Synchronizing Game and AI Design in PCG-Based Game Prototypes

Henri Bomström
henri.bomstrom@oulu.fi
University of Oulu
Oulu, Finland

Markus Kelanti
markus.kelanti@oulu.fi
University of Oulu
Oulu, Finland

Jouni Lappalainen
jouni.lappalainen@oulu.fi
University of Oulu
Oulu, Finland

Elina Annanperä
elina.annanpera@oulu.fi
University of Oulu
Oulu, Finland

Kari Liukkunen
kari.liukkunen@oulu.fi
University of Oulu
Oulu, Finland

ABSTRACT

Procedural content generation (PCG)-based game design aims to reach a new way of playing games by focusing gameplay around algorithmic game content generation. However, positioning interaction with PCG systems and generated content to the center of player experience poses design challenges for both game design and AI design. In order to create the wanted affordances, rich contextual information is required to make informed decisions on the generated content. While previous research has presented excellent developments on PCG's possibilities, further considering context and affordances in the early stages of prototyping may aid designers reach these possibilities in a more consistent manner. This study is set to discuss how context, affordances and the game's overall design can be considered during the prototyping process of PCG-based games. Misaligned game context and affordances can result in deeply rooted design issues that may later manifest as subpar gameplay experiences and increased development effort. These emergent issues are examined through a post-mortem case study to produce an extended PCG-based design process, featuring actionable steps, that takes context, affordances, and the game's overall design into account through meaningful play.

CCS CONCEPTS

• **Applied computing** → **Computer games**.

KEYWORDS

PCG, PCG-based game design, Prototyping, Video games, Video game design

ACM Reference Format:
Henri Bomström, Markus Kelanti, Jouni Lappalainen, Elina Annanperä, and Kari Liukkunen. 2020. Synchronizing Game and AI Design in PCG-Based Game Prototypes. In *International Conference on the Foundations of Digital Games (FDG '20), September 15–18, 2020, Bugibba, Malta.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3402942.3402989

FDG '20, September 15–18, 2020, Bugibba, Malta

ACM ISBN xxx-x-xxxx-xxxx-x
https://doi.org/10.1145/1234567.8901234

1 INTRODUCTION

Procedural content generation (PCG) refers to automatic content creation with algorithms [34]. Commercial digital games have featured PCG in limited roles of design throughout the history of digital games [37]. Early motivations for PCG, such as variability in content, replayability and assistance in human creativity, remain accurate to this day [23]. Moreover, content generation can provide unique benefits that would otherwise be unfeasible to implement, such as an aesthetic of exploring an endless world. However, by utilizing PCG in a narrow design role the player's core gameplay experience generally stays the same [25].

PCG-based game design is an iterative design process that aims to reach a new way of playing games by focusing gameplay around content generation. By allowing players to control aspects of content generation through game mechanics, they are able to form strategies around content generation, making gameplay revolve around it. [6, 25] However, binding content generation to game mechanics causes design challenges for both game design and AI design as the act of designing games shifts from hand authoring player experiences to creating ranges of meaningful content with algorithms [25]. The challenge in content generation remains to produce content that is both novel and valuable, matching the design task at hand [3, 19, 32].

Game design, a part of game development, usually begins with an idea that is refined into a prototype [2, 8, 11, 16–18]. Prototypes are used to evaluate a game's concept by creating mock-ups that demonstrate core game mechanics and functionality. PCG-based games can be challenging to prototype as their core gameplay, per definition [25], revolves around content generation. Instead of designing the game and its AI system in vacuum, the design phases are tightly coupled through the *context* provided by game design – in which the AI system's decisions need to make sense in – and the *affordances* based on the AI system's capabilities that would otherwise be unattainable [6, 25]. However, disjointed context and affordances can result in catastrophic failures [16] that leave the game in an unplayable state and result in a greater overall development effort.

Affordances, context and the game's overall design should be taken into account when designing PCG systems [19, p. 2]. However,

it is not clear *how* they should be taken into account. Furthermore, there is a void in detailing the process of formalizing designer knowledge in PCG-based game design. This paper i) provides an initial proposition for taking context and affordances into account during the design process and ii) extends the PCG-based game design process with actionable steps. This paper is organized as a case study [38], where the emergent problems of a PCG-based game prototyping process are used to frame the question of how context and affordances should be taken into account through practice. An initial version of the extended design process was created during the prototyping case, which was further modified to include the proposed solutions for considering affordances and context during design. Furthermore, theories such as meaningful play [17, chapter 3, p. 1-6] and essential experience [18, p. 20-21] were used to propose additional steps in order to better consider affordances and context during the design process.

The rest of this paper is structured as follows. First, the relevant background for prototyping PCG-based games is introduced, followed by the case study section describing the methods used, the design process behind the PCG-based game prototype, and the emergent problems raised. These problems are then discussed by focusing on the role of context and affordances in PCG-based game design. Afterwards, the results of this study are presented and discussion provided on the lessons learned, limitations, and the need for future research. Finally, conclusions are presented.

