Automated Software Transplantation for Procedural Content Generation

María del Mar Zamorano López

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

September 18, 2025

I, María del Mar Zamorano López, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Game Software Engineering (GSE) is a specialized branch of Software Engineering dedicated to the development of video games. Unlike traditional software, video game development presents unique challenges, including the management of diverse artefacts, differing development paces, limited code reuse, and difficulties in automated testing. Research in GSE has grown significantly since 2010, with contributions appearing in top Software Engineering venues.

Video games are complex products that blend art and programming, requiring input from designers, programmers, and audio engineers. The industry has rapidly expanded, with a wide variety of titles driving economic growth and cultural influence. However, all games share a common challenge: the immense need for content. Large-scale productions require vast amounts of high-quality assets, leading to costly and time-intensive development cycles. Smaller teams, constrained by limited resources, must carefully balance ambition and feasibility, while games designed for specialized purposes are restricted by domain-specific content requirements.

To address these challenges, researchers have explored Procedural Content Generation (PCG), an approach that automates or semi-automates the creation of game assets. This thesis introduces a novel technique, Procedural Content Transplantation (PCT), which applies Software Transplantation principles to PCG. In software, transplantation involves extracting functional components from one system and integrating them into another. Traditionally, it has been used for program repair, testing, security, and functionality enhancements. Applied to content generation, PCT extracts functional content from a game to integrate it into another content of a game.

PCT employs a search-based algorithm to identify and integrate suitable content fragments from existing games into new contexts, mitigating the challenges of manual content creation. To validate this approach, we designed and implemented the first search-based transplantation algorithm for video game content. Our evaluation, conducted in collaboration with a commercial game studio, assessed the effectiveness of PCT in generating non-playable characters (NPCs) and compared its results with a standard PCG method from the literature.

Impact Statement

The work presented in this thesis has an impact in the field of Game Software Engineering research, more precisely on Procedural Content Generation.

This research introduces Procedural Content Transplantation (PCT), a novel technique that leverages Software Transplantation principles to enhance Procedural Content Generation (PCG). By automatically identifying and integrating existing content fragments into new game contexts, PCT reduces the manual effort required for asset creation, lowering development costs and accelerating production timelines.

The development and evaluation of a search-based transplantation algorithm mark a significant advancement in game development methodologies. In collaboration with a commercial game studio, this approach was tested on non-playable character (NPC) generation, demonstrating its effectiveness compared to traditional PCG methods. The results suggest that PCT can improve content diversity and quality while maintaining design coherence, offering practical benefits for developers across the industry.

By bridging the gap between software engineering and creative game design, this research contributes to the broader field of automated game development. It opens new possibilities for scalable, efficient content production, making high-quality game creation more accessible to developers with varying resources. As PCT continues to evolve, it has the potential to shape future industry practices, supporting both large studios and independent developers in delivering richer, more engaging gaming experiences.

Acknowledgements

Completing this PhD has been a challenging yet incredibly rewarding journey, and I would not have reached this milestone without the support and encouragement of many individuals.

First and foremost, I would like to express my heartfelt appreciation to my family. Specially to my parents, Ángel and Esther, for their unwavering belief in me and constant encouragement, and also for their patience and understanding. They have been my rock since I can even remember, and without them I would not be here. Their endless love is what has made who I am.

I would like to express my deepest gratitude to my advisors, Federica Sarro and Carlos Cetina, for their unwavering guidance, invaluable insights, and continuous support throughout this process. Their expertise and mentorship have been instrumental in shaping both this research and my development as a scholar. Also, I would like to thank you for the emotional support through this journey, and believing always in me even when I could not.

A special thank you goes to my colleagues and friends at SOLAR group at UCL and SVIT group at USJ. The stimulating conversations, shared challenges, and camaraderie have made this journey all the more fulfilling. I am especially grateful to Raúl, África, Carol, Thanatad, Song, and Dave for their friendship and support.

On a personal note, I extend my heartfelt appreciation to my partner, Gvidas, and friends. I have fallen many times not only during my thesis but during my life, and you have always been there for me. All your love, support, and encouragement have been my anchor throughout this journey. To my closest friends, Laura and Esther, I hope you know how important you are to me, and I am deeply grateful for

having you and your families.

This thesis is as much a product of my efforts as it is of the collective support I have received, and for that, ‘thank you’ is not enough.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Game Software Engineering (GSE) [10] is a branch of Software Engineering which focuses on the development of video games. GSE specifically tackles the challenges arising from the fact that video games present characteristics that differentiate their development and maintenance from the development and maintenance of classic software [11, 12]. For example, developers contribute to video games vs. non-games by working on different kind of artefacts, develop and maintain artefacts at a difference pace [11], find it more difficult to reuse their code [11] and adopt automated testing [12]. The interest in GSE research has witness a continuous growth since 2010 [13], with several GSE work published in the most renewed SE venues including ICSE, FSE, MODELS, ASE [14].

Video games are complex software products where art and coding are combined during the development process to conform the final product. The industrial scene of video games development has seen a rapid expansion in the last decades, with video games becoming an economic motor and a worldwide phenomenon that has captured the general interest of all sectors of the population globally [15].

The industry of video games development can be split into three main different branches: AAA games, indie games, and serious games. The main players in the field develop AAA video games, which are produced and distributed by mid-sized or major publishers [16]. These games are typically more complex than other types of games, and have high development and marketing budgets, while being mainly constrained by time factors in the form of deadlines and firm release dates. On the

other side of the spectrum, it is possible to find the so-called indie games, created by individuals or smaller video game studios that lack the financial and technical support of a large game publisher. While indie games benefit from unbridled creativity, they often struggle with funding and issues typically associated to small development teams such as having a constrained extension or undesired functionality due to limited testing. Serious games are designed not only to entertain, but to also instruct, inform, or educate the players. We can find examples of serious games in Non-Governmental Organizations that develop games to raise awareness about societal issues, medical research associations that do so to simulate complex operations, or top-valued tech companies that use gamification to instruct new recruits in their processes. These games are usually restricted to the scope of their respective domain, and can vary wildly in their goals, complexity, and development times.

Regardless of the differences in their development goals and processes, all these games share a common issue in the form of their *need for content*. Nowadays, the creation of video games is a creative process that involves experts from many different crafts, including artists, designers, programmers, and audio engineers, among many other disciplines. Without the aesthetic components that create immersive game environments, the music scores and sound effects that evoke feelings and create memorable game experiences, or the interactive elements that the players can use as tools within the game, there can be no game at all.

Of all games, AAA games are usually the ones that require the most content both in amount and variety, often resulting in a long, manual, repetitive, and very costly content creation effort that is carried out by large amounts of game developers. On the other hand, in indie games, content creation is a delicate process constrained to the capabilities of the team. While being too ambitious with game content can cause delays in a project or even result in failure to publish a game, being too lax with the content can render a game unsuccessful. For small game studios, as it may happen with any start-up company, both scenarios can be deadly due to the limited available funding. Serious games are in the same spot as indie games with regard to their limits in content creation, but often for different reasons. In the case of serious

games, their very nature narrows their scope and their target players, and restricts the creation of content to that which is strictly necessary within the domain of the application. While this may be seen as beneficial, it is more often than not a severe hindrance in development, since the specificity of content often calls for unique solutions involving usually unavailable domain experts.

To tackle these development challenges, researchers have focused their efforts towards Procedural Content Generation [17], a field of work that has gained traction in the last decades. **Procedural Content Generation** (**PCG**) [1] is defined as the automated production of different components of video game. Through PCG, video game software engineers generate components and aspects of video games in an automated or semi-automated fashion, which is a more competitive approach than doing so through the traditional manual development process.

This thesis address the *need for content* from a new perspective applying Software Transplantation into Procedural Content Generation, which we named Procedural Content Transplantation (PCT). In medicine, *transplantation* is a procedure in which cells, tissues, or organs of an individual are replaced by those of another individual, or the same person [18]. In software, researchers understand transplantation as a procedure in which a fragment (organ) of a software element (donor) is transferred into another software element (host) [19]. Software transplantation has been successful for different tasks: program repair [20, 21], testing [22], security [23], and functionality improvements [24].

We rely on a search-based algorithm as the process of transplantation involves connecting the boundaries of the donor organ with those of the host, and with multiple potential connection points available from the donor, the search space created by the combination of boundaries and host is simply too vast to be thoroughly explored through brute force methods. Moreover, search-based approaches have been successfully applied for traditional software transplantation [19].

To show the viability of procedural content generation via transplantation we designed and realised the first ever search-based transplantation algorithm for video-game procedural content, dubbed IMHOTEP, and evaluated its effectiveness in

generating non-playable characters (NPCs) for the commercial video-game Kromaia. Our approach is named after IMHOTEP, who is considered by many to have written the Edwin Smith Papyrus (the oldest known manual of surgery).

To evaluate our proposal we have carried out an industrial case study in collaboration with the developers of the commercial video game Kromaia[1], and we have empirically assessed and compared the content generated by our approach and a PCG technique from the literature.

1.1 Motivation for Transplantation

While Procedural Content Generation (PCG) offers automated means to address the increasing demand for game content, most existing approaches rely on generating new artefacts from scratch. These methods often struggle to balance quality, diversity, and believability, particularly in content-heavy domains such as AAA or serious games. Moreover, many PCG techniques operate within tightly constrained rule systems or generative grammars, which can inadvertently limit the richness of the produced artefacts compared to manually crafted ones.

Transplantation provides a complementary perspective by leveraging existing game content as reusable building blocks. Instead of reinventing content anew, transplantation transfers functional or aesthetic fragments from a donor artefact into a host artefact, enabling the creation of novel yet coherent game elements. This process offers several advantages:

- **Reuse of Proven Content**: Borrowing from content that has already been designed, tested, and integrated into a game ensures higher quality and reliability compared to generating completely new assets.

- **Reduction of Development Costs**: By transplanting fragments, studios can mitigate the high costs of manual asset creation, especially in contexts with limited resources such as indie or serious game development.

- **Increased Variety**: Transplantation allows recombination of existing artefacts

[1] See the official PlayStation trailer to learn more about Kromaia: `https://youtu.be/EhsejJBp8Go`

in unforeseen ways, expanding the design space while preserving consistency within the game world.

- **Empowering Developers**: Unlike black-box generative methods, transplantation is not only an automated process but also a tool for developers. It gives them greater power over the generation process by allowing them to guide what gets reused, where it is placed, and how it is adapted. This balance between automation and human oversight ensures that the resulting content aligns with the creative vision of the development team.

- **Bridging Manual and Automated Processes**: Unlike purely generative methods, transplantation strikes a middle ground between handcrafted content and automated generation, enabling developers to retain creative control while benefiting from automation.

In essence, transplantation reframes the challenge of PCG: rather than asking "how do we generate new content?", it asks "how can we adapt and reuse existing content to create something new?". This shift aligns well with the realities of video game development, where assets are plentiful but costly to produce, and where creativity often emerges from the recombination of familiar patterns.

1.2 Contributions

The main contributions of the current PhD research are:

1. A comprehensive survey about the current state of Search-Based PCG (SBPCG) that also provides insights for future directions in this field (Chapter 3).

2. IMHOTEP, a Procedural Content Transplantation technique (Chapter 4).

3. An empirical study on the use of IMHOTEP in an industrial video game (Chapter 5).

4. A controlled experiment with humans to assess IMHOTEP and a PCG technique (Chapter 6).

1.3 Publications

The following publications are the work presented in this PhD thesis:

1. **The Quest for Content: A Survey of Search-Based Procedural Content Generation for Video Games** currently under review in ACM Surveys [2] and available on arXiv [25];

2. **Game Software Engineering: A Controlled Experiment Comparing Automated Content Generation Techniques** [26] was published in ESEM 2024 Technical Papers Track (Chapter 6), and received the *Best Paper Award* on its category. It has also won the *Best Video-Presentation* on the '2ª Happy Hour de la Red AI4Software' from Red de Investigación en Inteligencia Artificial Aplicada al Proceso de Desarrollo Software;

3. **Video Game Procedural Content Generation Through Software Transplantation** [27] was published in ICSE (SEIP) 2025 (Chapter 4, 5);

During my PhD I undertook additional research that is not part of this thesis:

1. **Evaluating Explanations for Software Patches Generated by Large Language Models** [28] was published in SSBSE 2023.

2. **Search-based Negative Prompt Optimisation for Text-to-Image Generation** [29] was published in EvoMUSART 2025.

1.4 Roadmap

This thesis is organized as following: Chapter 2 discuss the relevant background knowledge for our automated software transplantation in procedural content generation work. Chapter 3 presents the literature review undertaken as part of this thesis on Search-based Procedural Content Generation. Chapter 4 presents our automated software transplantation approach. Chapter 5 presents an empirical evaluation of our approach. Chapter 6 presents our controlled experiment comparing automated content generation techniques. Chapter 7 discusses the outcomes of the thesis and future lines of work, thus concluding the content for this thesis.

[2] https://dl.acm.org/journal/csur

Chapter 2

Background

This chapter provides the reader with an overview of relevant topics for our search-based transplantation algorithm for video-game procedural content.

First, we put our work in the context of Procedural Content Generation (Section 2.1), and more precisely into Search-Based Procedural Content Generation (Section 2.2). Next, we introduce the state-of-the-art of Automated Software Transplantation (Section 2.3).

Finally, we provide some clarifications about Video-game Development (Section 2.5) to understand Kromaia. Kromaia (Section 2.6) is the industrial video-game used for the empirical study and the controlled experiment of this thesis.

2.1 Procedural Content Generation

Procedural Content Generation (PCG) refers to the automation or semi-automation of the generation of content in video games. By content, we refer to every aspect of a game. This definition is broad given the large amount of content that a game usually needs, starting from the environment till the inner system logic of the game. To that extent, the literature groups the content in different content types [1]; game bits, game space, game system, game scenarios, game design, and derived content. Game bits, game space, game system, game scenarios, and game design, refer to elements inside the game such as vegetation [30] or sound [31] (game bits), environment or terrain [32] (game space), Non-Playable Characters [33] (game system), levels or puzzles [34, 35] (game scenarios), and rules or restrictions [36] (game design).

Derived content, on the other hand, is all the content that are generated because of a game, like videos of player experiences playing a game.

PCG can be carried out in two different manners, corresponding to two different stages of the lifetime of a game. Hence, offline PCG refers to content generated before the release of a game (at design time), and online PCG refers to content generated on the fly while the game is being played (at run-time).

PCG is a large field spanning many algorithms [37], which can be grouped in three main categories according to the survey of PCG techniques by Barriga et al. [38]: Traditional methods [39] that generate content under a procedure without evaluation; Machine Learning methods (PCGML) [40, 41, 42] that train models to generate new content; and Search-Based methods (SBPCG) [1, 2] that generate content through a search on a predefined space guided by a meta-heuristic using one or more objective functions.

Our work falls in the SBPCG category and it generates content of the NPC type. In the context of NPC generation using SBPCG, Ripamonti *et al.* [43] developed a novel approach to generate monsters adapted to players, considering the monster with more death rate the preferred by the player. To evaluate the monsters, they recreated an environment with the main aspects from a MMORPG [1] game. Pereira *et al.* [44] and later extended by Viana *et al.* [33] seek for generating enemies that meet a difficulty criteria. Pereira *et al.* and Viana *et al.* use the same research academic game in their experimental designs. Blasco *et al.* [45] focuses on generating spaceship enemies that are comparable to the ones manually created by developers. To generate spaceships, Gallota *et al.* [5] used a combination of Lindenmayer systems [46] and evolutionary algorithm. Gallota *et al.* as well as Blasco *et al.* use a commercial video game in their evaluation.

In the context of ML, to the best of our knowledge there is a gap in the generation of NPC. ML research focus on other aspects of video games, such AI [47] or graphical aesthetics [48]. The motivation of our work comes from the limitations that we detected in previous work. Previous work focused on speeding up development

[1] Massive Multiplayer Online Role-Playing Games

Table 2.1: Definitions of the key components for Search-Based Procedural Content Generation (SBPGC).

	Name	Short Description
Encoding	Direct	Less genotype-to-phenotype complexity conversion
	Indirect	Requires human effort on creating the conversion system genotype-to-phenotype
Objective Function	Direct - Theory Driven	The developers assess with their opinion to elaborate the objective function
	Direct - Data Driven	The objective function is based on information about relevant parameters extracted from artefacts
	Simulation - Static	The simulator agent does not change during the simulation
	Simulation - Dynamic	The simulator agent learns during the simulation
	Interactive - Implicit	Players are outright asked for their opinions
	Interactive - Explicit	Data is indirectly extracted or inferred from the observation of the actions of the players and the results of those actions

time. However, the influence of the developers on the generated content was limited. The generated content depended on randomness resulting on generated content not aligned with the intention of the developers. As a result, the generated content was either not used or used as secondary content.

2.2 Search-Based Procedural Content Generation

Search-Based Procedural Content Generation is a field of the more general Search-Based Software Engineering (SBSE) research area [49, 50] .

SBSE seeks to reformulate Software Engineering problems as 'search problems'. Given a search space of a particular problem, a search-based approach can look for an optimal or near optimal solution within a set of candidate solutions. The next paragraphs describe the most commonly used algorithms in SBSE (which are also summarised in Table 2.1), and how the two key ingredients of SBSE (namely representation and objective function) have been being applied in SBPCG.

Within the main algorithms used for SBSE, we find local search algorithms,

single-objective evolutionary algorithms, and multi-objective evolutionary algorithms [49, 50]. Local search algorithms receive a set of candidate solutions as input, and then determine one of the candidate solutions as the starting point for the search. In order to perform the search, they then iteratively move to a neighbour solution which slightly differs from the current solution and evaluate the new candidate solution according to the fitness function. Single-objective evolutionary algorithms use mechanisms inspired by the Darwinian concept of evolution. Evolutionary algorithms apply genetic operations such as the crossover or mutation of individuals over the candidate solutions to obtain new populations of candidate solutions, which are then evaluated according to a fitness function that targets a single objective. Multi-objective evolutionary algorithms work through the same Darwinian principles and genetic operations to search for the best candidate within a search space, but evaluate the candidate populations according to objective functions that consider more than one goal.

A search-based approach needs a representation of the particular problem that an algorithm can understand to perform the search (*i.e.* encoding). Regarding the representation of the problem, we find two main components, the *genotype* and the *phenotype*. The *genotype* is the data structure that the algorithm will process, and the *phenotype* is the data structure that will handle the evaluation part of the search. In other words, a phenotype is a conversion from a genotype. Based on the difficulty of the conversion from genotype to phenotype, we refer to 'direct encoding' or 'indirect encoding'. A direct encoding implies less genotype-to-phenotype complexity conversion than an indirect encoding, which requires a more complex conversion system. While an indirect encoding requires more human effort on creating the conversion system, it also provides freedom to represent the content. In SBPCG, an example of a direct encoding is a grid where each cell represents an element of the content. An evolutionary algorithm is capable of interpreting the grid and evolve it, and the final phenotype can be extrapolated just by looking at the genotype. An example of indirect encoding is a list of the details of the different elements that compose the content. A vector representing that list of details is likely to be evolved by an

evolutionary algorithm. However, there is a need for interpreting the details of the elements in the vector to generate the phenotype.

Finally, a search-based approach needs an objective function (or fitness function) that guides the search towards an optimal solution.

Regarding the objective function, SBPCG differentiates between three different types [2]: direct, simulation, and interactive. Direct objective functions are those that are based on the available knowledge of developers (that is, the developers themselves participate in the assessment of the objective function). Direct objective functions can be either theory-driven (meaning that the opinion of the developers is directly leveraged) or data-driven (meaning that information about relevant parameters is extracted from artefacts like questionnaires or player models). Simulation objective functions replicate real situations to estimate the behaviour of real players. Work in this area focuses mainly on developing more human-like agents, bots, and AIs to be used by objective functions. Simulation objective functions can be static, where the simulator agent does not change during the simulation, or dynamic, where agents that learn during simulation are used. Finally, interactive objective functions are those that involve players in the composition of the objective function. Incorporating human expertise in the objective function constitutes a broad research area itself, named human-in-the-loop [51, 52], which studies how to incorporate humans into the algorithm process. In SBPCG, interactive objective functions can be either explicit, when players are outright asked for their opinions, or implicit, when the data is indirectly extracted or inferred from the observation of the actions of the players and the results of those actions.

2.3 Automated Software Transplantation

Automated Software Transplantation (AST) is a technique that involves extracting functional components from one software system and integrating them into another. This process allows the reuse of code across different applications while handling differences in dependencies, architectures, and execution environments. AST has been applied for automated program repair, testing, security and functionality im-

provements.

On functionality transplantation, Miles *et al.* [53] and Petke *et al.* [54] proposed the first approaches that transplant software code in a same program (assuming that different versions of the programs are considered a same program). This seminal work has inspired follow up research to perform Automated Software Transplantation between different programs [19], or even different programming languages [55] and platforms [56], as summarised below. When transplanting within a same program, there is no need for adapters (*i.e.* alterations in organ or host to adapt the organ to fit into the host).

Sidiroglou-Douskos *et al.* [57] proposed a technique that divides the donor program by specific functionality, each piece is called a 'shard'. The approach insert the shard into the host without modifications, that is, the work from Sidiroglou-Douskos does not use adapters either.

On the other hand, Maras *et al.* [58] proposed a three step general approach, without implementing it, which applies feature localization to identify the organ; then code analysis and adaptation, and finally feature integration.

Wang *et al.* [59] instead of using feature localization, takes as inputs the desired type signature of the organ and a natural language description of its functionality. With that, the approach called Hunter uses any existing code search engine to search for a method to transplant in a database of software repositories. Further, Hunter generates adapter functions to transform the types from the desired type signature into the type signatures of the candidate functions.

Allamanis *et al.*'s SMARTPASTE [60] takes the organ and replace variable names with holes, the approach using a deep neural network fills the holes. Allamanis *et al.* [60] use Gated Graph Neural Networks [61] to predict the correct variable name in an expression.

Unlike Allamanis *et al.*, who puts holes into the organ, Lu *et al.* [62] introduced program slicing where the host is provided with a draft of the code with holes, or natural language comments. Similarly to Wang *et al.* [59] , program splicing looks into a database of programs to identify a relevant code to the current transplant task.

Finally, the approach selects the more suitable result found to fill the holes in the draft.

Barr *et al.* propose μSCALPEL [19], an automatic code transplant tool that uses genetic programming and testing to transplant code from one program to another. μSCALPEL uses test cases to define and maintain functionalities, small changes are made to the transplanted code, and code that does not aid in passing tests can be discarded, reducing the code to its minimal functioning form.

Subsequently, Marginean *et al.* proposes τSCALPEL [55] to achieve the transplantation between different programs and programming languages. Kwon *et al.* propose CPR [56] that transplants an entire program between different platforms. CPR realizes software transplantation by synthesizing a platform independent program from a platform dependent program.

To synthesis the platform independent program, CPR uses PIEtrace [63] to construct a set of trace programs, which captures the control flow path and the data dependencies observed during a concrete execution, and replaces all the platform dependencies with the concrete values that it observed during the concrete execution. Finally, CPR merges all these trace programs together to handle any input, by replacing the concrete values observed during the executions, with input variables.

2.4 General Software Development vs Game Software Development

From a software engineer's perspective, general software development and game development share foundational skills—like coding, debugging, and design patterns, but they differ significantly in goals and workflows. General software is typically built to solve real-world problems, prioritizing functionality, usability, and long-term maintainability. Whether it's a banking app or a logistics dashboard, the focus is on delivering reliable tools that perform clearly defined tasks, often following standardized architectures like MVC or microservices.

Game development, in contrast, aims to create immersive and emotionally engaging experiences. This shift in purpose demands a real-time performance

focus, which adds unique constraints around graphics, physics, and user input. Game developers rely on specialized engines like Unity or Unreal, use lower-level programming techniques for optimization, and architect systems around game loops and data-driven models such as Entity-Component-System (ECS). The work blends technical skill with artistic sensibility, where changes are often driven by gameplay feel rather than functional requirements.

Testing and collaboration also differ greatly. General software emphasizes automated testing—unit, integration, and end-to-end—to ensure reliability and regressions. In game development, testing is more experiential, relying heavily on manual playtesting, balancing, and user feedback. Furthermore, game development is highly interdisciplinary: engineers work closely with artists, sound designers, and narrative teams to bring the game world to life, requiring tools and systems to accommodate rapidly evolving creative input.

Overall, while both domains require disciplined engineering, game development introduces unique challenges that revolve around performance, storytelling, user experience, and cross-functional creativity. General software is built to be stable and scalable; games are built to be felt. A software engineer moving between the two must shift not only their technical focus but also their mindset—from solving problems to shaping experiences.

2.5 Model-Based Game Software Development

Video games are pieces of software that, like any other software, need to be designed, developed, and maintained over time. However, there are some particularities of video games that make them differ from traditional software, such as the artistic component of the video-game, the complexity of the rendering pipelines, the heterogeneous nature of video game development teams, and the abstract nature of the final purpose of a video game: fun [11, 64]. Hence, video games present characteristics that differentiate their development and maintenance from the development and maintenance of classic software. Examples of these differences can be found in how video game developers must contribute to the implementation of different

kinds of artifacts (e.g., shaders, meshes, or prefabs) or in the challenges they face when locating bugs or reusing code [11, 64]. Nowadays, most video games are developed by means of game engines. Game engines are development environments that integrate a graphics engine and a physics engine as well as tools for both to accelerate development. The most popular ones are Unity and Unreal Engine, but it is also possible for a studio to make its own specific engine (e.g., CryEngine [65]).

One key artefact of game engines are software models. Unreal proposes its own modeling language (Unreal Blueprints) [66], Unity proposes Unity Visual Scripting [67], and a recent survey in Model-Driven Game Development [68] reveals that UML and Domain Specific Language (DSL) models are also being adopted by development teams. Developers can use the software models to create video game content instead of using the traditional coding approach. While code allows for more control over the content, software models raise the abstraction level, thus promoting the use of domain terms and minimizing implementation and technological details. Through software models, developers are freed from a significant part of the implementation details of physics and graphics, and can focus on the content of the game itself (see Figure 2.1).

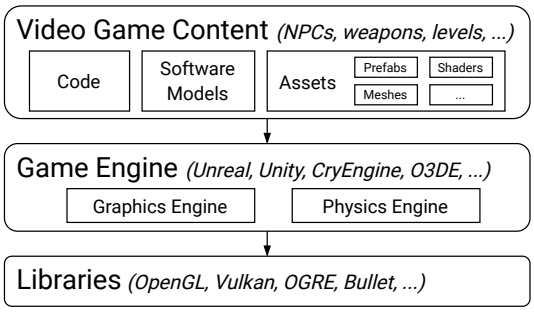


Figure 2.1: Overview of video game artefacts.

One of the challenges in software development is the environment used, as each environment and programming languages has unique characteristics. Software models, and more precisely Model Driven Engineering, study how to alleviate this problem by approaching software development from a platform-independent perspective through models. Video game developers must deal with this challenge as well and has motivated the research that combine software models and the domain

of video games.

The 2010 survey of Software Engineering Research for Computer Games [13] identified only one work that applied Model-Driven Development to video games [69]. That work coined the term "Model-Driven Game Development" and presented a first approach to 2D game prototyping through Model-Driven Development. Specifically, they used UML classes and state diagrams that were extended with stereotypes, and a model-to-code transformation to generate C++ code.

