Melete: Play testing and 3D environments for Mixed Initiative Artificial Intelligence as a method for prototyping video game levels

Sokol Murturi[1,2,*,†], Joseph Walton-Rivers[1] and Michael James Scott[1]

[1]Falmouth University, Faculty of Screen, Technology and Performance, Cornwall, United Kingdom
[2]Goldsmiths University of London, Department of Computing, London, United Kingdom

Abstract

Recent advancements in artificial intelligence and human-computer interaction have led to tools that help human authors collaborate creatively with computing systems. Many such tools are available for 2D video games, but exploration of these techniques in a 3D environment is necessary. This paper introduces Melete, a mixed-initiative artificial intelligence tool which uses the Wave Function Collapse algorithm and a human-in-the-loop design process to create and test 3D levels and environments for video games. Analysis by a panel of expert game developers in two design challenges—a battle royale game and a real-time strategy game—indicate that Melete enhances designer engagement and facilitates iterative design. However, it also draws attention to the importance of play testing in the design loop, kernel generation and the usability of such tools. The findings suggest that Melete's workflow helped designers to seed ideas, explore concepts, and refine implementations quickly with creative agency, but also suggest that further work is needed to integrate Mixed-Initiative Artificial Intelligence techniques into design tools effectively.

Keywords

game, game design, levels, environments, development tool, computational creativity, expert evaluation, qualitative

1. Introduction

The role of AI has been expanding over the last ten years. Fields such as medical diagnostic systems, financial trading algorithms, driverless cars, customer engagement systems, linguistics, audio, games, and countless other areas have adapted or implemented AI algorithms. However, these systems require expert knowledge to interpret and operate. This limits the deployment of these systems to industry experts with this technology [1, 2].

AI has been a key component in video game design and development for a long time. One of its primary applications is procedural content generation (PCG), where algorithms create various game elements such as art, levels, and rules. These algorithms are generally used in one of two ways defined by Togelius et al.[3] as, 'Online', within a game to dynamically produce content at runtime without direct human intervention. Or 'Offline' - generating content that is later refined by human designers, such as AI-generated terrain that is manually adjusted. Efforts have been made to provide user-friendly interfaces to these offline PCG algorithms [4, 5]. and the exploration of the interaction between humans and artificial intelligence is known as Mixed-Initiative Artificial Intelligence (MIAI) or Co-creative AI.[6]

MIAI pipelines are collaborative environments where users work with artificial intelligence agents, seamlessly blending their contributions to the creative process[7]. MIAI pipelines have been utilized in various creative fields, including art, music, dance, drawing, and game design [8]. Most existing MIAI

Synergy 2025 HHAI-WS 2025: Workshops at the Fourth International Conference on Hybrid Human-Artificial Intelligence (HHAI), June 9–13, 2025, Pisa, Italy.

sokol.murturi@falmouth.ac.uk (S. Murturi); Joseph.Walton-rivers@falmouth.ac.uk (J. Walton-Rivers); michael.scott@falmouth.ac.uk (M. J. Scott)
https://www.falmouth.ac.uk/staff/sokol-murturi (S. Murturi); https://www.falmouth.ac.uk/staff/joseph-walton-rivers (J. Walton-Rivers); https://www.falmouth.ac.uk/staff/dr-michael-scott (M. J. Scott)
0000-0001-9466-8981 (S. Murturi); 0000-0002-3406-0584 (J. Walton-Rivers); 0000-0002-6803-1490 (M. J. Scott)

pipelines discussed in this paper focus primarily on level development for 2D games, such as Mario, The Legend of Zelda, and physics-based puzzles, such as Cut the Rope. Although these games have a strong presence in academic research, applying state-of-the-art MIAI techniques to an industry settings requires systems that support level design and prototyping for 3D environments.

Developing levels in 3D which is commonly reserved for AAA studios[9] presents several challenges, one of the most significant being play testing. Play testing is a crucial part of the game development cycle and is increasingly recognized as standard practice. It is widely employed in Games User Research (GUR) to enhance usability, player experience, and the development life cycle itself[10]. GUR, an interdisciplinary field rooted in psychology and Human-Computer Interaction (HCI), enables developers to refine designs through designer evaluations that provide deeper insights into possible player behavior and interaction.

However, many MIAI pipelines focus primarily on level generation without integrating play testing mechanisms, limiting their potential to support comprehensive game development workflows, particularly for complex 3D environments. To allow developers to create meaningful applications of MIAI pipelines it is crucial to establish and ensure their effectiveness as tools for creative endeavors.

The primary contributions of this paper are:

- The introduction of a novel MIAI pipeline, Melete. That can be used for prototyping map layouts for 3D environments.
- Data analysis of an expert review, that highlights the importance of play test integration of MIAI pipelines.

In this paper, we explore existing MIAI pipelines and their limitations, with a particular focus on their application in game design. We provide a range of recent examples to showcase the diverse applications of MIAI Pipelines. Following this, we present our approach to validating our own MIAI system, Melete, along with an in-depth look at its 3D designer-in-the-loop creative process. This includes an overview of the Wave Function Collapse (WFC) algorithm, emphasizing its texture synthesis approach for kernel generation. We then present our expert analysis and qualitative validation approach for Melete, highlighting the critical role of play testing in MIAI pipelines. The paper concludes with recommendations for future research and potential improvements to MIAI pipelines.

2. Background

2.1. Procedural content generation

PCG has emerged as a highly active research field over the past two decades, focusing on the generation of game levels, environments, and other virtual content. Originally rooted in academia, PCG is now widely used across the gaming industry, with popular titles like Valheim, Dwarf Fortress, Diablo 4, and Starfield relying on it for level generation. Beyond level design, PCG systems have expanded to generate text, stories, textures, and even entire games.

Despite its widespread adoption, the field still lacks standardized evaluation frameworks for assessing the quality and effectiveness of new PCG approaches[11, 12, 13, 14, 3] In Withington et al. [15] the authors address this challenge by developing a novel taxonomy of PCG evaluation methods and conducting a survey of recent research, analyzing current practices through this framework.

The authors identify a key limitation in their analysis: the treatment of Mixed-Initiative PCG systems, where human designers collaborate with AI during content creation. While they include an “Evaluated through Mixed-Initiative Creation” category, this classification is limited to evaluations involving human-in-the-loop interactions, where user data is collected either during or after the level design process. This gap underscores the need for more refined evaluation approaches tailored to the unique challenges of Mixed-Initiative systems and how PCG systems also not be evaluated in the same way as PCG systems.

2.2. Human Computer Interaction

HCI examines how users interact with software, hardware, and digital interfaces, aiming to create systems that align with human capabilities and needs. MIAI systems aim to integrate artificial intelligence into these frameworks. Shneiderman et al. [16] introduced a framework to aid in the development of digital interactive tools for creative problem-solving, a well-established area in creativity research (e.g.[17, 18]). In exploring the potential to enhance individual creativity through new tools, Shneiderman et al. [16] considered both widely used general-purpose and other design tools like text editors and spreadsheets and other specialized applications in architecture, graphic design, and engineering.

Before integrating AI into design tools, the need to research user interaction with Creativity Support Tools (CSTs) was already identified by the HCI community [19]. In his call for researchers to explore future developments in CSTs, Shneiderman et al. [19] also emphasized the evolving nature of tool evaluation and the importance such evaluation plays in establishing the value of CSTs to end users. The development of MIAI pipelines brings to the front the need to identify HCI and PCG methods of validation.

HCI researchers have implemented a broad spectrum of methodological approaches to validating pipelines. Quantitative methods mainly focus on analysing the outputs of systems or conducting surveys[4]. These quantitative approaches allow for the rapid testing of multiple systems in a relatively short period of time. In contrast, qualitative methods can provide insights into the users' thought processes and provide nuanced information about how the user perceives the CST. The deployed techniques include structured interviews, speak-aloud interviews, expert interviews and grounded theory[20].

With this in mind, it is important to note that the design of MIAI pipelines and their features is a generative process, with incremental contributions being added over time by many academics, for example, the use of level culling methods being integrated into MIAI pipelines[4, 20, 21] to help inform designers about the structural issues of the artifacts they are designing during their interaction with the MIAI pipeline, although this trend is not observed, or systematically analyzed through quantitative measures. These insights provided by experts align closely to research through design(RtD)[22], where the foucs of scholars is argued by authours such as Zimmerman et al. [23] and Stolterman et al. [24] should be making the "right thing" to advance research in the "right" direction.

2.3. Mixed-Initiative Artificial Intelligence

One of the many challenges that MIAI researchers face, is that of where and how does AI fit into complicated and creative processes. In Margarido et al. [25] the authors address the challenges in human-computer collaboration during game design, particularly focusing on the communication gap between human designers and computational agents. Some considerations for MIAI pipelines as a whole claimed that the role of AI in human-AI collaboration relies on the capability of AI [26] (i.e., AI as a teammate when AI performance is better than human performance and AI as external stimuli when AI performance is worse than human performance). In Melete, these considerations have been made using the Wave Function Collapse (WFC) algorithm, which allows designers to give Melete, the MIAI pipeline/agent in this case a simple design that can be rapidly iterated on by the agent to present possible designs to the authour which they may like. While the roles of AI emphasize the positive impact of AI in human-AI collaboration, Pandya et al. and Zhang et al. [27, 28] showed that human-AI collaboration can produce better outcomes when the AI contributes a better performance than the human. If the AI performs at the same level or worse than a human, it is likely to lead to less favorable results. While MIAI pipelines offer significant benefits for rapidly prototyping and tackling complex tasks, novice users are more prone to making mistakes. This underscores the need for experts to thoroughly evaluate MIAI pipelines before they are introduced or exploring concepts with novice users.

Some recent studies investigated the role of AI and the impact of AI on ideation in human-AI collaboration. Liao et al. [29] presented three potential roles of AI-based inspirations in ideation that are closely related to the interaction paradigm providing suggestions described in the following:

- AI as representation creation (providing inspirations by suggesting texts or images)
- AI as an empathy trigger (supporting the designer's descriptive thinking)
- AI as engagement (helping the designer avoid fossilization[In this context fossilization refers to the learning and repeated implementation of incorrect habits when it comes to writing/design] and perform typical design actions)

A short Taxonomy of MIAI systems was presented by Lai. et al[30] in which they present some of the previous works of MIAI systems and identify some of the core areas of interest for MIAI pipelines which included: 2D Images, 2D models, 3D Assets, Animation, Code, Dialogue, Game Design, and Level design. Melete as an MIAI pipeline focuses on level design, thus, we will focus on these types of MIAI pipelines. An example of the difference between game design and level design is "Evaluation of a Recommender System for Assisting Novice Game Designers,"[31] in which the authors introduce an AI-driven game design assistant that suggests game mechanics, whereas examples such as Sentient Sketchbook[20] focus on topology aspects of levels.

2.3.1. Testing MIAI in Game Development and the role of playtesting

The majority of game development tools such as Unity, Unreal, Godot, Scratch, etc., include playtesting functionality built into their design and as stated earlier play testing is an essential component in GUR[10]. In this sub-section we will present other MIAI system papers and explore their implementation and whether they have a playtesting functionality, in addition to how they were tested.

One of the most prevalent and arguably one of the keystone papers of MIAI systems, the Sentient Sketchbook[20] is a tool designed to assist game designers in creating levels. It simplifies the design process by using map sketches while automating playability checks, evaluations, and visualizations of key game play properties. Although the Sentient Sketchbook is the only other system that in later versions has a 3D visualization component, it does not have a method for testing the output of the interaction beyond visual inspection. That being said, it was one of the first academic explorations of MIAI pipelines for level design. To test the usability of Sentient Sketchbooks, a small-scale expert analysis was conducted.

Another example of MIAI in game design is 3Buddy[32], which is an MIAI tool designed to function as a digital peer, offering creative stimuli during level design rather than merely executing commands. It generates content using the Legend of Grimrock 2 level editor, which is a 2D top-down editor without playtesting integration. In order to analyze the HCI aspects of the tool, a user study was conducted, involving six participants who completed a four-question Likert scale questionnaire. While three participants were experts and the results were promising, the small sample size limits statistical inference. Additionally, the use of a Likert scale may have led to a loss of nuanced data, affecting the depth of insights gained from the expert participants. Papers such as Liapis et al.[33] make recommendations for questions that could be put towards asking qualitative research in MIAI pipelines. As an approach to validating the usability of MIAI pipelines both these approaches show promise; however, in our research we have opted for a qualitative approach.