2 BACKGROUND

2.1 Procedural content generation

Content generation has deep roots in analog games and the early motivations of replayability and assisting creativity map well to contemporary PCG [23]. Earlier digital games such as Elite [1] and .kkrieger [31] used PCG to great effect in circumventing hardware restrictions through deterministic PCG [34] by expanding content from minimal representations. Determinism in PCG refers to the relationship between the inputs and outputs of an algorithm – whether it always produces the same output with the same inputs. Stochastic PCG on the other hand produces varying content between runs. Varying game content has been used to improve replayability [23, 34] and to create an aesthetic of exploring endless worlds in games such as Rogue [7], Diablo [4], and Minecraft [14]. However, these algorithms are not random, despite PCG sometimes being synonymized with random generation. The produced content still has structure and is in some way selected as purely random content would be unplayable [33].

Sometimes the produced content is undesirable or not even playable. Producing coherent and playable content is crucial for player experience [34] and producing unplayable content has been aptly regarded as a catastrophic failure [39]. However, some content is more tolerant to failures than others. The necessity of content, whether the player needs the specified content to progress, can be used to gauge whether failures are tolerated and to what extent. Similarly, the selected approach, such as constructive or generate-and-test, affects how generation itself is viewed. Constructive approaches create content instances in one pass, making sure the output is correct. Generate-and-test approaches on the other hand create content instances and test them against some criteria based on the instance's properties. [34] Furthermore, generate-and-test approaches can be further divided into more popular and novel approaches such as search-based [34], utilizing evolutionary algorithms, and experience-driven PCG [36], employing player models to guide generation. The selection of approach depends on how the selected algorithm should behave as a constructive approach trades novelty for predictability and performance. With this on mind, PCG algorithms can be further divided into online and offline methods depending on whether generation occurs during gameplay [34]. Search-based approaches are preferred for offline generation when execution time is not as big of issue, while other approaches are preferred for online generation [37].

Most search-based solutions utilize evolutionary algorithms to optimize content, where a population of solutions is evaluated against some criteria. Each content class is processed via its genotype, the genetic structures which are operated on by the algorithm, and phenotypes, the end result of how a genotype "looks" when evaluated. Genotypes can be though of as the instructions from which a level is built and the phenotype can be thought of as the complete level. Content representation is an important issue as it shapes the algorithm's search space, the group of its possible solutions. [34] Next, evaluation functions assign fitness to each instance and the algorithm ranks them according to it. Afterwards, the population starts to reproduce with operators such as selection, crossover and mutation and the population is replaced with the newly created one. [13, p. 8-9] This process is repeated as many times as needed or until some criteria is met. However, search-based methods cannot guarantee a completion time or the quality of the result. Evaluation functions consume most of the algorithm's runtime, and evaluation methods such as simulation can be costly. [34, 36]

From a design point of view, content generators themselves can be seen to fall into optimization, constraint satisfaction, grammatic generation, content selection or constructive approaches, and selecting the correct one is an important design choice [22]. Each approach usually presents at least some kind of constraints over content generation to represent designer knowledge on playability and player experience. Constraint satisfaction in itself is also a relatively popular approach as it allows the designers to specify content in logical statements, resulting in a space of solutions that fulfils the criteria set for content. Examples of constraint driven approaches can be seen in the Variations Forever game [20] and the Tanagra [27] mixed-initiative level generation tool. Finally, a grammar based approach can be seen in Launchpad [28], the content generator used by games such as Rathenn [24] and Endless Web [25].

Content generators should be evaluated to understand the generator's capabilities and whether it actually produces content matching the designer's intent [19, p.215-217]. Evaluating generators can be challenging as most of the existing PCG-systems are tailored to the needs of individual games [32]. However, generator evaluation mostly focuses on generative space, the eventual outputs that the generator will produce given certain parameters [21], and expressive range, the style and variety of level the generator can produce [26]. A generator's expressive range can be evaluated by determining the appropriate metrics, generating content, visualizing its generative space, and by analyzing the impact of parameter

changes [26]. Furthermore, there are multiple positive attributes shared by all generators: speed, reliability, controllability, expressivity and diversity, and creativity and believability [19, p. 6-7]. Lastly, it should be made sure that the generator is both interfaceable and controllable [32] for both players and designers, depending on the selected approach on control over the generator [22].

2.2 Procedural content generation-based game design

PCG-based games closely tie their core game mechanics to the underlying content generator in a way that profoundly affects the game's dynamics and aesthetics [25]. Currently, there are only a few famous examples of PCG-based games. As an example, Galactic Arms Race [9] utilizes online content generation to create armaments for combating space ships through player statistics, allowing players to steer generation through behaviour. However, the indirect control over content generation limits how deeply players can utilize generation for different strategies. A different approach is selected for Endless Web [25], perhaps the most famous example in PCG-based games, as players are provided direct control over generated content while providing visibility on the generator's internal state. In this way players are able to steer content generation to a favorable direction and form strategies around it in order to reach their goals.