More recent work presents work that intended to minimize errors, time, and cost in multi-platform video game development and maintenance [70, 71, 72], or suggest the use of business process models as the modelling language for video games [73].

In the intersection between software models and evolutionary computation, Williams *et al.* [74] use an evolutionary algorithm to search for desirable game character behaviours in a text-based video game that plays unattended combats and that outputs an outcome result. The character behaviour is defined using a Domain-Specific Language. The combats are managed internally and are only driven by behaviour parameters, without taking into account a spatial environment, real-time representation, or visual feedback (which takes into consideration the physical interaction of the characters, variation in the properties, etc.).

Another work that focuses on the intersection between software models and evolutionary computation is Avida-MDE [75], which generates state machines that describe the behaviour of one of the classes of a software system (Adaptive Flood Warning System case study). The resulting state machines comply with developer requirements (scenarios for adaptation). Instead of generating whole models, Avida-MDE extends already existing models (object models and state machines) with new state machines that support new scenarios. The work in Goldsby and Cheng *et al.* [75] does not report the size of the generated state machines; however, the ones shown in the paper are around 50 model elements, which is significantly smaller than the more than 1000 model elements of the models of a commercial video game such as Kromaia.

The work mentioned above focus on generating new content from models,

which differs with our proposal of using MDE to transplant model fragments between models.

2.6 The Kromaia Video Game

In this section we introduce Kromaia, the commercial video game investigated, in collaboration with its developers, in the empirical study and the controlled experiment presented in this thesis.

Each level of Kromaia consists of a three-dimensional space in which a player-controlled spaceship has to fly from a starting point to a target destination, reaching the goal before being destroyed. The gameplay experience involves exploring floating structures, avoiding asteroids, and finding items along the route, while basic enemies try to damage the spaceship by firing projectiles. If the player manages to reach the destination, the ultimate antagonist corresponding to that level (which is called *boss*) appears and must be defeated in order for the player to complete the level. Kromaia's boss is the NPCs target content that we aim to automatically create via PCT. Bosses can be built either using C++ code or software models. The upper part of Figure 2.2 depicts a boss fight scenario in which the player-controlled ship (item A in the figure) is battling the NPC Serpent (item B in the figure), which is the final boss that must be defeated to complete Level 1. The bottom part of Figure 2.2 illustrates the two possible development approaches for the Serpent boss (model-driven Vs. code-centric).

Even though Figure 2.2 shows excerpts of the implementation of the Serpent both in the form of software models and code, it is not necessary to realize both in order to implement this content. Developers can mix both technologies by developing different parts of the boss using one or the other approach indistinctly, but they are also free to implement the content using software models exclusively or to do so purely via code. However, the heterogeneous nature of video game development teams - comprised mainly of programmers [76], but also counting game designers, artists, UI designers, and QA engineers within their ranks - possibly favours the use of software models over code thanks to the higher abstraction level of the

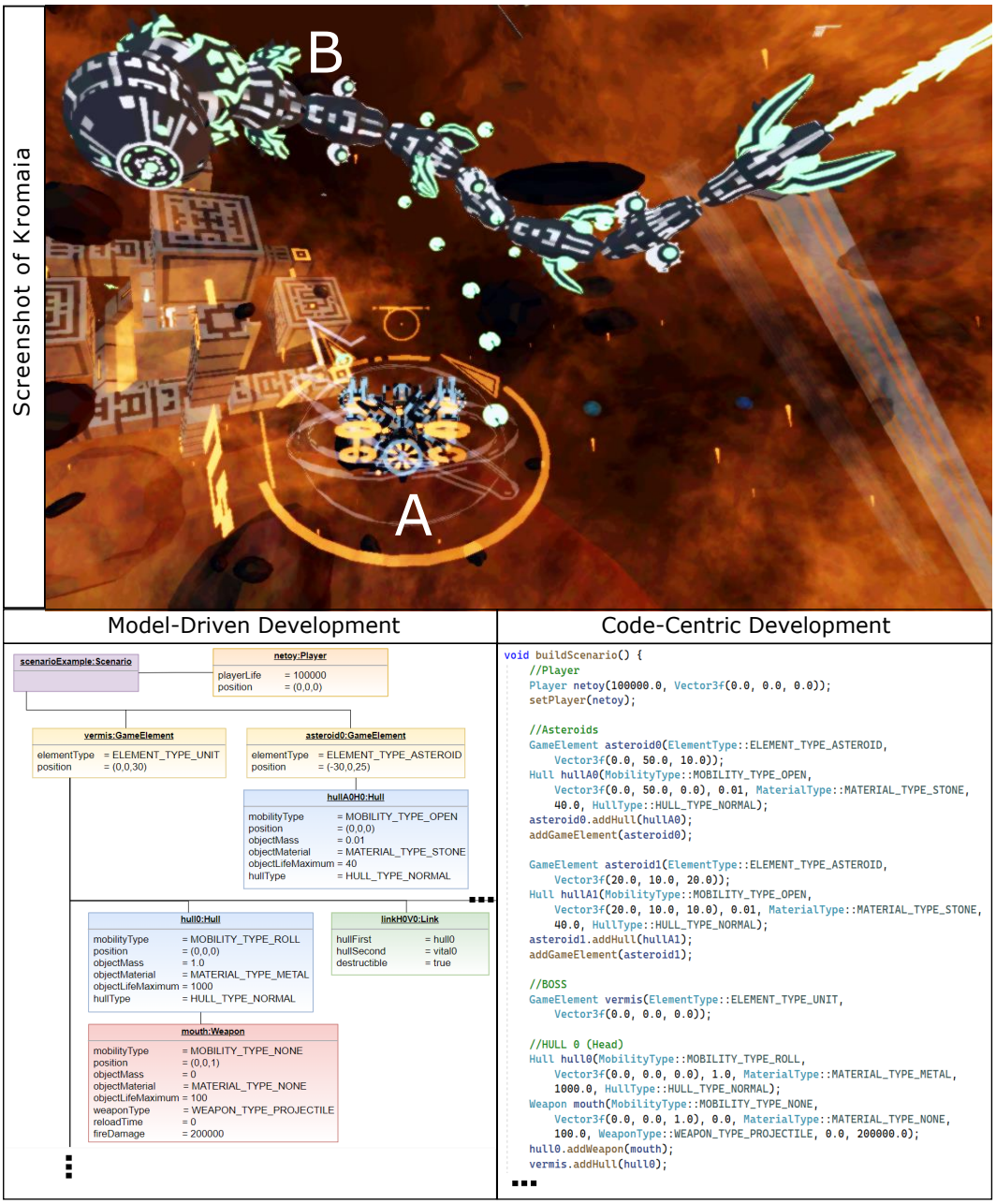


Figure 2.2: Model-Driven Development vs. Code-Centric Development in the context of Kromaia

former (combined with their detachment from more technical implementation details) which empowers less tech-focused roles to embrace a more active participation in development tasks. Also, previous work [77] showed that video game developers make fewer mistakes and are more efficient when working with models rather than code.

In Kromaia, the elements of the game are created through software models,

and more specifically, through the Shooter Definition Model Language (SDML). SDML is a DSL model for the video game domain that defines aspects that are included in video game entities: the anatomical structure (including their components, physical properties, and connections); the amount and distribution of vulnerable parts, weapons, and defences; and the movement behaviours associated to the whole body or its parts. SDML has concepts such as hulls, links, weak points, weapons, and AI components, and allows for the development of all types of video game content, such as bosses, enemies, or environmental elements. The models are created using SDML and interpreted at runtime to generate the corresponding game entities. In other words, software models created using SDML are translated into C++ objects at runtime using an interpreter integrated into the game engine [45]. More information on the SDML model can be found on-line at `https://youtu.be/Vp3Zt4qXkoY`.

2.7 Conclusion

To the best of our knowledge this thesis propose the first automated software transplantation approach in the field of content generation for video games. Our proposal allows the transplantation between different types of content. More precisely, in this thesis, we transplant organs from scenarios to NPCs. We have demonstrated that in this context a simulation-based objective function yield superior outcomes compared to the test-based objective function that previously attained the most favourable results in traditional software engineering transplantation (μSCALPEL [55]).

A previous work from us also generated NPCs using SBSE [45], speeding up development time. However, in our previous work the influence of the developers on the generated content was limited. The generated NPC depended on randomness resulting on generated NPCs not aligned with the intention of the developers. As a result, the generated content was either not used or used as secondary content. In fact, the limitations of our previous work were the inspiration for moving to transplantation. The transplant-based approach of this work keeps control in the hands of the developers (who choose the organ to transplant) and helps to explore

the latent content that exists in the video game.

Our research introduces a fresh perspective on content generation through the use of transplantation, which sets it apart from traditional procedural content generation (PCG) methods. Transplantation enables the seamless integration of various content types, facilitating in our work the transplant of elements from scenarios to NPCs.

Chapter 3

Literature Review

3.1 Introduction

This chapter surveys the current state of **Search-Based PCG** (**SBPCG)** and provide insights for future directions in this field. The ultimate aims is to encourage further applications of Search-Based Software Engineering to Game Software Engineering as we may have just scratched the surface of its application in this area. In the following subsection we further detail the scope of our survey.

3.1.1 Scope

Traditional methods include diverse methods, such as pseudo-random number generators, cellular automata, generative grammars, fractals, or noise. Developers must design an approach for a specific type of content which produces useful elements for that type of content without an evaluation or a learning process. Thus, TM do not follow diverse content, and require human domain-specific knowledge. The generation of vegetation is probably the most successful case of video game PCG through Traditional Methods, with the development of tools such as SpeedTree [78]. The usage of SpeedTree in games, both AAA games and indie games, is widespread. SpeedTree was first launched in 2002, and since then there has been no similar successful tool. Traditional methods have mostly been used for offline generation; we found no references to TM online generation. The fact that they have been restricted to offline use hinders their use for replay-ability[1].

[1]The potential for continuing playing after the first completion of a game.

Machine Learning methods train models to generate content based on training data, thus reducing the need for human domain-specific knowledge. ML online generation is still not widespread, most of the work on ML is offline generation. The use of ML methods for PCG (PCGML) was recently reviewed in 2018 by Summerville *et al.* [40], who could not identify any successful case study used by the industry. One of the open challenges of using PCGML is the lack of training content, which is not always accessible, thus requiring additional effort from the development team to create the training content in advance. While the use of Deep Learning (a subfield of ML) has provided some advances in PCGML as reviewed by Liu *et al.* [41], its application to PCG is still limited by two main factors. First, DL approaches resist the specification and enforcement of explicit constraints, such as setting up the number of rooms in a dungeon. Constraints are important for game developers to achieve their vision of the content. Furthermore, DL has issues with the interpretation of results [79], making it difficult for developers to understand the design patterns that appear in the generated content. The understandability of the patterns beneath the content is important for the development, since it allows developers to generate the content exactly as expected in the design. This is perhaps less of a problem in AAA and indie games, where creativity is encouraged to a certain degree, but a major issue for serious games.

Search-based methods do not suffer from the limitations of TM, and are therefore less expensive to apply for many types of content. SB methods also do not suffer from the limitations of ML methods, since they are easier to constrain and provide more accessible explanations of the generated results. In fact, some of the weaknesses of other methods become a strength in SB methods. For instance, in SB methods, the objective function can use constraints to guide the search. SB approaches for PCG generate content that is evaluated within the algorithm prior to its use. During the evaluation phase, content is appraised to decide whether to use it, discard it, or recombine its building blocks to generate further content.

The scope of this survey puts the focus on SB methods, since they mitigate or outright remove a series of limitations suffered by TM and ML methods alike. In

2010, Togelius *et al.* [2] surveyed SB methods for PCG (SBPCG) reviewing the very first research work in this field. They identified several research challenges for SB methods: Suitability of the generated content, avoidance of catastrophic failure, and improvement of the key components of the SB methods.

Our study reveals that in 2025 many of those challenges are still open, whereas new types of contents and concepts have appeared such as surprise search [3] or quality diversity [80]. Thus, making this field an exciting avenue for future research.

In this chapter we survey the current state of SBPCG by analysing a total of 121 articles, published between 2010 and 2023, that have applied at least one Search-Based method to procedural content generation. Based on the analysis of this article we evolve previous taxonomies [2, 1] in a new one, which is able to capture the most recent content proposed in the literature, we describe the current work according to this taxonomy, and conclude by providing insights for future directions in this field.

3.2 Survey Methodology

This survey gathers and categorizes research work published in the field of PCG for Game Software Engineering. More precisely, this work puts the focus on the context of SBPCG. In the following subsections, we present our search methodology in detail, along with a description on the selected publications.

3.2.1 Search Methodology

Inspired by Martin *et al.* [81], our search methodology follows three steps: First we perform a preliminary search followed by a repository search, then we apply the selection criteria, and finally, we conduct a snowballing process.

Our preliminary search has two goals. The first one is to assess whether there is a sufficient amount of publications in this field since the latest survey on Search-Based PCG [2] from 2011. The second one is to define the keywords that will be used for the repository search. The results of the preliminary search define the following keywords: (‘pcg’ OR ‘automatic generation’ OR ‘procedural’) AND (‘videogame’ OR ‘game’) AND (‘search-based’ OR ‘evolutionary’ OR ‘genetic’ OR ‘local search’ OR ‘tabu search’ OR ‘Monte Carlo Tree Search’ OR ‘mcts’)

We conduct the repository search on Scopus, a database that indexes papers from ACM, IEEE, Springer and Elsevier, among others. We have gathered publications from 2011 to 2023, since the last survey on Search-Based PCG [2] released on 2011, which covers work published until 2010. We restrict the publications based on the words defined during the preliminary search. We run the query on the title of the article, the abstract and the keywords associated to each publication.

The inclusion criteria used in this survey ensure that the publications present Search-Based algorithms with the aim of generating content in the area of video games. For example hybrid approaches such as 'Neuroevolution', that is, the use of an evolutionary algorithm to evolve a neural network (or any other ML method) that will create the content, are not included. Similarly, approaches that evolve other agents assessing content rather than creating it, are not included either. To verify that the publications found during our search meet the inclusion criteria, we examine the publications by applying the following three stages process:

1. Title: we remove publications that are clearly irrelevant from the title.

2. Abstract: we inspect the abstract and remove publications that are clearly irrelevant according to the scope defined in Section 3.1.1.

3. Body: publications that pass the previous two steps are excluded if their content is not relevant to the scope of this survey (Section 3.1.1)

The publications studied in this survey are the result of the application of this selection process. Sections 3.4 to 3.8 discuss in detail the studied publications.

To reduce the risk of missing relevant publications from the literature, we apply one level of backwards snowballing [82]. More precisely, we inspect the publications cited in the related work of each publication that passed the previous selection process.

In addition, we request feedback from the authors of the work included in our survey, as done in previous surveys [81].

Table 3.1: Number of publications retrieved at each step of our literature search.

Step	Number of publications	Added publications
Preliminary search	318	107
Snowballing	196	13
Author feedback	1	1

3.2.2 Selected Publications

Table 3.1 provides the number of publications we retrieved at each stage of our search, and specifies the number articles that at each step meet the inclusion criteria for this survey, and were therefore included in the survey.

As a result of the examination of 498 publications, we retained 121 unique publications that are in the scope of our survey, i.e., articles describing content generation for video games by applying at least one SB technique. These 121 publications have been published in 43 different venues. The list is available in our online appendix [2], along with a classification of the publication venues according to the CORE ranking portal [3], and the JCR ranking portal [4].

3.2.3 Threats to Validity

To mitigate the threat of missing relevant information in our literature survey we undertook a number of mitigation actions, as detailed below. 0po-Two authors examined all articles independently, in order to ensure reliability and reduce researcher bias. The results were compared at the end of the process, and any inconsistency was resolved by a joint analysis and discussion. Moreover, to ensure that our survey is comprehensive and accurate, we contacted the authors of the publications collected. We asked them to check whether our description about their work is correct. Based on their feedback, we revise our survey as well as included 1 additional publication. We carefully describe the search process we followed and make additional data available in our online appendix, so that future studies can reproduce, replicate, and extend our work.

[2] https://solar.cs.ucl.ac.uk/os/sbpcg/Venues.xlsx
[3] https://www.core.edu.au/conference-portal
[4] https://jcr.clarivate.com/jcr/home

3.3 SBPCG Taxonomy

The main aim of creating an SBPCG taxonomy is to correctly identifying and classify the different type of contents that have been described in published work on SBPCG. This taxonomy is then used to discuss each of the articles within a given category. This will ease the analysis and comparison of work aiming at automatically generating new type of content. In this section, we explain the methodology by which we construct the taxonomy that we use to classify existing SBPCG work.

Figure 3.1 summarizes the steps we followed elaborating the taxonomy. In Step 1, *Harmonizing*, we analysed the taxonomies from two previous studies and looked for similarities, thus getting a starting point for an harmonized taxonomy. Then in Step 2, *Test of time*, we create a preliminary categorisation of the work that we analyse in our survey. We then expand this categorization by including additional subcategories to better reflect the current state of the work in SBPCG in Step 3, *Subdividing*. Finally, in Step 4, *Check Empty Categories*, we analysed why some types of contents that was discussed in the previous surveys, have not been subsequently get any traction in SBPCG, thereby leading us to the removal of some categories. Further details for each of the steps are provided in the following paragraphs.

Previous PCG surveys have adopted a similar approach, by proposing their own taxonomy. Our taxonomy stems from the analysis of these previous work, and also includes new types of content according to the needs of newer articles published from 2011 to 2023. Specifically, we used two surveys (namely Togelius *et al.* [2] and Hendrikx *et al.* [1]) as the starting point for our taxonomy as they are more generic and cover more types of content than other surveys [40, 41]. Togelius *et al.* [2] focused on Search-Based techniques (as we do) proposed till 2011; while the 2013 survey by Hendrikx *et al.* [1] focused on the most common and some emerging techniques for PCG (including 11 Search-Based ones). Table 3.2 summarise and compare the two previous surveys and our own by specifying their publication venue, the publication time frame of the articles discussed by each survey, and their main

Figure 3.1: Summary of the process we followed to construct a taxonomy of procedural content categories. We started from two existing taxonomies ([1, 2]).

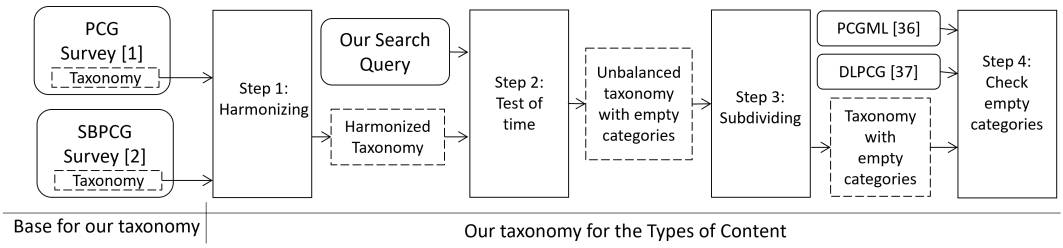


Table 3.2: Summary of the three surveys.

Authors	Published in	Cover articles from	Number of articles	Focus
Togelius *et al.* [2]	ToG	2005 - 2010	25	SBPCG
Hendrikx *et al.* [1]	TOMM	1991 - 2011	80	PCG
This survey	*under review*	2011 - 2023	118	SBPCG

focus[5].

In the following paragraphs, we explain the taxonomies used in both the previous surveys, and the process we followed to derive the one we use in this survey.

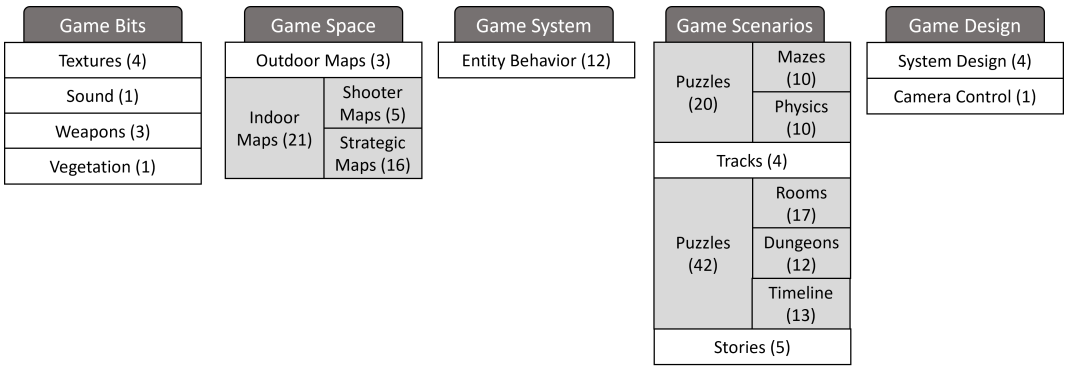
Togelius *et al.* [2] divided SBPCG content into two categories: considering whether the content was *necessary*, that is, whether the content was needed for a game to be played, or *optional*, that is, whether the content was not strictly needed for the game to be played and could be avoided during gameplay. The types of content on each category may vary depending on their purpose in a game. This survey built the taxonomy considering research articles published until 2010.

Hendrikx *et al.* [1] proposed a more structured classification for PCG methods. Instead of constructing their taxonomy from the results of a literature search, they asked themselves about the contents of a game. Then, they looked into the different types of content to analyse the techniques applied within each category.

We have studied the detailed descriptions of both taxonomies, appreciating significant similarities between them (see Fig. 3.1 Step 1). Both taxonomies overlap in 10 out of 12 types of content of the sub-classification of the taxonomy of Togelius *et al.* [2]. More precisely, there are three types of content with the same name

[5]These information have been extrapolated from the content of the surveys, as they are not explicitly discussed in the surveys.

Figure 3.2: Structure of our proposed taxonomy of procedural content categories. The subcategories highlighted in grey are those that have been introduced in this taxonomy based on the articles we found, thus extending previous existing taxonomies.



(buildings, puzzles and levels) in both taxonomies, and we argue that vegetation subsumes trees, indoor maps subsumes maps, outdoor maps subsumes terrains, storytelling is equivalent to storyboards, story is equivalent to narrative, and system design subsume rules and mechanics.

To build a first version of our taxonomy we start from the those of Hendrikx *et al.*. Although their focus is different from ours, their taxonomy is more comprehensive than those by Togelius *et al.*'s one, and we observe that it better fits the more recent articles we found in our search till 2023. We augment this taxonomy with three types of content from the taxonomy of Togelius *et al.* (*i.e.*, weapons, tracks, and camera control) because we found recent articles that tackle those types of content. Weapons fit in game bits, tracks fit in game scenarios, and camera control fits in game design.

Then, we analysed our literature search results and categorised the articles found according to this preliminary taxonomy (see Fig. 3.1 Step 2). Such an analysis pointed out two aspects that must be taken into account in devising our new taxonomy: difference in the amount of work related to different type of content, and decaying vs. emerging type of content. Firstly, the number of work within the different types of content vary significantly (see Fig. 3.1 Step 3). In particular, there are two types of content (levels and indoor maps) that independently comprise more work

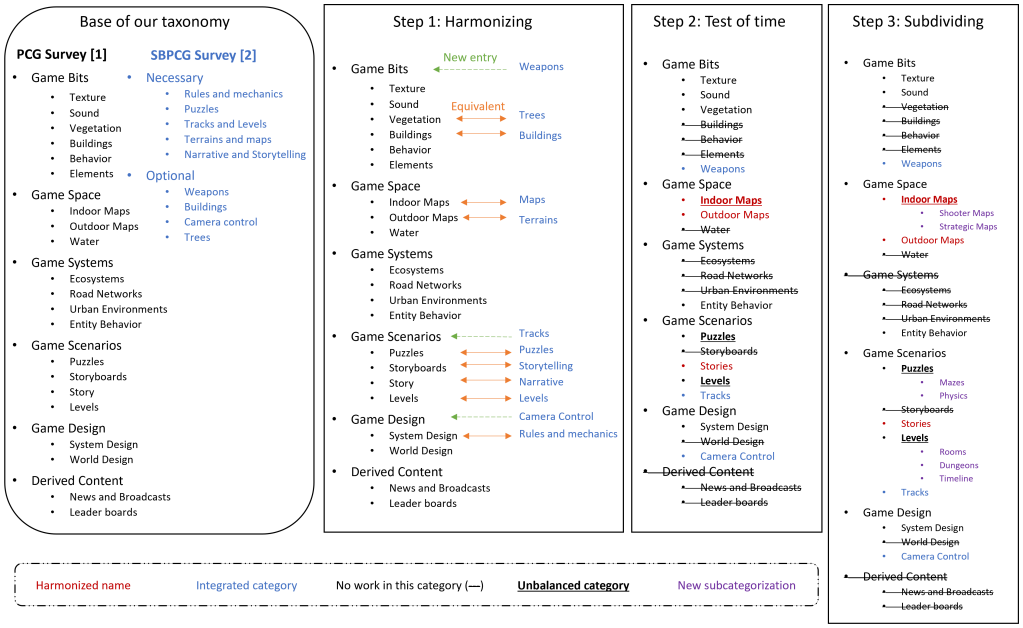


Figure 3.3: Evolution of the categories on each step of the process to construct our taxonomy.

than those presented by the rest of the types of content combined. Looking into those types of content we have identified differences over the content that allow a further sub-categorization. We therefore propose a further sub-categorization of levels, indoor maps, and puzzles. To that extent, we have divided levels into the timeline, room, and dungeon subcategories; indoor maps into the shooter maps, strategic maps, and floor subcategories; and puzzles into the physics and mazes subcategories (see Fig. 3.2). We argue that this new classification is better adapted to the current research in SBPCG. Secondly, one of the general categories mentioned above (*i.e.* derived content) and half of the types of content defined in the taxonomy subcategories have not been tackled in the last decade by SBPCG works. The derived content category addresses the generation of content after the development of a game. This type of content is usually related with the interaction of players with a game. Derived content is out of our scope as we relate to the generation of content for a game before its release or during its gameplay.

From the analysis of the literature we also noticed that there are types of content that have not been tackled by SBPCG. To speculate about the reason (see Fig. 3.1 Step 4), we have examined two more recent surveys of PCG [40, 41], which tackle

PCG from the point of view of PCGML. PCGML handles patterns better as it is based on learning, while SBPCG needs a manual configuration of the constraints in order to be able to follow patterns. Maybe there are types of content that require so much effort in the form of constraints that researchers resort on learning the constraints.

In the last ten years there are types of content that have not been tackled by neither SBPCG nor PCGML (*i.e.* buildings, behaviour, elements, water, ecosystems, road networks, urban environments and world design). We ran an informal search in Google Scholar for each type of content, in order to understand these types of content, with the following query: 'procedural generation' + < type of content >. The results show that works that study these types of content are addressed through Traditional Methods, such as, noise generators, fractal structures or L-systems [83]. Here we find an opportunity for SBPCG and PCGML. TM approaches have been successful but we did not find arguments proving their efficiency over SBPCG or PCGML techniques.