Automated play testing in MIAI Automated playtesting presents two interesting areas of possible research in MIAI systems; the importance of providing information to the designer, and a possible avenue for allowing MIAI systems to directly interact with the designers actions.

One approach to validating the usability of MIAI pipelines has been to use automated playtesting to ensure that the quality of the levels that are being produced can be used in a video game context. Roposseum[4]is an example of how physics-based puzzle solvers can help ensure designers create playable puzzles for the cut the rope game, which is a 2D puzzle game for mobile phones.

Other examples of automated playtesting include MIAI systems such as Tanagra[34] used to design 2D platformers and has the unique ability to override the designers input to ensure level playability. In "Integrating Automated Play in Level Co-Creation"[21], the authors enhance the Morai Maker[35], a mixed-initiative level editor for Super Mario Bros, by integrating automated pathing and introducing

an A* reachability check to ensure level traversal. This is a similar approach to how the Sentient Sketchbook approached resource placement and in both examples an A*-based analysis to approximate difficulty/fairness was used.

Another example of automated playtesting is presented in the paper "Toward Co-Creative Dungeon Generation via Transfer Learning"[36]. The authors present another possible avenue for research with automated playtesting, where they address the challenge of acquiring co-creative training data necessary for Procedural Content Generation via Machine Learning (PCGML) systems, to effectively collaborate with human designers. The authors address this via helping users design levels for both Mario and Zelda style 2D games. To overcome the difficulty and time consumption associated with collecting such data, they propose approximating human-AI interaction data and employing transfer learning techniques. Examples of this can be found in "Mixed-Initiative Level Design with RL Brush"[37], where the authors introduce RL Brush, a level-editing tool for tile-based games that facilitates mixed-initiative co-creation by integrating reinforcement learning (RL) models to provide AI-generated suggestions during human level design. The tool is applied to the classic puzzle game Sokoban, a 2D Zelda style game allowing users to manually design levels while receiving AI-driven recommendations to enhance playability and complexity. This approach involves adapting co-creative knowledge learned from one game to facilitate dungeon room generation in another game, specifically focusing on co-creative Zelda dungeon room generation. To validate their MIAI tool automated playtesting to assess the quality of output which restricts the designer from playtesting the environment in the tool they are using to design with. Automated playtesting for information presentation is one area that we have currently not explored with Melete. However its' use in the academic space presents an interesting avenue for further research.

2.3.2. Large Language Models

The papers "Level Generation through Large Language Models" [38] and "Consistent Game Content Creation via Function Calling for Large Language Models" [39] explore the integration of Large Language Models (LLMs) into what can be considered MIAI pipelines for game development. Both systems generate 2D levels, with one focused on Sokoban puzzles and the other on top-down dungeon layouts. However, neither system incorporates a playtesting mechanism, meaning users cannot directly test the environments they design within these frameworks. While this remains an under-explored area in MIAI research, the use of LLMs presents a novel approach to content generation by enabling designers to input text-based ideas, which AI then translates into potential level topologies to support the design process.

Play testing included Anhinga[40], presents a unique approach to play testing, where users can play a selection of levels from the original Snakebird (a 2D Game) and request the Exhaustive Procedural Content Generation (EPCG) system to suggest the single best modification to the current level based on an evaluation function. While the system qualifies as mixed-initiative PCG, the authors note that it is not yet fully co-creative due to limitations in human-AI interaction. Whereas Lode Enhancer[41] and Kumaran et al.[42] present approaches to playtesting 2D games after the interaction process are completed. Germinate[43] is an example of playtesting of 2D games during the design process.

Many of the limitations presented by the paper mentioned above have been addressed by Melete. The key areas in which Melete differentiates itself are:

- 3D and 2D level creation
- Interaction loop
- Playtesting

Many of the systems discussed in this section have primarily focused on 2D level creation, limiting their applicability to more complex spatial design tasks. While Melete does support 2D level generation, this version extends beyond those constraints by leveraging MIAI for 3D level creation—an area that has received comparatively less attention. Additionally, prior systems often offer limited, "black-box"

interactions, where users are presented with opaque suggestions from the agent. In contrast, Melete emphasizes transparency and user agency: users actively guide the system by interacting with the Kernel and selecting from the agent's outputs, fostering a more collaborative design process. Moreover, many previous approaches neglect the iterative nature of real-world game development, which heavily relies on continuous playtesting and refinement. Melete is explicitly designed to support this iterative workflow, aligning with the practices emphasized in GUR and contemporary game development environments.

3. Melete

This section outlines the implementation of Melete, which consists of three key stages of user interaction. In the first stage, data entry, the user engages with the WFC algorithm by designing a kernel to train it, described in section 3.1. Next, the user enters an interactive loop with the algorithm's output, where they can refine the design, adjust elements, load assets in real time, and playtest their work. In the final stage, the user can save the completed design. Figure 1 illustrates Melete's overall structure.

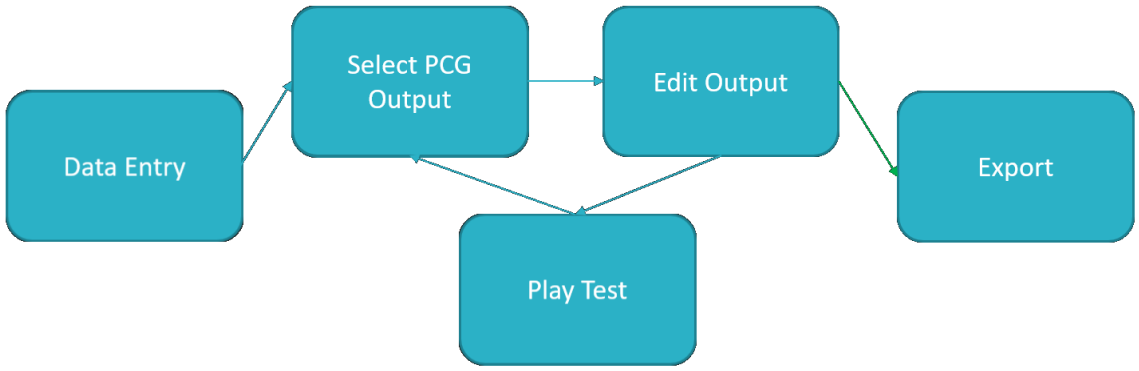

Figure 1: Melete Overview

3.0.1. Data Entry

When the designer begins using Melete, they are presented with an empty "Canvas," which functions as the data input section for the WFC algorithm. Through this user interface, the designer can populate the canvas with gameplay semantics.

To populate the canvas with game-play semantics, the user interacts with the canvas within the Unity editor and a predetermined palette is used to fill the Canvas.

Melete's implementation of WFC is distinct from other examples of the algorithm. In most cases, WFC represents concrete data, with each tile corresponding to a specific component of an artifact. In contrast, Melete abstracts this information, using colors to represent data instead. This approach allows gameplay semantics to be embedded directly into the data representation. Although users are not required to understand the theoretical foundations of the WFC algorithm, it will help the user manipulate the output of the algorithm. Despite this, the flexibility of Melete makes it an effective example of an MIAI pipeline for rapid content development, enabling users to engage with it from a genre-agnostic perspective. Its abstract representations are highly versatile, applying to both 2D and 3D games, and can incorporate gameplay behavior scripts, such as arrays of 3D models programmatically instantiated during the interaction cycle.

In the current iteration of Melete, the gameplay semantics are constructed offline, which means users cannot edit the underlying data representations during run-time. The data present in the palette represents:

- Orange: Areas outside of buildings.

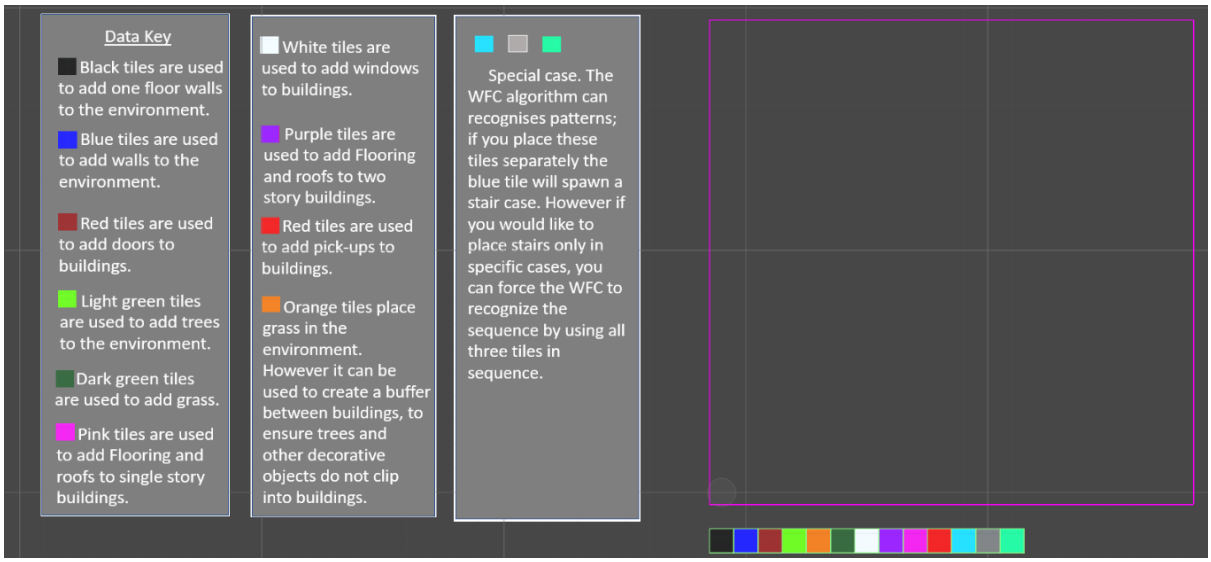

Figure 2: Melete Empty Data Set

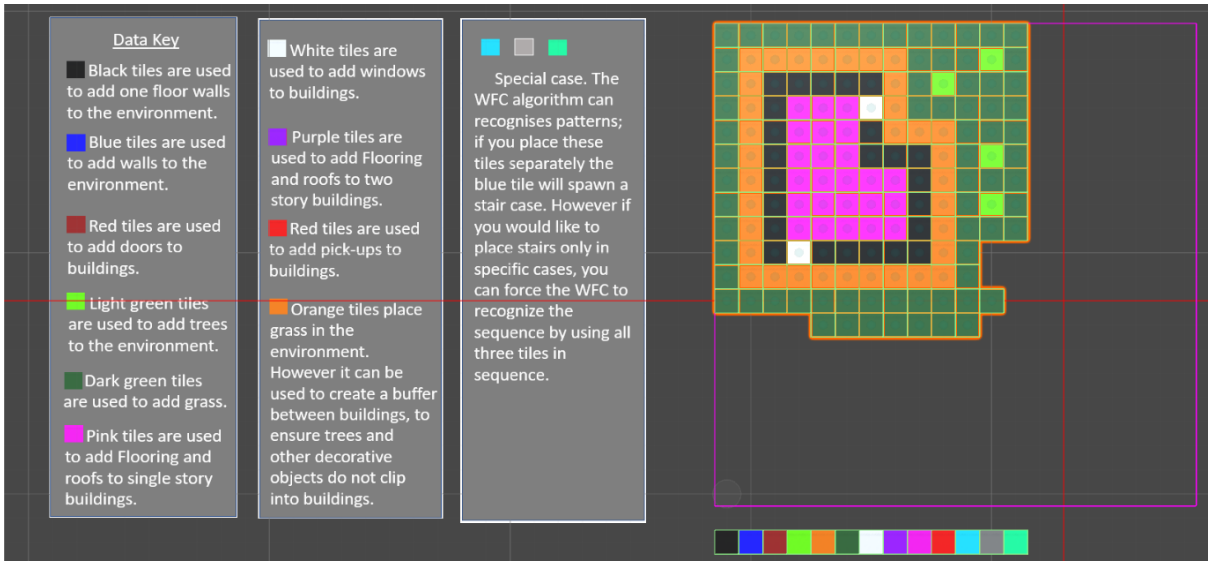

Figure 3: Melete Entering Data

- Dark Green: Grassy areas.
- Light Green: Trees.
- Black: Single-story building walls.
- Blue: Two-story building walls.
- White: Windows.
- Pink and Purple: Wood and concrete floors for the interior of buildings.
- Maroon: Entrances and exits.
- Red: Power-ups.
- Teal, Grey, and Cyan (combined): A special case scenario recognized as a staircase or ramp. Melete only identifies this combination as such when all three colors appear together.