The PCG-based game design loop (figure 1) describes an iterative design process where game design and AI design are worked in tandem. Entering the design process can be done from either side as AI design affords possible core game mechanics, while game design provides the necessary contextual information for the AI system's decisions. Game design provides the context in which the AI system's decisions need to make sense in and the AI system in return creates affordances to be used through game mechanics in order to form player strategies around content generation. Both game design and AI design are affected by domain knowledge such as AI architectures, game design convenientions and knowledge domains. Early iterations are encouraged as both design activities need to be fleshed out as they feed each other. [6, 25] PCG systems afford unique game dynamics when leveraged to a high enough degree [22] and can lead to a new way of playing games or to new game genres [6]. However, in order to leverage PCG to its full degree, the AI being utilized must support player exploration without merely creating an illusion of intelligence. [6]

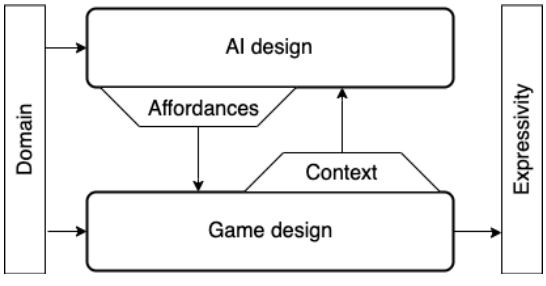

Figure 1: The PCG-based game design process [6, 25] is an iterative practice that combines game design and AI design.

The mechanics, dynamics and aesthetics (MDA) framework [10] specifies that designers approach games through mechanics, dynamics and aesthetics. Players on the other hand experience gameplay in reverse order through the game's aesthetics, dynamics and mechanics (figure 2), making game design an indirect way of creating experiences. Game mechanics represent the game's components at the level of data structures and algorithms. When players interact with these rules, or the rules interact with each other, dynamics arise to represent the system's behaviour during play. Finally, aesthetics depict the "desired emotional responses" elicited by the player during interaction with the game system [10] – representing how players experience the game and whether they found it meaningful [6]. This paper uses the design lense of essential experience [18, p. 20-21] as a tool to derive what experience the player is to have, what is essential to that experience, and how can that experience be captured.

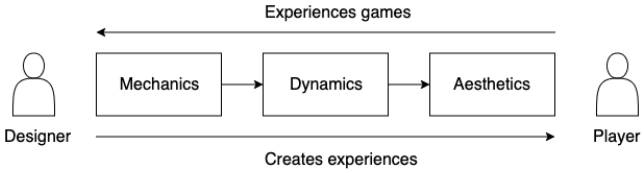

Figure 2: The mechanics, dynamics and aesthetics framework [10].

The indirect nature of designing PCG-based games is further emphasized by the lack of authorial control over player experiences. When compared to traditional game design, designing experiences via hand authored content shifts to creating ranges of meaningful content with algorithms as interaction with generated content is in the centre of the player's experience [6, 22, 25]. There are also several different trade-offs that can be made, such as how designer knowledge is represented in the generator, in which stage the generation occurs, how players interact with the generator and whether the generator purposefully generates certain experiences [22]. However, there are design metaphors such as the tool, material, designer and expert that help choose the correct approach for utilizing PCG in game design [12].

2.3 Context, interactivity and meaning

Creating meaningful content is challenging. Meaningfulness is a concept that is often discussed with content generation as it helps to understand generated content in terms of gameplay experiences. Without meaning provided by context, games can feel empty and hollow, despite being filled with interactivity and varying generated content [3, 32]. In Super Mario [15], for example, the player may complete the level while jumping constantly. However, the act of jumping in itself is not very exciting without gaps and enemies [29]. By providing context for actions, content can change the meaning behind jumping.

A game's design manifests as a *space of possibility* that defines all the possible actions and meanings that can emerge during gameplay. This interactive space provides a context where meaning is created. The relationship between the player's action and the system's response assigns meaning to the action. Meaningful play occurs when

this relationship is both discernable and integrated to the game's overall context. Player actions can be seen to be *discernable* when the game system communicates the outcome of a player's action in an understandable way. Similarly, player actions are *integrated* when the system communicates their effects on the rest of the game. [17, chapter 3, p. 1-6]

Together game design and AI design, with their context and affordances, form the game's system and its space of possibility, from which meaning and gameplay experiences are formed through interaction. Designed interaction has an internal structure and a context that assigns meaning by providing a choice for the player, who in turn takes action, which turns into an outcome [17, chapter 6, p. 1-14]. Furthermore, game systems consist of objects, attributes, actions, relationships and spaces. Objects refer to the system's parts and attributes describe their properties. Objects can perform certain actions within the system and affect each other through relationships. Lastly, the system operates in a certain space. [8, 17, 18]

Both context and the relation between player action and system outcome shape the meaning conveyed during gameplay. As interaction with AI systems is in the heart of PCG-based games, it is essential that various parameters for different generative algorithms are meaningfully based on player actions [32] and that players are able to understand the effect they have on the game world [25]. Additionally, player engagement with PCG offers ways for changing the meaning conveyed by games such as directly manipulating a game's context [5] or allowing players to find different approaches through challenges by navigating the game's generative space [24].

