good for larger maps where regions would need to be edited to produce variations for use in a changing game world.

Users seemed hesitant initially with tool 2 and the context aware tool as they were not as used to it compared to a brush-based tool.

For ID: 8 – they did not paint the biomes using the specific palette instructions for tool 1 and 2, reducing their times taken.

For ID: 7 & 8 – repainting the red sub zone, they deleted the entire sub zone for the context aware tool and redefined the entire zone again instead of just deleting and re painting the objects – potential lack of learnability or misunderstanding of task.

The context aware tool on large scenes had long loading times due to the high amount of raycasts – could be fixed by using heightmap based placement using the terrain.

Users had difficulty with creating and adding new zones, trying to delete or modify existing zones.

Appendix 2 - L

Excluded Data

Tool 1	Small Scene	Medium Scene	Large Scene	Large Scene Red Biome
ID: 7				7:15 – redid entire scene on accident (excluded from the average)

ID: 8 (Painted entire scene using one palette - excluded from the average)	0:24	0:42	2:36	1:00

Tool 2	Small Scene	Medium Scene	Large Scene	Large Scene Red Biome
ID: 8 (Painted entire scene using one palette - excluded from the average)	0:15	0:18	2:04	1:19

CAAPT	Small Scene	Medium Scene	Large Scene	Large Scene Red Biome
ID: 7				2:06 (deleted and redefined red zone entirely - excluded from the average)
ID: 8				2:26 (deleted and redefined red zone entirely - excluded from the average)