Once the user completes their initial design, this information is sent to the WFC algorithm for analysis. An interesting feature of Melete is its ability to recognize special-case scenarios and place objects accordingly. This leverages the WFC algorithm's feature-preserving behavior from the training data. Designers can use this to create specific structures, like staircases, by encoding these features

in the training data. For instance, a combination of cyan, grey, and teal can signify where a staircase should appear. In figure 3, the system identifies this special-case scenario and places a staircase in the environment.

This palette is derived from the palette used in the data entry stage of Melete. Since the palette data is independent of the WFC algorithm, this allows for a genera-agnostic approach to PCG and game design, allowing the user to represent not only 2D and 3D environments but also depending on the user's needs different genres of games, for example, top-down Legend of Zelda style 2D games or in this case a third person battle royal style environment.

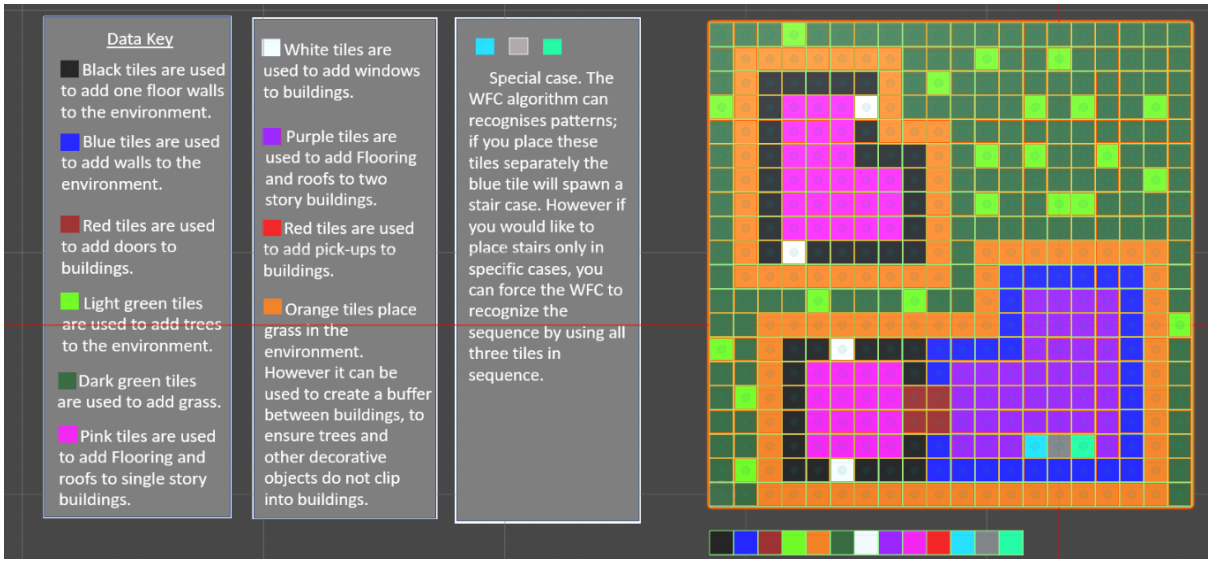

Figure 4: Completed Data Input

3.1. Wave Function Collapse

The WFC algorithm is a constraint-solving technique that uses user-defined generative grammar to generate images. Originally implemented Maxim Gumin, described in[44] WFC is inspired by the quantum mechanical WFC formula, as presented by Schrödinger and further explored in Max Born's research on quantum mechanical systems [45]. However, as far as the implementation of WFC in computing the algorithm itself servers as a constraint solving algorithm [46].

The algorithm requires an input image provided by the user before the preprocessing phase. Which in the case of Melete is the data input portion It then analyzes this image and generates similar outputs based on the input data. figure 5 illustrates this process, showing three input images on the left, their corresponding example outputs on the right, and the constraint-solving process in between.

The use of WFC for programmatically generating content can be divided into two distinct phases. The first is the pre-processing phase, where an image provided by the user is analyzed and interpreted into data that the algorithm can use to generate content. The second is the generation phase, where the extracted information is used to create new images similar to the input.

During the pre-processing phase, the input image is divided into small square sections, typically using a value of two for the square size. These sections are then used to create tiles, which are groups of pixels representing small portions of structural information from the input image. These tiles are cataloged, weighted, and assigned rules, forming the foundation for content generation in the next phase.

To create tiles, the WFC algorithm compares all possible square sections of the input image based on their pixel information. Each tile is assigned a name derived from its pixel composition, and squares are rotated to account for all possible rotational transformations. These transformations are added to the catalogue regardless of whether they appear in the original input image. Once all tiles have been named

and catalogued, duplicate tiles with identical pixel information are removed from the catalogue and instead assigned weighted values, increasing the likelihood of their occurrence in the generated output.

After defining all possible tiles from the original image, the algorithm assigns rules that determine how tiles can connect to one another. To verify connectivity, the WFC algorithm overlaps each tile with every other tile in the catalogue, assessing compatibility based on the number of possible offsets for that tile.$((2(n-1)+1)^2)-1$.

If the pixel values match in the overlapping areas, a placement rule is established for that tile. Once all necessary information about the input image has been gathered, the WFC algorithm moves on to the generation phase. This phase begins by selecting a random point within the output grid and placing a randomly chosen tile from the catalogue at that position. Once the initial tile is placed, the algorithm starts filling in the rest of the output.

To track which tiles need to be placed and which have been resolved, the WFC algorithm stores coordinate positions in a Boolean array. As tiles are placed, their corresponding Boolean values switch from true to false. This process helps the algorithm monitor the outer edge of the solution while also identifying tiles that have been fully surrounded by correctly placed neighbors. The algorithm then focuses on solving the remaining grid by analyzing tiles that have been placed and have at least one neighboring true value. This approach ensures that the algorithm identifies the tile with the lowest number of possible solutions.

To determine the tile with the fewest placement options, the WFC algorithm checks each tile's rule index and compares it to the values of surrounding tiles. With each tile placement, the algorithm both solves that tile and reduces the number of acceptable rules for adjacent tiles.

In rare cases where an unsolvable position arises, local backtracking can be applied. The algorithm identifies problematic tiles with neighboring true values and refers to the Boolean array to remove all false values adjacent to a true value. This backtracking process allows the WFC algorithm to adjust the solution and continue resolving the output. The algorithm terminates once no further valid placements remain.

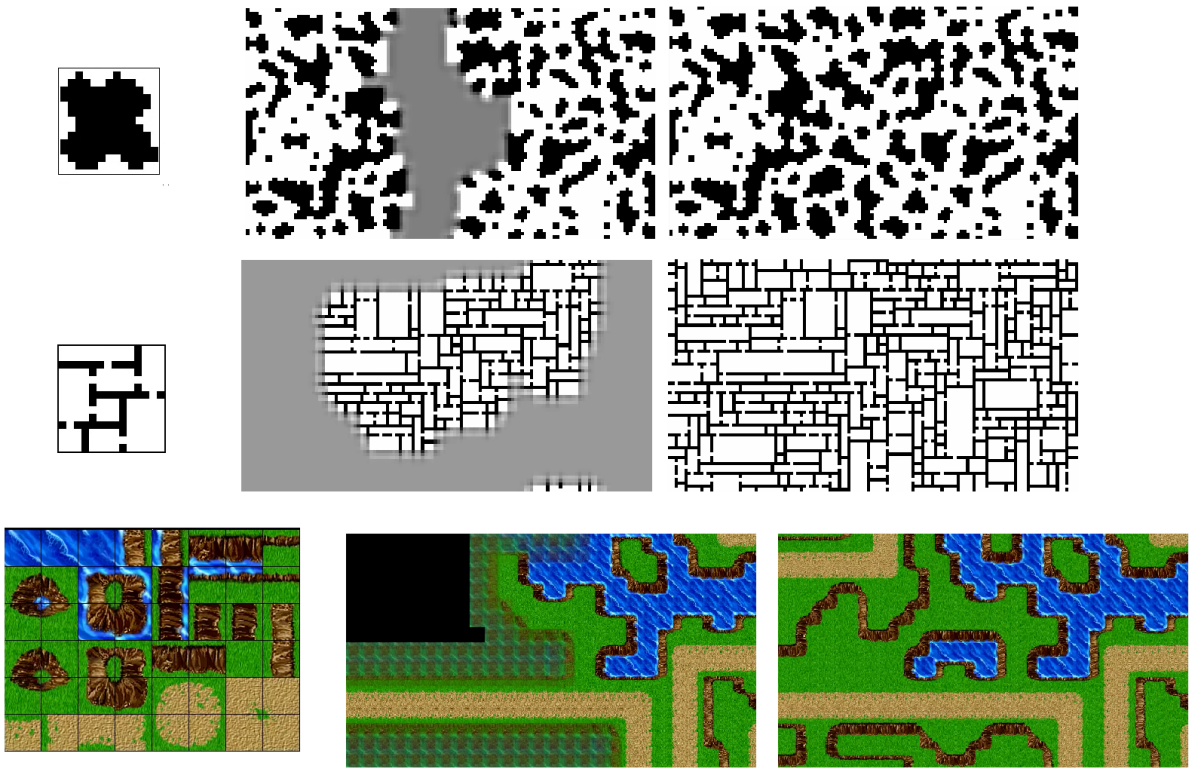

Figure 5: WFC Example Input and Output

WFC is influenced by the computer graphics technique known as texture synthesis. While texture synthesis methods are typically used to repair and correct digital images, WFC differs in that it does not merge or average adjacent pixels. This distinction allows it to be a useful component of an MIAI system such as Melete as it preserve gameplay semantics embedded within the training image.

The WFC algorithm takes a probabilistic approach to procedural content generation (PCG). While other probabilistic PCG methods exist, they differ from WFC in their implementation. Traditional systems rely on programmatically defined generative grammar, whereas WFC enables users to visually create a representation of generative grammar. This grammar is influenced by both spatial relationships and the probabilistic proximity of other rules within the input data, following the principles of the Wave Function Collapse formula.

3.1.1. Data Input and preprocessing

Before the WFC algorithm can generate output images, it requires user-provided data. During the data input stage, the user interacts with a blank canvas and a palette. The palette contains game objects, which serve as templates in Unity and are used by the algorithm to construct the final output.