2.4 Digital prototypes

Game design processes begin by stating an idea or a solution to a design problem that the designer wishes to explore. The second step is to formalize said idea or solution by writing it down or creating a prototype in order to realize the idea closer to practice. Once the idea has been formalized, it needs to be tested based on various criteria if it solves the design problem at hand or needs to be discarded. This is done by evaluating test results and determining whether the proposed solution is good enough – if not, then start again. [8, 17, 18]

Game design is a distinct part of the whole game development process. The game development life cycle (GDLC) features three main phases: pre-production, production, and post-production [2, 11, 16]. However, Ramadan & Widyani [16] provide a finer-grained approach (figure 3) that also considers prototype maturity. The authors present six steps: initiation, pre-production, production, testing, beta, and release. The steps are divided into production cycles that consist of pre-production, production and testing phases. The goal of initiation is to produce a rough concept of the game, followed by a prototype for assessing the proposed concept in the pre-production step. The prototypes are further refined in the production step, where formal details and refinement into a complete prototype take place. The testing, beta, and release steps focus on ensuring usability and playability and delivering the final product to the public. Their model situates game design to the pre-production phase of game development. This paper is scoped to the initiation and pre-production phases of the GDLC where design concepts are formed and digital prototypes built to assess game design concepts.

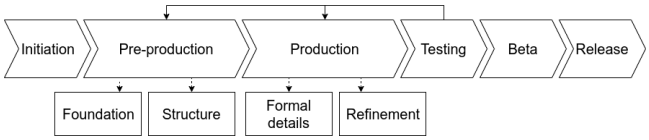

Figure 3: The game development life cycle [16] consists of six steps: initiation, pre-production, production, testing, beta, and release. Prototype maturity stages, the foundation, structure, formal details and refinement, are represented under their respective steps.

3 CASE STUDY: ACTIONABLE STEPS IN PCG-BASED GAME DESIGN

This study is conducted as a post-mortem case study [38] to highlight the emergent problems faced during a PCG-based game prototyping process. The original goal of this study was to help designers formalize their ideas into prototypes during the first steps of the GDLC [16] by extracting actionable steps for the PCG-based game design process. Preliminary extensions were made to the PCG-based game design process (figure 1) based on previous literature, e.g. [6, 10, 17–19, 25, 34] and further modifications included the "design lense" of *essential experience* [18, p. 20-21] and the theory of *meaningful play* [17, chapter 3, p. 1-6].

The resulting artefact was evaluated by using it to create a prototype PCG-based game. An artificial evaluation setting was selected with unreal users, real systems and real problems [30] to achieve a summative view of the artefact's efficacy. Evaluation with real users was deemed unfeasible with the artefact's level of maturity and the availability of game designers, but real problems and real systems could be used by prototyping a PCG-based game. The evaluation observed the design process itself – while remaining indifferent to what is being designed.

The prototype is a simple dungeon crawler that mimics *Endless Web's* [25] portal mechanics, allowing transitions in generative space through game mechanics. In this case, the player transitions through portals that affect the number of enemies spawned, environmental hazards, number of rooms, corridor tiles, amount of available food, and impassable terrain. The player is encouraged to navigate the game's expressive range to find goals hidden to certain combinations of generator parameters. The game's AI system uses online search-based methods to create 2D levels that are evaluated by utilizing the portal controlled parameters in combination with designer specific metrics such as level traversability, average room size, map grid cell usage and room density.

The prototype was constructed over two iterations by following the preliminary version of the extended design process. The first iteration focused on conceptualizing the game's idea and design in the form of mechanics, dynamics and aesthetics, while employing the design lense of essential experience. The game's PCG system was integrated during the second iteration and the content generation-based mechanics were implemented alongside

it. However, most of the produced levels resulted either in catastrophic failures or otherwise subpar level instances. During AI design, sensible ranges for individual generation parameters had to be tweaked iteratively so that the produced content supported the wanted aesthetics. In practice, designer specific evaluation functions were written to ensure that content fulfils the requirements presented in the game design phase. The generator's controllability was adjusted by visualizing its expressive range through measuring each level by its leniency and linearity similarly to how expressive range was visualized for the Launchpad generator [26, 28]. In order to reach a playable state, the levels produced by the AI system had to be playtested shortly. Gameplay feedback was found to be crucial during the design process, as the need to test the current design arose during both the game design and AI design phases.

An early stage prototype was completed by utilizing the extended design process. However, emergent problems were faced during transitions between game design and AI design that could not be resolved solely by redesigning existing parts of the process. By following the proposed process as is, the game's context and affordances were not taken into account well enough, causing issues with parameterizing algorithms, catastrophic failures, and mismatching content. While the levels were playable and fulfilled their requirements on paper, they did not produce the wanted gameplay experiences and felt disconnected from the players actions. Corrective measures were taken to restructure the generated content by examining the relationships between different level elements.