Further explanation about the general categories, types of content, and subcategories can be found within the following sections, which compose the survey *per se*. The Game Bits Section (Section 3.4) addresses textures, sounds, weapons, and vegetation. The Game Space Section (Section 3.5) addresses indoor maps (and their subcategories into shooter maps, strategic maps, and floors) and outdoor maps. The Game Systems Section (Section 3.6) addresses non-playable characters. The Game Scenarios Section (Section 3.7) addresses puzzles (and their subcategories into mazes and physics), levels (and their subcategories into timeline, room, and dungeon), tracks, and stories. The Game Design Section (Section 3.8) addresses system design and camera control. In the online appendix [6], we list the articles discussed in this survey by reporting on their publication year and venue, the two main characteristic of the search-based approach (*i.e.* encoding and objective function) and content type they investigate (see Figure 3.2).

[6] https://solar.cs.ucl.ac.uk/os/sbpcg.html

3.4 Game Bits

We start our survey by discussing Game Bits. Game bits are the smallest units of game content, or in other words, the most basic pieces that can be used to build a game. Game bits on their own, that is, when considered independently and out of the context of a particular game, do not hold any value for the players of the game. In that sense, game bits are the most basic building blocks that are in turn used by the developers of a game to generate other types of content. For instance, textures are one of many different game bits that can be used to construct game scenarios. We analyse work with respect to textures, sounds, weapons and vegetation. Textures are the images and materials of the elements of a game. Textures are in accordance with the artistic style of the game. Sounds encompass the music and sound effects of a game. Music is an important element to create the game atmosphere. Sound effects report feedback to the player regarding their actions or changes in the environment. Weapons are game bits that are used by the players to face adversities in a game. Vegetation creates an aesthetic engaging environment. In addition, this game bit help players as hiding place, as raw material, or guide players about directions and climate changes.

3.4.1 Texture

Textures have been mainly addressed by the Graphical Computation community [84]. The generation of textures for video games tackle a wide range of challenges. For instance, there are work that generate the shape of the elements of a game [85]. Kowalski *et al.* [85] generated novel shapes for chess-like games motivated by previous work related to chess-like games that used the rules of chess to generate novel games (see Section 3.8.1). Their work tackled the generation of the whole collection of pieces for a game, as well as each individual piece separately. The evaluation indicated that keeping the balance between the collection and the individuals moved the results towards one of the objectives.

Other work have tackled the generation of textures by evolving the colour palette [86] or the complex materials [87] that compose an element of a game. Players tend to associate the appearance of an element of a game with its aim, that

means, that a change in the appearance of an element of a game would affect the perspective of the player. Liapis [86] studied the use of an evolutionary approach to modify the colour palette of Pokémon keeping their shape. The new colour would fit those Pokémon into different Pokémon categories just by their appearance.

Brown *et al.* [88] evolves camouflages patterns for game assets. Inspired by a real military uniform pattern, a genetic algorithm is able to generate new patterns assessed by the environment. The evaluation used computational vision observation of the pattern in the environment, and assessed its capacity to camouflages. They also conducted a human evaluation that corroborated the blend capacity assessed previously by a computer.

The work on textures shows how the search in a large space leads to provide new ideas over this type of content, and the human evaluation corroborated how new content would be feasible in a video game.

3.4.2 Sound

The application of algorithms for sound composition tasks is not a new challenge. Research in algorithmic composition started in the last decade and has a long history [89]. Its application on games is not new either [90]. After the survey of Togelius *et al.* [2], Plans *et al.* [31] brought Experience-Driven PCG and sound together. Plans *et al.* generated music based on the experience of players while playing. The actions of players are the inputs of the approach, and affect the music at runtime. Their results sustained the idea that music affects gameplay.

A key factor over the sounds is the effect on the players, generating new sound content seems to require some human verification. This may be the reason why there is not so much SBPCG work on this type of content.

3.4.3 Weapons

Previous work on procedural generation of weapons tackled the challenge through approaches guided by human players [91, 92, 93]. The authors of these work developed a 2D commercial game where the weapons were generated based on the interaction of the players with the game. More precisely, the game generated

weapons based on how often the players interacted with a weapon in particular.

McDuffe *et al.* [94] also applied an interactive objective function for generating weapons. In contrast to the work referenced in the prior paragraph, McDuffe *et al.* generated weapons for an academic 3D multiplayer game. The weapons were evolved through the study of implicit evaluations of each weapon provided by players. The measured factors were the time that players had equipped each weapon and the number of kills obtained by players with each weapon. From the point of view of the developers, the results suggested that the approach generated interesting weapons.

Inspired by the work of McDuffe *et al.* [94], Gravina *et al.* [95] introduced a weapon generation approach for a 3D commercial game. They evolved weapons with the aim of obtaining balanced weapons, where balance was calculated through an objective function that contemplated the distribution of kills of each weapon. In addition, they applied the same objective function to evolve and improve existing weapons. In contrast to the work by McDuffe *et al.* [94], the objective function simulated matches among bots, which did not need rendering to do the simulation thus accelerating the evaluation. The authors also ran an experiment with human players to assess the quality of the generated weapons. The received feedback showed that the approach generated weapons that were interesting, fun to play, and balanced.

Based on their previous work, Gravina *et al.* [3] continued with the generation of balanced and effective weapons, where balance still considered the distribution of kills of a weapon and effectiveness considered whether a weapon could actually kill an enemy or not. Throughout the paper, they explored the usage of a constrained surprise search approach to generate weapons. Constrained surprise search is a Feasible Infeasible-2population constrained optimization algorithm which looks for "surprising" results, that is, diversity on the output. Gravina *et al.* [3] compared the performance of their approach against a single-objective search approach and a constrained random search approach. The comparison exposed that the objective search approach obtained more feasible results than the other two approaches. However, the surprise search approach obtained more diversity in the results than the other two

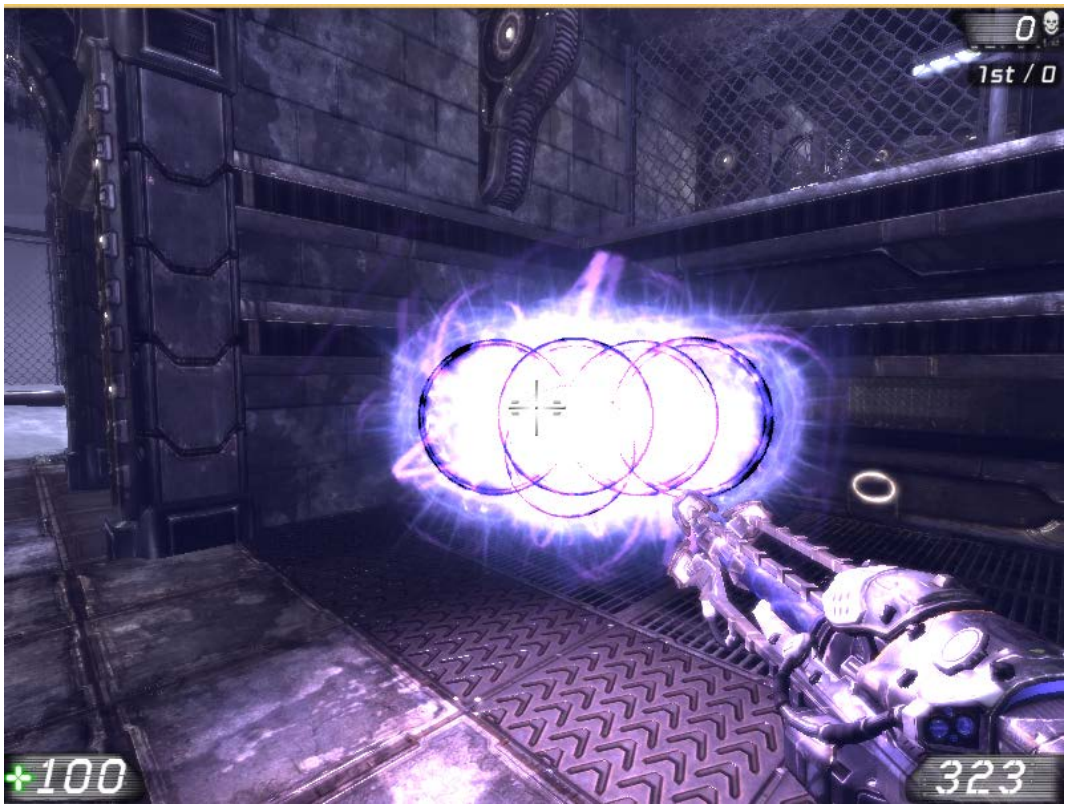


Figure 3.4: Example of weapons evolved via constrained surprise search (source [3]).

approaches. Overall, the results of this work highlighted that the constrained surprise search approach was capable of quickly and reliably generating diverse weapons.

We observe a successful use of on-line generation for weapons where the search is guided through players feedback to create the content that would suit those players. On the other hand, weapons generation has moved also towards generating diverse content that could be unexpected and not only suiting players preferences.

3.4.4 Vegetation

Vegetation includes wherever plants appear in any digital environment, including indoor plants. Vegetable, like any digital object, can be interactive or not. To date, almost all vegetation is noninteractive. The importance of detail representation of vegetation, depends heavily on the target realism of the game and the resolution of the hardware. Empirically, vegetation is well-suited for TM: certainly, the game industry has overwhelming voted for it with their keyboards for it to date. Evidently, the industry deems the coarse-grained realisation of vegetation satisfactory. The

canonical TM is SpeedTree, which provides a development environment to create and modify vegetation, as well as a wide catalogue. The development environment allows the user to visualize the vegetation in different seasons, and to add wind, and light, among the different options. SpeedTree is widely used in cinema and video games, two of the most popular 3D game engines [96] (Unity and Unreal) integrate it. Perhaps, TM's success has stymied ML work in the space; in any case, we are unaware of any PCGML for vegetation.

The uniformity of TM vegetation, TM's weakness, inspired one group of SBPCG researchers to look to improve the diversity of noninteractive vegetation. Mora *et al.* [30] has proposed a novel, SBPCG approach to generate vegetation. Their approach uses an evolutionary algorithm to simulate the life cycle of the flora. Universe 51 game is a planetary exploration game, one with photo realistic environments which enhance the fun and interest of exploring an alien landscape. The authors integrated their flora generator into Universe 51 and play-tested it. Their goal is to increase the naturalness of a digital environment by simulating change in the flora. There are two big questions unaddressed by Mora *et al.*, the computational cost of adding this dynamism, and the benefit in terms of player satisfaction. Perhaps this is why, at the time of writing, no game publisher has adopted it.

Interactive flora is even less explored. To our knowledge, there is only one game that does so. Petalz [97], a game which allows players to interact with flowers creating new content. Interactive objects, by definition, as noted in the introduction, have behaviours, which, in turn, introduce constraints, like weight, that are well-suited for SBPCG.

Flora generation has been dominated by TM techniques, however there is no evidence that prove why other techniques have not gained a similar widespread. Answering the two open questions that we have aroused could enlighten the reason.

3.5 Game Space

Game spaces (or maps) are the environments of a game, that is, a one-, two-, or three-dimensional area that can be filled with game bits in a relative position and

direction. Game spaces do not specify linear gameplay, meaning that they do not need a start and end point. We can distinguish two main types of game spaces: outdoor and indoor maps. Outdoor maps are large spaces, usually with different ecosystems, that require a certain amount of time to traverse. In these spaces it is common to use vehicles or teleport systems. Specific to outdoor maps is the topography of their terrain, which is a depiction of the elevation of the map. Most works in the field of outdoor maps deal with the topography of the terrain of the map, and often referred to it just as terrain. Indoor maps are maps that are contained in a limited space within which the player can move. For example, shooter maps rely on buildings and objects to create suitable combat spaces, while strategic maps are designed to require the management of different kinds of assets in real-time. The assets are the different kinds of units, resources, buildings, and any other video game elements that a player can position, manoeuvrer, manage, or otherwise control during play.

3.5.1 Outdoor Maps (Terrains)

In 2009, Frade *et al.* [32] coined the term Genetic Terrain Programming (GTP), which refers to the use of SBPCG in order to generate terrain for video games. Their approach used an interactive objective function that involved humans to guide the search. The results highlighted two limitations regarding user fatigue and the inability to perform zoom over the generated pieces of terrain.

In order to address these limitations, Frade *et al.* continued their work with a new version of GTP, named Automated Genetic Terrain Programming (GTPa) [98, 99]. These works modified their previous approach to tackle these limitations by avoiding user fatigue and enabling the zoom feature over the generated terrain. In order to avoid user fatigue, the authors proposed two distinct direct objective functions, guided by different objectives in each of the two works, that did not involve humans. In 2012, the same authors combined the two functions from their GTPa previous work into a new direct objective function that was the result of the sum of the formerly individual objectives [4]. The approach decreased overlaps in the results, but generated terrain with smaller amplitude. The results were released in an open

Figure 3.5: Screenshot of Chapas video game where the terrain was generated online by a TP (source [4]).

database[7] for future research purposes [100].

In 2016, Pech *et al.* [101] proposed a novel approach for generating terrains by incorporating elements into pre-existent terrains. To incorporate those elements, Pech *et al.* [101] introduced the use of an architectural element in the objective function, which is isovist. An isovist is the volume of a space visible from a given point in space. Through this perspective, Pech *et al.* [101] were able to introduce elements such as hidden areas in a terrain. This novel approach benefited terrains that were previously unplayable.

Terrains generation has evolved from the use of an interactive objective function to a direct objective function to avoid user fatigue. It is interesting to notice that there is no comparison between the use of an interactive objective function or a direct objective function, and it gives an opportunity to explore simulation-based objective function to address the issue of user fatigue.

[7] `https://sourceforge.net/projects/tps-db/`

3.5.2 Indoor Maps

3.5.2.1 Shooter Maps

The previous SBPCG survey [2] shows that SBPCG work addressed the challenges of generating content as tracks for racing games, rules for board games, weapons for space shooters, levels for platform games and maps for real-time strategy games. The work of Cardamone *et al.* [102] is the first to address the challenge of generating First Person Shooter (FPS) maps. Despite FPS is one of the most popular game genres, only three studies [103, 104, 105] investigated FPS games before, by targeting the generation of a new form of content; that is, generating the behaviour of Non-Playable Characters.

Maps are the heart and soul of FPS games. Cardamone *et al.* [102] argue that generating FPS maps poses a bigger challenge than generating maps for other games. They also acknowledge that FPS maps should favor gameplays that reward skillful use of complex tactics, and force players to vary their tactics so they cannot use the same patent trick all the time to win.

Outside of PCG, researchers have analysed FPS maps and proposed design patterns for FPS maps [106]. These design patterns might be useful for guiding the automated search of new FPS maps. Nevertheless, Cardamone *et al.* [102] do not leverage the former design patterns but rely on bots for the objective function instead.

The work of Cardamone *et al.* [102] generated complete and playable FPS maps for Cube 2 [8], an open-source game from 2004. They learned that direct encoding works better than indirect encoding for this kind of map. However, they point out two main limitations of their work: (1) the dependency on the control logic of the default bots of the game, and (2) the lack of validation by players, that is, they evaluate the maps in terms of synthetic measures (fighting time and map space), but they do not claim that their measures correspond with human players' judgments.

Three years later, Lanzi *et al.* [107] revisited the challenge of generating maps for FPS. Their work shares the encoding, the case study, and the synthetic evaluation

[8] `http://sauerbraten.org`

with the work of Cardamone *et al.* [102]. Nevertheless, Lanzi *et al.* [107] aimed for a different goal. Whereas Cardamone *et al.* addressed fast-paced action maps, Lanzi *et al.* addressed match balanced maps. Balancing a match between two players is one of the seminal problems of video game development. To do so, Lanzi *et al.* [107] proposed a novel objective function computed on the basis of the statistics collected from a simulated match between two bots. Their work succeeded, in three scenarios, generating maps that balanced the match between pairs of bots with different skills.

Olsted *et al.* [108] brought human players to the work of Cardamone *et al.* [102]. The novelty of their work was the use of human players as objective function. Humans played the maps and used votes to rank them. The approach was evaluated in an academic context with the FPSEvolver video game, a video game developed by the authors for the evaluation. Almost every player agreed that the maps improved in quality as they played. However, players expressed that maps felt quite flat in comparison to maps of real games such as the popular Counter-Strike and Call of Duty.

Loiacono *et al.* [109, 110] were the first to explore multi-objective algorithms for generating FPS maps. As in previous work, they use the same encoding as Cardamone *et al.* [102] and Lanzi *et al.* [107] (*i.e.* a static simulation through bots to guide the search) and the same case study (*i.e.* Cube 2). However, Loiacono *et al.* [110] collect statistics from the simulation for multi-objective search. Their objectives are the balance of the map, the pacing on which the players are engaged in fights, the average length of kill streaks, the fighting time, the shooting time, ability of loose enemy's sight during fight and loose enemy's sight enough time to stop the fight. Their evaluation suggest that multi-objective evolution can provide a good insight of what happen with human players. The same objectives computed using bots and evaluated with human players provide significant agreement.

Shooter map generation approaches have addressed several challenges, starting with the use of single and multi objective search algorithms. They have also used simulation and interactive objective functions. Finally, their evaluations used both academic and commercial video games. One of the works unveil players concern

that the content felt quite flat, and this issue seems to remain unaddressed in the literature.

3.5.2.2 Strategic Maps

In the previous SBPCG survey [2], it is possible to appreciate that SBPCG works tackled map generation for strategic games. Those work generated strategic maps at the scale of academic games [111, 112]. Togelius *et al.* [111, 112], authors of those works, used a semi-direct encoding, crossover and mutation operators, and a five-objective function. The objectives were obtained individually for each game, and were derived from the play style and rules of each specific game under evaluation. Four of these objectives (surface, asymmetry, resource distance, and resource clustering) were of the direct type, and the fifth (A* base distance) was of the simulation type. In 2013, Togelius *et al.* [113] extended their previous work with further experiments and included human players in the evaluation. Human players appreciated that the asymmetry objective generated unbalanced maps.

In the last ten years, SBPCG moved from academic strategic games to commercial strategic games. Lara-Cabrera *et al.* [114] were the first that generated strategic maps at the scale of commercial games (in particular, they did so for Planet Wars, an indie game). To do that, they went beyond the state of the art by leveraging bot-based simulations to guide the search [114, 115]. Specifically, they identified the indicators that should be monitored during the simulation to calculate the objective function [116, 117]. Those indicators, once again specific to the game being evaluated, addressed map balancing (territorial imbalance, growth imbalance, and enemy imbalance), resource management dynamism (game length, conquering rate, reconquering rate, and peak difference), and player confrontations (battle rate and destroyed enemies). These works claimed that they successfully generated strategic maps, however, none of them involved humans in the evaluation to evaluate the quality of the results or to find out whether the results were aligned with the expectations of the players.

In 2013, Lara-Cabrera *et al.* continued their work on strategic maps from a different perspective: The aesthetics of strategic maps [118, 119, 120], that is, the

spatial distribution of the elements of the map and their features (size and number of elements). In contrast to their previous work, the authors based their research on direct objectives through which they assessed the geometric, morphological, and topological properties of the maps with the purpose of generating procedural content related to aesthetic aspects of the game. The results of these works were evaluated through the usage of automated and semi-automated techniques along with the support of a human expert. Contrary to previous work [116, 117], the evaluation of these approaches intended to measure the degree of the quality of the aesthetics of the maps according to the defined objectives, with the aim of studying whether the generated maps were aligned with the expectations of the players.

The aesthetics of strategic maps have also been addressed by other authors before. Through a constructive method, Johnson *et al.* [121] used a Cellular Automata (CA) algorithm for generating maps. A CA algorithm is a discrete model with self-organizing properties that consists of a grid containing cells that can exist within a finite number of states. The algorithm works by setting a state in each cell of the grid and traversing the grid through an iterative process. However, CA algorithms lack control and cannot be easily adapted for generating other maps. For this reason, Mahlmann *et al.* [122] generated maps using a search-based approach that incorporated a control mechanism to the CA algorithm approach. To that extent, they use a direct objective function that generates maps for an abstract version of another strategic game (Dune 2).

The most recent work by Lara-Cabrera *et al.* [123] is the first one that tackled level balancing for a 3D academic strategic game (*i.e.* Paintbol). The authors defined a balanced level as a level that does not provide an initial advantage for one of the two participating teams over the opposing team. In contrast to their previous work on balancing content [114] (with a simulation objective function), Lara-Cabrera *et al.* [123] used an indirect encoding and a direct objective function that analysed the defensiveness, ranking, and dispersion indicators. Those indicators generated balanced levels based on the results of the objective function. The results of this work suggested that, by applying different parameters in the evolutionary algorithm,

better maps were generated. In particular, the rank and elitist selection methods generated better maps than roulette selection according to the measured indicators.

Other authors also tackled the generation of balanced maps [124, 125, 126, 127]. Those approaches used search-based techniques to obtain the balanced maps rather than generating them from scratch. Barros *et al.* [124] balanced the maps through the initial position of the players. Kowalski *et al.* [125] and Franco *et al.* [126] evolved the positions of the assets that would be placed in the map. Ma *et al.* [127] also placed the assets in the map, but using a multi-objective approach The results by Barros *et al.* [124], by Kowalski *et al.* [125] , by Franco *et al.* [126], and by Ma *et al.* [127] generated playable and balanced maps.

One of the main goals in strategic map generation is the balance in the maps generated. Several approaches have addressed this goal from different perspectives (e.g., different representations and objective functions) but only one work has provided a human evaluation that concludes that asymmetry does not work for balanced maps. The other main goal addressed by strategic map generation is based on the aesthetics of the maps. We find it interesting that contrary to the balance goal, researchers have taken into account human expectations but they have only investigated the use of direct objective functions.

3.6 Game Systems

Game systems bring the virtual worlds of video games closer to the human world in order to provide the players with a sense of immersion. This is achieved through complex models that include, among others, entity behaviours, also known as Non-Playable Characters or NPCs. NPCs are essential for the experience of the players, since they generate the illusion of a virtual world along with the opportunity to create interactions between the environment and the players.

In 2013, Guarneri *et al.* [128] described an approach to automatic generate NPC monsters through an evolutionary algorithm. The goal of the approach was to obtain a diversity set of new monsters from a starting population defined by the developer. This approach was later applied by Norton *et al.* [129] on another video game genre.

With the same goal, Ripamonti *et al.* [43] developed a novel approach to generate monsters adapted to players. This approach records the number of times a player kill each type of monster, considering the monster with more death rate the preferred by the player. The evaluation used a simulation to test the generated monsters, meeting authors' expectations on diversity, coherence, and difficulty.

Pereira *et al.* [44] instead of diversity seek for generating enemies that meet a difficulty criteria. In that sense, the objective function looks for enemies that are closer to the difficulty stated in the search. The results with human players indicates that the generated content matched the desired difficulty. Viana *et al.* [33] extended the work by Pereira *et al.* [44] introducing quality diversity methods, to improve the diversity of the enemies generated. As Pereira *et al.*, the results show how the generated content matched the desired difficulty.

In 2021, Blasco *et al.* [45] looked for generating, and later improving [130], spaceship enemies which quality is comparable to manually content created by developers. The approach has a novelty as they worked with software models, instead of code. Model-Driven Engineering has the ability to provide an abstraction level in the development process. The results show how the approach was capable to generate content comparable to the manually created by developers in 5 hours, compared to ten months that took to the developers. On other hand, to generate also spaceships, Gallota *et al.* [5] used a combination of Lindenmayer systems [46] and evolutionary algorithm. Their results suggest that the approach generated spaceships that meet some human preferences.

Some work in the literature tackle battle formations on games where the player must defeat a coordinated group of NPC enemies. Players do not find challenge on the static strategy behind battle formations, as they just learn the optimal way to win. Thus, Ruela *et al.* [131, 132, 133] presented a co-evolutionary approach for generating balanced and challenging battle formations. They evaluated the effectiveness of their proposal offline (i.e., outside the game itself) by comparing the search-based generated battle formations against the battle formations built by human players. The results showed how the generated battle formations could win

Figure 3.6: In-game screenshot of a spaceship generated using the approach by Gallota *et al.* [5] (source [5]).

against the formations of active players, improving the challenge for the players.

A novel work in the literature by Brown *et al.* [134] proposed a generative approach towards city discovery in four different role playing video games based on the social structures and networks of the NPCs. As a result, the designed algorithm devised the placement of cities and NPCs based on the intern complex relationships of NPCs in order to generate a more realistic video game environment. The evaluation showed the approach to be human competitive.

Most of the work on this type of content focus on enemies NPC, and only one of them on social aspects of NPCs, which provides space to explore more types of NPCs. On other hand, an interesting strategy has been presented by the use of Model-Driven Engineering to generate new content, that could be likely extended to more content types in future work.

3.7 Game Scenarios

Game scenarios describe the goals of a game, and the way and order in which game events unfold. Normally, the events of a game are set in motion as a result of partial or total completion of the game goals, as well as through the interaction between

the players and the game. Game scenarios are described by game developers, and are often transparent to the players of the game. Game scenarios can be classified into three different types of content: puzzles, tracks, levels, and stories. Puzzles are problems to which the player must find a solution. The solution can be based on previous knowledge or on a systematic exploration of the space of the possible solutions embedded in the problem. Puzzles can be found in a wide variety of game genres. Some different kinds of puzzles are mazes and physics. Mazes are puzzles defined as a network of paths and hedges through which the player has to find a way. Physics puzzles introduce the laws of the physics into games. Players need to apply the laws of physics for the sake of solving the problem. Tracks are cyclical game scenarios, and are usually found in racing games. Levels are logical separators that enable advancement within the different sequences of a game. The advancement is often based on the successful completion of game objectives by the player. Levels can take up many different forms as follows: Rooms are levels where the players must interact with a set of game elements available within a particular section of a game; Dungeons are levels composed by a succession of rooms. The player must pass through the different rooms to complete the goals of the level, Timelines are levels that are linearly designed and are usually found in platform games. Stories are the narrative elements that compound the game. Stories present the events of the game to the player affecting directly their experience. The following paragraphs discuss the relevant search-based research work in each area.

3.7.1 Puzzles

3.7.1.1 Mazes

A previous work on maze generation by Ashlock *et al.* [135] presented an evolutionary approach to generate mazes. In the approach, the authors used a direct encoding and a simulation-based and direct theory-driven objective function. In 2011, the same authors [136] proposed new types of encodings (direct, chromatic, indirect positive, and indirect negative) and features that could be used to construct the objective function. Their results showed that the different encodings and objective functions were feasible, and that the selection of one of them would depend on the desired type

of maze.

In the same year, Ashlock *et al.* [137] presented a work that can generate mazes with two possible solutions, depending on the character that addresses the challenge, named as dual mazes. In this approach, the authors used features for the objective functions from previous work [136], but modified the chromatic and indirect positive encodings. The results suggested that the direct encoding generated more diverse mazes and that the indirect encoding found better solutions according to the objective function. Based on this work, McGuinness *et al.* [138, 139] generated large mazes using small mazes as tiles. The novelty of these works [138, 139] resided in the objective function, which was adapted in order to provide the developers with more control over the tiles and the final maze. McGuinness *et al.* evaluated the results of their work by using their approaches to generate large mazes according to designs provided by developers.

In 2012, McGuiness [140] ran an experiment to statistically compare different encodings from previous work [139]. The author argued that the encoding is an important factor to the final visual representation of the mazes. The results suggest that the visual representation of the mazes is very different depending on the encoding, even when the mazes are similar according to the measurements that the author used in the analysis. More recently, McGuinness [141] built up on previous work by incorporating a direct encoding and features from previous studies in the objective function. In this work, McGuinness [141] tackled maze generation with a novel search-based approach adapted from the Monte Carlo Tree Search technique. The results revealed that the mazes generated through this approach were intuitive and qualitatively different from those generate by using only evolutionary computation.