In the example presented in figure 2, The palette below the data entry section has the following tile types: *Orange, Dark Green, Light Green, Black, Blue, White, Pink, Purple, Maroon and Red.*

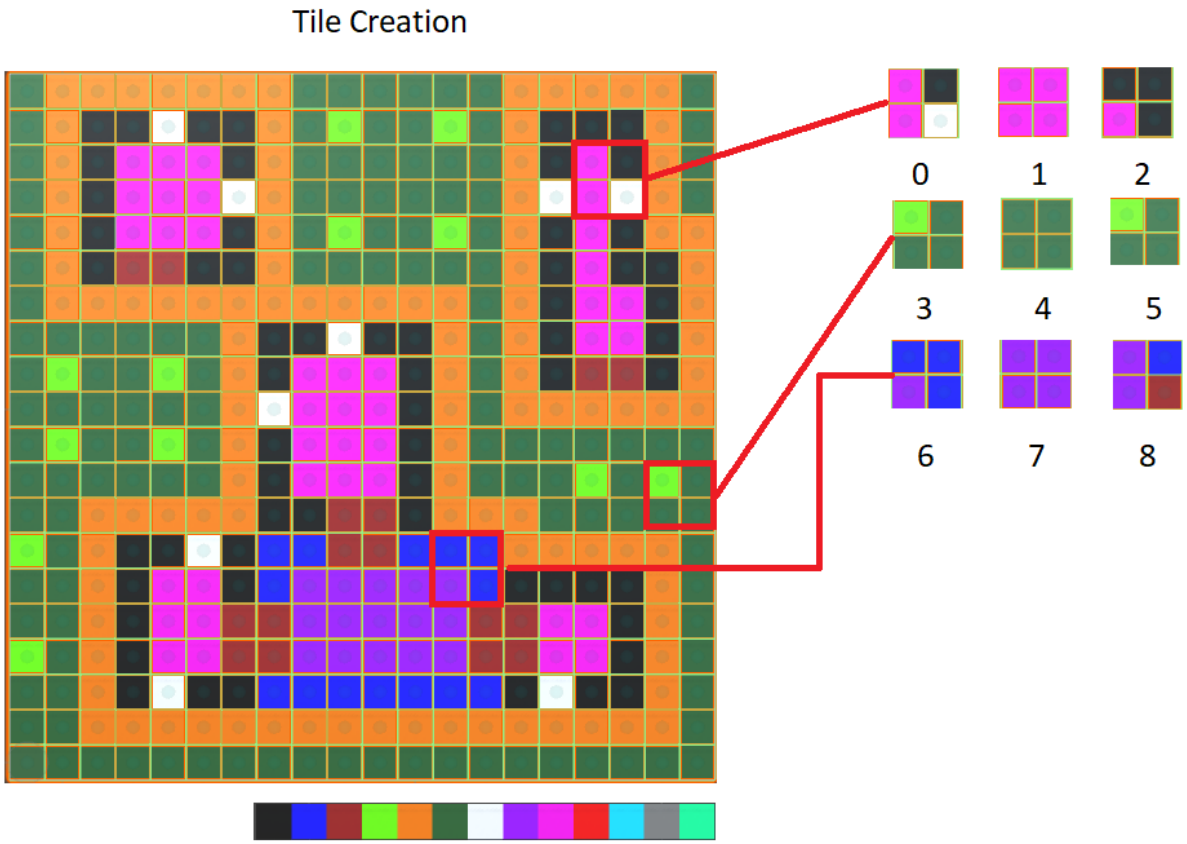

Figure 6: Tile Creation

After the user has defined the tile-set size and creating the training data, the WFC algorithm then goes through the map and makes the tile-sets and assigning each of the tile-sets into an array. It is important to note that in the figure 6 the tile-set 5 is a duplicate. The WFC does not update the array of possible patterns with duplicate tile-sets.

3.1.2. Creating Adjacency Rules

From the array of patterns created, the BuildPropagator() function is used to create an index data structure; this index is used by the system to describe how tile-set should fit next to one another.
'0' : ['1','2',"]
'1' : ['0','2',"]
'2' : ['1','2',"]
'3' : ['4',"",""]
'4' : ['3',"",""]
...
This example is quite rudimentary/crude. However, it does illustrate the basic principle. If two tile-sets can overlap one another, this information gets added to the index data structure stored as a valid rule.

3.1.3. Generation Phase

During the WFC generation phase, grid locations for the output grid; are continuously selected and assigned. The user specifies the size of the output grid, and as the constraint solver works through the grid, it keeps track of the tile-sets it has placed and their adjacency rules. Also, marking the areas of the grid that have had tile-sets assigned to them and storing grid locations that need to have tile-sets set.

The way this is achieved is by programmatically storing this information in a table. The variables stored in this table are Boolean values; these values represent whether or not the algorithm has assigned a pattern to a location within the output. The WFC output grid where true statements represent areas that have not had a tile-set assigned to them, and false statements have had tile-sets assigned.

As tile-sets are selected and placed within the WFC grid output, this action serves two purposes. It reduces the unassigned variable's index bringing the WFC algorithm closer to zero entropy and, in turn, reducing the possible choices that can connect to the existing tile-sets which have already had tile-sets assigned. In the event, the WFC algorithm encounters a conflict. It can locally backtrack on its output; instead, a global reset occurs.

As the WFC algorithm is filling in its output grid, it is simultaneously identifying unsigned tiles with the lowest entropy, which refers to the lowest possible combinations of tile-sets that will fit in a given position on the output grid. The grid location with the low entropy cost identified by the Observe() function is selected. This function then chooses a tile-set that will fit into that position, given the weighted average of tile-sets presented in the source data.

After the observation function has chosen and placed a tile-set, the WFC output must be updated. After placing a tile-set, the WFC algorithm then ensures that the placement of this new tile-set does not break existing rules; it does this by checking the valid rule array in the Propagation() function.

The propagation function does this by looping through all the output locations that have not reached zero entropy until there are no locations left to update. Whilst it is going through each of the locations on the output grid, it checks all of the neighbors of the currently selected location to ensure that the pattern placed is still a valid option. If it isn't, it flags that location for updates and removes its current patterns from the possible options for that location.

As stated earlier, with the placement of each tile-set, the 'entropy' or left over tiles of the output area in turn reduces. Once the entire output grid has reached zero entropy or there are no tiles left to solve, the system is ready to produce the final output.

3.1.4. Interaction Loop

Once the data entry stage is completed and the initial kernel entry is finalized, Melete can use the WFC algorithm to generate outputs. The user then enters the interaction loop, which can be divided into four distinct stages: output selection, output mapping, output editing, and output testing. While in the interaction loop, the user can switch between these stages at any time, allowing for a dynamic and iterative design process.

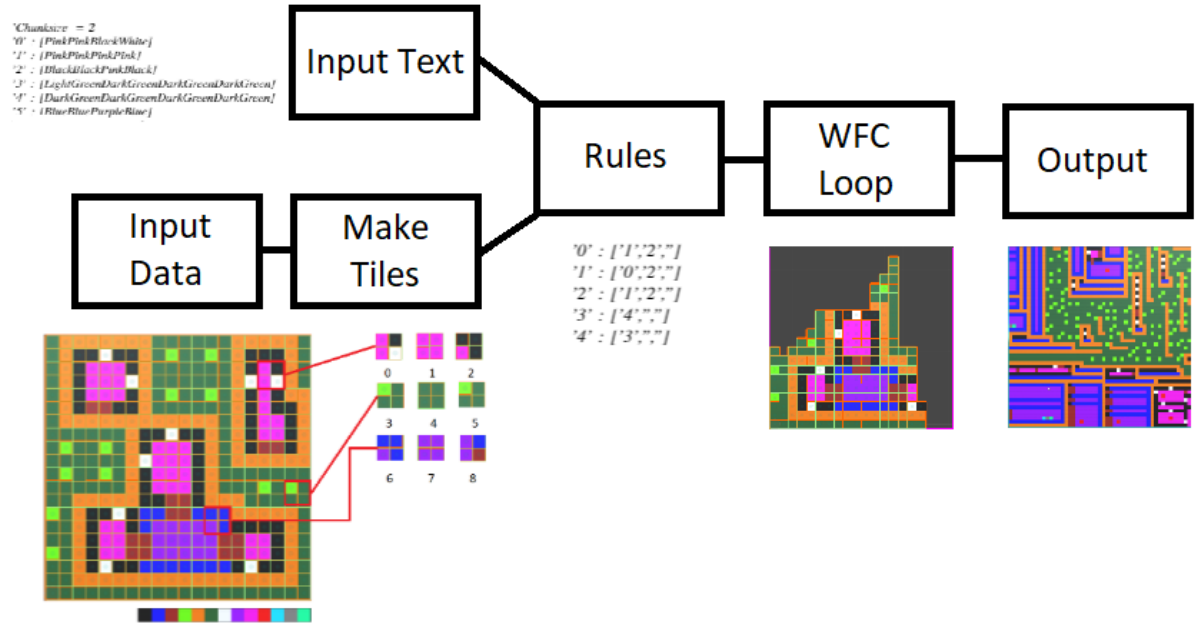

Figure 7: WFC Overview

3.1.5. Output Selection

During the output selection stage of the interaction loop, the user can cycle through multiple outputs generated by the WFC algorithm, each based on the original kernel created during the data entry stage. The user can review these outputs and save the ones they find most appealing for further refinement.

3.1.6. Output Mapping

Once the user chooses an output that matches the type of level they want to create, they move on to the output mapping stage in Melete. In this stage, Melete fills in the selected output with gameplay elements and environmental objects. As shown in figure 9, the system generates a preview of the environment, placing trees, grass, and other game objects. These are scaled, rotated, and positioned based on the user's input and chosen design style. Melete also applies textures to the environment according to the user's settings. This stage gives the user a complete view of the level, helping them better understand how the gameplay might look and feel.

3.2. Output Editing

In most cases, PCG algorithms may generate outputs that are close to what the user envisions but may still contain errors. To address this, the user can disable the output mapping stage and switch to the output editing stage. This stage allows the user to fine-tune the output to meet their needs while eliminating much of the repetitive work involved in level design.

During the output editing stage, the user interacts with a palette UI located on the right-hand side of the output. This interface enables them to directly edit the WFC output. Additionally, the user can modify the kernel used by the WFC algorithm based on their updated design. These changes not only refine the current output but also serve to improve future generations of the WFC algorithm, aligning them more closely with the user's evolving design preferences.

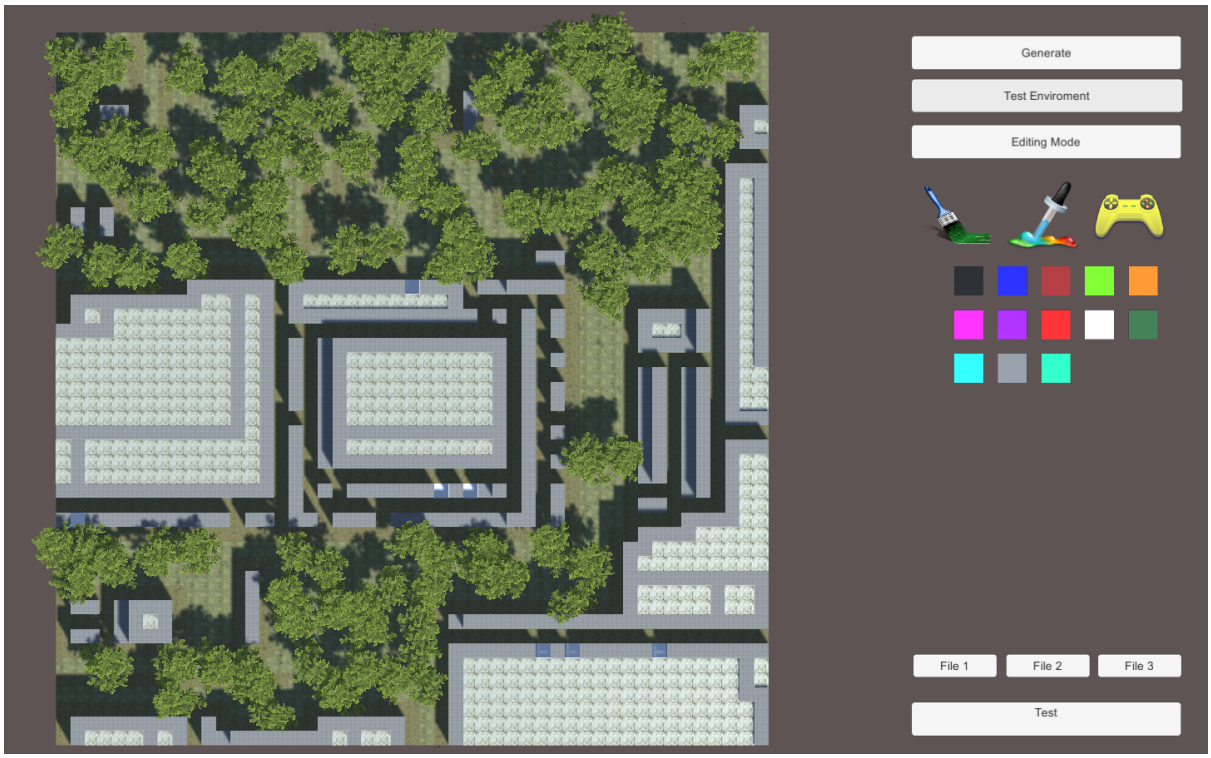

Figure 8: Birdseye environment

3.2.1. Output Testing

In this stage, the user can play-test and explore their design using a player character, seamlessly switching between editing and testing. Gaining this insight before releasing their design helps facilitate an iterative design process.

The dynamically updated 3D view enables immediate testing of modifications, providing an interactive way to explore and evaluate the environment. At any point during the interaction loop, the user can choose to view the design from the player's perspective, ensuring a more polished and player-centric result.