As an example, the prototype was designed to elicit an aesthetic of discovery and a dynamic of opponent play. To put it simply, the player would explore an endless selection of different levels and fight monsters while at it. However, it was discovered that the produced content lacked *meaning*. A higher level of structure was missing from produced content as level elements were not connected in any meaningful way. Evaluation functions were revised to promote exploration by taking into account average walking distance between starting location and portals, placing beneficial items in further away rooms, positioning environmental hazards to create chokepoints, and by having enemies guard items and portals. However, this approach is not optimal as it focuses on content attributes directly, without taking into account the larger context of the game. Instead, content can be designed and evaluated through the meaning it creates and the experience it produces. Nelson & Smith experienced a similar problem [19, p. 153-156] with produced levels and note that the overall design goal can be approached through player experience instead of focusing multiple evaluations on specific content instances. Furthermore, the authors note that content's properties can be expressed through their intended usage, for example through the gameplay a level produces.

The problems presented in this paper are not claimed to be new, as the development of games such as *Mismanor* and *Prom Week* present examples of mismatching affordances and game design requirements. Furthemore, it has been explicitly noted that AI design and game design should be iterated early on as content generation based mechanics cannot be implemented without designing AI, which in turn cannot be designed without a rough concept of the game system. [6] However, in this case, the aspect of interest becomes *when* these problems surface and *how* they can be addressed. When designers begin creating prototypes, they can run into problems with context and affordances at the very first stage of prototyping and addressing them as soon as possible would be beneficial in preventing escalated design issues further down the line.

4 SYNCHRONIZING GAME DESIGN AND AI DESIGN

There is no silver bullet for in-depth consideration of context and affordances for every possible design process – yet. However, the meaning created through interaction with the resulting game system makes it possible to *validate* whether context and affordances have been taken into account well enough and to *constrain* generation to fit the game's overall frame of reference. The extended PCG-based game design process (figure 4) visualizes how game context and AI affordances can be synchronized. On a high level, the proposed design process can be thought to operate on two levels. On the first level, game design and AI design are used to construct the game's system. The second level examines the resulting system's behaviour and the meaning it produces. The proposed design process is based on the existing PCG-based game design process (figure 1) and features additions and changes, indicated with a gray color, that aim to help designers create early stage prototypes.

The steps on the game design side (figure 4) are presented through the MDA framework. As a modification, the "design lense" of essential experience [18, p. 20-21] has been added to the aesthetics component. The lense is used to set a goal for the player's experience and can be used to guide the overall gameplay experience towards this goal. In this case, the lense operates by determining the aesthetics that contribute to the wanted experience, which in turn further drive the search for supportive dynamics and mechanics. An example of this would be to set the essential experience as exploration and derive an aesthetic of discovery out of it. Discovery can be supported by the dynamic of building strategies around content generation, which in turn can be achieved through online and directly controllable approaches to PCG [22].

The initial game design derived through mechanics, dynamics and aesthetics serves as context for AI-afforded mechanics. The boxes containing context and affordances have been expanded to describe what game design and AI design provide for each other. The explanations serve as a reminder that the counterparts share a common goal of creating a coherent whole. In this way context informs AI design of what should be generated and within which constraints. Affordances on the other hand provide an explanation on how generation occurs within constraints.

The steps for AI design side (figure 4) represent a search-based approach. Based on the provided game context, content representation and data structures are created, evaluation functions designed, and solutions searched. After a generation pipeline has been established, the generator should be evaluated to determine whether produces content matching the designer's intent. The selected approach to PCG doesn't have to be search-based and can be substituted with other approaches as long as the resulting generator provides the necessary inputs and outputs, can be controlled to a reasonable degree, and, at least preferably, supports evaluating its attributes.

When the produced game system reaches a playable maturity level, it becomes increasingly important to study its behaviour and