Approaches other than evolutionary computation or Monte Carlo Tree Search have also been applied in the last decade. First, Kim *et al.* [142, 143] proposed a search-based approach to generate 'perfect' mazes, that is, mazes with no loops or inaccessible areas. The approach takes as input the desired metrics for the maze and selects the algorithm that better suit the metrics in order to generate the mazes. Once the mazes are generated they are evaluated by the desired metrics and by a set of

measure metrics. The difference between the values obtained by the mazes for the desired metrics and for the measure metrics act as the objective function and as the stop criteria for the search. Secondly, Pech *et al.* [144, 145] proposed an approach based on the evolution of Cellular Automata (CA) rules that would be in charge of the generation of mazes. The authors argued that evolving CA rules is faster than evolving mazes, because their proposed CA was able to generate a variety of mazes that met the set of rules evolved.

The work on mazes appeared in the literature since the last survey has focused their effort on the representation of the maze problem. This work point out how different types of representation are feasible to generate mazes, but depending on the encoding the results will vary, and while the generated content may be similar based on measurements it can still exhibit visual differences. One open challenge within maze generation is the validation with humans in commercial video games.

3.7.1.2 Physics

Physics-based puzzles are a type of content that has not been tackled by SBPCG before 2013. Shaker *et al.* [17] are the pioneers in this area. They proposed an evolutionary algorithm based on a indirect encoding and a direct objective function. Their results suggested that this technique generated promising puzzles to be played. Afterwards, Ferreira *et al.* [146, 147] also presented an evolutionary algorithm for generating physics-based puzzles. However, the encoding and objective function by Ferreira *et al.* [146, 147] differ from those proposed by Shaker *et al.* [17]. Ferreira *et al.* [146, 147] used a direct encoding and simulation in the objective function, with the main purpose of stability. The results showed that their approach generated stable puzzles.

Kaidan *et al.* [148] extended these prior work, modifying the objective function to adjust the levels according to the player. Kaidan *et al.* did a preliminary validation with human players. In a subsequent work, Ferreira *et al.* [6] built upon their previous work in the field [147]. In contrast to the work by Kaidan *et al.* [148], their approach improved the encoding to allow for more features, such as duplicated blocks. Moreover, the objective function for their approach also took into account

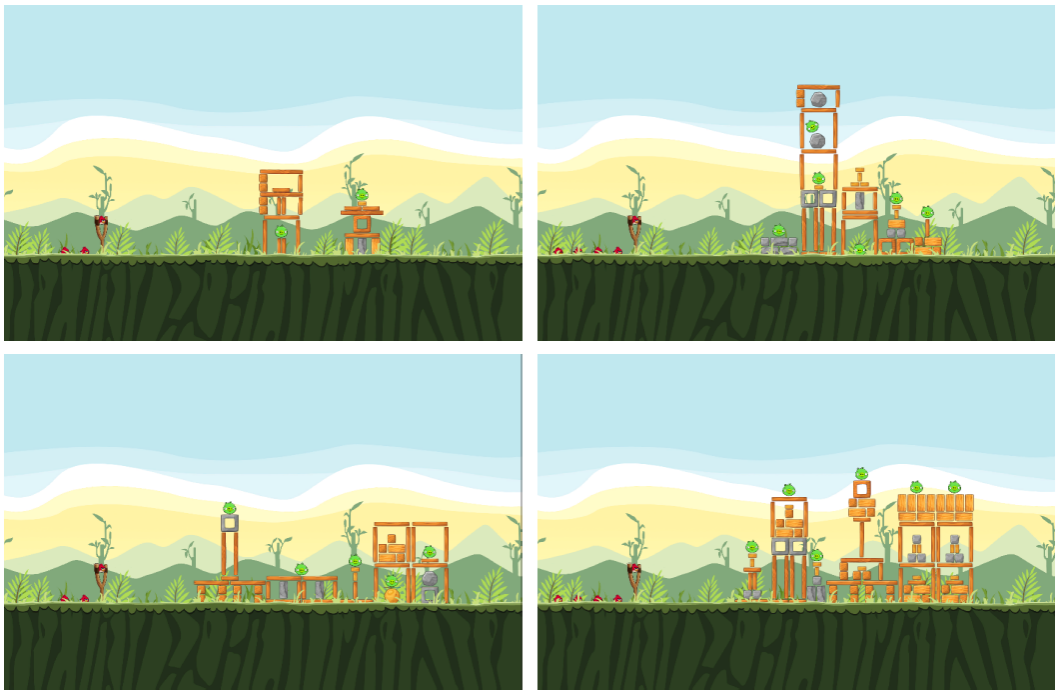


Figure 3.7: Example of levels generated with the GA of Ferreira *et al.* [6] (source [6]).

the feasibility of a puzzle, not only its stability. The results showed that the approach generated relevant puzzles for the first episode of the game used as case study (*i.e.* Angry Birds).

The direct encoding used in previous work limited the structure of the puzzles [148]. Due to this limitation, Calle *et al.* [149] proposed a novel evolutionary approach to generate stable free form puzzles. In order to reduce the cost to evaluate the objective function, before the simulation, a candidate needed to meet two criteria (distance to the ground and overlapping of blocks). Their results highlighted that SBPCG had potential for generating physics-based puzzles, and the need for problem-specific knowledge. Lately, the same authors [150, 151] reduced the cost to evaluate the simulations of the objective function through the usage of a physics engine instead of a game engine.

In contrast with the work mentioned above, other authors [152, 153] tackled the n-body physics problem. To that extent, they proposed an evolutionary approach which generated puzzles according to a given difficulty (easy, medium or hard). The approach by Lara *et al.* [152] presented three different objective functions, based on

different criteria (intersections, gravitational acceleration, and simulations), and a preliminary analysis was run with human players. The results suggested that none of the three objective functions rated the difficulty of the generated content in the same manner as human players did. Lopez-Rodriguez *et al.* [153] further validate the work by Lara *et al.* with human players. They found out that the automated approach tended to rate the generated content as higher difficulty when compared to the difficulty ratings provided by the human players.

Physics is a novel type of content that we have identified in our survey. It has gained enough attention and several authors have investigated the use of different encodings and objective functions. Some authors recommend the use of indirect encoding to avoid structure limitations on the puzzles. A work with human validation noticed how human and the approach differed over the difficulty of the generated physics puzzles. Further investigating the reason behind this difference could lead to interesting insights.

3.7.2 Tracks

Some work in the field of tracks [154, 155], included in the previous survey [2], tackled track generation through an evolutionary algorithm that generated racing tracks for an academic 2D racing game. In order to generate the racing tracks, the authors based their approaches on a bot-simulation objective function to guide the search.

More recent works, such as the one presented by Loiacono *et al.* [156], aimed to optimize the fun value of the game through the maximization of the potential diversity of race tracks in the game, namely, through the maximization of the differences between the available race tracks in the game. The diversity of a particular race track was assessed by a multi-objective function that measured the curvature of the track through a direct objective, and the speed profile of the track through a simulation objective. Loiacono *et al.* performed a preliminary validation with humans, which suggested that there is a statistically significant alignment between the results provided by the approach presented and the preferences of human players.

Prasetya *et al.* [157] also worked towards the optimization of the fun of a game

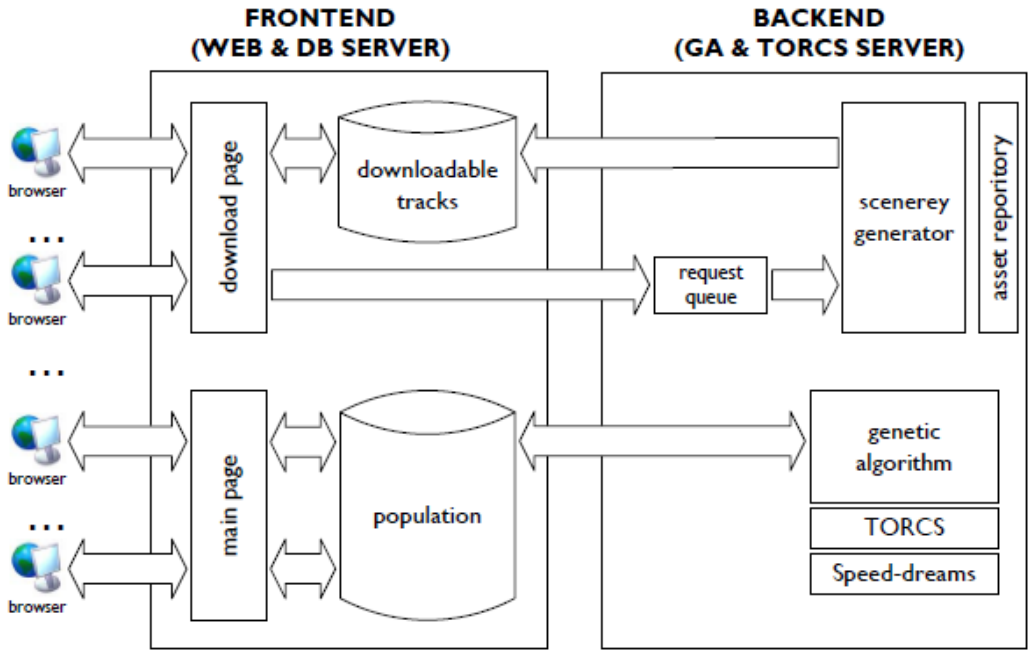


Figure 3.8: Architecture of TrackGen (source [7]).

through its tracks. They differed from the work of Loiacono *et al.* mainly in two aspects. Firstly, they used a semi-direct encoding instead of an indirect encoding. Secondly, they compared the performance of two different search algorithms (Tabu Search versus a Genetic Algorithm). Among the two search algorithms, the Genetic Algorithm had less average generation time than Tabu Search. Prasetya *et al.* performed an evaluation with humans comparing the tracks generated by their approach against man-made tracks. Their results suggested that the generated tracks were measured up to man-made tracks in terms of the measured fun.

With the aim of aligning content generation results with the preferences of the human users of a game, Cardamone *et al.* [158, 7] were the first to introduce an interactive objective function in the track generation process. Through their approach, a population of tracks was generated by an Evolutionary Algorithm. The Evolutionary Algorithm was guided through the assessment of the tracks, provided by human participants after each iteration of the algorithm. After the experiment, humans stated improvements in the quality of the tracks, and that the process produced interesting tracks.

The work in tracks share a common goal, that is 'fun'. In order to achieve this objective, human intervention is present either in the evaluation or in the objective function.

3.7.3 Levels

3.7.3.1 Rooms

We have noticed the need of this subcategory because in the last ten years, the amount of works that fall into this type of content ,and the diversity of techniques (such as Evolutionary Algorithms [159], Multi-objective Algorithms [160], Quality Diversity [161]), has increased. As an example, in 2012, Togelius *et al.* [162] proposed a preliminary approach that tackled this type of content through a hybrid approach that used Evolutionary Algorithms and Answer Set Programming (ASP), which has been used before for system design generation [163]. However, in more recent work, the core of the generation of this type of content has shifted towards the usage of the General Video Game AI (GVGAI) framework. Two main elements are used to build this framework: (1) the Video Game Descriptive Language, and (2) the General Video Game Playing Competition. The Video Game Descriptive Language (VGDL) [164] is a textual description language that has been used to represent two-dimensional games. The General Video Game Playing Competition [165], which started in 2014, is an event that explores the challenge of creating controllers for general video game play, where a single agent must be able to play many different games. The GVGAI framework provides a series of different games based on VGDL, as well as the game-independent agents to play the generated room levels for those games.

Neufeld *et al.* [166] used an Evolutionary Algorithm to evolve the rules used by a room generator from GVGAI based on ASP. These rooms were evaluated through a simulation objective function that calculated the difference of average scores obtained by vanilla Monte Carlo Tree Search and a random player. Their results showcased the benefits of the approach, however, Drageset *et al.* [167] identified the computational cost of translating VGDL games into ASP rules as a drawback. Hence, Drageset *et al.* proposed a purely evolutionary approach, named Meta Generator,

based on a more elaborated simulation-based objective function that uses three of the agents provided by GVGAI. The evaluation compared the Meta Generator against both random and constructive generators using the same objective function. The results highlighted that the Meta Generator obtained higher scores for the objective function than the other generators.

Zafar *et al.* [168] also proposed an evolutionary approach, more precisely, a Feasible Infeasible Two Population algorithm which differ from previous work in the metrics used for the objective function, measuring aesthetics and the difficulty of the rooms. Their results suggested that the levels obtained were aesthetically and challenging. With the same aim, Petrovas *et al.* [169, 170] proposed a genetic approach, with a complex direct fitness function, Combined Compromise Solution.

Another proposal from Zafar *et al.* [171] used design patterns to generate rooms. The approach selected patterns from a collection of design patterns and used them as input for the evolutionary process. The objective function also included a factor in the equation related to design patterns. The experiments were run with different agents from GVGAI, and concluded that the agents had better performance on rooms generated with design patterns.

Walton *et al.* [172] addressed the room generation from the perspective of the developers instead of the player. Their proposed approach takes as input a level designed by the developer, and uses a Feasible Infeasible Two Population algorithm to generate new levels. This approach was evaluated by developers judging its capacity to facilitate their job. The developers found more inspiring the results from the evolutionary algorithm than random generation. However, the approach lacks of diversity, which limits its use.

In order to tackle this issue, which is common to different approaches, a novel evolutionary strategy, named Illumination Algorithm [173], was exploited. Illumination Algorithms find high performing solutions in different sections of the search space instead of maximizing one solution as evolutionary algorithms usually do. Charity *et al.* [174] incorporated this idea in their approach, and used the mechanics from some of the games provided by the GVGAI framework to create multiple,

relatively high quality states for a GVG-AI level that demonstrate combinatorial variations of a game's mechanics. Their results showed that this approach generated satisfactory rooms with a single mechanic or with a controlled combination of mechanics.

The use of Quality Diversity methods has grown also in the field of room generations. Green *et al.* [161] introduced Constrained MAP-Elites, generating rooms based on different human play-styles called 'Personas'. Using Personas to generate content can encourage a player towards new challenges. Alvarez *et al.* [175] and Charity *et al.* [176] proposed a co-creative approach, where MAP-Elites approach proposes new rooms and users guide the generative process. Users can also modify the generated content influencing the evolutionary process.

In contrast with the work study above, Bhaumik *et al.* [177] were curious about the performance of more Search-Based strategies. Due to that reason, Bhaumik *et al.* [177] compared eight different search-based algorithms, including Tree Search Algorithms and Optimization Algorithms. Those algorithms include Breadth First Search, Depth First Search, Greedy Best First Search, Monte Carlo Tree Search, Hill Climbing, Simulated Annealing, Evolution Strategy, and Genetic Algorithm. Their results suggest that Optimization Algorithms generally performs faster than Tree Search Algorithms.

More recently, Bailly *et al.* [178] proposed the inclusion of Wave Function Collapse (WFC) into a genetic algorithm. They introduce WFC to generate rooms targeting specific play experiences. Their approach used a simulation to measure novelty, safety, and complexity of the generated rooms. The results compare those metrics within a pure genetic search, a brute force search, and their genetic WFC approach, obtaining this last the highest fitness.

Most of the work in this category is related to the General Video Game AI, and its interest on the General Video Game Playing Competition. However, after a human-based evaluation unveiled that the generated content lacks of diversity, researchers seems to have moved towards addressing this issue. This also suggests that a human evaluation is strongly recommended for all the approaches based on

diversity.

3.7.3.2 Dungeons

Dungeons play an important role in video games. In the last ten years, the interest has not decreased. One of the challenges of generating dungeons is the reduction of the generation time. Pereira *et al.* [179] claimed that a tree structure reduce the need for validation and fixing time that graph/grid approaches required, such as the one proposed by Valtchanov *et al.* [8]. Later, Pereira *et al.* [180] conducted an experiment with human players to validate the generated dungeons. The results showed that the dungeons were enjoyable and challenging.

Font *et al.* [181] used a graph approach to reduce the generation time, which differs from the work by Valtchanov *et al.* [8] for two aspects. First, the approach used a context-free grammar representation to avoid the generation of syntactically non-valid individuals. Secondly, Font *et al.* [181] reduced the search space by dividing the approach in two steps: Generating the structure of the dungeon first, and the detailed elements that are more time consuming, such as monsters or chests, afterwards.

Another challenge in the generation of dungeons is the diversity of the results. Tackling this challenge, Ruela *et al.* [182, 183, 184] proposed a single-objective approach, which later derived into a multi-objective approach due to the fact that a single-objective approach that combines different objectives tends to prioritize one in detriment of the others. Melotti *et al.* [185] proposed a variation of a multi-objective approach combined with Deluged Novelty Search Local Competition, which separated the search space into niches, allowing for the control of the differentiating characteristics of the niches through a distance function.

On a different perspective, Liapis [186] addressed dungeon generation using intertwining segments as representation. And, inspired by Liapis representation, Viana *et al.* [187] generate dungeons using barriers as novelty, which are mechanics that force players to follow a path. The diversity of the dungeons are measured by map linearity, mission linearity, leniency, and path redundancy. Their results analysed through expressive range suggest that the approach achieve the diversity on

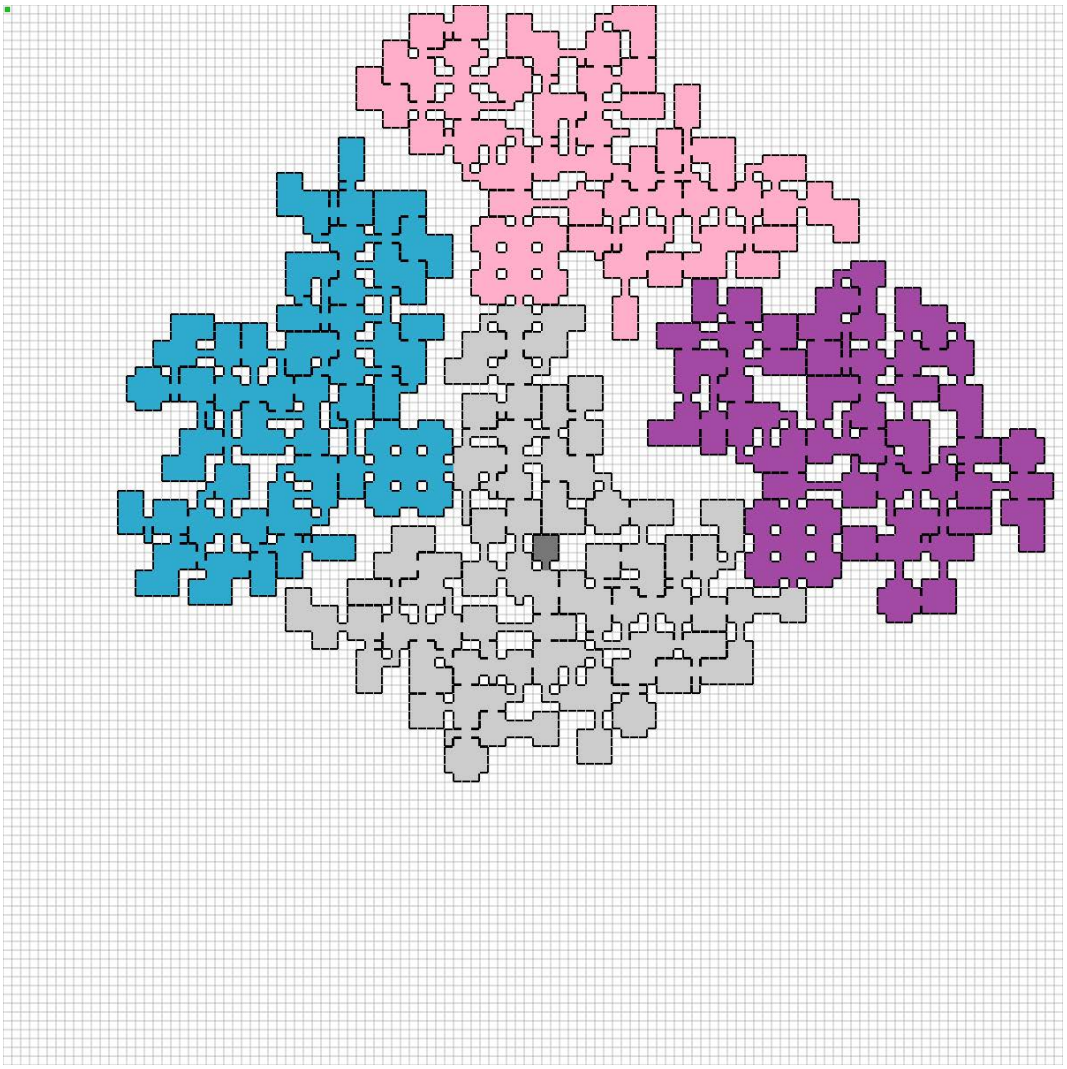


Figure 3.9: Example map generated for the Multi-Region Map Experiment - different regions colored (source [8]).

dungeons generation.

In addition to diversity, Ruela *et al.* [184] designed an experiment to compare the performance of four well-known algorithms against their own algorithm. They concluded that it is not possible to define an overall winner algorithm, and that the use of each algorithm will depend on designer preferences. The baseline algorithms were: Improved Strength Pareto Evolutionary Algorithm (SPEA2), Pareto Archived Evolution Strategy (PAES), Non-dominated Sorting Genetic Algorithm II (NSGA-II), and Multi-objective Cellular (MOCell). The main limitation of the novel algorithm proposed by the authors was its time consumption, which limited its usage to offline

PCG.

To scale up the resources of a game, Brown *et al.* [188, 189] proposed the integration of a dungeon editor that players can use into a commercial game. The proposed search-based approach generated different rooms within a space and then connected the rooms. The connection was possible due to the hard constraint that each new room must overlap with an existing one. This work serves as an indicative that hard constraints are an advantage that search-based methods provide as we mention in the ML constraints (Section 3.1). Brown *et al.* also allowed the integration of objects and enemies into the dungeon. A Petri net method filled the dungeon with objects, and a second evolutionary approach placed enemies throughout the dungeon. On other hand, Harisa and Tai [190] generate dungeon levels for based on game designer preferences of game pacing. The experiments showed the error between the designer preferences and the generated content with results between 1.16-18.53%.

The work on dungeon generation has identified two challenges for this type of content. First the need to reduce the generation time, and secondly the diversity of results. Due to those challenges the generation of dungeons has mainly remained an offline generation. However, there is work on online generation, which makes use of hard constraints in the approach. There is no outcome from online approaches about generation time or diversity.

3.7.3.3 Timeline

Most of the work under this category used a well known platformer video game as a case study, Infinite Mario Bros, which is a clone of Super Mario Bros. However, we have decided to name this subcategory as 'Timeline levels' because a more general name provides the potential of covering a wide range of genres, such as platform games, runner games, or endless games. Shaker *et al.* [191] is a good example of a generic approach of this category. The approach generated linearly the sequence of actions for different games. They validated the approach simulating the sequences.

An advantage of the timeline levels is that they reflect the difficulty curve of games. A difficulty curve is a graphical representation of how the difficulty fluctuates during the game. Several approaches has addressed this challenge on platformer

games [192, 193, 194]. Adrian *et al.* [192] was the first one that used the difficulty curve designed by developers as objective function. The experiments showed that the difficulty curve of the generated timeline levels were close to the designed difficulty curve. A preliminary validation with 22 players supported the similarity with handmade timeline levels. Inspired by Adrian *et al.*, Atmaja *et al.* [195] applied the same idea into a scrolled vertically shooter game. Moghada *et al.* [193] also generate the rhythm of the level, fitness with difficulty curve by human.

Different representation approaches has been studied for timeline levels, more precisely for Mario Bros. In 2011, Mourato *et al.* [196] found out that the use of a grid as a detailed representation of timeline levels had the risk of consuming substantial resources. To reduce the consumption of resources, Dahlskog *et al.* [197, 198] divided Mario Bros levels into vertical slices. Each slide was a micro-patterns that they extracted from the original video game. The combination of the slides generated the resultant timeline levels, named as 'scenes'. Dahlskog*et al.* [198] validated the approach finding in the generated scenes combinations of micro-patterns that expressed meso-patterns that were originally in the game. Later, Green *et al.* [199] 'stitched' the scenes generating long timeline levels. In 2022, Moradi *et al.* [200] introduced Estimation Distribution Algorithm to also create meso-patterns.

Based on the work of Dahlskog *et al.*, Green *et al.* [201] generated scenes that served as tutorials for a specific mechanic. The approach used a feasible infeasible two population algorithm with two objectives function. The infeasible population objective function measured the aesthetics, e.g. a pipe on Mario Bros requires two consecutive slides. The feasible objective function compared the performance of two agents, a 'perfect' agent and a limitated agent. When the limitated agent failed on completing the level, it meant that a special mechanic is needed, and therefore the level could be a tutorial. Khalifa *et al.* [202] compared three approaches which generate timeline levels for one mechanic, including the work of Green *et al.* [201]. They results showed that the approach that guarantee the mechanic on the scene had three disadvantages: It was the slowest approach, required human intervention, and relied on agents failures.

We have seen that in more recent years, new techniques such as illumination techniques (see Section 3.7.3.1) have appeared. Warriar *et al.* [203] presented a first attempt of Illuminated approach for platform video games. The results highlighted three opportunities for improvement: A more visual human-designed aesthetic levels, better features that define fun, diversity and controllability on the content, and how to keep a computationally low consumption with an entire map of playable levels. In this direction, Withington [80] presented a preliminary comparison of Quality Diversity algorithms MAP-Elites [173] and SHINE [204], however neither approach stood out above the other.

Research on timeline generation has worked towards three different aspects. The first one is the use of a difficulty curve designed by developers to approximate the generated content to the desire curve. Second is the effect of using different representations. The last one is seeking diversity over the results. These three aspects has been addressed separately, where future work it is needed to investigate their combination.

3.7.4 Stories

Addressing the generation of stories is a complex task for procedural content generation. The reason is the difficulty to obtain a cohesive story from evolutionary operators. However, the use of trees or grammars with small pieces of stories are known to work well together [205]. Based on the use of trees and the recent works to reach diversity through Search-Based PCG, Fredericks *et al.* [206] introduced a novel idea that combines those two elements into a genetic improvement algorithm. The work is still under development, and currently lacks of evaluation. In this direction, Alvarez *et al.* [207] and De Lima *et al.* [208] has also generated narrative quests making use of grammars [207], and trees [208].

By the hand of Alvarez *et al.* [209, 210] Quality diversity methods has also been applied in stories generation. Alvarez *et al.* [175] first proposed a co-creative approach to generate rooms (See 3.7.3.1) that has been adapted and applied for stories [210].

Even with the difficulties that stories present to PCG, novel ideas have been

applied for this type of content. Most of the work move toward achieving diversity in the results.

3.8 Game Design

Game design is the core of a game. Game design defines the gameplay of a game by conceiving and designing its rules and structure. A change in game design could generate a whole new game. Game design decisions affect all the content in the previous sections. We can further distinguish game design into two aspects: (1) System design, which refers to the rules and mechanics that define a game; (2) Camera control, which refers to the placement and behavior of the camera in the game, or in other words, how the player will visualize and experience the game.

3.8.1 System Design

As seen in the previous SBPCG survey [2], system design has been addressed by different approaches.