In this version of the training data, we have used a combination of cyan, grey, and teal to indicate where a staircase should be placed. As shown in figure 10 above, the system has recognized a special case scenario, resulting in the placement of a ramp or staircase within the environment.

In figure 11, we see that part of the special case scenario has been placed within the environment. However, since the cyan element is not included in the map output, no stairs or ramp was generated.

Melete is an innovative tool designed to assist human designers in the level design process. By utilizing the WFC algorithm and a unique user interface, Melete enables users to create a 2D representation of a level, which is then analyzed and stylistically replicated by the algorithm. Users can further manipulate the WFC output to refine the design. The final step in Melete is the export process—if the user is satisfied with the level, they can export an XML document containing the map structure. This document can then be read by another system, allowing seamless integration of the designed map into a game.

4. Method

When reviewing existing literature on MIAI pipelines, two distinct approaches emerged: evaluating the PCG system's output and conducting user studies to assess user experience. Given our focus on the role of playtesting within the interaction loop, we determined that expert analysis would provide the most nuanced feedback regarding participants' experiences with Melete.

Figure 9: Play Testing

To design the questions, we reviewed existing studies, though many MIAI systems mentioned user studies without reporting the specific questions used. We tailored our questions based on the available literature, resulting in the following:

- What did you like about the system?
- What did you dislike about the system?
- How easy was it to use the tool?
- Did you have enough control over the design process?
- Was the play testing mode useful?

To explore playtesting's impact on the MIAI pipeline, we presented participants with two distinct design challenges to highlight its role during the design process. We chose one top-down perspective and one over-the-shoulder perspective, aligning with the design considerations for battle royale and real-time strategy (RTS) games, respectively. These genres provided enough contrast to reveal how playtesting influenced design workflows.

To reach a consensus among experts, we conducted a semi-structured focus group after the Melete sessions. The follow-up questions were:

- Do you think this is a useful tool?
- What was your workflow with the system?

For consistency, experts were introduced to Melete through a 15-minute recorded lecture and tutorial. This session explained MIAI systems, provided an overview of the wave function collapse (WFC) algorithm, and outlined its core functionality. After which they where given 15 minute sessions with Melete to tackle each of the design challenges presented above. The primary research was available during this time to answer any questions the participants had and provided the participants with a hand out that included the main system controls.

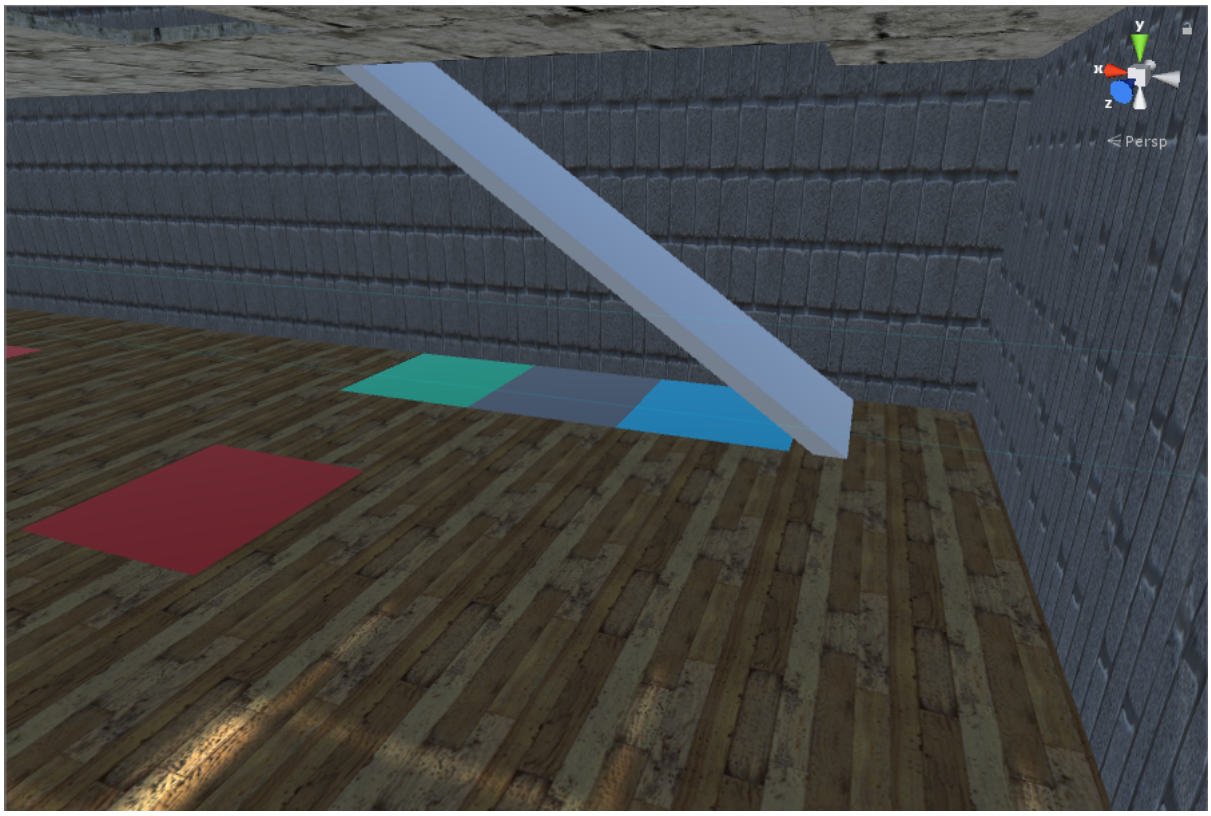

Figure 10: Object Rules

4.1. Participants

The participants selected for this experiment were two experts academic designers, and one industry expert designer, and two academic experts in AI.

5. Results and Data analysis

In this section we will highlight some of the areas of interest from the results of our expert analysis of Melete and our focus group results analysis.

5.1. Question 1: What did you like about the system?

5.1.1. Creativity

One of the most notable similarities across both sessions was the emphasis on creativity. Participants described the emergent structures generated by Melete as compelling and inspiring. Participant 1 encapsulated this sentiment by stating that "the emergent complexity of some environments can make inspiring designs," reflecting how the procedural generation capabilities of the system provided new design avenues. Similarly, Participants 4 and 5 acknowledged the value of the "messy" terrain and the satisfaction derived from interacting with procedural content generation (PCG). The narrative across both design sessions suggests that Melete is perceived as a catalyst for creative exploration, enabling designers to uncover novel configurations that may not have been considered through conventional design processes.

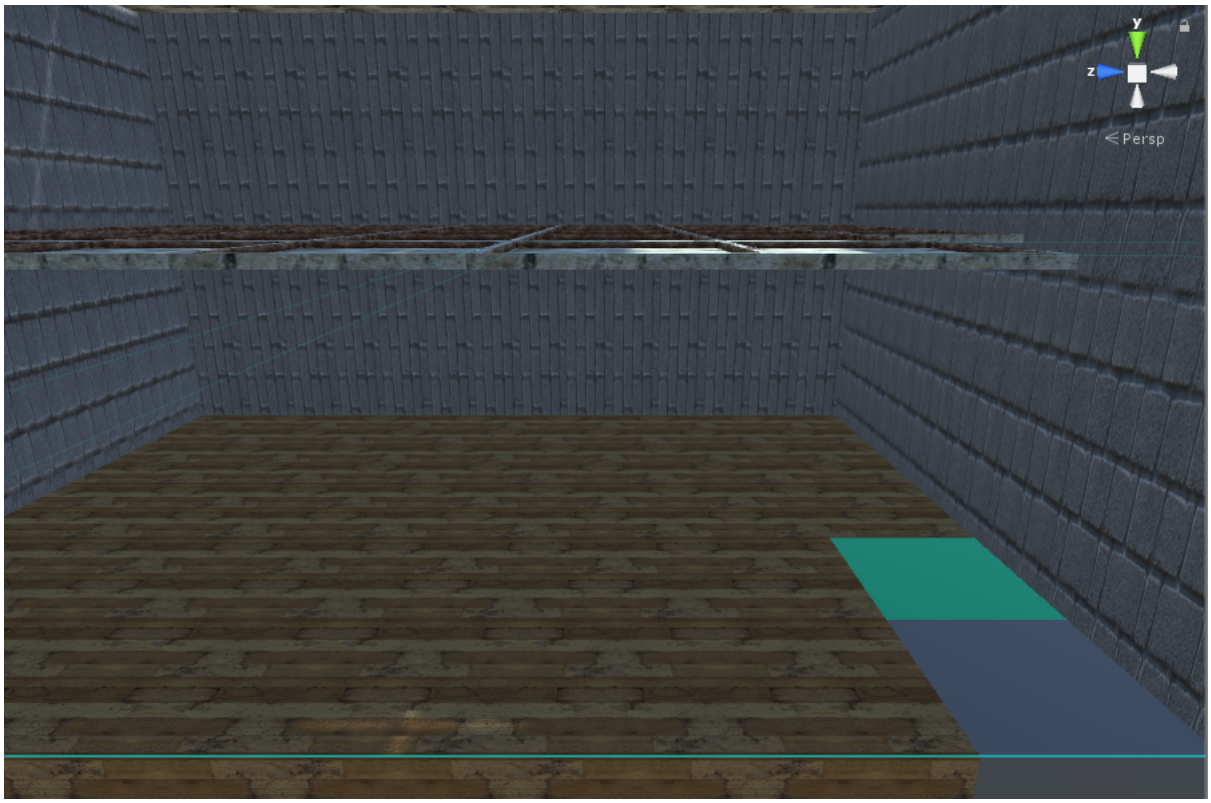

Figure 11: No pattern recognized

5.1.2. Collaborative

Another recurring theme was the perception of collaboration between the designer and the system. Participant 5 in both sessions underscored the experience of co-creating with Melete, describing it as a "nice feeling of collaboration." This reflects a broader narrative in which the designers did not merely view Melete as a passive tool but rather as an interactive partner in the creative process. The system's ability to generate unexpected yet innovative results contributed to this collaborative sensation, reinforcing the idea that Melete serves as an extension of the designer's thought process rather than just a utility.

5.1.3. Design Iteration

Both sessions underscored the utility of Melete for rapid prototyping and iteration. In the BR session, participants appreciated the tool's ability to facilitate prototyping and quickly modify generated maps. Participant 4 noted how useful it was for generating prototyping environments, while Participant 5 highlighted the ease of editing generated maps. The RTS session also emphasized real-time editing capabilities, with Participant 4 remarking that "being able to do live editing was cool." This suggests that, regardless of the game genre, Melete is valued for its ability to streamline the level design process and offer real-time adjustments that enhance workflow efficiency.

5.1.4. Novelty and Inspiration

The perception of novelty in Melete's output emerged as a key aspect of its appeal. In the BR session, Participant 1 emphasized how wave function collapse produced "compelling and often novel" spaces. Similarly, Participant 2 recognized its potential for generating non-essential tile-based positional objects. The RTS session echoed this sentiment, with Participant 3 highlighting the flexibility of tile colouring and diversity of tile types. This suggests that Melete's adaptability across different game genres fosters

Study	Q1	Q2	Q3	Q4	Q5
Lode enhancer[41]	What did you like about the system?	What did you dislike about the system?	How could enhancing/scaling levels improve level designers work flow and fulfilling their own design goals	How did you use the persistence function?	What other AI/ML functionality would you like to see in a tool like this.
Procedural generation in educational games [42]	How would you rate the ease of generating new narratives using the tool?	How engaging and unexpected did you find the generated game?	To what extent did the interaction in the game accurately represent and respond to your intent	Overall how pleased are you with the game generated by the tool	-
Creative sketching partner[47]	Did you see any relation between the choices you had for visual and conceptual similarity slide bar and your creativity?	Was it more inspiring when the system output was less/more similar to your input?	In which iteration where you inspired most, can you explain why	Do you have any comments on the interaction of a computational partner to inspire you	-
Boosting computational creativity with human interaction in mixed-initiative co-creative tasks [33]	To what extent is the produced item a high quality example of its genre?	To what extent is the produced item an example of the artefact class in question	To what extent is the produced item dissimilar from what is currently created by the human user?	To what extent would the produced item be of use to end users?	How does the produced item match the human users frame?
A mixed initiative casual creator[43]	What is your overall impression of the tool?	How easy was it to use the tool?	Were you able to use the tool without unnecessary effort?	Did you feel you had enough control over the generated games?	Was it fun?
Alone we can do so little[48]	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported
Friend, Collaborator, student, manager: How to design an ai-driven game level editor[35]	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported
Towards pattern based mixed initiative dungeon generation[49]	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported
Sentient Sketchbook [20]	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported

innovative level design, presenting designers with unique configurations that might not emerge through traditional methods. Both sessions highlighted how Melete's unpredictability contributed to creative inspiration. In the BR session, participants were particularly intrigued by how the system produced unconventional structures—such as a two-story building with excessive staircases—that they would not have conceptualized independently. Similarly, in the RTS session, Participant 4 appreciated the "messy" terrain that suited a destroyed urban battlefield setting. Participant 5 described how Melete "throws curveballs in its outputs" that stimulate creative thinking. This theme underscores the role of Melete in broadening design perspectives and encouraging designers to explore unconventional spatial arrangements.