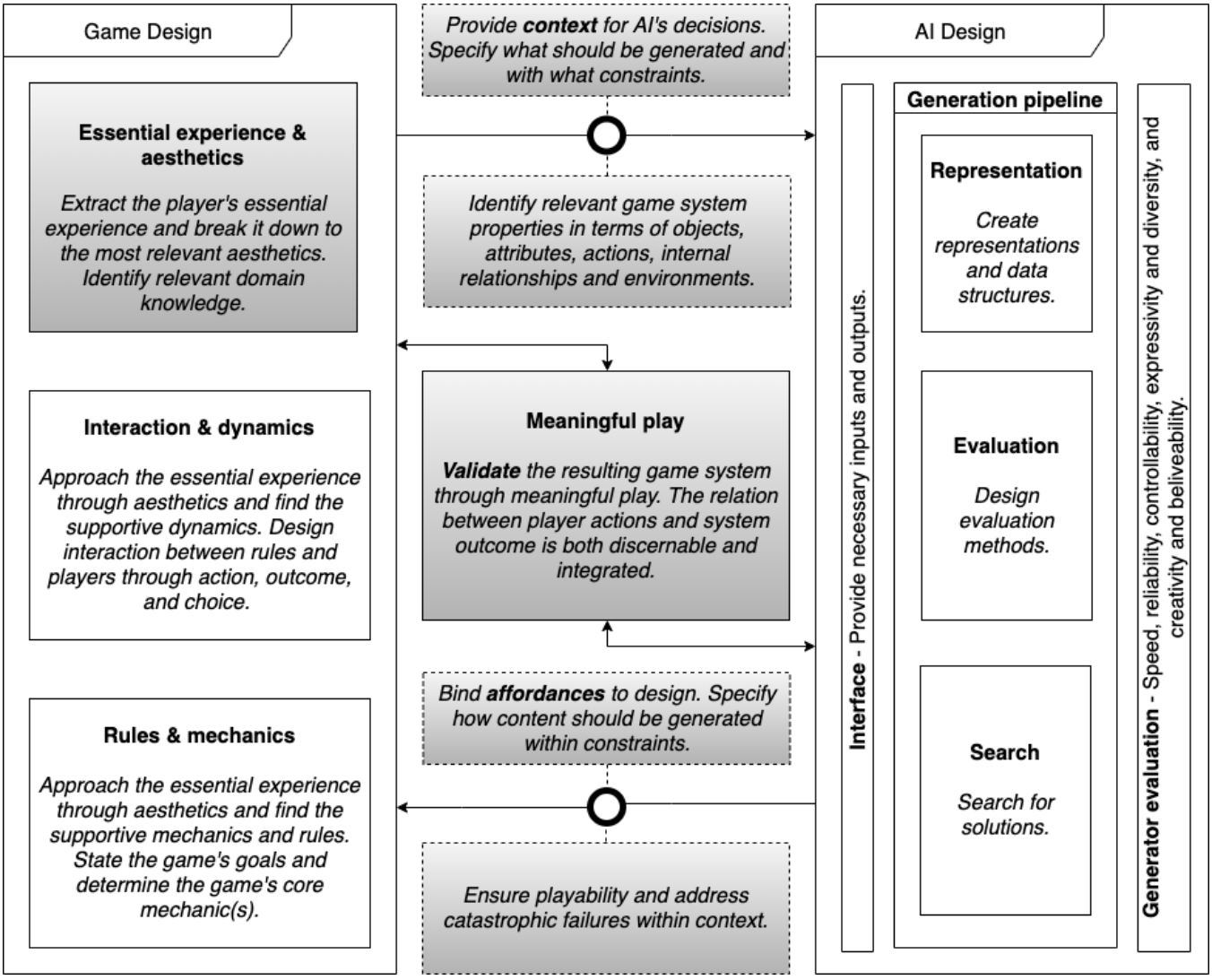

Figure 4: The extended PCG-based game design process features additions and changes, indicated with a gray color, that aim to help designers create early stage prototypes.

the resulting gameplay experience. The first stage prototype, the foundation, can be reached by first utilizing mockup content to ensure the game is mostly playable. When transitioning to the second prototyping stage, the structure, a functioning PCG-system is required to demonstrate core gameplay and core game mechanics. However, interaction with the resulting system should also support the wanted gameplay experience and produce meaning.

Meaning can be used to bring the afforded game mechanics closer to the game's context. As an example, the levels created by the prototype in section 3 suffered from a lack of meaning as they seemed disconnected from the game's overall frame of reference. The prototype featured a mechanic of stepping through colored portals that alter generator parameters affecting the structure of generated levels. The portals took the player to another level but it was not clear why a specific portal should be chosen or if the action had any effect due to the natural variance of generated levels. As an example of establishing meaning, the portal mechanic can be made more discernable – to better communicate the outcome of an action – by changing the color of floor, wall and enemies based on current location in the generator's search space [25]. Since all generated levels differ from each other slightly, the change in the system's state can be explicitly communicated by changing content coloring or through other suitable means. Additionally, the portal mechanic can be made more integrated – to better communicate how the action affects the rest of the game – by having the portals indicate how close the player is to the next goal, allowing players to gauge whether progress has been made. Furthermore, the goal indication would also assure the player that the goal will be eventually reached when enough portals have been traversed.

To summarize, the prototyped game system may consist of any configuration of generated content as long as interaction with it produces meaningful play and supports the wanted gameplay experience. Game design and AI design can be thought of as two parts that form a whole. Interaction with the created game system can be used as a validating mechanism as it should result in meaningful play and support the wanted gameplay experience. The more mature the prototype becomes, the more the system's behaviour and meaningful play matters. If the system does not support the wanted gameplay experience or create meaning, the two design counterparts can be reiterated and the generated content constrained to fit the game's overall frame of reference.

5 DISCUSSION

Inspired by the emergent problems of a PCG-based game prototyping process, this paper has presented actionable steps that may better allow designers to consider game context and AI affordances during prototyping efforts of PCG-based games. The results are presented as an extended version of the PCG-based game design process that attempts to explicate the first steps taken between the initiation and pre-production phases of game development, and provides a way to consider game context and AI affordances during prototyping efforts.