Browne [36] presented the Ludi system, which used Evolutionary Computation combined with a simulation-based objective function, with the aim of measuring aesthetic aspects of the game. This system led to the first fully computer-invented games to be commercially published. Togelius *et al.* [211] introduced the concept of 'fun' measurement through a Hill-Climbing approach. Both approaches used simulation-based objective functions that have been used in recent works to tackle system design from different points of view.

Aligned with the work by Togelius *et al.*, Halim *et al.* [212] guided the search with the aim of optimizing entertainment. The objective function measurements included the duration of game, the intelligence required to play the game, the dynamism exhibited by the pieces, and the usability of the play area. Halim *et al.* validated their approach by conducting two experiments: one with a neural network-based AI to measure controller learning ability, and another with human users to measure the entertainment provided by the game. The results suggested that the evolved games were more interesting and better than the randomly generated games.

Kowalski *et al.* [9, 213] proposed different approaches to generate games based

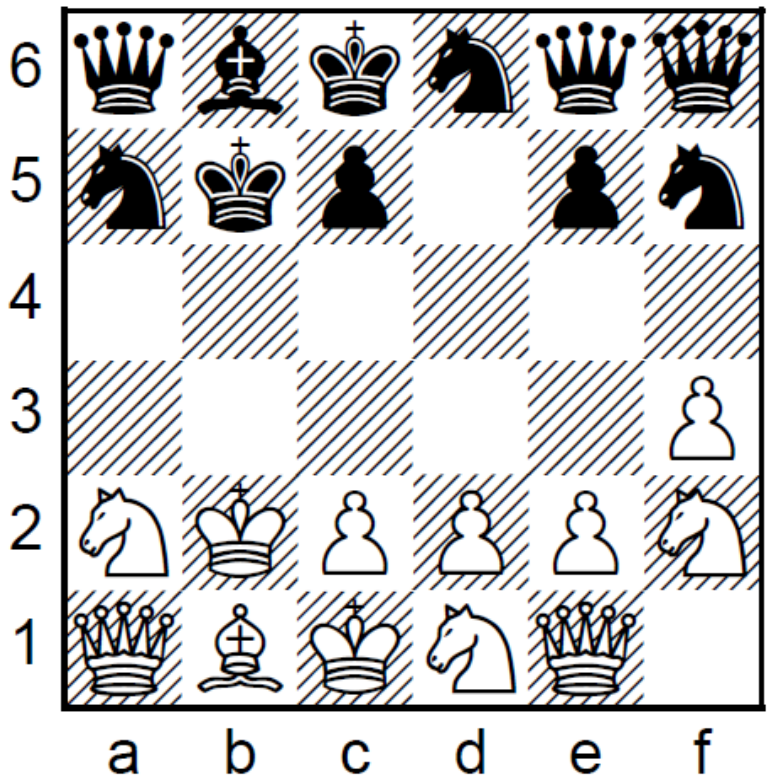


Figure 3.10: One of the evolved games presented by Kowalski *et al.* [9] (source [9]).

on chess. In 2015, Kowalski *et al.* [9] presented an objective function based on simulation to generate playable and balanced games. A hand-made evaluation function analyzed the playout histories and checked the balance, game tree size, pieces importance, and complexity of the game rules. In 2016, Kowalski *et al.* [213] generated games that were closer to chess and more constrained than those generated by their previous work. The objective function increased the number of agents in the simulation, representing players with various degrees of intelligence. The experiment had two parts; the first compared the different agents of the simulation, and the second compared the results with human-made chess-like games. The results showed that the approach obtained playable and balanced games that were similar to the high quality human-made games.

In 2023, Volden *et al.* [214] proposed a genetic algorithm to generate rules for a kindergarten serious game. The approach measures the difficulty of the game with the generated rules. Their results approximated the target difficulty.

In recent years, three different goals have been addressed by System Design approaches. First, tackling aesthetics, secondly, optimizing entertainment, and last, playability and balance. The latest search strategies rely on Quality Diversity, which is a promising opportunity for generating this type of content.

3.8.2 Camera Control

Previous work on automatic generation of camera control content [215, 216] showed that the search space of this type of content is rough to be explored. In addition, it has also been highlighted that the objective function for generating camera control content is computationally expensive, which reduces the number of evaluations available for the search process.

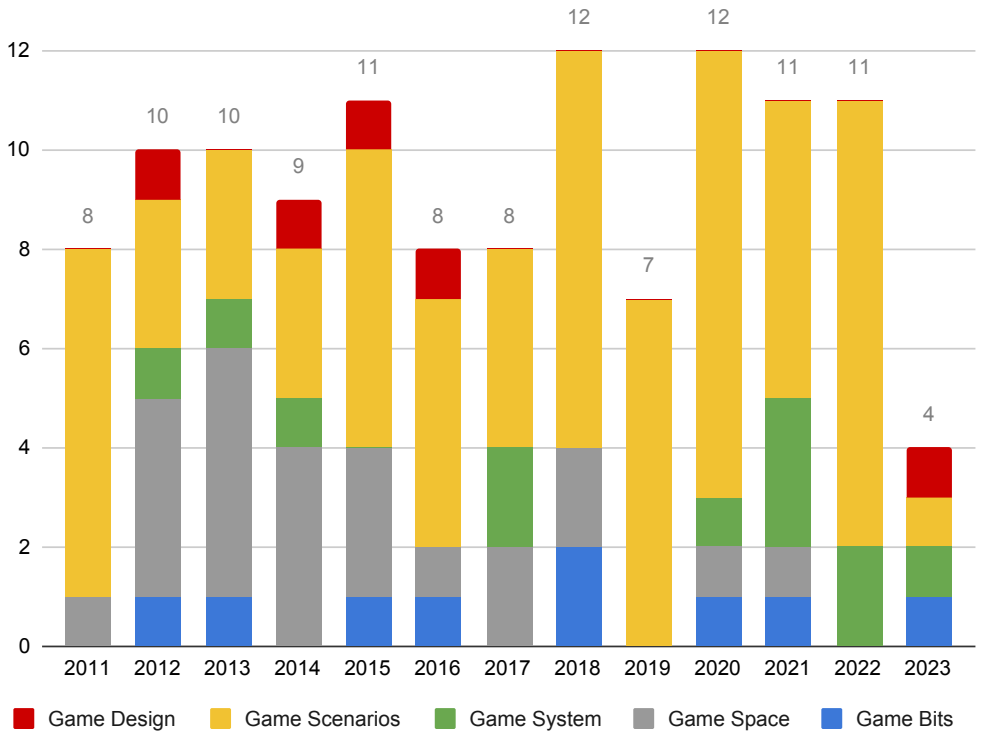
Preuss *et al.* [217] addressed automatic camera control through a niching and restart Evolutionary Algorithm. Niching extends Evolutionary Algorithms to multimodal domains, locating multiple optimum candidates where the Evolutionary Algorithm loses population diversity, converging to a unique solution. Convergence is an issue that Preuss *et al.* tackled through a restart mechanism when the approach reached this point. Preuss *et al.* improved their previous approach [218] by adding a constraint that reduced search space, which is one of the problems in content generation of automatic camera control. Their evaluation compared their approach with the state-of-the-art algorithms. Their results suggested that their novel approach and another similar approach (CMA-ES) performed better than other prior approaches.

The novel approaches to camera control perform better than the previous approaches, at least in academic settings, which makes it interesting to further assess the performance of commercial games.

3.9 Future Directions

Throughout the pages of this section, we analyse the different research challenges that are open in the field of work of PCG for video games. To that extent, the following subsections present the challenges, their reason to be, and the open problems for each challenge, as well as recommendations and potential future research lines in the identified areas.

Figure 3.11: Number of articles published per year for each category of the taxonomy studied in this survey.



3.9.1 Content Opportunities

Our survey has shown that the number of publications is not balanced across the content types. In Figure 3.11, we show the number of articles published per year that tackle the generation of content for each category analysed in our survey. While Game Scenarios and Game Spaces arise as the most popular content types, it is also possible to identify three exceptionally unpopular content types: Game Bits, Game Systems, and Game Design. Regarding the Game Bits category (Section 3.4), which comprises the most basic building blocks of a game, we find a total of 8 works. Within the category, we find the textures (4 works), sound (a single work), weapons (3 works), and vegetation (a single work) subcategories. Regarding the Game Systems category (Section 3.6), which deals with bringing virtual worlds closer to the human world through the study of NPCs, we find a total of 11 works. Finally, regarding the Game Design category (Section 3.8), which defines the core of a game in the form of its rules and structures, we find again a total of 4 works dealing

with system design (3 works) and camera control (a single work). The study of sounds and camera control are, then, the most neglected content types among already unpopular categories. However, we could not find a clear reasoning coming from the community for the lack of work in those particular areas, so we can only speculate why they are being neglected. Games and video game genres appearing in a recent report on the status of the video game industry[9] include those types of content, so their necessity in video game development becomes apparent. As a matter of fact, we have considered that some of these content types have been neglected because they are considered too important and integral to the success of a video game: interactions with NPCs, for instance, help players with video game immersion; and poor sound effects can lead to a worsening of the user experience, which can in turn lead to poor reviews and poor sales as a ultimate consequence. This may be a reason that calls for a special treatment, and for a careful development by hand. We also theorize that there may be too much diversity in the possible outcomes of PCG in those areas, which makes it easier for researchers to focus on other content types.

In any case, the existence of fewer studies in those areas implies that there are less improvements on the current complexity of their generation process and on the complexity of the needed constraints. This issue could in turn mislead the directions of researchers that favour other content types, since the current state of research makes it possible to believe that Search-Based approaches are not suitable for these content types even when nothing appears to indicate that those areas are harder to generate through SBPCG than other content types. In that sense, we believe that these content types should not go unnoticed by the research community any longer. Even if there is an inherent diversity to the content, many SBPCG approaches allow constraints to the objective function or different degrees of customization that may be useful to tailor the results to the necessities of each case study. In addition, as already discussed, details as textures, sounds, or interactions with NPCs are essential to captivate players and provide quality to a video game. A game that lacks these content types is directly headed to a failure. Hence, the procedural generation of

[9] https://www.wepc.com/news/video-game-statistics/

these content types could not only help with the economic cost and time to market of a video game, but also with the outright feasibility of a game where developers lack the means to build some of these content types.

Finally, regarding the suitability of SBPCG for these content types, we want to encourage researchers to avoid steering their research to those fields of work that are already explored, and raise awareness about the opportunities of research in the field. In particular, we recommend to revisit approaches that have proven their potential towards PCG applications, and to explore their application towards researching the neglected content types. Overall, we recommend to approach all content types equally, as they are equally important for the development of a video game.

3.9.2 Online PCG

The content of a video game can be added at two different points of the development process. Those two points are either before the release of the game, offline generation, or during play, online generation. Most of the works presented in this survey work with offline generation approaches. We speculate the reason behind this lies in the content generation time issue: the current reality is that approaches take too much time to generate content to be viable in online contexts. This problem affects offline generation to a lesser extent because there is more flexibility with the time for content generation during the development process. This issue will stand for online content generation as long as approaches lack the necessary speed to not paralyze the experience of the player.

The content generation time issue finds its roots mainly in the inefficiency of the approaches and the computational resources that those approaches use. As an example, approaches that use an indirect representation of the content require less resources, however, indirect representations require a transformation process that turns the indirect representation used in the approaches into the final content that is incorporated in the video game. On the other hand, simulation objective functions require more time than direct objective functions. Games that actually use online generation do so through the combination of preexisting elements, with

the disadvantage that the content is then limited to designs that have already been established by the developers offline.

The challenge lies in the application of current approaches to online generation. To that extent, it would be necessary to identify applicability problems of current approaches, improve the necessary generation times, and optimize the device resources used by a game to dedicate more computing time to the generation of content. In addition, to study these issues with a greater level of detail, it becomes necessary to know the time that takes to generate the content. However, most of the presented works in our survey do not provide the times associated to the generation time. We recommend to report the generation time in the results of future work in this field.

Finally, we believe that it would be possible to reduce or even avoid these challenges through the usage of remote servers. Ideally, the remote servers would run the approaches while the players play the game, generating the content in parallel. In theory, this would allow for online generation while the players make full usage of their playing time. Exploring the usage of remote server SBPCG for online content generation has been largely neglected in research in the field, save for works in tracks [158] that explores remote generation of content, and hence, it remains a promising direction for solving the challenge.

3.9.3 Solvability, Playability, Fairness, and Diversity

Through the literature in the field we identify solvability, playability, fairness, and diversity as the measurements that identify the basis of player expectations towards a game. Solvability is understood as the characteristic that defines whether a problem presented in the game content can be completed or solved (e.g. going from point A to point B in a level). Playability is understood as the measurement that defines the extent to which content can be exploited by human players. Fairness deals with the perception of the player when dealing with content (e.g. distribution of resources in a strategy game, or the probability of an event occurring in the game). Diversity describes the variety of the content so that players do not receive similar content. All of these factors affect the overall feelings of players towards a game, and influence the decision on whether to keep playing or not. For instance, it is important for a

game to be challenging but not impossible to finish, and unfair game mechanisms generate frustration in players.

Due to the reasons listed above, these measurements are commonly used to evaluate the results produced by approaches and to guide them in procedural content generation. However, they are not the only means to evaluate the results of the objective functions in the available research. To that extent, we have observed multiple research directions regarding the objective function in use for the evaluation of the generated contents. In that sense, while some works produce objective function scores based on the above metrics, other works retrieve their evaluation from the behaviour of simulators that intend to substitute human players, and finally, the works that obtain the best results in the literature make direct use of humans as the objective function function in what is known as interactive objective function.

While interactive objective function leads to the results better aligned with the expectations of players, it is still not a perfectly adequate objective function. As a result of experimentation, human participants tend to fatigue, which leads to a worsening of the objective function over time and a ceiling effect in the results that hinders the potential of research in the field. In addition, the fatigue of human participants limits the application of approaches to advanced or complex case studies, which we can observe in the fact that most of the studies in this field evaluate the approaches over severely tailored academic case studies.

As a potential research direction, we recommend revisiting objective functions and their application to the research field. In that sense, we believe that researchers should explore improvements to interactive objective function through the incorporation of mechanisms to avoid fatigue. To that extent, it would be possible to explore hybrid objective functions that combine metrics or simulators (or both) with interactive objective function. The combination would see metrics or simulators working for a while on their own, guiding approaches towards preliminary results that could be assessed afterwards by human hands. Exposing humans to short interactions with an evolutionary approach at suitable times would avoid fatigue, thus allowing the application of the approaches to the more complex industrial case studies, opening

promising lines of research in real-world environments.

In addition to the above, in the case of offline generation, if results are not satisfactory, it is possible to generate new content or involve developers to refine the content manually. This is not possible for online content generation. Moreover, the prior survey [2] expressed concerns with the limitation of diversity that may be caused by approaches looking for the best possible results. To overcome this limitation, research in the past few years has seen a surge in quality diversity approaches [219], a young and promising field that has attracted the attention of PCG researchers (on categories such as weapons, strategic maps, timeline or rooms), and is yet to be applied to several content types in both offline and online generation. A hybrid objective function that combines different measurements with human evaluation, along with the possible generation of content in remote servers mentioned as part of another future direction (Section 3.9.2), would help with the evaluation of content generated online, and with the acquisition of feedback directly from the players, who are the best source of information for indicating the viability and diversity of the generated content.

3.9.4 Bricolage

The main goal of PCG is to help developers during the development process of the content of a game. However, not all generated contents are fit to be directly included in a game. In that sense, the content that is created, stored, but never used creates a waste of resources. In addition, discarded content becomes a potential source of frustration for developers, who may evaluate those contents in terms of working hours, or even see them as promising ideas thrown away, only unfeasible because of the amount of time that it would take to fix them by hand so that the content can make the cut into the game.

However, we could tackle this issue from a new angle, considering the contents discarded after a content generation process as material that can be potentially used to refine and adapt the process to generate suitable content for the game. To that extent, it would be possible to consider content as a sum of components rather than as a whole, and to evaluate the suitability of each component for the game independently.

In that way, it would be possible to explore the reuse of components from existent content to novel content, thus avoiding waste of resources and the frustration of the developers.

The reuse of components would also not be limited to discarded content, on the contrary, it would be possible to take into account components coming from all the existing content for a video game. For instance, reuse of components could leverage components from content that is under development in order to generate refined versions or variants of the content during the development process, allowing developers to have a wider choice of content. Reuse of components could also use components from content that has been already approved into a game, gaining access to a library of components that count with the endorsement of the developers of the game. In addition, though the usage of reuse of components, developers could be more involved with the generation process, choosing their favored components to refine the directions of the approaches. Our recommendation for this research direction is to build approaches, both for online and offline PCG, that take developer involvement into consideration and that empower the reuse of components of the existing content.

3.9.5 Statistical Rigor

We have identified two common practices towards evaluating the presented approaches. The first of them is the usage of an objective function as a measure to evaluate the outcomes and reliability of the approaches. However, trusting the objective function requires a prior evaluation of the objective function, and causes the reliability of content generation systems to fall on the reliability of the objective function. The second one is to present the approaches to events in order to compare their efficiency. In this case, the results of the approaches are set against each other, limiting the comparisons between approaches to the rules of the event and not specific measurements. In such scenarios, it becomes necessary to improve the execution of the approaches and their comparison. In that sense, it would be advantageous to have a baseline that enables a fair comparison between works and equal opportunities for progress in the community, as well as a fair comparison of certain aspects that could

not be compared otherwise.

In addition to the above, the criteria followed by events as a means of evaluation is not uniform. While some events leverage results from bots to evaluate the approaches, the usage of feedback provided by real players is gaining momentum in the field in the form of interactive objective function. We can find an example of the latter in an event where a jury provides an assessment of the results of an approach based on the performance of players. Many good practices in this field of work, such as A-B testing, could be leveraged to build hybrid interactive systems that help with the involvement of players and developers while avoiding fatigue. Overall, we advocate for building approaches that allow practitioners and developers alike to get closer to the target public, to ensure that the generated content can live up to the expectations of players.

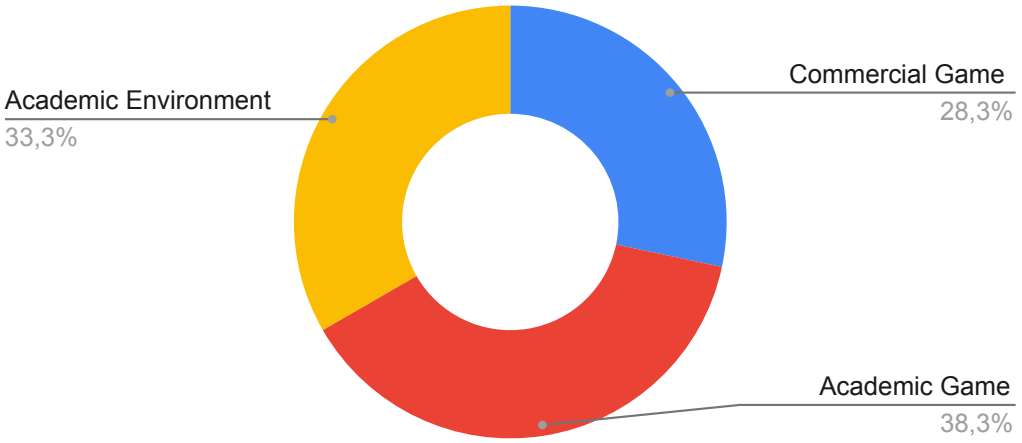
Finally, through this survey, we found out that many studies refer to artifacts and results not publicly available, which severely hinders research replication. Following the principles of open science[10] to which the general Software Engineering community adheres to, we strongly recommend the publication of prototypes, approaches, artefacts, and research results. This would not only help with research replication, but it would also influence the growth of the community and the rigor of the presented research.

3.9.6 Industrial Content

The majority of the studies presented in this survey focuses on academic games or academic environments as case study (see Figure 3.12). Those academical games are clones of industrial games, simplifications or prototype versions of original games. Academic environments are environments built explicitly for the research purposes of the researchers. Plenty of research questions arise from the observed scenario: Is there a need of more detailed generators? Is it difficult to adapt the representations used in research to industrial content? Is the content of industrial games more complex, making the quality of the generated content insufficient in industrial scenarios? Whatever the case, the reality is that research in the field

[10] https://github.com/acmsigsoft/open-science-policies

Figure 3.12: Percentage of articles where the case study involves academic games, academic environments or commercial games.



is rarely applied to real-world video games. With such a scenario, the research community remains disconnected from the industry, causing poor communication of results between the novelties in SBPCG research and the final target users of those approaches, the players. In other words, current research results do not reach players. For example, Ruela *et al.* [131, 132, 133] could not empirically assess their proposal involving humans due to the lack of access to the original game developers, and the effort and time that human players would have had to put in for evaluation

This issue leads to what we believe is one of the major opportunities for research in the field. If we are able to avoid this disconnection by applying the approaches to real-world industrial case studies, we might be able to obtain feedback from players, which might represent a very large and very valuable source of information for developers and researchers alike. This source of information could be used to guide automated approaches and to manually refine the generated content. However, in order to apply the approaches to industrial content, we must avoid fatigue. In addition to the exploration of hybrid interactive approaches, in blockbuster games with millions of active players, it would also be possible to research mechanisms to share the fatigue load of the fitness function. To that extent, our recommendation is to identify the necessities of developers working in industrial contexts, and to adapt the proposed approaches so that they can work over commercial content.

3.9.7 Interaction between SBPCG and other techniques

Throughout the survey, we have focused on pure SBPCG approaches that generate content. However, there exist work that tackle PCG through the interaction between SBPCG and other techniques, specifically ML-based techniques. In the recent years the interaction between SBPCG and PCGML has gained interest due to a new research line called latent variable evolution [220]. Latent Space (LS) [221] allows to learn the shape of search spaces where later the approach can search more effectively. This is important, for example, for online PCG, where the time of the search matters. On the other end, Quality Diversity (QD), a novel research field in SBPCG, has already gain attention from the PCGML community exploring the interaction between QD and LS [222].

3.10 Conclusion

The high demand for video game content has led to an increased interest in PCG, and its investigation has gained momentum in the past decades. Throughout the pages of this chapter, we have surveyed the updates in the state-of-the-art stemming from the 10-year gap since the last surveys in PCG for video games were published. To that extent, we have built a taxonomy based on the two prior surveys [2, 1], and gathered and categorized novel research in SBPCG. As a result, we have reported herein on new work in Game Bits, Game Space, Game Systems, Game Scenarios, and Game Design. Despite the undeniable advances in research in the field, we consider that there are still many unexplored topics, and that some of the research challenges and recommendations proposed 10 years ago are yet to be studied in depth. Through our work, we have identified plenty of open research challenges regarding content opportunities, the speed of online PCG, the characteristics (solvability, playability, fairness, and diversity) of the generated content, the possibilities enabled by content bricolage, the inclusion of statistical rigor in this field of research, and the need for applying research to industrial content. Along with the open research challenges, we have presented recommendations and identified several potential future research lines in the areas under study. Overall, this survey presents a concentrated and

comphrensive report on the latest work in SBPCG , effectively assessing the status of research in the field and providing a renewed point of view for the ongoing discussion over SBPCG in video games.

Chapter 4

Our Proposal: Imhotep

4.1 Introduction

In this chapter, we propose a new angle to tackle video games content generation inspired by transplantation techniques [19], which we named Procedural Content Transplantation (PCT).

Current PCG approaches work as follows: developers provide initial content (usually human-generated content) into an algorithm to work with. Afterwards, the algorithm (Traditional, Machine Learning, or Search-Based methods) will generate new content. Only a few traditional methods have succeeded in providing tools used by the industry to randomly generate vegetation (e.g., SpeedTree in Unreal and Unity).

Our PCT proposal introduces for the first time the transplantation metaphor for video-games. In our approach, the developers of a game will select an organ (a fragment of video game content) from a donor (video game content), and a host (another video game content) that will receive the organ. The organ and the host will serve as inputs for a transplantation algorithm that will generate new content for the game by automatically combining the organ and the host. Our hypothesis is that our transplantation approach can release latent content that results from combining fragments of existing content. Furthermore, our transplantation approach provides more control to developers in comparison to current PCG approaches that are solely based on random generation, leading to results that are closer to developers'

expectations.

Moreover, we propose the use of video game simulations ($S_{Imhotep}$) to guide the search, based on the intuition that it is possible to harness video games' NPCs to run simulations that provide data to asses the transplantation.[1]

To the best of our knowledge, this is the first work that leverages transplantation to generate video game content, obtaining more favourable solutions than current SBPCG in an industrial setting. In summary:

- Our results show that procedural content generation through transplant (*i.e.* PCT) has significantly outperformed classic content generation in the evaluation of this work, opening a new road towards tackling content generation.

- Our transplantation approach has produced the highest number of successful transplants, to date - almost double than those found in previous work. Moreover, the transplants are carried out in an real-world industrial context in contrast to the academic context of other work.

- Our work returns control to the hands of the developers through organ selection. The generated content is more in line with the intent of developers, as discussed in the focus group.

- Our work reveals that harnessing simulations rather than test suites leads to significantly better results. This may empower software transplantation researchers to reconsider the usage of test suites in their work.

- Our analysis of the results reveals interactions between organs that are a promising line of research to advance the field of software transplants.

For replicability, reproducibility and extension of our work, we made IMHOTEP's source code and the data of our study publicly available at `https://github.com/SOLAR-group/IMHOTEP`.

[1] In fact, within video games, it is typical to find NPCs that serve as companions to the player, adversaries to defeat, or inhabitants of the virtual world. These NPCs have pre-programmed behaviours that could be used in game simulations. For instance, in a first-person shooter game (like the renowned Doom video game), NPCs explore the game scenarios in search of weapons and power-ups to engage in combat with other NPCs or the player.

4.2 Our Proposal: IMHOTEP

This section explains how IMHOTEP makes use of evolutionary computation [223] and software models to transplant organs within the current content of video games to create new content (*i.e.*, Kromaia bosses in our case study). To facilitate the comprehension, we also provide the reader with an example of transplantation for a simplified version of a Kromaia 'boss' inspired by the 'Serpent' boss shown in Figure 2.2 with letter B. Given the popularity of software models for video-game development (see Section 2.5), we designed IMHOTEP to work with models. Although our running example uses the SDML models of Kromaia, our approach is generic and can be used with other modelling languages because it exploits the idea of boundaries between model elements.

Figure 4.1 shows an overview of IMHOTEP. On the left is the input to our approach selected by the developers, namely the organ to be transplanted from the donor and the host to which the organ will be transplanted (input selection 4.2.1). Afterwards, IMHOTEP automatically detects the points of the organ that allows the transplantation (Boundary detection 4.2.2) and the points where the organ can be inserted into the host (Boundary mapping 4.2.3). To initialize the population of the evolutionary algorithm, the organ is cloned and transplanted to a random point (Initialize population 4.2.4). Genetic operations generate potential solutions for transplantation (Genetic operations 4.2.5), while the objective function assesses the quality of all of these potential solutions (Objective funtion 4.2.6). This process of generating and assessing is repeated until a specific stop condition is met. When the evolutionary algorithm finishes the execution, we obtain a ranked list based on the given objective function of the best transplants between organ and host. Next, we describe in more detail each step of IMHOTEP.