5.2. Question 2: What did you dislike about the system?

5.2.1. PCG

Participants across both sessions raised concerns about the effectiveness and efficiency of the PCG algorithm, emphasizing the need for significant manual intervention to achieve believable environments. Participant 1 noted in the first session that substantial ”re-generation and artifact cleaning” was necessary, indicating that the automated generation process produced inconsistencies requiring correction. In the second session, they further elaborated that a ”small kernel size” limited the algorithm’s ability to replicate larger formations beyond immediate adjacency. Similarly, Participant 2 expressed dissatisfaction with the tile/grid-based nature of Melete, stating that ”Currently, it’s very tile-based in all three axes.” This observation suggests that the rigid, grid-like Melete’s approach constrained organic or free-form level design, which could reduce immersion and variety in both BR and RTS experiences. Together, these insights highlight the inefficiencies in the Melete, which ultimately hindered automation and increased the need for manual corrections. Reinforcing the idea that MIAI Pipelines do have a place in design as purely “Online PCG” systems may not be sufficient.

5.2.2. User Interface and training

The usability of the UI emerged as a significant point of friction among participants. Both Participant 4 and 5 found the interface unintuitive and cumbersome. Participant 4 described a ”disconnect in the UI,” indicating that the interaction flow between different states was problematic. Participant 5 echoed these sentiments, stating that the UI and input controls were ”a little fiddly” and required effort to remember the correct sequences to perform desired actions.

5.2.3. Navigation and Camera Control Issues

Difficulties with navigation and camera controls further compounded usability challenges. Participant 5 described the camera and movement controls as ”sluggish,” noting that they interfered with their ability to inspect levels effectively. This sluggishness hindered both the design process and playtesting, making adjustments and refinements slower than necessary. Participant 4 suggested that incorporating a minimap for better navigation would be beneficial. Additionally, RTS participants pointed out issues with playtest agent movement, with Participant 3 stating that ”the movement of the agent in playtest is not very accurate.” This inaccuracy diminished the reliability of play testing in that genera, making it difficult to evaluate gameplay mechanics effectively.

5.3. Question 3: How easy was it to use the tool?

5.3.1. Ease of Use

In the BR session, Participant 1 noted, ”However, as a concept, the cyclic iteration process could become easy to prototype spaces with.” Similarly, Participant 5 stated, ”Fairly easy - I understood the concept quite quickly. Most of the issues came from trying to remember the interface elements and pressing the wrong buttons.” Despite understanding the process, they faced difficulties with interaction mechanics.

In the RTS session, similar sentiments were echoed. Participant 1 acknowledged, ”The current layout is not intuitive, but the tool itself was straightforward enough to work with.” However, challenges arose due to the scale of the sample map and limited tile options, leading to misaligned structures. Participant 2 further criticized the UI, stating, ”A bit cumbersome - a single pallet that was selectable would be preferred over separate select buttons, index, and selectable palette.”

5.3.2. UI and Workflow Challenges

In the BR session, Participant 4 commented, ”There could be some more easing of the user experience with tooltips, less complex inputs, and fewer clicks in between different desired actions.” Similarly,

Participant 1 stated, ”The current user interface is hard to use, with several steps and often unintuitive interfaces.”

The RTS session participants had similar complaints but focused more on specific aspects of the UI. Participant 1 noted, ”The current layout is not intuitive,” while Participant 2 emphasized the need for a more efficient tile selection system: ”A bit cumbersome - a single pallet that was selectable would be preferred over separate select buttons.”

5.3.3. User Adaptation Over Time

Despite these challenges, both groups demonstrated adaptability as they became more accustomed to the tool. Participants developed workarounds to streamline their workflows and mitigate usability issues. In the BR session, Participant 5 remarked, ”Most of the issues came from trying to remember the interface elements and pressing the wrong buttons.” Over time, they found ways to navigate these difficulties.

A similar pattern emerged in the RTS session, where Participant 5 stated, ”The concept is simple, and I’m starting to get used to using it (tactics for feeding it, etc.), but some of the fiddly interaction aspects are a bit more difficult than needed.” This demonstrates that while initial usability was an obstacle, continued use led to increased familiarity and efficiency.

Participant 4 suggested, ”An RTS probably needs two distinct enemy bases that the units originate from, but having to manually add these into the map after random generation was harder than just doing this by hand.” The ability to combine hand-designed bases with procedural generation was identified as an optimal solution. In contrast, the BR session emphasized difficulties with navigation and workflow efficiency. Participants suggested improvements such as mini-maps and smoother transitions between perspectives.

Participant 4 noted, ”There could be some more easing of the user experience with tooltips, less complex inputs, and fewer clicks in between different desired actions.” Similarly, Participant 5 observed, ”Most of the issues came from trying to remember the interface elements and pressing the wrong buttons.” These comments indicate that BR participants were primarily concerned with refining the user experience rather than genre-specific functionality.

Participant 4 stated, ”I understand it’s a prototype, so I’m basing my evaluation on that: there could be some more easing of the user experience with tooltips, less complex inputs, and fewer clicks in between different desired actions.” This suggests that they were willing to overlook some issues, provided they could be resolved in later iterations.

5.4. Question 4: Did you have enough control over the design process?

5.4.1. Control Over the Design Process

Participant 1 from the RTS session described the outcomes as ”substantially variable,” leading to unintended elements such as long walls stretching across the map. This unpredictability constrained their ability to fully utilize the tool’s potential. On the other hand, BR participants acknowledged similar artifacts but found them more manageable within their design framework. Participant 1 from BR mentioned that while artifacts existed, the ”wave function collapse algorithm” still allowed for some consistency in design replication, reinforcing the idea that the tool offered more usable outcomes in this context.

5.4.2. PCG

Participant 4 articulated this challenge: ”Not being able to think about things like resource placement or cover was a bit odd for the task of designing an RTS map.” This sentiment underscores a key limitation in procedural generation: while it can inspire creativity, it lacks the ability to automatically generate maps without requiring significant post-processing adjustments.

Conversely, BR participants perceived the tool as more adaptable to their needs. The design requirements of BR maps, which often emphasize large-scale layouts with scattered elements rather than precise resource placements, allowed the procedural system to function more effectively. While some participants noted issues with the algorithm, they generally felt they had enough control over adjustments to make the results useful. For instance, Participant 4 noted that the tool was adequate for "grayboxing a level," even if additional features like power-ups and complex structures were needed. This perspective suggests that MIAI tools such as Melete may be more effective in early-stage design.

5.4.3. Control

The perceived level of control over the design process was another differentiating factor. BR participants overwhelmingly felt they had enough influence over the final output. Participant 5 stated, "I fed the thing and had the final say on the output," reinforcing the notion that despite MIAI system's inherent randomness, they still felt like the designers in control of the process. In contrast, RTS participants had a mixed reaction. Some, like Participant 5, saw the tool as merely an inspiration mechanism rather than a rigid determinant of the final design, comparing it to "other techniques for driving creativity, like randomization." However, others, like Participant 2, simply answered "no" when asked if they felt in control.

5.5. Question 5: Was the play testing mode useful?

5.5.1. Play-test Navigation

Participants in both sessions found playtesting effective in identifying map artifacts and rendering inconsistencies. BR Participant 1 stated that playtesting allowed for rapid prototyping. Whilst, in the RTS session, the same participant confirmed that playtesting effectively outlined artifacts and outliers, helping refine map structures. Both genres benefited from playtesting's diagnostic capabilities.

5.5.2. Perspective

BR participants found playtesting essential due to their game's reliance on movement, exploration, and 3D space awareness. Participant 5 described playtesting as "essential" despite usability challenges, while Participant 4 stressed the need for a minimap to maintain spatial awareness during level design. In contrast, RTS participants found it less relevant since their game is typically viewed from a top-down perspective. Participant 4 stated that ground-level playtesting was "less useful" in an RTS context, as the genre rarely adopts that perspective.

5.6. Expert Focus Group discussion

5.6.1. Question: do you think this is a useful tool?

During the focus group the participants where asked: "Do you think this is a useful tool?" Participants generally recognized the tool's potential, particularly in mixed-initiative design, and iteration-based workflows. However, they also identified key limitations, especially in its current prototype form.

Participant 1 highlighted the significance of the tool's integration with a mixed-initiative approach, noting that its iterative feedback system could be a core innovation. However, they also pointed out that artifacts frequently emerged, making it sometimes more practical to design elements by hand rather than relying on the tool. Similarly, Participant 2 acknowledged its potential in rule system creation but expressed concerns over its lack of flexibility in handling structured map designs.

Participant 4 provided additional insight, emphasizing that while the tool was effective at generating randomized content, it struggled in structured environments, such as RTS maps where controlled layouts are necessary.

5.6.2. PCG

Participants found that the tool was more useful for certain game genres than others. Games that benefit from randomized elements, such as RPGs, dungeon crawlers, and BR maps, were seen as more compatible with the tool's capabilities. In contrast, RTS games, which require precision and structured layouts, posed significant challenges for the tool's procedural approach. Participant 4 noted that while the tool was useful for generating semi-randomized elements, it struggled with more formalized gameplay structures. For example, an RTS typically requires two symmetrical bases with controlled pathways, but the tool often produced chaotic noise instead of structured lanes. They suggested that while the tool could generate useful background elements, it would require additional manual intervention to be viable for RTS game design. Participant 5 echoed this sentiment, stating that while the tool had potential across various design challenges, it was less suited for RTS levels due to its tendency to generate enclosed rather than open spaces. Nonetheless, they believed the tool had utility in all examined scenarios, albeit to varying degrees.

5.6.3. Strengths in Generating Semi-Randomized Noise

Despite its limitations in structured map design, the tool was seen as particularly strong in generating procedural "background noise." Participants found it effective in creating environmental elements such as forests, ruins, and scattered features, which are difficult to craft manually without appearing artificial. Participant 4 reinforced this perspective, explaining that while the tool did not excel at structured RTS maps, it performed well in generating randomized features for RPGs and BR games. The tool's ability to create organic-looking randomness was its key strength.

5.6.4. Appreciation for Mixed-Initiative Design

Several participants valued the mixed-initiative approach, which allowed users to refine and adjust generated content instead of relying solely on automation. However, they also noted that the frequent appearance of artifacts sometimes made manual design a more effective alternative.

Participant 1 discussed how iterative refinement and feedback loops could be central to the tool's usability. They proposed that adding elements such as kernel size variability and randomized placements could significantly improve the tool's effectiveness.

Overall, the MIAI tool has notable potential but is hindered by its current limitations. It is most effective in generating semi-randomized environmental features but struggles with structured gameplay elements and large tile-based maps. While participants valued its mixed-initiative approach, they also encountered usability challenges that sometimes made manual design preferable.

5.6.5. Question 2 focus group: What was your workflow with the system?

During the focus group the participants where asked: "What was your workflow with the system?" The use of the Melete among participants was characterized by an iterative, trial-and-error approach where play, experimentation, and exploration were essential. Due to the unpredictable nature of the Melete's outputs, users engaged in cycles of input adjustment, observation, and refinement.