Traditional prototyping approaches can be ill-fitting for PCG-based games as integrating content generators can change the way prototypes are played [3]. The first prototyping stage, the foundation, aims to validate whether the current idea provides a good foundation for a game and is tested via mockups [8, 16], and may not yet require a functioning AI system. However, this approach serves to validate secondary gameplay, such as combat with enemies, without realizing the unique design affordances of PCG. The second prototyping stage, the structure, validates whether the functioning implementation works as intended. However, it might not be enough to only strive for functioning prototypes. Despite the prototype in section 3 working as intended on paper, the prototype did not support the wanted aesthetics and overall gameplay experience as the game's context and the provided affordances did not match. In addition to aiming for a functional prototype, an extra step should be taken towards synchronizing the game's context and the provided AI affordances.

The act of generating content in itself can be seen as somewhat trivial as the real challenge remains in generating meaningful content [3, 32]. Unless the generated content matches the game's overall frame of reference, it may feel disconnected from the rest of the game and negatively impact player experience. With PCG-based games, players take actions to influence content generation through game mechanics. The outcomes of these actions are experienced through the generated content instances. However, generated content is likely to better match the game's context when the core mechanics are designed to be both discernable and integrated. In other words, game context and AI affordances may be synchronized by communicating the effect of player's action in an understandable way and by communicating the action's outcome on the rest of the game.

The proposed extensions (figure 4) in the PCG-based game design process were added to help traverse the path from an idea to an implementation and to take the additional step towards meaningful play. Game design, providing context, and AI design, affording new mechanics, have a common goal to form the game's system. Interaction with this system through mechanics results in dynamics [10] and creates meaning through the relationship between player actions and the system's outcomes [17, chapter 3, p. 1-6]. After constructing a functional prototype, meaningful play acts as a validating mechanism for the resulting game system's behaviour. Through this mechanism, designers can constrain content generation to fit the game's overall frame of reference and help weigh the technical choices for different design approaches. By considering context and affordances through meaning early on, gameplay and design issues may be avoided further down the development process.

This paper takes part in the discussion on how game design practices can be examined in further detail through the relationship between game context and deeply integrated AI components. The proposed solution differs from previous research in both perspective and relation to the GDLC as a whole. Despite previous research's excellent work with PCG-based games and more mature prototypes, providing concrete tools for early stage prototypes might still provide additional insight on how traditional prototyping approaches fit PCG-based games. Furthermore, previous research has provided laudable discussion on meaningful content (e.g. [6, 24, 25]). However, meaningless content has been mostly discussed as a problem. In this paper, meaning has been seen as a tool that can be used to strive for better prototypes.

The problems presented in this paper are not claimed to be new or specific to PCG-based games. However, the presented problems may also be encountered in this domain early on during the prototyping process. Demonstrating core game mechanics and functionality in PCG-based prototypes is challenging as mockups fail to provide the necessary support required for player engagement with PCG. In the presented case, the design processes' open ended steps were found to be effective in guiding the first design iterations and validating the resulting system's behaviour through meaningful play. However, further empirical evidence is required to draw stronger conclusions.

For future research, the proposed extensions to the PCG-based game design process must be subjected to further evaluation as the artefact is a socio-technical solution and requires human interaction to prove its effectiveness [35] in solving problems. In the presented case, the design processes' open ended steps were found to be effective in guiding the first design iterations and validating the resulting system's behaviour through meaningful play. However, the next step in future research is to subject the proposed extensions to further empirical evaluation in a practical context. Further evaluation should determine how the extended process performs in PCG-based game prototyping and whether it helps in working with PCG during the design process. Additionally, the resulting prototypes should be used to study player engagement with prototype PCG systems and determine the role of meaning on improving player experience in PCG-based games. Lastly, further studies should be conducted on how traditional prototyping models fit PCG-based games and how meaning affects these games during the different stages of the GDLC. Despite limitations, this paper has offered insight on prototyping PCG-based games and on how

context, affordances and meaning can affect prototyping during the first steps of the GDLC.

6 CONCLUSIONS

PCG-based game design poses unique challenges for both game design and AI design. Originally, this paper's goal was to extract actionable steps for the PCG-based game design process. However, emergent design issues were faced during prototyping that regarded the unalignment of game design and AI design. This paper set out to explore these issues through the concept of meaningful play. The key discussion point of this study is that during prototyping game design and AI design can be thought to form a whole, and by interacting with the resulting game system, its functionality can be validated through meaningful play. This paper contributes to the discussion around PCG's role in game design by examining it from a non-traditional perspective. The results of this paper are presented as an extended version of the PCG-based game design process that features actionable steps and addresses the need to consider game context, affordances, and the game's overall design. The proposed solution can lower the required PCG-literacy for designers and enthusiasts alike in translating their ideas into functioning prototypes. However, the preliminary nature of the proposed solution calls for further and varied evaluations before being subjected to real world problems. Finally, this paper aids in facilitating discussion on the nuanced relationship between game design and deeply integrated AI elements, and how designers can formalize their ideas to prototypes in practice.

ACKNOWLEDGMENTS

This study is supported by the ITEA3 VISDOM project and Business Finland. The authors are grateful for this support as this study would not have been possible without it. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of ITEA3 or Business Finland.