4.2.1 Input selection

IMHOTEP allows the developers to identify a source model content (donor) with the organ (a fragment of the source model content) that will be transplanted, and a target model content (host). Donor, organ and host will be strictly related to the meta-model used by each context where IMHOTEP wants to be applied.

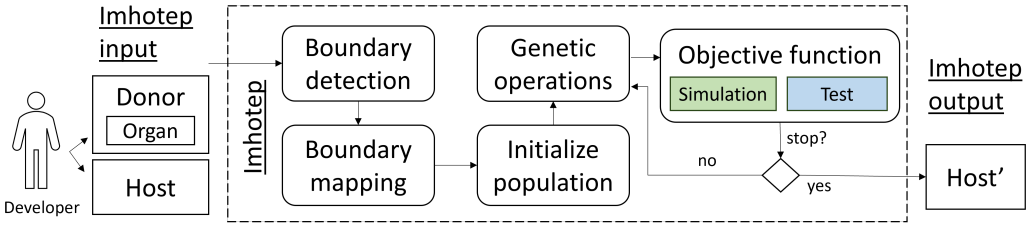


Figure 4.1: Overview of IMHOTEP, our proposal for PCT.

In our running example we present a simplified version of the meta-model, and the corresponding concrete syntax of the model (see Figure 4.2 *Metamodel*) from Kromaia. In such model 'Hulls' serve as the structural framework that define the anatomical composition of the models. For example, the boss presented in Figure 2.2 (identified as 'B') has its body built by hulls. 'Weak points' are conceptual elements that possess the vulnerability to be harmed. 'Weapons' are tangible items capable of causing harm through direct contact, such as discharging projectiles like bullets. Hulls, weak points, and weapons are attached between them through 'Links'.

In our example, the source donor model is a simplified version of the original Kromaia 'boss' 'Serpent'. Figure 4.2 *Input* **Donor** shows the graphical representation of the donor's model . It also shows with dashed lines the elements selected as organ. The host is a model of a regular enemy that could appear in Kromaia. Figure 4.2 *Input* **Host** shows the graphical representation of the host model.

4.2.2 Boundary detection

To transplant an organ into a host we need to find a way to connect them. To this end we exploit the boundaries between the model elements of the organ and the host. The study of boundaries between elements in software models has been ongoing for over ten years, with the aim of managing variability within models [224, 225]. A boundary is a connection point capable of connecting two distinct model elements within a model. The connection is restricted by the rules of the metamodel. In the simplified example in Figure 4.2 *Metamodel*, the Source and Target meta-relationships are the boundaries between the model elements of the models conforming to that metamodel. In other model languages, there will be other meta-relationships with other names that will be the boundaries.

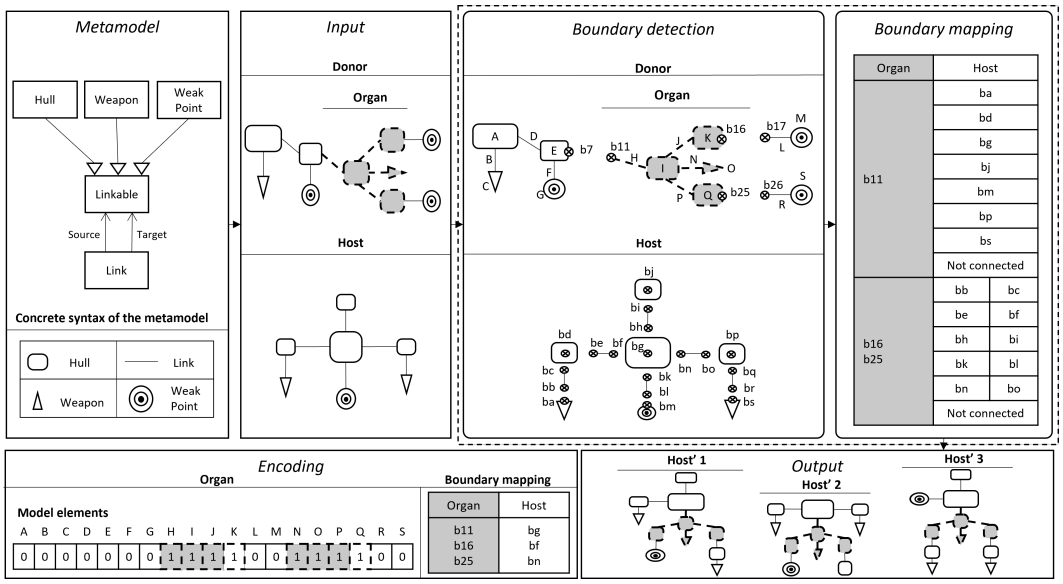


Figure 4.2: Overview of IMHOTEP on a running example.

IMHOTEP automatically identifies the boundaries of the selected organ, and all the boundaries of the host. In our running example, the boundaries of the organ are the connection points between donor and host. The elements that connect with the rest of the donor are H, K, and Q. Figure 4.2 *Boundary detection* **Donor** shows the donor, differentiating each element of the model with a letter from A to S, and the selected organ (namely, H, I, J, K, N, O, P, Q) with its boundaries (which are b11 for the H element; b16 for the K element, and b25 for the Q element). While, the host boundaries are all the points where its model elements connect. Figure 4.2 *Boundary detection* **Host** shows all the boundaries of the host of our running example: The host has a total of 19 boundaries identified by a tag from ba to bs.

4.2.3 Boundary mapping

In the boundary mapping step, IMHOTEP determines a mapping between the organ and the host boundaries. For each boundary in the organ, IMHOTEP considers all compatible boundaries of the host, including the possibility of not connecting the boundary to the host boundaries. The boundary compatibility is determined by the metamodel.

The table on the Figure 4.2 *Boundary mapping* shows a boundary mapping between the organ and the host of the running example. The boundary b11 is a

boundary from a 'Link' from the model and according to the metamodel it can connect to any 'Hull', 'Weapon', and 'Weak Point'. The boundaries b16 and b25 are both 'Hulls' and they can connect with any 'Link'.

4.2.4 Initialize population

In evolutionary algorithms, a population is a collection of possible solutions for a problem. The encoding is the problem representation that an algorithm is capable to understand.

In our work, the encoding requires a binary vector that represents the organ in the donor, and the boundary mapping (see Figure 4.2 *Encoding*). In the binary vector, each element from the model is a position in the vector. If a position in the vector has a '1', it means that the element from the model is part of the organ. On the other hand, each boundary from the organ gets assigned a compatible boundary from the host. The initial population of IMHOTEP contains individuals composed by the host and the organ placed in a random position (*i.e.* a random mapping between the organ boundaries and the compatible organ boundaries).

4.2.5 Genetic operators

IMHOTEP uses traditional genetic operators (namely, selection, crossover, and mutation) to generate new individuals (*i.e.* candidate solutions). Specifically, we use the ranking selection, which ranks the individuals based on the objective function and retains the top ones in the current population. We use a single, random, cut-point crossover, which selects two parent solutions at random, and determines a cut point uniformly at random to split them into two sub-vectors. Then, the crossover creates two children solutions by combining the first part of the first parent with the second part of the second parent for the first child, and the first part of the second parent with the second part of the first parent for the second child. Finally, the new offspring is mutated by changing any value of the encoding uniformly at random with a certain probability. Figure 4.2 *Output* shows an example of new individuals that could results from our running example. For simplicity, these individuals have unaltered organs, but illustrate different boundary mappings between organ and host.

4.2.6 Objective function

Our work proposes to harness video games' NPCs to run simulations that provide data to assess the transplants (*i.e.* to compute the value of the objective function assessing the quality of each transplant). Specifically, we propose to use the content generated via transplantation (each individual in the population) into a simulation of the video game. Such a simulation produces a data trace of the events that have occurred. Using the data from the trace, we can check how well aligned are the events with the intention of the developers. In our case study, the simulation is a duel between a spaceship and a boss. The simulation generates data about the duel, such as the damage inflicted. The intention of the developers may be that the duel ends with the victory of the spaceship with a remaining life of less than 10%. Our proposal does not require ad hoc development of simulations. In fact the simulations leverage mainly the NPCs (but also more video game elements, such as scenarios or items like weapons or powerups),which are usually developed anyway during for most types of video games. In other words, NPCs are integral components of most video game genres such as First-Person Shooter, Real-Time Strategy, or Racing Games. This use of simulations has two advantages: it makes the use of simulations cheaper (*i.e.* it does not involve additional development costs) and it facilitates fidelity to the video game compared to ad hoc development.

In our case study, IMHOTEP compute the objective function value for each individual in the population, through a simulation of a game battle between the boss generated via transplantation (*i.e.* the candidate solution, also referred to as Host') and an NPC spaceship. Since all these elements, as well as the scenarios and items such as weapons or powerups already belong to the game itself, no extra development is needed to run the simulation. Note that from now we can refer to the simulation-based version of IMHOTEP as $S_{Imhotep}$, to differentiate it from a more traditional objective function based on test-suite-compliance (referred as to $T_{Imhotep}$ herein).

Once a simulation is executed, we need a way to quantify its *quality*. One thing that differentiates video games from traditional software is that the basic requirement

of video games is 'fun'. 'Fun' is an abstract concept and the developers are in charge of interpreting it when creating a game. In fact, different developers may have different interpretations, also depending on the intended users of a given video game. For some, 'fun' is achieved with a difficult game that is very rewarding when progress is made (e.g., Dark Souls [226]), while for others, 'fun' is achieved by effortlessly killing enemies (e.g., Dynasty Warriors [227]). Therefore, we argue that such an intent is key for the evaluation of new generated content. Hence, to evaluate the quality of the candidate solutions generated by IMHOTEP we take into account the percentage of simulated player victories ($F_{Victory}$) and the percentage of simulated player health left once the player wins a duel (F_{Health}), which are commonly used metrics in the literature. Specifically, we compute $F_{Victory}$ and F_{Health} according to Blasco *et al.* [45], as described below:

$F_{Victory}$ is calculated as the difference between the number of human player victories (V_P) and the optimal number of victories (33%, according to the developers of Kromaia and their criteria) ($V_{Optimal}$):

$$F_{Victory} = 1 - (\mid V_{Optimal} - V_P \mid / V_{Optimal}) \tag{4.1}$$

F_{Health}, which refers to completed duels that end in spaceship victories, is the average difference between the spaceship's health percentage once the duel is over (Θ_P) and the optimal health level that the spaceship should have at that point ($\Theta_{Optimal}$, 20%, according to the developers):

$$F_{Health} = 1 - (\sum_{d=1}^{V_P} (\mid \Theta_{Optimal} - \Theta_P) \mid / \Theta_{Optimal}) / V_P \tag{4.2}$$

The $F_{Victory}$ and F_{Health} criteria are combined (*i.e.* averaged) in the objective function $F_{Overall}$ which guides the evolutionary search, as follows:

$$F_{Overall} = min\left(Validity, \sum_{i=1}^{N} F_i / N\right) \tag{4.3}$$

where $Validity$ is a crucial part to take into account the validity of newly generated

models by using a run-time interpreter which is already part of the game. In fact, such validation step is needed to discard models with inconsistencies. When a model is stated as non-valid by the interpreter the value of Validity will be 0. $F_{Overall}$ value is the minimum between Validity and the average value of $F_{Victory}$ and F_{Health}, thus it can assume a value in [0, 1].

To execute $T_{Imhotep}$, the Kromaia developers supplied a domain-specific test suite comprising 243 test cases, selected based on their expert knowledge of the system. The objective function of $T_{Imhotep}$ was evaluated by subjecting each individual to all 243 tests, recording the number of successful outcomes, and subsequently normalizing this value to the interval [0, 1]. Accordingly, an individual passing all tests attains an objective function score of 1, whereas an individual failing all tests receives a score of 0.

4.2.7 Software engineering reflections and tooling requirements

IMHOTEP is a novel system for Procedural Content Transplantation (PCT), designed to automate the reuse of game content by transplanting functional fragments (organs) from one game element (donor) into another (host). Grounded in software engineering principles, it addresses the high cost and complexity of manual content creation while positioning itself as both a technical tool and a methodological contribution to Game Software Engineering, emphasizing reuse, abstraction, and empirical validation.

The process begins with knowledge elicitation, where transplant candidates are defined. A donor provides the source model, an organ represents the reusable fragment, and a host is the target model that integrates the organ. This step ensures that transplantations are meaningful and contextually relevant, establishing the foundation for automated integration.

Integration is then performed through transplantation, where IMHOTEP uses boundary detection and mapping to identify how content fragments can be embedded into hosts. This approach mirrors software engineering practices such as interface matching and component integration, ensuring compatibility between donor fragments and host systems. By automating this process, IMHOTEP applies model-

driven engineering and design exploration to streamline the generation of playable content.

To ensure quality, IMHOTEP employs simulation-based evaluation rather than relying solely on static test suites. In-game simulations measure metrics such as victory rate and player health to assess how well transplanted content aligns with design goals. This reflects runtime validation and feedback-driven refinement, similar to practices found in DevOps and continuous integration/continuous deployment (CI/CD).

The implementation of IMHOTEP within a game development context requires careful consideration and adaptation to the specific needs of each project. Since the application of IMHOTEP is highly dependent on context, development teams must begin by defining an appropriate meta-model. This meta-model establishes the foundational rules and structures that will govern the integration process. The donor, organ, and host models must all adhere strictly to the rules of the defined meta-model in order to ensure consistency and compatibility throughout the system.

Once the meta-model has been defined, the development team must determine how organs are selected and how encoding and boundary functionalities are applied. These design decisions directly influence the operation of the evolutionary algorithm, which requires further customization to align with the chosen encoding scheme. Adaptations are also necessary for the genetic operations and the objective function, ensuring that they function effectively within the defined parameters of the project. Together, these steps allow IMHOTEP to be integrated seamlessly into game development workflows while remaining flexible enough to accommodate varying project requirements.

Chapter 5

Empirical Evaluation

In the previous chapter, we introduced our proposed approach, named IMHOTEP. In this chapter, we present the evaluation performed to validate our proposal.

To evaluate IMHOTEP we have carried out an industrial case study in collaboration with the developers of the commercial video game Kromaia[1]. Kromaia has been released on PC, PlayStation, and translated to eight different languages. In particular, in the Kromaia case study, we were able to assess the effectiveness of IMHOTEP to transplant 129 different *organs* extracted from the *scenarios* of Kromaia into five of its NPCs *bosses* that act as hosts, generating new video game *bosses*, for a total of 645 successful transplants. This is higher than previous work in the literature, which achieved at most 327 successful transplants [228].

We compare the quality of the 645 bosses generated by using IMHOTEP to the same number of bosses generate by using a search-based PCG approach from the literature [5], which is the most relevant state-of-the-art of a comparable nature, and those generated by a variant of IMHOTEP that uses test-suite as objective function (namely, $T_{Imhotep}$), in line with the traditional software transplantation literature. To perform the comparison, we rely on the concept of game quality and its automated measurement, which is widely accepted in practice [229].

The results show that, out of the three approaches, the content generated through the IMHOTEP obtains the best results: It yields 1.5x better results than $T_{Imhotep}$ and 2.5x better results than baseline. The statistical analysis shows that the differences

[1]See the official PlayStation trailer to learn more about Kromaia: `https://youtu.be/EhsejJBp8Go`

are statistically significant, and the magnitude of improvement is always large.

5.1 Experimental Design

In this section we explain the design of the experiments we perform to empirically evaluate IMHOTEP by using the commercial video game Kromaia. We present the research questions that we aim to answer, the evaluation method, and the implementation details.

5.1.1 Research Questions

IMHOTEP proposes a new angle for video game procedural content generation, and for this reason we need to assess how it compares to the established practice for PCG . This motivates our first research question:

RQ$_1$: *How does $S_{Imhotep}$ perform with respect to the current practice for PCG?*

To answer RQ1, we had to identify the most relevant and close work in the PCG literature. We identify the work by Gallota *et al.* [5] as the most representative benchmark for our study. Indeed, Gallota *et al.* proposed a hybrid Evolutionary Algorithm for generating NPCs, which combines an L-system with a Feasible Infeasible Two Population Evolutionary Algorithm. We choose Gallota *et al.* as PCG baseline because (1) it is of the same nature of IMHOTEP (*i.e.* it uses evolutionary computation), (2) it is specific for spaceships that can play the role of bosses which is comparable to the content of our case study, and (3) it achieves the best state-of-the-art results for this type of content.

Moreover, since we are the first to propose the use of a simulation-based objective function to guide the search for transplantation it is natural to compare it with the established practice in the software transplantation field, which instead relies on the use of a test suite to guide the transplantation. This motivates our second research question:

RQ$_2$: *To what extend using a simulation-based objective function to guide the transplantation is more effective than a test-based one for* IMHOTEP*?*

To answer RQ2 we empirically compare IMHOTEP guided by the simulation-based objective function described in Section 4.2.6 (which we refer to as $S_{Imhotep}$)

with a test-based variant of IMHOTEP (which we refer to as $T_{Imhotep}$). Specifically, $T_{Imhotep}$ uses an objective function based on the number of test cases that are passed by the transplanted software. The reason for considering this variant is that in traditional software transplantation the best results have been achieved by using the test suite as the objective function. In order, to run $T_{Imhotep}$, the Kromaia's developers provided us with a test suite relevant to the game, consisting of a total of 243 tests selected based on their domain knowledge. Therefore the value of $T_{Imhotep}$'s objective function was computed by running each individual through the 243 tests, recording the number of tests passed and normalizing this value in a scale of [0, 1]. An individual which passes the 243 tests will obtain an objective function score of 1, on the contrary if it does not pass any test it will obtain an objective function score of 0. As in $S_{Imhotep}$, each individual also needs to constitute a valid boss (i.e., solution), receiving a score of 0 if it does not represent a valid one according to the run-time interpreter (see Section 4.2.6).

5.1.2 Methodology

Figure 5.1 provides an overview of the process we followed to empirically assess IMHOTEP and answer RQs 1 and 2 for the Kromaia's case study. The top (white background) part shows the assets of the game itself (content) and the game development (test suite) that are used by the approaches. The middle (grey background) part shows inputs and outputs for each of the approaches compared herein. The bottom (white background) part shows the evaluation criteria used to assess the results.

5.1.3 Algorithms' Settings

As described in Section 4.2, developers need to select host and donors as input for IMHOTEP. In our empirical study, Kromaia's developers identified as hosts five different bosses (i.e., Vermis, Teuthus, Argos, Orion, and Maia), which constitute the full set of original bosses from Kromaia. While, as donors, they considered all Kromaia's scenarios and were able to identify 129 organs within them. Each host has more than a thousand model elements, while donor's organs have an average of 255 model elements. Then we run IMHOTEPwith the parameters shown in Table 5.1.

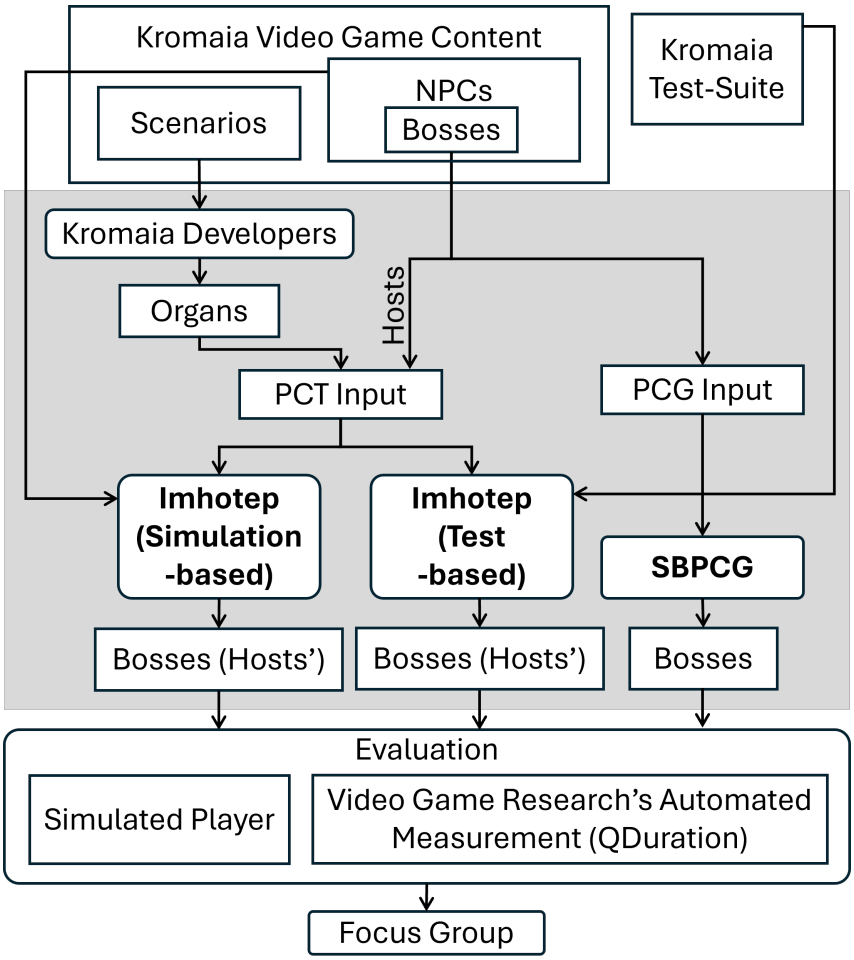


Figure 5.1: Overview of the evaluation process.

We established the stop condition at 2 minutes and 30 seconds, ensuring enough time to obtain suitable solutions.[2] At the end of the evolutionary process, each organ was successfully transplanted to each boss by IMHOTEP, which provided the developers with a total of 645 new bosses (5 hosts * 129 organs) (note we obtain 645 solutions from $S_{Imhotep}$ and 645 from $T_{Imhotep}$).

We executed the SBPCG benchmark by using the parameters presented by the original work and for a total of 129 times for each one of the 5 different hosts, so to obtain the same number of generated individuals (*i.e.* 645).

For all approaches we executed 30 independent runs to account for random variation [230]. Hence, we performed a total of 58,050 independent runs (645*3*30)

[2]The focus of this paper is not to tune the values to improve the performance of the approaches when applied to a specific problem, but rather to compare their performance in terms of solution quality on a level playing field.

Table 5.1: IMHOTEP parameter settings

Parameter description	Value
Stopping criterion	2m 30s
Population size	100
Number of parents	2
Number of offspring	2
Crossover probability	1
Mutation probability	1/150

for our experiment.

The implementation uses the Java(TM) SE Runtime Environment (JDK 1.8) and Java as the programming language. All experiments were run using two PCs with the following specifications: Intel Core i7-8750H, 16GB; and 2x Intel(R) Xeon(R) CPU X5660, 64GB.

5.1.4 Evaluation Measures

To compare the solutions provided by the SBPCG benchmark and the two variants of IMHOTEP (*i.e.* $S_{Imhotep}$ and $T_{Imhotep}$), we rely on the concept of game quality and its automated measurement through simulated players. The results by Browne *et al.* demonstrated the validity of this approach, which is now widely accepted in the research community [229]. Therefore, we need two ingredients to run our experiment: The simulated player and the automated measurement.

The simulated player, developed by the developers of Kromaia, possesses the ability to mimic human player behaviour. Our approach incorporates their algorithm, utilizing it to simulate battles between the generated bosses and the simulated player. Within these simulations, the simulated player confronts the boss, strategically targeting and destroying its weak points. Meanwhile, the boss operates in accordance with its anatomical structure, behavioural patterns, and attack/defensive dynamics, aiming to overcome the simulated player. Both entities within the simulation actively strive to emerge victorious, eschewing draws or ties, and ensuring a definitive win.

The automated measurement is $Q_{Duration}$ which was proven to achieve good results [229]. The duration of duels between simulated players and bosses units is expected to be around a certain optimal value. For the Kromaia case study, through tests and questionnaires with players, the developers determined that concentration

and engagement for an average boss reach their peak at approximately 10 minutes ($T_{Optimal}$), whereas the maximum accepted time was estimated to be 20 minutes ($2 * T_{Optimal}$). Significant deviations from that reference value are good design-flaw indicators: short games are probably too easy; and duels that go on a lot longer than expected tend to make players lose interest. The criterion $Q_{Duration}$ is a measure of the average difference between the duration of each duel (T_d) and the desired, optimal duration ($T_{Optimal}$):

$$Q_{Duration} = 1 - \frac{\sum_{d=1}^{Duels} \frac{|T_{Optimal} - T_d|}{T_{Optimal}}}{No.ofDuels} \tag{5.1}$$

Based on the equation above, the higher the $Q_{Duration}$ of a given approach, the better the solutions it produced.

5.1.5 Statistical Analysis

To measure whether there is any statistical significance difference between the results obtained by the different approaches we perform the Wilcoxon Ranked-Sum test (a.k.a. Mann–Whitney U test) [231] setting the confidence limit, α, at 0.05, and applying the Bonferroni correction (α/K, where K is the number of hypotheses) when multiple hypotheses are tested. We performed a one-sided test since we are interested in knowing if our proposed approach, $S_{Imhotep}$, would be better than the others. In such a case, the one-sided p-value interpretation would be straightforward. Specifically, for RQ1 we test the following null hypothesis: *The distribution of $Q_{Duration}$ values produced by $S_{Imhotep}$ is not better than that produced by the SBPCG benchmark*. If the test rejects the Null Hypothesis, the alternative hypothesis would be accepted: *The distribution of $Q_{Duration}$ values produced by $S_{Imhotep}$ is better than that produced by the SBPCG benchmark*. Similarly for RQ2 we test the following null hypothesis: *The distribution of $Q_{Duration}$ values produced by $S_{Imhotep}$ is not better than that produced by $T_{Imhotep}$*. If the test rejects the Null Hypothesis, the alternative hypothesis would be accepted: *The distribution of $Q_{Duration}$ values produced by $S_{Imhotep}$ is better than that produced by $T_{Imhotep}$*.

We consider the effect size to assess whether the statistical significance has

practical significance [232]. We use the Vargha and Delaney's $\hat{A}_{12}$ non-parametric effect size measure, as it is recommended to use a standardised measure when not all samples are normally distributed [232], as in our case. $\hat{A}_{12}$ measures the probability that an algorithm A yields greater values for a given performance measure M than another algorithm B, based on the following equation: $\hat{A}_{12} = (R_1/m - (m + 1)/2)/n$, where R_1 is the rank sum of the first data group we are comparing, and m and n are the number of observations in the first and second data sample, respectively. Values between $(0.44, 0.56)$ represent negligible differences, values between $[0.56, 0.64)$ and $(0.36, 0.44]$ represent small differences, values between $[0.64, 0.71)$ and $(0.29, 0.44]$ represent medium differences, values between $[0.0, 0.29]$ and $[0.71, 1.0]$ represent large differences.