5.6.6. Iterative Exploration as a Core Workflow

Participants engaged with the tool in an iterative manner, repeatedly testing different inputs, analyzing the outcomes, and adjusting their approach accordingly. Participant 1 described their workflow as a "play-exploration cycle" where they engaged in trial and error to understand how their inputs translated into visual outcomes. The unpredictability necessitated logical reasoning and puzzle-solving elements, making the experience feel like a game rather than a straightforward design tool. They expressed a desire for greater accuracy in input-to-output correspondence, believing that increased success in replicating intended results would shift the process from play to structured design.

Similarly, Participant 5 noted that their approach evolved as they became more familiar with the system's rules. Initially, their engagement was exploratory, but over time, they developed a more intuitive sense of the tool's behavior. They described the process as creative level design, where they threw out ideas, observed results, and refined their approach through multiple iterations.

5.6.7. Play testing

A recurring theme among participants was the necessity of toggling between the designer and player views to evaluate how the generated environments functioned in a gameplay context, this back-and-forth movement was seen as essential.

Participant 4 relied heavily on shifting perspectives, generating a random map, exploring it from a player's viewpoint, and then returning to the editor to refine elements that stood out. This iterative cycle was particularly useful for assessing emergent gameplay features, such as discovering how randomly generated structures (e.g., hallways and forests) contributed to level dynamics. However, they noted that while the tool excelled at creating organic, procedurally generated content, more structured, intentional designs required manual intervention. Participant 5 echoed this sentiment, stating that transitioning between views was fundamental to their process.

Overall, participants engaged with the tool through an iterative, experimental process, leveraging play and exploration to understand its capabilities. While the system was well-suited for MIAI. Switching between designer and player perspectives was a crucial but sometimes cumbersome aspect of the workflow. Participants expressed a strong desire enhanced usability features to improve efficiency. Despite these limitations, they recognized the tool's potential for rapid prototyping.

6. Discussion

The two sessions—Battle Royale (BR) and Real-Time Strategy (RTS) — provides insight into the potential usefulness of Melete as a design tool. The qualitative data reveal overarching themes across both generas while also highlighting key differences in the system's application and user reception. By examining participants feedback, we can discern the narrative underpinning their experiences and expectations when using Melete.

6.1. Q1

In response to the question "What did you like about the system?", the two sessions illustrate Melete's value as a tool for fostering creativity, facilitating collaboration, and supporting rapid prototyping. While both groups found Melete beneficial, their priorities differed based on game genre. BR participants emphasized iterative design and immersive visualization, whereas RTS participants valued the real-time editing and terrain flexibility. While many of the similarities highlight Melete's strengths, key differences in the sessions reveal the varying demands of different game genres. The BR session placed greater emphasis on playtesting and visualization, with participants valuing the 3D environment for interactive exploration. The third-person camera controller was seen as an advantage in this setting. Conversely, RTS participants prioritized real-time editing, finding value in the ability to modify maps dynamically, which was also available to them in session one. This divergence indicates that while Melete is effective in both scenarios, different genres require different functionalities, with RTS designers highlighting the importance of being able to playtest with similar camera functionality as that of the intended genera.

6.2. Q2

In response to the question "What did you dislike about the system?" Across both BR and RTS groups, common pain points emerged regarding the inefficiencies of PCG, the UI, and navigation and camera control issues. The excessive manual iteration required due to PCG constraints, the UI flow, and sluggish camera movement collectively hampered the design and testing processes. While participants still found

aspects of the system enjoyable, these persistent limitations negatively impacted usability and efficiency across both game genres. Addressing these concerns would be crucial for improving user experience, streamlining workflow, and enhancing automation within the game development environment. An additional challenge identified by participants was the lack of accessible, on-screen guidance. One participant noted that while an introductory tutorial was available, it was not easily accessible for reference during actual usage, making it difficult to recall necessary instructions. This lack of immediate guidance contributed to inefficiencies in navigating the interface, increasing frustration and reducing usability. Suggesting that not only do we need to consider how we develop MIAI tools, but how do we approach training users to use them effectively as a possible avenue for study.

6.3. Q3

The third question we asked the experts during the BR and RTS sessions about the system was, How easy was it to use the tool? Experts found the tool's core functionality understandable but noted usability challenges. BR participants struggled with UI complexity, navigation, and workflow inefficiencies, though minor improvements like tooltips could ease use. RTS participants found some tasks, like adding bases, more cumbersome than necessary and suggested prefabs for efficiency. Both groups agreed that streamlining the UI, refining interactions, and adding genre-specific tools would significantly enhance usability. Despite initial challenges, participants adapted over time, indicating Melete's strong potential with targeted refinements.

6.4. Q4

The fourth question we asked the experts during the BR and RTS sessions about the system was, did you have enough control over the design process? Experts had mixed views on design control. In the BR session, participants generally felt they had enough control despite grid-based constraints and occasional artifacts. The system was valued for prototyping, though some game-specific features were lacking. RTS participants had more varied experiences, with some finding the tool effective while others faced unpredictable outputs and limited control over key design elements. Both groups agreed Melete is a useful prototyping tool but requires refinement and manual intervention. This highlights the need for genre-specific tailoring to ensure sufficient control and predictability for different game design paradigms.

6.5. Q5

The fifth question we asked the experts during the BR and RTS sessions about the system was, Was the play testing mode useful? Experts agreed that playtesting mode was valuable but more effective for BR than RTS. BR participants praised its role in rapid space prototyping, though navigation challenges, like the lack of a minimap and clunky character controls, limited its usability. RTS participants found it helpful for visualizing maps and spotting artifacts but noted its limited relevance due to the genre's top-down perspective. Both groups saw playtesting as essential for refining designs, though BR participants highlighted its broader potential for other genres, like dungeon crawlers. Overall, playtesting was more impactful for exploration-heavy game formats like BR.

7. Conclusion

As highlighted in the background research and echoed by AI and design experts in this study, research into HCI, PCG, and their integration within MIAI pipelines remains in its early stages and demands greater focus. Understanding how different PCG algorithms impact MIAI workflows based on design challenges could reveal significant obstacles. Additionally, exploring if and how MIAI users interact with PCG algorithms may uncover opportunities in other areas of research such as LLM integration in MIAI pipelines. From an HCI perspective, designing effective interaction loops could empower designers

to collaborate more effectively with AI in creative contexts. Another critical challenge is establishing robust validation methods for MIAI pipelines. Advancing these areas will require collaborative efforts across multiple disciplines, including AI, UX/UI, and HCI, to create impactful tools and frameworks for both academia and industry.

In conclusion, in this paper we have introduced Melete, a novel MIAI pipeline used for the development of 3D virtual environment. Our expert analysis highlighted the critical role of playtesting in the design process, emphasizing the need for genre-specific playtesting components within MIAI interaction loops. Therefore, based on this paper's analysis it is not only important to include playtesting tools within MIAI pipelines, there must also be considerations as to genre-specific playtesteing tools being included in any MIAI pipeline that focuses on the creative applications of MIAI systems. By addressing usability challenges and enhancing user control, Melete demonstrates the potential for MIAI pipelines to transform game design workflows, enhance designer engagement, facilitate iterative design processes, and rapid prototyping of 3D environments. We further provide recommendations for future research, encouraging exploration into user interaction, validation methods, and cross-disciplinary collaboration to advance the development of effective and adaptable MIAI tools.

References

[1] S. Amershi, M. Cakmak, W. B. Knox, T. Kulesza, Power to the people: The role of humans in interactive machine learning, AI magazine 35 (2014) 105–120.

[2] T. Lubart, How can computers be partners in the creative process: classification and commentary on the special issue, International journal of human-computer studies 63 (2005) 365–369.

[3] J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survey, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 172–186.

[4] N. Shaker, M. Shaker, J. Togelius, Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 9, 2013, pp. 215–216.

[5] H. Yu, Reflection on whether chat gpt should be banned by academia from the perspective of education and teaching, Frontiers in Psychology 14 (2023) 1181712.

[6] S. Deterding, J. Hook, R. Fiebrink, M. Gillies, J. Gow, M. Akten, G. Smith, A. Liapis, K. Compton, Mixed-initiative creative interfaces, in: Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, 2017, pp. 628–635.

[7] L. Mamykina, L. Candy, E. Edmonds, Collaborative creativity, Communications of the ACM 45 (2002) 96–99.

[8] S. J. Russell, P. Norvig, Artificial intelligence: a modern approach, Pearson, 2016.

[9] P. Toprac, B. Heng, H. Siddharthan, A. Tseng, S. Abraham, E. Vouga, Automation of aaa game development using ai (2025).

[10] A. Drachen, P. Mirza-Babaei, L. E. Nacke, Games user research, Oxford University Press, 2018.

[11] A. Canossa, G. Smith, Towards a procedural evaluation technique: Metrics for level design, in: The 10th International Conference on the Foundations of Digital Games, sn, 2015, p. 8.

[12] M. Cook, J. Gow, G. Smith, S. Colton, Danesh: Interactive tools for understanding procedural content generators, IEEE Transactions on Games 14 (2021) 329–338.

[13] A. Liapis, 10 years of the pcg workshop: Past and future trends, in: Proceedings of the 15th International Conference on the Foundations of Digital Games, 2020, pp. 1–10.

[14] A. Summerville, J. R. Mariño, S. Snodgrass, S. Ontañón, L. H. Lelis, Understanding mario: an evaluation of design metrics for platformers, in: Proceedings of the 12th international conference on the foundations of digital games, 2017, pp. 1–10.

[15] O. Withington, M. Cook, L. Tokarchuk, On the evaluation of procedural level generation systems, in: Proceedings of the 19th International Conference on the Foundations of Digital Games, 2024, pp. 1–10.

[16] B. Shneiderman, Supporting creativity with advanced information-abundant user interfaces, Springer, 2001.
[17] S. G. Isaksen, D. J. Treffinger, Celebrating 50 years of reflective practice: Versions of creative problem solving, The Journal of Creative Behavior 38 (2004) 75–101.
[18] S. G. Isaksen, D. J. Treffinger, Creative problem solving, The Basic Course. New York: Bearly Limited (1985).
[19] B. Shneiderman, Creating creativity: user interfaces for supporting innovation, ACM Transactions on Computer-Human Interaction (TOCHI) 7 (2000) 114–138.
[20] A. Liapis, G. N. Yannakakis, J. Togelius, Designer modeling for sentient sketchbook, in: 2014 IEEE Conference on Computational Intelligence and Games, IEEE, 2014, pp. 1–8.
[21] A. Hoyt, M. Guzdial, Y. Kumar, G. Smith, M. O. Riedl, Integrating automated play in level co-creation, arXiv preprint arXiv:1911.09219 (2019).
[22] W. Gaver, What should we expect from research through design?, in: Proceedings of the SIGCHI conference on human factors in computing systems, 2012, pp. 937–946.
[23] J. Zimmerman, J. Forlizzi, S. Evenson, Research through design as a method for interaction design research in hci, in: Proceedings of the SIGCHI conference on Human factors in computing systems, 2007, pp. 493–502.
[24] E. Stolterman, The nature of design practice and implications for interaction design research, International Journal of Design 2 (2008).
[25] S. Margarido, P. Machado, L. Roque, P. Martins, Let's make games together: Explainability in mixed-initiative co-creative game design, in: 2022 IEEE Conference on Games (CoG), IEEE, 2022, pp. 638–645.
[26] F. A. Figoli, F. Mattioli, L. Rampino, Artificial intelligence in the design process: The Impact on Creativity and Team Collaboration, FrancoAngeli, 2022.
[27] R. Pandya, S. H. Huang, D. Hadfield-Menell, A. D. Dragan, Human-ai learning performance in multi-armed bandits, in: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, 2019, pp. 369–375.
[28] G. Zhang, A. Raina, J. Cagan, C. McComb, A cautionary tale about the impact of ai on human design teams, Design Studies 72 (2021) 100990.
[29] J. Liao, P. Hansen, C. Chai, A framework of artificial intelligence augmented design support, Human–Computer Interaction 35 (2020) 511–544.
[30] G. Lai, F. F. Leymarie, W. Latham, On mixed-initiative content creation for video games, IEEE Transactions on Games 14 (2022) 543–557.
[31] T. Machado, D. Gopstein, A. Wang, O. Nov, A. Nealen, J. Togelius, Evaluation of a recommender system for assisting novice game designers, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 15, 2019, pp. 167–173.
[32] P. Lucas, C. Martinho, Stay awhile and listen to 3buddy, a co-creative level design support tool., in: ICCC, 2017, pp. 205–212.
[33] A. Liapis, G. N. Yannakakis, Boosting computational creativity with human interaction in mixed-initiative co-creation tasks (2016).
[34] G. Smith, J. Whitehead, M. Mateas, Tanagra: A mixed-initiative level design tool, in: Proceedings of the Fifth International Conference on the Foundations of Digital Games, 2010, pp. 209–216.
[35] M. Guzdial, N. Liao, J. Chen, S.-Y. Chen, S. Shah, V. Shah, J. Reno, G. Smith, M. O. Riedl, Friend, collaborator, student, manager: How design of an ai-driven game level editor affects creators, in: Proceedings of the 2019 CHI conference on human factors in computing systems, 2019, pp. 1–13.
[36] Z. Zhou, M. Guzdial, Toward co-creative dungeon generation via transfer learning, in: Proceedings of the 16th International Conference on the Foundations of Digital Games, 2021, pp. 1–9.
[37] O. Delarosa, H. Dong, M. Ruan, A. Khalifa, J. Togelius, Mixed-initiative level design with rl brush, in: Artificial Intelligence in Music, Sound, Art and Design: 10th International Conference, EvoMUSART 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 10, Springer, 2021, pp. 412–426.
[38] G. Todd, S. Earle, M. U. Nasir, M. C. Green, J. Togelius, Level generation through large language