REFERENCES

[1] Acornsoft. 1984. Elite.
[2] Saiqa Aleem, Luiz Fernando Capretz, and Faheem Ahmed. 2016. Game development software engineering process life cycle: a systematic review. *Journal of Software Engineering Research and Development* 4, 1 (2016), 6.
[3] Calvin Ashmore and Michael Nitsche. 2007. The Quest in a Generated World.. In *DiGRA Conference*.
[4] Blizzard Entertainment. 1997. Diablo.
[5] Michael Cook and Simon Colton. 2014. A rogue dream: Automatically generating meaningful content for games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.
[6] Mirjam P Eladhari, Anne Sullivan, Gillian Smith, and Josh McCoy. 2011. AI-based game design: Enabling new playable experiences. *UC Santa Cruz Baskin School of Engineering, Santa Cruz, CA* (2011).
[7] Epyx. 1980. Rogue.
[8] Tracy Fullerton. 2008. Game design workshop: a playcentric approach to creating innovative games. (2008).
[9] Erin Jonathan Hastings, Ratan K Guha, and Kenneth O Stanley. 2009. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games* 1, 4 (2009), 245–263.
[10] Robin Hunicke, Marc LeBlanc, and Robert Zubek. 2004. MDA: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, Vol. 4. 1722.
[11] Christopher M Kanode and Hisham M Haddad. 2009. Software engineering challenges in game development. In *2009 Sixth International Conference on Information Technology: New Generations*. IEEE, 260–265.
[12] Rilla Khaled, Mark J Nelson, and Pippin Barr. 2013. Design metaphors for procedural content generation in games. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1509–1518.
[13] Melanie Mitchell. 1998. *An introduction to genetic algorithms*. MIT press.
[14] Mojang. 2011. Minecraft.
[15] Nintendo. 1985. Super Mario Bros.
[16] Rido Ramadan and Yani Widyani. 2013. Game development life cycle guidelines. In *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 95–100.
[17] Katie Salen and Eric Zimmerman. 2004. *Rules of play: Game design fundamentals*. MIT press.
[18] Jesse Schell. 2008. *The Art of Game Design: A book of lenses*. CRC press.
[19] Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games*. Springer.
[20] Adam M Smith and Michael Mateas. 2010. Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 273–280.
[21] Adam M Smith and Michael Mateas. 2011. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 187–200.
[22] Gillian Smith. 2014. Understanding procedural content generation: a design-centric analysis of the role of PCG in games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 917–926.
[23] Gillian Smith. 2015. An Analog History of Procedural Content Generation.. In *FDG*.
[24] Gillian Smith, Elaine Gan, Alexei Othenin-Girard, and Jim Whitehead. 2011. PCG-based game design: enabling new play experiences through procedural content generation. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. 1–4.
[25] Gillian Smith, Alexei Othenin-Girard, Jim Whitehead, and Noah Wardrip-Fruin. 2012. PCG-based game design: creating Endless Web. In *Proceedings of the International Conference on the Foundations of Digital Games*. 188–195.
[26] Gillian Smith and Jim Whitehead. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. 1–7.
[27] Gillian Smith, Jim Whitehead, and Michael Mateas. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on computational intelligence and AI in games* 3, 3 (2011), 201–215.
[28] Gillian Smith, Jim Whitehead, Michael Mateas, Mike Treanor, Jameka March, and Mee Cha. 2010. Launchpad: A rhythm-based level generator for 2-d platformers. *IEEE Transactions on computational intelligence and AI in games* 3, 1 (2010), 1–16.
[29] Adam Summerville and Michael Mateas. 2016. Super mario as a string: Platformer level generation via lstms. *arXiv preprint arXiv:1603.00930* (2016).
[30] Ying Sun and Paul B Kantor. 2006. Cross-Evaluation: A new model for information system evaluation. *Journal of the American Society for Information Science and Technology* 57, 5 (2006), 614–628.
[31] .theprodukkt. 2004. .kkrieger.
[32] Julian Togelius, Alex J Champandard, Pier Luca Lanzi, Michael Mateas, Ana Paiva, Mike Preuss, and Kenneth O Stanley. 2013. Procedural content generation: Goals, challenges and actionable steps. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[33] Julian Togelius, Emil Kastbjerg, David Schedl, and Georgios N Yannakakis. 2011. What is procedural content generation? Mario on the borderline. In *Proceedings of the 2nd international workshop on procedural content generation in games*. 1–6.
[34] Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186.
[35] John Venable, Jan Pries-Heje, and Richard Baskerville. 2012. A comprehensive framework for evaluation in design science research. In *International Conference on Design Science Research in Information Systems*. Springer, 423–438.
[36] Georgios N Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing* 2, 3 (2011), 147–161.
[37] Georgios N Yannakakis and Julian Togelius. 2014. A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games* 7, 4 (2014), 317–335.
[38] Robert K Yin. 2017. *Case study research and applications: Design and methods*. Sage publications.
[39] Adeel Zafar and Hasan Mujtaba. 2012. Identifying catastrophic failures in offline level generation for mario. In *2012 10th International Conference on Frontiers of Information Technology*. IEEE, 62–67.