5.2 Results

In this section, we present the results obtained by running IMHOTEP and the SBPCG benchmark on Kromaia. Table 5.2a shows the mean values and standard deviations for $Q_{Duration}$ for each IMHOTEP variant and the SBPCG benchmark, while Figure 5.2 shows the results in form of boxplots, grouped per host (*i.e.*, the boss of Kromaia used in our experiment, namely Argos, Maia, Orion, Teuthus, and Vermis) and overall. Each boxplot represents the distribution of $Q_{Duration}$ values (obtained as average of 30 independent runs) for each of the 645 solutions obtained from transplantation IMHOTEP ($S_{Imhotep}$ and $T_{Imhotep}$) and SBPCG. We can observe that both variants ($S_{Imhotep}$ and $T_{Imhotep}$) obtained better results than the SBPCG benchmark. Specifically, $S_{Imhotep}$ yielded the best results, followed by $T_{Imhotep}$ and then SBPCG. The variants obtained an average value of 44.85% in $Q_{Duration}$, with $S_{Imhotep}$ being the variant that obtained the best results overall (53.31% in $Q_{Duration}$). $T_{Imhotep}$ obtained 36.39% in the overall $Q_{Duration}$, which also outperformed SBPCG. SBPCG obtained the worst $Q_{Duration}$. Overall, the results reveal that leveraging simulations as objective function pays off in the context of PCT, yielding 1.5x better results than the $T_{Imhotep}$ and 2.5x better results than the SBPCG benchmark.

When analysing whether there is statistical significant differences among the

Table 5.2: RQ1-RQ2. (a) Mean value and standard deviation for $Q_{Duration}$ obtained by each approach per boss and overall. (b) Wilcoxon test and Vargha-Delaney $\hat{A}_{12}$ results obtained by comparing $S_{Imhotep}$ Vs. SBPCG (RQ1) and $S_{Imhotep}$ Vs. $T_{Imhotep}$ (RQ2) per boss and overall. $\hat{A}_{12}$: Large – L.

(a) Mean and standard deviation

	$S_{Imhotep}$	$T_{Imhotep}$	SBPCG
Boss	Mean ± StDev	Mean ± StDe	Mean ± StDe
Argos	43.92 ± 9.30	32.17 ± 6.94	20.15 ± 1.86
Maia	43.08 ± 12.09	29.52 ± 9.34	8.43 ± 1.81
Orion	48.86 ± 8.69	31.41 ± 6.83	32.97 ± 0.85
Teuthus	60.78 ± 7.38	46.33 ± 10.54	19.53 ± 1.88
Vermis	69.90 ± 10.52	42.50 ± 12.96	25.48 ± 3.31
Overall	53.31 ± 14.26	36.39 ± 11.72	21.31 ± 8.32

(b) Wilcoxon / $\hat{A}_{12}$

	RQ1	RQ2
Boss	$p-Value$ / $\hat{A}_{12}$	$p-Value$ / $\hat{A}_{12}$
Argos	$3.25x10^{-23}$ / 0.99 (L)	$1.28x10^{-18}$ / 0.85 (L)
Maia	$3.25x10^{-23}$ / 1.0 (L)	$6.64x10^{-18}$ / 0.85 (L)
Orion	$4.01x10^{-23}$ / 0.98 (L)	$4.95x10^{-22}$ / 0.95 (L)
Teuthus	$3.25x10^{-23}$ / 1.0 (L)	$3.60x10^{-18}$ / 0.87 (L)
Vermis	$3.25x10^{-23}$ / 1.0 (L)	$8.86x10^{-23}$ / 0.95 (L)
Overall	$1.41x10^{-107}$ / 0.98 (L)	$6.58x10^{-93}$ / 0.82 (L)

results obtained by $S_{Imhotep}$ and Base. We found that the obtained p-values for $Q_{Duration}$ are always lower than $4.01x10^{-23}$ (see Table 5.2b). This is below the significance threshold value, so we can comfortably state that $S_{Imhotep}$ provides significant better values for $Q_{Duration}$ with respect to Base. We also observe that all the A12 effect size values are large (see Table 5.2b), thus confirming the practical magnitude of such a difference. Thus, we conclude that: **Answer to RQ**$_1$ $S_{Imhotep}$ performance far surpasses SBPCG with statistically significant difference and large effect size in all cases, exhibiting a remarkable overall enhancement of 250% over SBPCG.

As for the comparison between $S_{Imhotep}$ and $T_{Imhotep}$ (RQ2), we observe that all the p-values achieved when comparing the $Q_{Duration}$ distributions provided by the two IMHOTEP variants are smaller than the significance threshold, thus indicating that the difference in solution quality is statistically significant in favour of $S_{Imhotep}$, and always with a large A12 effect size (see Table 5.2b). Therefore, we conclude that: **Answer to RQ**$_2$ $S_{Imhotep}$ provides significantly better results than $T_{Imhotep}$ in the context of automated content generation through transplantation, with a large effect size in all cases examined. The efficacy of $S_{Imhotep}$ demonstrates a 150% enhancement overall compared to the outcomes of $T_{Imhotep}$.

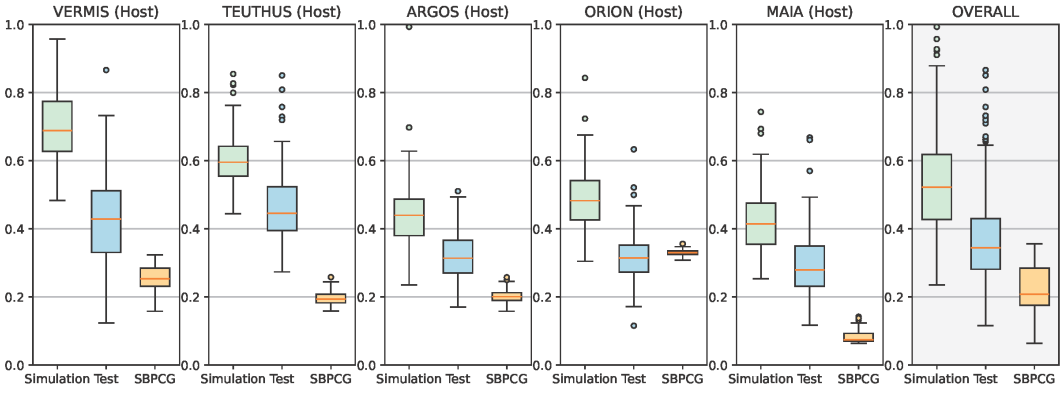


Figure 5.2: Results of IMHOTEP ($S_{Imhotep}$ and $T_{Imhotep}$) and the SBPCG benchmark in terms of $Q_{Duration}$.

5.3 Discussion

To begin with, our work revolves around the transplantation of organs between two very different types of content in video games: scenarios and bosses. One may wonder why not transplanting organs between contents of the same type, such as between bosses. Technically, it should also be a smaller challenge to transplant organs among the same type of content due to the similarities and shared structures. However, video games put the focus on fun, which is many times achieved by avoiding repetition. Since the number of bosses is usually very limited in video games, transplanting between bosses could lead to repetition, hurting fun and creating negative play experiences for the players. In contrast, scenarios provide an abundant and promising source of organs that can withstand repetition, since it is frequent for a relevant portion of a scenario to not be explored by a player during a game: while players spend most of the time playing within scenarios, the focus of scenarios on completing goals combined with their sheer extension renders them difficult to explore in full. Hence, reusing between bosses and scenarios is more original and relevant for fun.

Since transplanting an organ to a host contributes to generating new desirable content, one might consider performing more than one transplant on the same host to continue creating novel content. In its current state, our approach allows for only one organ to be transplanted at a time, but it should be possible to repeatedly transplant the same organ onto the same host, or to consider chains of transplants

where desirable combinations of organs can be identified and transplanted in bulk into a host. However, upon analysing the results, we have detected various interactions between organs that may help guide an approach that considered multiple transplants:

Organ dependencies occur when an organ requires for another organ to be present in the host to work properly. For instance, a spike weapon must be mounted on a hull belonging to the body of a boss and cannot appear by itself. In other words, a spike weapon organ depends on the existence of a hull organ to be able to be included in the boss.

Organ incompatibilities happen when an organ should not appear in the host under any circumstances. For instance, consider attaching a black hole organ to a hull belonging to the boss. The black hole organ destroys everything it touches, so it would instantly end the boss without triggering the end condition for the game, since the battle is considered as completed only when the player is the one responsible for ending the boss. This would actively block player progress, which is undesirable for the game.

Organ synergies are found when the functionality of an organ benefits from the existence of another organ in the host. For instance, adding one or more weapons to a hull where a weak spot is located protects the boss from the player, building a more interesting challenge.

Organ discordances take place when the functionality of an organ is hindered by the existence of another organ in the host. For instance, annexing a hull with a mobile arm to another hull with a laser may cause the laser beam to be intermittently blocked, decreasing its attack capabilities.

So far, the literature on software transplantation does not tackle or even identify interactions between organs. Studying these organ interactions is a line of work to advance the concept of transplantation both in video games and in the general software domain. As part of our evaluation(see Figure 5.1), we also carried out an informal focus group where we surveyed two developers from Entalto [233] and two developers from Kraken Empire [234]. All of them are seasoned video game developers who devote most of their working hours to realising the software behind

different commercial games. We asked them to express anonymously their content preferences, presenting them with the Kromaia's new content produced by either IMHOTEP or by the SBPCG benchmark (note that the source of the generated content was masked to the developers to avoid influencing their answer, *i.e.* they did not how the content was generated). The results showed an unanimous preference for IMHOTEP-generated content. Furthermore, they indicated that they would use it as primary content for the game rather than secondary.[3] Until now, previous PCG work has generated only results used as secondary content. In that sense, the possibility of using generated content as primary content represents an advancement in PCG. Developers justify this choice by arguing that the content generated by IMHOTEP aligns better with the vision of the game, whereas the SBPCG-generated content feels more random in purpose even when reusing content that was created within the context and vision of the game by the developers. These results have been confirmed in a subsequent larger empirical user-study [26] dedicated to compare content generated via IMHOTEP (more generally referred to as content reuse) and traditional search-based procedural content generation. In fact, this study reveled that developers favour the transplantation approach as they feel that it enhances the underlying content and yields superior outcomes compared to PCG [26]. The developers acknowledged content reuse in form of transplants as a natural progression of the initial original content, while PCG was unfavorably labeled as content that lacked the touch of professional developers.

5.4 Threats to Validity

To tackle possible threats to the validity of our work, we follow the classification suggested by De Oliveira *et al.* [235].

5.4.1 Conclusion Validity

To minimize *not accounting for random variation*, we run each of the approach (*i.e.* $S_{Imhotep}$, $T_{Imhotep}$and SBPCG) 30 times. Also, we make sure to assess the same

[3]Primary content is that which conforms an essential part of the experience of the players, while secondary content is that which does not directly affect the main experience but contributes to creating the atmosphere of the game (for instance, distant decoration).

number of solutions (*i.e.* 645 new bosses) for each of the approaches, so to make the comparison fair. In order to address the *lack of good descriptive statistics*, we present the standard deviation and a box-plot of the results. We also applied statistical significance tests (Mann-Whitney U) and effect size measurements ($\hat{A}_{12}$) following accepted guidelines [230]. We tackled the *lack of a meaningful comparison baseline* by comparing IMHOTEP to a recent and most relevant Search-Based PCG approach as a benchmark, as detailed in Section 6.1.

5.4.2 Internal Validity

We provide the source code and the artefacts used in our experiments to allow for reproduction and replication and avoid the *lack of discussion on code instrumentation*. We handled the *lack of real problem instances* by using a commercial video game as the case study for our evaluation and by working closely with its developers in a real-world industrial setting. Likewise, the problem artefacts (donor, organs and hosts) were directly obtained from the video game developers and the documentation itself.

5.4.3 Construct Validity

To prevent the *lack of assessing the validity of cost measures*, we made a fair comparison between the two variants of our approach and the SBPCG benchmark. Furthermore, we used a metric for the evaluation that has been widely adopted and *validated* by the research community [229].

5.4.4 External Validity

To mitigate the lack of *generalization* threat, we designed our approach to be generic and applicable not only to our industrial case study but also for generating content in other different video games. To apply IMHOTEP to another case study, it is necessary an encoding for the transplantation of the content, and leverage the NPCs to obtain the simulation of the objective function. To avoid the *lack of a clear object selection strategy* in our experiment, we have selected the instances from a commercial video game, which represents real-world instances. In fact, IMHOTEP can be applied where NPCs are available. NPCs are usually available in popular game genres such as

car games (rival drivers), FPS games (bots), or RTS games (rival generals). For those cases were there is no NPC, the developers should ponder the trade-off of the cost of developing the NPCs and the benefits of generating content with our approach. Our approach should be replicated with other video games before assuring its generalization.

Chapter 6

Controlled Experiment

While theoretical frameworks as we have seen in previous chapters provide a foundational understanding, empirical studies offer the necessary validation and refinement, which is crucial for effective implementations. As in other disciplines dealing with human behaviour (*e.g.*, social sciences or psychology), empirical research allows building a reliable knowledge base in software engineering [236, 237]. By empirically investigating the user experience of video game techniques, researchers can unveil both the strengths and limitations of existing approaches, paving the way for advancements that align more closely with the diverse needs and preferences of developers and players. Through rigorous experimentation and analysis, empirical studies serve as the cornerstone for fostering innovation and pushing the boundaries of what is achievable within video game techniques.

There are studies that establish the particularities of the study of the quality of video games compared to other software developments [238]. Video games have characteristics that are difficult to measure and define, such as 'fun' or 'entertainment'. Thus, it makes automating tests challenging due to the multitude of options available to players. Moreover, it is considered that the users under study must have specific characteristics and that not random profiles can be useful in the testing of this type of artefacts.

In this chapter, we aim to empirically assess and compare content generated by two different content generation techniques along with two different user profiles (players and developers). We study two automated approaches for generating content,

namely Procedural Content Generation (PCG) and Procedural Content Transplantation (PCT), and whether their use has an impact on the quality of the generated content. We do so by analyzing the commercial video game Kromaia released on PlayStation 4 and Steam. Specifically, we invite participants (developers and players) to play with content generated by PCG and PCT in the game, and then evaluate their experience in terms of video game specific percieved quality measures, namely 'difficulty', 'design','fun', and 'immersiveness'[6]. Participants were not told how the content was generated. We conducted three distinct sessions, one for players and the other two for developers, in order to investigate whether the profile of the participants assessing video games influences their perception. A total of 44 participants took part in the experiment, assessing the generated content in two scenarios of the game. The results show that the participants perceive the content generated by PCT to be of superior quality in comparison to the content generated by PCG: PCT obtained better results than PCG in 77% of the cases based on *difficulty*, in 34% cases for *design*, in 28% cases for *fun* and 5% for *immersiveness*.

Our findings challenge three prevailing trends in Game Software Engineering (GSE). Firstly, there is a perception that content reuse leads to repetitive game content, which is typically frowned upon by developers. However, our research indicates that subjects actually prefer content generated through PCT. Secondly, previous content generation experiments have involved only the players, neglecting the input of developers. Our results demonstrate no significant differences between players and developers. This suggests that the input of developers is also relevant for content generation. Furthermore, developers are shown to provide more detailed feedback. Lastly, 73% of previous content generation experiments have missed important factors such as hypotheses formulation, statistical analysis, or the inclusion of a replication package. We have not found any reasons for neglecting the aforementioned practices, and hence, our work encompasses all of the above - including replication, which has been overlooked in 100% of previous studies. We hope that our research will inspire future research in GSE to comply with empirical best practices.

6.1 Experimental Design

In this section we present the experiment design following Wohlin's guidelines [237] for reporting software engineering experiments.

6.1.1 Objective

The research objective has been organized using the Goal Question Metric (GQM) template to define the objectives originally presented by Basili and Rombach [239]. Our goal is to **analyze** different techniques for content generation, namely Procedural Content Generation (PCG) and Procedural Content Transplantation (PCT), **for the purpose of** comparison, **with respect to** perceived quality; **from the point of view of** of more and less experienced players and developers; **in the context of** content generation for an existing video game.

6.1.2 Research Questions and Hypotheses

The research questions and null hypotheses are as follows:

RQ1 - Does the **Technique** used to automatically generate software in video games impact the perceived *Quality* of the game? The corresponding null hypothesis is $H_{0,1}$: The **Technique** does not have an effect on the perceived *Quality* of the game.

RQ2 - Do evaluators with different profiles evaluate the quality of the game differently? The corresponding null hypothesis is $H_{0,2}$: The **Evaluator's profile** does not have an effect on the evaluation of the *Quality* of the game.

The hypotheses are formulated as two-tailed, as this is the first time these RQs are studied and there is no reason to assume that one approach is better than the other.

6.1.3 Variables

In this study, the factor under investigation is the content generation technique (**Technique**) used to automatically generate content, *i.e.*, final bosses, for an existing video game. There are two alternatives: PCG or PCT, which are the two different techniques used to generate a final boss that will be played with and evaluated by different kind of human participants. Since the goal of this experiment is to evaluate the effects of using different techniques to generate content for an existing commercial video game, we selected response variables related to the quality perceived by partic-

Table 6.1: Response variables and correspondent items in the evaluation questionnaire

Response variable	**Related Items in the evaluation questionnaire**
Boss difficulty	*Item*1. I think the boss difficulty is high
Design	*Item*2. The boss is perfectly integrated in the game *Item*3. I liked the design and behavior of the boss *Item*4. The boss I fought seemed to me to have a good balance between difficulty and playability
Fun	*Item*5. I enjoyed playing against the boss *Item*6. When the time was up, I was disappointed that I could not continue playing against the boss
Immersiveness	*Item*7. At no time did I want to give up while facing the boss *Item*8. At some point I was so involved that I wanted to talk directly to the video game

ipants playing the generated content. We selected *Quality* as the response variable to evaluate the effects of using different procedural content generation techniques in a commercial video game. We decomposed the analysis of quality into different dimensions: *difficulty*, *design*, *fun* and *immersiveness*, based on previous work [6].

To evaluate difficulty we used three response variables: *Game duration*, *Won rate* and *Boss difficulty*. *Game duration* is the average time spent by each participant in their games. The value of this variable was calculated by dividing the time each participant spent playing with a boss by the number of games played against that boss. *Won rate* is the percentage of games won by a player out of all games played against a boss, calculated by dividing the number of games won by the number of games played against a boss. We measured *Boss difficulty* based on the participant's answers to an explicit question about the difficulty of the game in a 7-item Likert-type questionnaire with different items. Different items in this questionnaire were used to measure the response variables *Design*, *Fun*, and *Immersiveness*. Each of these variables correspond to specific items in the questionnaire. The participants rated their degree of agreement with the statements of each item, with a value of 1 corresponding to totally disagree and 7 to totally agree. We average the scores obtained for these items to obtain the value for each variable. Table 6.1 shows the specific items of the questionnaire, used for the calculation of each of these response variables.

For the evaluation of each boss in the game, the participants also answered an open-ended question in which they could provide additional comments. We considered two response variables to quantify the qualitative information contained in these comments: *Comment length*, defined from the number of characters in the comment, and *Comment type*. To define the type of comment, the comments were classified into five categories by assigning them a numerical value from 0 to 4: 0, no comments; 1, comments not related to the evaluation of the boss; 2, comments on the difficulty of the boss evaluated; 3, comparisons between the bosses played; and 4, detailed analysis of the evaluation.

In order to establish the different evaluator profiles among the participants, we conducted different sessions of the experiment with specific groups of participants: potential players and experienced developers. In addition, a demographic questionnaire was designed to take into account the degree of experience both playing and developing video games, in particular, playing video games with similar characteristics to the one being evaluated. The groupings of participants in sessions by participant profile (player or developer) and the participants' responses to the demographic questionnaire were used to define three blocking variables: **Profile**, **Game development**, and **Gamer profile**. The objective was to analyze whether and how the experience in video game development and the profile as a player could influence the evaluation of the quality of the game elements.

The blocking variable **Profile** has two alternatives, player or developer, depending on the previous grouping of participants in sessions by profile. This variable also allows the study of the differences between the sessions held and the demographic profiles of the participants. To define the alternatives for the blocking variable **Game development**, the weekly hours that the participants dedicated to developing software for video games were taken into account. The variable will have two alternatives: 1, for participants who do not dedicate more than 10 hours per week to developing video games, and 2, for those who dedicate 10 hours or more to developing video games each week. The blocking variable **Gamer profile** is used to distinguish participants with a player profile that is closer to the target audience of

the video game being analyzed from participants with less related profiles, such as casual players or those who are not interested in video games. In order to define the alternatives of **Gamer profile** we considered the scores given by the participants to the following questions:

1. How many hours do you play video games per week? (1, Less than 5; 2, between 6 and 10; 3 between 11 and 20; 4, between 31 and 30; 5, between 31 and 40; and 6 more than 40.)

2. How would you rate your overall experience with video games (knowledge, playing time, skills)? (1, No experience; 2, Little experience; 3, Medium experience; 4, Very experienced; and 5, Expert in the area)

3. How would you rate your overall experience with shooter video games (Examples: Call Of Duty, Doom, Quake)? (1, No experience; 2, Little experience; 3, Medium experience; 4, Very experienced; and 5, Expert in the area)

4. What difficulty do you usually choose when playing video games? (1, Easy; 2, Normal; 3, Hard; 4, Extreme)

We defined three alternatives for the variable **Gamer profile** according to the sum of the scores given by the participants to the questions: 1, for participants scoring no more than 33% of the 20 possible points, 2 for participants scoring between 33% and 66% of the possible points and 3, for participants scoring 66% or more of the possible points. Participants in the third alternative of the variable could be considered the most similar to the target audience of the game, while participants in the first alternative would represent participants more distant from this audience.

6.1.4 Design

We chose a Two-Treament crossover design with two sequences using two different evaluation tasks: T1, evaluate a boss created using PCT, and T2, evaluate a boss created using PCG. The participants were randomly divided into two groups (G1 and G2). In the first period of the experiment, the participants of G1 perform T1 and the

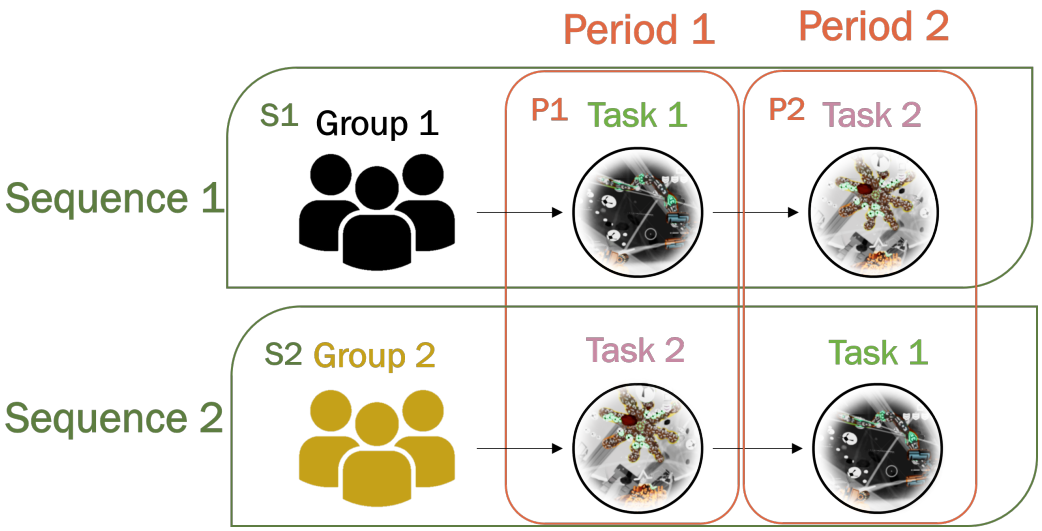


Figure 6.1: Two-Treament crossover design of our experiment.

participants of G2 perform T2. In the second period, the participants of G1 perform T2 and the participants of G2 perform T1.

This repeated measure design enhances the experiment's sensitivity, as noted by Vegas *et al.* [240]. Considering the same participant evaluating both alternatives, between-participant differences are controlled, thus improving the experiment's robustness regarding variation among participants. By using two different sequences (G1 evaluating PCT first and PCG afterwards, and G2 evaluating PCG first and PCT afterwards) the design counterbalances some of the effects caused by using the alternatives of the factor in a specific order (*i.e.*, learning effect, fatigue). We study the effects of the factors period, sequence, and participant to validate of this experiment.

To verify the experiment design, we conducted a pilot study with two participants. The pilot study facilitated an estimate of the time required to complete the tasks and questionnaires, the identification of typographical and semantic errors, and the testing of the online environment used to create the experiment. The participants in the pilot study did not participate in the experiment.

6.1.5 Participants

We selected the participants using convenience sampling [237]. A total of 46 participants with different knowledge about developing and playing video games performed the experiment, but only 44 decided to submit their answers and confirmed their agreement to be part of this study. In this study, the participants included 12 professionals related with video game development and 34 third year undergraduate students who are taking a course in *Software Quality* from different technology programs at a higher education institution (Universidad San Jorge). In particular, part of those students are specifically studying video games design and development.

The experiment was conducted by two instructors. During the experiment, one of the instructors gave instructions and managed the focus groups, and both instructors clarified doubts and took notes.

6.1.6 Experimental Objects

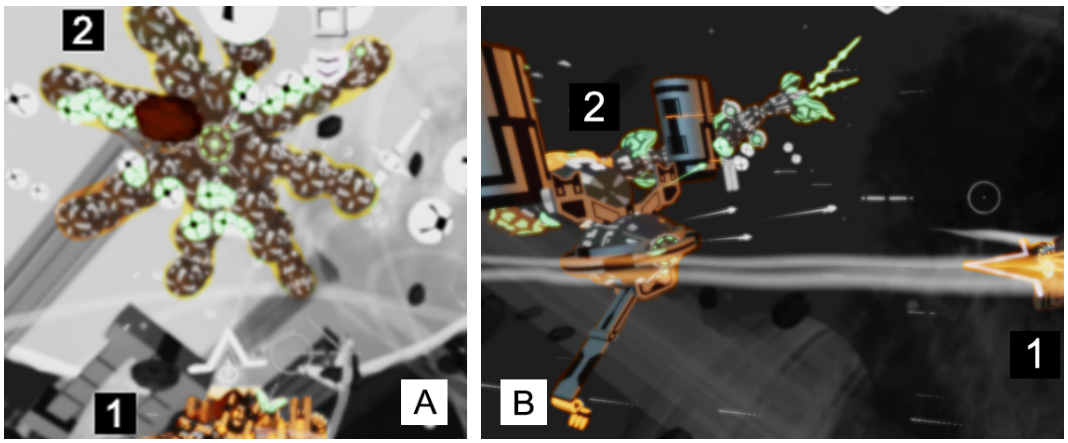


Figure 6.2: (A) PCG boss. (B) PCT boss.

In the experiment, the participants evaluate content (bosses created for an existing video game). Participants must defeat these bosses by piloting and shooting from a spaceship. Figure 6.2 shows the spaceship used by the player and the two bosses used during the experiment. The player's spaceship is highlighted in orange (see 1 of Figure 6.2), while the bosses are in black and green (see 2 of Figure 6.2). The scenario where the player fights the boss is the grey part, and the white balls are projectiles exchanged between the player's spaceship and the boss. The two bosses

shown in Figure 6.2 (PCG boss and PCT boss) are the two best bosses obtained with PCG and PCT according to the game's development team.

For the execution of this experiment, a video game engineer who was involved in the development of the game developed a test scenario based on scenarios from the original game. In this scenario, the participants of the experiment can (1) learn how to operate the game controls, (2) learn how to fight an original boss from the game, and (3) fight the bosses that they will have to evaluate.

For data collection, we prepared two forms using Microsoft Forms (one for each experimental sequence) with the following sections:

1. An informed consent form that the participants must review and accept voluntarily. It clearly explains what the experiment consists of and that the personal data will not be collected.