models, in: Proceedings of the 18th International Conference on the Foundations of Digital Games, 2023, pp. 1–8.
[39] R. Gallotta, A. Liapis, G. Yannakakis, Consistent game content creation via function calling for large language models, in: 2024 IEEE Conference on Games (CoG), IEEE, 2024, pp. 1–4.
[40] N. Sturtevant, N. Decroocq, A. Tripodi, C. Yang, M. Guzdial, A demonstration of anhinga: A mixed-initiative epcg tool for snakebird, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 16, 2020, pp. 328–330.
[41] D. Bhaumik, J. Togelius, G. N. Yannakakis, A. Khalifa, Lode enhancer: Level co-creation through scaling, in: Proceedings of the 18th International Conference on the Foundations of Digital Games, 2023, pp. 1–8.
[42] V. Kumaran, D. Carpenter, J. Rowe, B. Mott, J. Lester, Procedural level generation in educational games from natural language instruction, IEEE Transactions on Games (2024).
[43] M. Kreminski, M. Dickinson, J. Osborn, A. Summerville, M. Mateas, N. Wardrip-Fruin, Germinate: A mixed-initiative casual creator for rhetorical games, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 16, 2020, pp. 102–108.
[44] I. Karth, A. M. Smith, Wavefunctioncollapse is constraint solving in the wild, in: Proceedings of the 12th International Conference on the Foundations of Digital Games, 2017, pp. 1–10.
[45] J. R. van Meter, Schrödinger–newton 'collapse'of the wavefunction, Classical and Quantum Gravity 28 (2011) 215013.
[46] I. Karth, A. M. Smith, Wavefunctioncollapse is constraint solving in the wild, in: Proceedings of the 12th International Conference on the Foundations of Digital Games, FDG '17, Association for Computing Machinery, New York, NY, USA, 2017. URL: https://doi.org/10.1145/3102071.3110566. doi:10.1145/3102071.3110566.
[47] P. Karimi, J. Rezwana, S. Siddiqui, M. L. Maher, N. Dehbozorgi, Creative sketching partner: an analysis of human-ai co-creativity, in: Proceedings of the 25th international conference on intelligent user interfaces, 2020, pp. 221–230.
[48] N. Shaker, M. Abou-Zleikha, Alone we can do so little, together we can do so much: A combinatorial approach for generating game content, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 10, 2014, pp. 167–173.
[49] A. Baldwin, S. Dahlskog, J. M. Font, J. Holmberg, Towards pattern-based mixed-initiative dungeon generation 20506 (2017) 1–10. doi:10.1145/3102071.3110572.

A. Appendix

	Session 1 Battle Royale	Session 2
Participant 1	When wave function collapse generates building spaces, the emergent structures and spaces feel compelling and often novel - there's an element of unconventional square matrix space design which can feel surreal and engaging. For square matrix environments, like Manhattan style cities this could construct some believable and complex scenes	Program does successfully generate multi-level systems, so long as WFC arrangement is set up correctly. The emergent complexity of some environments can make inspiring designs that a human could further interpret.
Participant 2	I like the way it learns patterns and rules that can then be replicated automatically in-game. It shows some promise as a way of helping create non-essential tile-based positional objects. and landscape/areas	System has promise but up to a limit to be useful in its current form.
Participant 3	The immersive 3d environment for checking the world design is really attractive.	The flexibility of the colouring process and the diversity of tile types
Participant 4	I generally like level editors and I enjoy building levels, so this was something that I like doing for fun, and also partially what I was comparing the system to. In that sense it's enjoyable to use, and I can also see it being useful from the perspective of generating graybox prototypes. One example of this was the totally random generation of a two story building that had tons of staircases. This isn't something that I would have ever made on my own, but it made me think divergently about what this would be like in a Battle Royale context, which was cool.	Being able to do live editing was cool. I did like that I could conceivably have used this to think about both the unit level from the third person perspective and then also from a further remove map perspective. I liked that it generated a bunch of "messy" terrain in the center of the map, which seems like it would fit what I'd want for a destroyed urban battleground setting.
Participant 5	It's satisfying to feed an PCG system and see what it produces - a good sense of agency attached to my actions. The loop is nice and quick, also, so I can tinker and see the results satisfyingly quickly. Editing the generated maps is also nice - a sense of collaboration with the system/myself.	The sense of feeding a PCG system and seeing the results is quite satisfying. It's a nice feeling of collaboration with the system/myself. I give it what I think will help with the level generation and it throws curveballs in its outputs that I think can stir creativity.

Table 1
Question 1: What did you like about the system?

	Session 1	Session 2
Participant 1	As with most waveform collapse systems, a lot of re-generation and artefact cleaning is necessary to make a believable environment; a lot of itteration, trial and testing is needed to make a quite low complexity scenes	Likely due to a small kernal size, forms larger than immediate adjacency are not effectively replicated. Successful replication of layout complexity usually only comes after a large enough number of generations that the affordance of the automation are lost, though some pipeline optimisation could help make this tool become more of a utility.
Participant 2	Currently its very tile-based in all three axes. Hard to see how it could improve on existing tile based solutions outside of the ability to change non-essential tiles and neatly handle multiple gridheight (e.g 8 or more) or granular height (such as on a 3D terrain, or multiple grids and items that don't fit a specific regular grid.	Lack of ability to define areas as to be randomised and areas where a specific arrangement is required.
Participant 3	Cannot find any at the moment	The movement of the agent in play test is not very accurate
Participant 4	The disconnect in the UI was a problem sometimes. This was most apparent because of the wide array of button presses to get to various editing states. There was also an issue with me wanting to have a little more onscreen guidance in the editing screen rather than having it as an intro I couldn't easily refer back to. I also found myself wanting the ability to seamlessly switch perspectives, maybe using something like a minimap, to let me know where I was in the playtest scene. This would help me to do what I do in actual prototyping, which is to go to an area, take a look around from the perspective of a player, make some changes based on an intuition, and then come back in to see how it feels.	This was a lot harder to design for an RTS because it didn't have as many of the affordances I'd need to think about in a game like this - resource nodes, defensible positions, spawns, etc. I could sort of imagine where these would be, but it was harder to fit the genre constraints for RTS into this system than it was Battle Royale. This was especially the case in terms of sight lines. The camera perspective in a battle royale is much different from the one in an RTS (which tends to be further remove isometric), so thinking about interesting sight lines and landmarks was a lot harder in practice.
Participant 5	The UI and inputs are a little fiddly and I found it tricky to remember what does what and what order to click things in to take my desired action. The character /camera controller is also a bit sluggish and gets in the way a little - I want to be able to more easily inspect my level.	It's a bit fiddly to interact with - the UI and input bindings. In some ways, the "extra steps" on generating a level could slow things down I suppose, but that's mitigated by the fun of the experience.

Table 2
Question 2: What did you dislike about the system?

	Session 1	Session 2
Participant 1	The current user interface is hard to use, several steps and often unintuitive interfaces. However, as a concept, the cyclic iteration process could become easy to prototype spaces with	The current layout is not intuitive, but the tool itself was straightforward enough to work with. The scale of the sample map and the limited number of tiles more often produces structures unaligned with the prompt than not.
Participant 2	Straigghtforward - a little flakey from a user perspective(e..g having to select the sample button before selecting the tile type to be placed. Hard to see though how it could use meaningfully a pallete of hundreds of tiles as would be present in any grid-based or 2d isometric game.ric	A bit cumbersome - a single pallet that was selectable would be preferred over seperate select buttons, index and selectable pallete
Participant 3	It's very intuitive and easy to use	fairly easy to understand and use
Participant 4	I understand it's a prototype, so I'm basing my evaluation on that: there could be some more easing of the user experience with tooltips, less complex inputs, and fewer clicks in between different desired actions. However, keeping in mind that these would be fairly simple to add, I think that the process was very straightforward and usable once I understood the basic interactions.	This was again made harder by the tool. An RTS probably needs two distinct enemy bases that the units originate from, but having to manually add these in to the map after random generation was harder than just doing this by hand. This is where a pre-fab option might be handy. I could completely hand design two different factional bases, and then use the proc gen to make a battlefield, and then combine the two, which would be the best of both worlds to me.
Participant 5	Fairly easy - I understood the concept quite quickly. Most of the issues came from trying to remember the interface elements and pressing the wrong buttons (etc.).	The concept is simple and I'm starting to get used to using it (tactics for feeding it, etc.), but some of the fiddly interaction aspects are a bit more difficult than needed.

Table 3
Question 3: How easy was it to use the tool?

	Session 1	Session 2
Participant 1	A free editor gives me control u the the limitations of grid based prefab placement system, the waveform collapse algorithm can follow replicate design elements fairly consistently, though rarely without artefacts	As the outcomes of the product were substantially variable and more often produce construct unwanted biproducts such as long walls stretching the entire map, careful selection of tiles felt like it began to restrict the maximum complexity this tool can bring with the curent pallet range for the task set.without careful
Participant 2	No. I could see it being useful after revision of the tool as a quick teaching tool or prototyping tool - but not for production work apart from the ability to populate random tile sets that were not gameplay specific.	No
Participant 3	Yes	Yes
Participant 4	From the perspective of grayboxing a level, yes. There are things I can think of that would be necessary for this type of game (powerups, more complex architectures, etc.), but I think that I could use this to slap together a level that could be shown to environment artists and programmers to give them a sense for what I wanted.	I think it was adequate, but not being able to think about things like resource placement or cover was a bit odd for the task of designing an RTS map.
Participant 5	I think so. I fed the thing and had the final say on the output.	I think so - it starts and ends with me. It's hard to know how much the bit in the middle is influencing my final decisions, but it's not unlike other techniques for driving creativity, like randomisation.

Table 4
Question 4: Did you have enough control over the design process?

	Session 1	Session 2
Participant 1	It can provide a compelling use-case for rapid space prototyping with prefabs in unity, with some limitations. This could be used for certain flavour of dungeon crawler or complex generation prototypes	Yes, it provides a clear line of what the map actually feels like when rendered and clearly outlines artefacts and experiential outliers
Participant 2	Yes - though it would be nice to have the two combined so changes happened immediately. having to quit out , change and run again is a serious bottleneck in game development. tools that require this don't hang around long.	Yes
Participant 3	That's the most interesting part IMO	Yes, very useful to visualise the 3d world
Participant 4	Yes, although see the above comments about wanting a minimap to help me understand where my player-perspective character is in the level I'm designing. This was the hardest part of the process to me, and having to constantly recenter between changes threw off my design process(which I set as making a level with a forest and a large central two story building).	Less useful than in the previous experiment. Very rarely do you get that sort of perspective in an RTS (and I assume the game would have to be specifically designed around it), so it gave me less information than the battle royale genre, which is almost always in that perspective.
Participant 5	It's essential for getting a sense of the level but the character controller, camera, and UI make it a bit hard to use. It does need to be there though.	Yes, it's needed to see the results from the ground level. The controls could be improved.

Table 5
Question 5:Was the play testing useful?