2. A demographic questionnaire that was used for characterizing the sample and defining the blocking variables.

3. Specific information on how to download and use the game's test environment that will be used to perform the experiment, and instructions on how to use the game environment.

4. Specific instructions on how to access the boss fight and the evaluation questionnaire about the game experience against the boss. This section was repeated three times in the questionnaires, once for each boss played by the participants: first against the original boss, and then against the two bosses generated with the techniques we compared (PCG and PCT).

The experimental objects used in this experiment (the test scenario, the generated bosses, and the forms used for the questionnaires), as well as the results and the statistical analysis, are available in the replication package at .

6.1.7 Experimental Procedure

The experiment was carried out in three different sessions. In the first session, the experiment was conducted face-to-face with the group of students. In the second

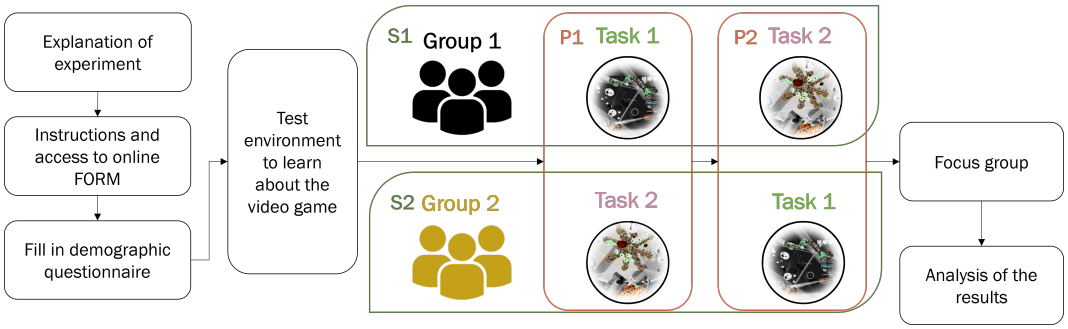


Figure 6.3: Our experimental procedure.

and third sessions, the experiment was conducted online with professionals. During the online session, all participants joined the same video conference via Microsoft Teams, and the chat session was used to share information or clarify doubts. The experiment was scheduled to last for 100 minutes and was conducted based on the experimental procedure described below:

1. An instructor explained the context of the experiment, the parts of the session and clarified that the experiment was not a test of the participants' abilities. (5 min)

2. The participants received clear instructions on where to find the links to access the forms for participating in the experiment and about the structure of these forms. The participants were randomly divided into two groups (G1 and G2). (10 min)

3. The participants accessed the online form, and they read and confirmed having read the information about the experiment, the data treatment of their personal information, and the voluntary nature of their participation before accessing the questionnaires and tasks of the experiment. (5 min)

4. The participants completed a demographic questionnaire. (5 min)

5. The participants received specific information on how to download, navigate through the files (see Figure 6.4), and use the test environment that will be used to conduct the experiment. They downloaded and used the test environment to

learn how to pilot the ship they will had to use to fight different bosses during the experiment. (15 min)

6. The participants received specific instructions on how to access a fight with an original boss of the game. After playing against the boss as many times as desired, the participants completed the evaluation questionnaire about the experience of playing against the original boss. (15 min)

7. The participants performed the first task. They received specific instructions on how to access a fight with the boss to evaluate. The participants of G1 played against the boss generated with RGC while the participants of G2 played against the boss generated with PCG. After playing as many times as desired against the assigned boss, all participants completed the evaluation questionnaire about the game experience against the boss played. (15 min)

8. The participants performed the second task. They received instructions on how to access a fight with the boss to evaluate. The participants of G1 played against the boss generated with PCG while the participants of G2 played against the boss generated with PCT. After playing as many times as desired against the assigned boss, all participants completed the evaluation questionnaire about the game experience against the boss played. (15 min)

9. One instructor conducted a focus group interview (see Table 6.2 about the tasks, while the other instructor took notes. (15 minutes)

10. Finally, a researcher analyzed the results.

6.1.8 Analysis Procedure

We have chosen the Linear Mixed Model (LMM) [241] for the statistical data analysis. LMM handles correlated data resulting from repeated measures, and it allows us to study the effects of factors that intervene in a crossover design (period, sequence, or participant) and the effects of other blocking variables (*e.g.*, in our experiment, profile, game development practice, and gamer profile) [240]. In the

Table 6.2: Focus group questions.

	Question
Question 1	Do your results match those of the simulators?
Question 2	Do you consider that there is an objective way to measure which is the best boss? which one(s)?
Question 3	Do you consider it necessary to have a specific profile to be able to evaluate the quality of a video game? Which one? Does it depend on the development phase of the video game you are in (design/implementation/testing/deployment/maintenance)?
Question 4	Do you consider that the questionnaire they have made takes into account the profile of the subject who completes it?
Question 5	Have you noticed any difference between the content generated by the two techniques applied?

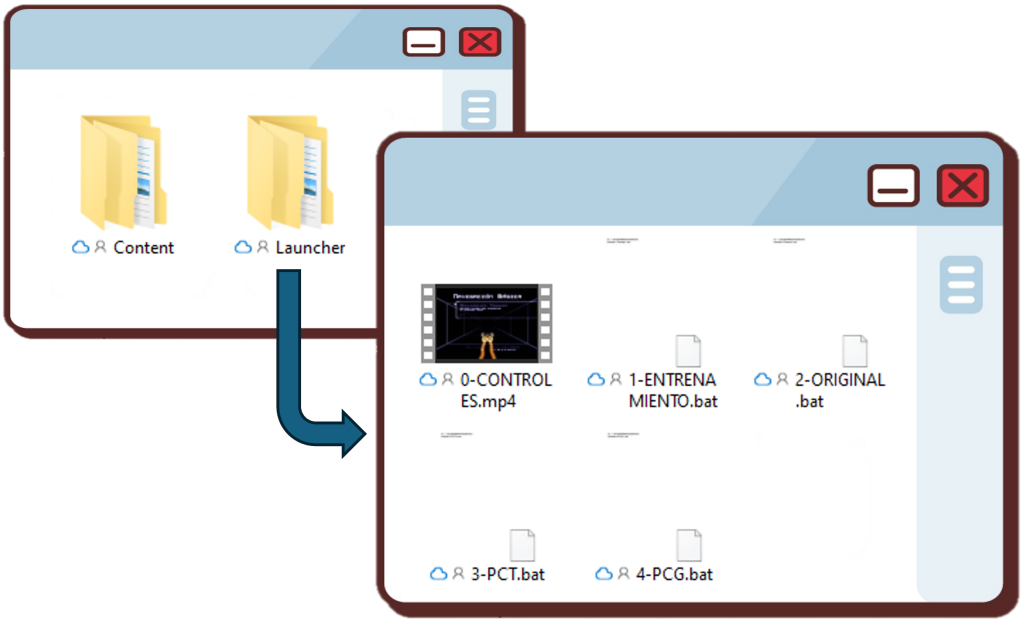


Figure 6.4: Files provided for the experiment.

hypotheses testing, we applied the Type III test of fixed effects with unstructured repeated covariance. This test enables LMM to produce the exact F-values and p-values for each response variable and each fixed factor.

In this study, **Technique** was defined as a fixed-repeated factor to identify the differences between using PCG or PCT, and the participants were defined as a random factor $(1|Subj)$ to reflect the repeated measures design. The response variables (RV) for this test were as follows: *Game duration*, *Won rate*, *Boss difficulty*, *Design*, *Fun*,

and *Immersiveness*, which were related to participants' perceived quality of the boss; *Comment length* and *Comment type*, which were used to determine differences in participants' comments.

In order to take into account the potential effects of factors that intervene in a crossover design in determining the main effect of **Technique**, we considered **Group** to be fixed factor with two alternatives: G1 and G2, corresponding to the two different sequences in which the bosses are evaluated. The first group of participants (G1) played and evaluated the boss generated with RGC, and then played and evaluated the boss generated with PCG. The second group of participants (G2) played and evaluated the boss generated with PCG, and then played and evaluated the boss generated with RGC.

In order to explore the potential effects of the blocking variables related to the evaluators' profile to determine the variability in the response variables, in the statistical model we also considered as fixed factors the blocking variables **Profile**, **Game development**, and **Gamer profile** and the combination of this variables with the principal factor **Technique**.

We tested different statistical models in order to find out which factors or blocking variables, in addition to **Technique**, could best explain the changes in the response variables. Some of these statistical models are described mathematically in Formula 6.1. The starting statistical model ($Model\,0$) reflects the main factor used in this experiment, **Technique** (Tech.)and the random factor $(1|Subj)$. We also tested other statistical models (*e.g.*, $Model\,1$, $Model\,2$, and $Model\,3$) that included the one or more of the additional fixed factors (AF) considered in the experiment (**Group**, **Profile**, **Game development**, or **Gamer profile**) or their interactions with the factor **Technique** ($Tech.*AF$) which could have effects on the response variables.

$$\begin{aligned}
&(Model\,0) \quad RV \sim Tech.+(1|Subj.)\\
&(Model\,1) \quad RV \sim Tech.+AF+Tech.*AF+(1|Subj.)\\
&(Model\,2) \quad RV \sim Tech.+AF_1+AF_2+CF_3+AF_4+(1|Subj.)\\
&(Model\,3) \quad RV \sim Tech.+AF_1+AF_2+Tech.*AF_1+(1|Subj.)
\end{aligned} \tag{6.1}$$

The statistical model fit of the tested models for each variable was evaluated based on goodness of fit measures such as Akaike's information criterion (AIC) and Schwarz's Bayesian Information Criterion (BIC). The model with the smallest AIC or BIC is considered to be the best fitting model [242, 243]. The assumption for applying LMM is the normality of the residuals of the response variables. To verify this normality, we used Kolmogorov-Smirnov and Shapiro-Wilk tests as well as visual inspections of the histograms and normal Q-Q plots. To describe the changes in each response variable, we selected the statistical model that satisfied the normality of residuals and also obtained the smallest AIC or BIC value.

To quantify the differences in the response variables due to the fixed factors considered, we calculated the Cohen *d* value [244], which is the standardized difference between the means of the response variables for each factor alternative. Values of Cohen *d* between 0.2 and 0.3 indicate a small effect, values around 0.5 indicate a medium effect and values greater than 0.8 indicate a large effect. We selected histograms and boxplots to describe the results graphically.

To verify that the group of measures associated with each response variable or fixed factor is consistent, we applied Principal Components Analysis (PCA) to the set of measures collected from the task sheets. PCA allows analyzing the structure of the correlations in a set of variables, identifying and establishing subsets of variables that have something in common with each other, but not with the rest. PCA produces components, which are new random variables that summarize the patterns of each subset of variables and are not correlated with each other [245, 246]. If the group of measures selected to define a variable (*e.g.*, the results of items 2, 3, and 4 to define variable *Design*) are in a single PCA component, the information from the measures is correlated and can be reduced to one variable, which would support the consistency of the proposed grouping of measures. On the other hand, if the measures used to define different variables are in different PCA components, we can interpret that they explain different aspects of the information contained in the measures and that there is no strong correlation between them.

6.2 Results

Principal Component Analysis (PCA) is a non-parametric method that starts from the diagonalization of the correlation matrix of a set of metrics. PCA was applied to the set of measures used to define the different response variables and factors of this work. In this work we applied PCA twice, one to the measures used to define the blocking variables, and another to the measures used to define the response variables. The results of this PCA executions are in the replication package. To determine the convenience of the application of PCA, we followed the recommendations of Tabachnick and Fidell [245] and Hair et al. [246] based on the factorability of the correlation matrix: the determinant of the correlation matrix of the factors considered is 0.019 (greater than 0.00001), the matrix is positive definite, the KMO index is 0.717 (greater than 0.7), and the p-value of 0.000 in the Bartlett sphericity test rejects that the correlation matrix is the identity matrix. To improve the interpretation of the components, a Varimax rotation with Kaiser normalization was performed [245, 246]. Each component extracted by PCA is a new random variable that summarizes the information of a subset of variables [245, 246]. In general terms, the extracted PCA components were consistent with the subsets of measures selected to define each variable.

The application of PCA to the measures used to define the response variables produced four components. The first component groups mainly the responses to the questions used to define the *Design* and *Fun* response variables, implying similar results in both variables. The second component groups the response variables related to the comments made by the subjects. The third component groups *Won rate* and *Boss Difficulty*, and the fourth component represents the *Game duration*. The responses to the questions used to define immersiveness were part of all of the previous PCA components, but they did not define clearly a single factor.

The application of PCA to the measures used to define the blocking variables **Profile**, **Developing games**, and **Gamer Profile**, produced two components, one defined mainly by the factor **Profile** and the other grouping the responses to the questions used to define **Gamer Profile**. The variability of the factor **Developing**

Table 6.3: Mean and standard deviation ($\mu \pm \sigma$) values of the dependent variables for the factor (Technique) in each alternative of the fixed factors. The light, medium and dark gray highlight indicates a small, medium or large effect.

	Technique	Profile		Developing Games		Gamer Profile			Group	
		Players	Developers	More than 10 h/Week	Less than 10 h/Week	Target Audience	Neutral	Non Target Audience	G1 (PCT-PCG)	G2 (PCG-PCT)
Game Duration	PCT	*4.24±2.85*	4.18±3.23	4.38±1.52	*4.05±3.27*	*4.57±1.95*	4.57±4.36	3.22±2.22	5.33±2.77	4.16±2.93	4.32±2.83
	PCG	*2.01±1.76*	2.19±2.02	1.54±0.55	*2.39±2.06*	*1.34±0.68*	1.58±0.54	2.01±1.38	2.13±2.34	2.21±2.28	1.79±0.93
	All	3.12±2.61	3.18±2.85	2.96±1.83	3.22±2.83	2.95±2.18	3.07±3.33	2.62±1.92	3.73±3	3.19±2.77	3.05±2.44
Won rate	PCT	*0.32±0.37*	0.33±0.39	0.29±0.33	0.3±0.39	0.36±0.35	0±0	0.25±0.32	0.5±0.39	0.41±0.38	0.22±0.34
	PCG	*0.71±0.39*	0.7±0.4	0.73±0.4	0.6±0.42	0.9±0.26	0±0	0.68±0.36	0.95±0.16	0.76±0.4	0.66±0.39
	All	0.52±0.43	0.52±0.43	0.51±0.42	0.45±0.43	0.63±0.41	*0±0*	*0.46±0.4*	*0.72±0.37*	0.59±0.42	0.44±0.42
Boss Difficulty	PCT	*5.41±1.68*	5.28±1.59	5.75±1.91	5.39±1.73	5.44±1.63	2.8±1.48	5.86±1.42	5.61±1.38	5.48±1.31	5.33±2.03
	PCG	*3.05±2.09*	2.84±2	3.58±2.31	3.61±2.25	2.06±1.34	6.2±1.79	3.43±1.96	1.72±0.9	2.96±2.16	3.14±2.06
	All	4.23±2.23	4.06±2.17	4.67±2.35	4.5±2.18	3.75±2.26	4.5±2.37	4.64±2.09	3.67±2.28	4.22±2.18	4.24±2.3
Design	PCT	*4.72±1.66*	4.53±1.64	5.22±1.66	4.63±1.79	4.88±1.42	4.6±2.23	4.73±1.7	4.74±1.54	4.17±1.61	5.32±1.53
	PCG	*3.53±1.47*	3.54±1.48	3.5±1.5	3.67±1.45	3.29±1.51	3.27±1.46	3.57±1.4	3.56±1.62	3.3±1.47	3.78±1.45
	All	4.13±1.67	4.04±1.63	4.36±1.78	4.15±1.69	4.08±1.65	3.93±1.91	4.15±1.64	4.15±1.67	*3.74±1.59*	*4.55±1.67*
Fun	PCT	*4.35±1.99*	4.13±2.05	4.96±1.76	4.18±1.98	4.66±2.03	4.2±2.17	4.29±1.96	4.47±2.09	4.09±1.92	4.64±2.07
	PCG	*3.4±1.81*	3.38±1.89	3.46±1.67	3.39±1.73	3.41±2.01	2.1±1.34	3.57±1.65	3.56±2.04	3.04±1.8	3.79±1.79
	All	3.88±1.95	3.75±1.99	4.21±1.85	3.79±1.89	4.03±2.09	3.15±2.03	3.93±1.82	4.01±2.09	3.57±1.91	4.21±1.96
Immersiveness	PCT	4.35±1.98	4.09±2.16	5.04±1.23	4.11±1.96	4.78±2.01	3.6±1.98	4.43±1.75	4.47±2.28	4.17±1.84	4.55±2.16
	PCG	4.16±1.81	4.06±1.78	4.42±1.94	4.16±1.66	4.16±2.1	3.4±2.27	4.38±1.58	4.11±1.97	4.07±1.71	4.26±1.94
	All	4.26±1.89	4.08±1.96	4.73±1.62	4.13±1.8	4.47±2.04	3.5±2.01	4.41±1.65	4.29±2.11	4.12±1.76	4.41±2.03
Comment Length	PCT	200.5±275	120±136	415±417	205±321	193±177	121±164	202±357	*221±193*	236±346	161±167
	PCG	177±223	86±807	336±156	160±156	144±155	123±170	177±170	136±133	148±172	160±135
	All	201±275	*103±112*	*375±311*	182±251	168±165	*122±157*	*189±273*	*179±169*	192±274	161±150
Comment Type	PCT	2.68±1.55	2.41±1.6	3.42±1.17	2.64±1.59	2.75±1.53	1.6±1.82	2.38±1.6	3.33±1.19	2.61±1.62	2.76±1.51
	PCG	2.55±1.62	1.94±1.63	3.67±1.16	2.32±1.7	2.56±1.71	1.6±2.19	2.38±1.75	2.67±1.5	2.09±1.62	2.76±1.73
	All	2.68±1.55	*2.17±1.62*	*3.54±1.14*	2.48±1.64	2.66±1.6	*1.6±1.9*	*2.38±1.65*	*3±1.37*	2.35±1.62	2.76±1.61

games, related to video game development time, is represented by the two previous components to similar degrees. This means that the variability it contains is explained by both factors, but not only by one of them. We decided to include the factor separately in the statistical analysis even though this result confirms positive correlations with the other two factors under consideration.

6.2.1 Changes in the Response Variables

There were differences in the means and standard deviations of all of the response variables related with the boss quality perceived by the subjects depending on which **Technique** was used to create the played boss. However, the differences in *Immersiveness* were small and there were also no large differences due to the factor **Technique** in the variables related to the subjects' comments. Table 6.3 shows the values for the mean and standard deviation of all the response variables considered (*Game duration*, *Won rate*, *Fun*, *Boss difficulty*, *Design*, *Fun*, *Immersiveness*, *Comment length*, and *Comment type*) for each one of the **Techniques** compared: PCG

and PCT, and for each one of the alternatives of the blocking variables and factors considered as fixed factors in the statistical analysis: **Profile**, with two alternatives (Players and Developers); for **Developing games** with two alternatives: subjects who perform video game development tasks for less than 10h per week (<10h/week) and subjects who dedicate more than 10 hours per week to these activities (>10h/week); **Gamer Profile**, with three alternatives: subjects with a player profile close to the target public of the game in which the evaluated bosses are contextualized (3), subjects with a player profile neutral (2) and subjects with a profile far removed from the target audience (1); and **Group**, whose two alternatives reflect the sequence in which subjects have played and evaluated the bosses generated with each technique (G1: PCT-PCG, G2: PCG-PCT). Note that Table 6.3 also shows the values of means and standard deviations by combination of the factor **Technique** with these variables. This allows us to illustrate both the effects that these variables have on the evaluation of a boss and the effects that they can have on the evaluation of the differences of bosses performed with different techniques. In Table 6.3 the pairs of values are shaded according to the effect size of their differences. The darker the shade, the larger the difference in the values of the response variables across the alternatives of the factors and blocking variables considered. Additionally, the italicised text highlights the statistically significant comparisons.

To quantify the differences in the response variables due to each factor or blocking variable, we analyzed the Cohen *d* values. Table 6.4 shows the Cohen *d* values of the response variables for all of the fixed factors considered in the statistical analysis. Positive values indicate differences in favor of the first alternative of the factors and negative values indicate differences in favor of the second alternative of the factor. Values indicating a small, medium or large effect due to a factor are highlighted in light, medium and dark gray, respectively. In the case of the blocking variable **Gamer Profile**, with three alternatives, the table shows the Cohen *d* values of all two-to-two comparisons of these alternatives. The values are shown in an order triad, where the Cohen *d* values between alternatives 1 and 2, 1 and 3, and 2 and 3 of the blocking variable are shown in this order.

Table 6.4: Cohen d values for the response variables for each fixed factor. Gamer Profile: 1=Non Target audience, 2=Neutral, and 3=Target audience.

	Technique (PCT vs PCG)	**Profile** (Players vs Developers)	**Developing Games** ($< 10h/week$ vs $\geq 10h/week$)	**Gamer Profile** (1vs2, 1vs3, 2vs3)	**Group** (G1vsG2)
Game duration	0.941	0.086	0.103	(0.203,-0.213,-0.448)	0.051
Won rate	-1.024	0.010	-0.434	(-1.265,-2.166,-0.667)	0.353
Boss difficulty	1.248	-0.272	0.339	(-0.067,0.363,0.448)	-0.009
Design	0.760	-0.194	0.039	(-0.128,-0.125,0.002)	-0.497
Fun	0.501	-0.235	-0.125	(-0.418,-0.417,-0.044)	-0.335
Immersiveness	0.102	-0.347	-0.177	(-0.527,-0.379,0.060)	-0.151
Comment Length	0.209	-1.456	0.061	(-0.261,0.338,0.046)	0.141
Comment Type	0.168	-0.910	-0.541	(-0.460,-0.936,-0.405)	-0.257

The effect size of a factor measure through the Cohen *d* value is related to the percentage of non-overlap between the distributions of the response variables for each alternative of the factor. Higher effect size correspond with greater percentages of non-overlap and larger differences. The histograms in Figure 6.5 illustrate the differences in *Won Rate* (left), *Design* (center),and *Immersiveness* (right) depending on the Technique use to generate the boss evaluated. In the *Won Rate* histogram, the non-overlapping parts are around 39%, which corresponds to a very large effect size and to a Cohen *d* value of more than 1. In the *Design* histogram, the non-overlapping parts are around 30%, which corresponds to a large effect size and to a Cohen *d* value of around 0.8. However, in the *Immersiveness* histogram, the non-overlapping parts are around 5%, which corresponds to a negligible effect size and to a Cohen *d* value around 0.

According to the Cohen *d* values of the response variables for **Technique** (first column of Table 6.4), we can affirm that the effect size of this factor for *Game Duration*, *Won rate*, and *Boss Difficulty* was large, with Cohen *d* values of 0.941, -1.024 and 1.248, respectively. The signs of these values indicate that the subjects' *Game duration* were longer with the PCT boss than with the PCG boss, but that the

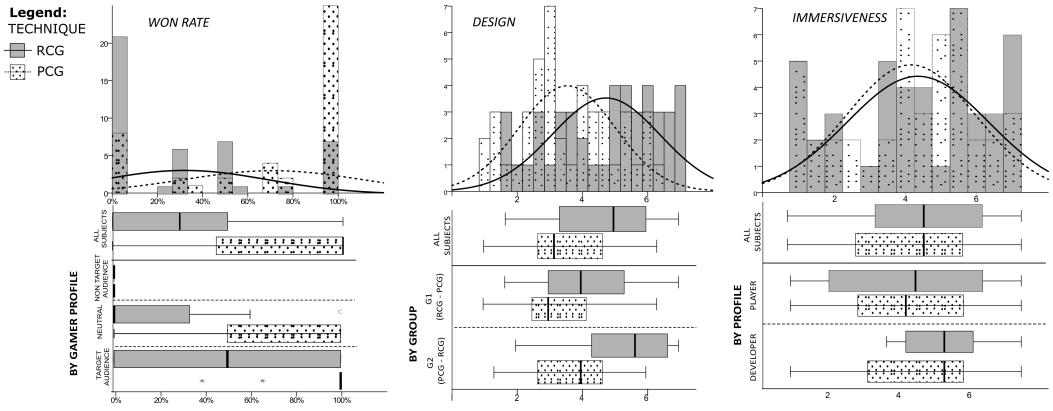


Figure 6.5: Histograms with normal distributions and box plots for Won Rate, Design and Immersiveness, with boxplots by the alternatives of Gamer Profile, Group and Profile respectively

Won rate is significantly lower, they win less often because the *Boss difficulty* of the PCT boss is higher than the PCG boss. The effect size of the factor **Technique** in favor of the PCT boss was medium for *Design* and *Fun* and negligible for the rest of variables with Cohen *d* values of less or around 0.2. Table 6.4 also shows the Cohen *d* values of the response variables for the fixed factors considered in the statistical analysis. The first six rows of the table show how the blocking variables has no effects on all the response variables related to the quality perceived by subjects and that these effects are only large in the case of **Gamer Profile** for *Won rate*.

The bottom part of Figure 6.5 shows ten pairs of box plots, arranged in rows and columns, illustrating the differences in *Won Rate*, *Design*, and *Immersiveness* due to some of the fixed factors considered. The first row of pairs of box plots corresponds to all of the subjects, and illustrates the differences in the response variables due to **Technique**. The following rows corresponds to the alternatives of the blocking variables considered in each response variable, and illustrates the differences due to this variable and its combination with **Technique**. The boxplots in the bottom left of Fig. 6.5 illustrate the large effects of the factors **Technique** and the blocking variable **Gamer Profile** in *Won rate*. The box plots in the bottom right of Fig. 6.5 illustrate the negligible effects of **Technique** (All subjects), and the medium effects of **Gamer Profile** in *Immersiveness*. The blocks of boxplots by fixed factor, after the first row of boxplots, also show the absence of differences of the blocking variables combined

with Technique, since the differences between one boss and the other do not depend on the alternative of the variable considered. For *Won rate*, in all alternatives, the won rate is higher or equal with the PCG boss than with the PCT boss, but for *Design* or *Immersiveness*, PCT boss outperforms PCG boss.

The forth column of Table 6.4 shows that the blocking variable **Gamer Profile** has effects in all the response variables except in *Design*. Cohen *d* values of *Won rate*, *Fun* or *Immersiveness* indicate that subjects with a profile farther away to the target audience (Alternative 1 of the variable) have a much lower *Won rate* than subjects closer from the target audience, in fact they didn't actually win any games (see the sixth column of the second row of Table 6.3). Subjects with non target audience profile also score worse on *Fun* or *Immersiveness* variables. In *Fun* and *Immersiveness* the differences between alternatives 2 and 3, neutral subjects or subjects closer to the target audience respectively, are negligible.

The values of the second column of Table 6.4 shown that the factor **Profile** has large effects on *Comment length* and *Comment type* in favor of developers. Developers made longer and better quality comments than players. The Cohen *d* values of the last two rows of the table, corresponding to the variables related to the quality of the subjects' comments, indicate that the best comments also come from subjects who spend more time **developing games** and from subjects with a **gamer profile** that is closer to the target audience.

6.2.2 Hypothesis Testing and Response to the Research Questions

The statistical linear mixed models used to explain the statistical significance of the changes in the response variables are different for each one of them. We selected the statistical models that obtained higher values for the AIC and BIC fit statistics from among all those that do verify the normality of the residuals. In addition, the use of the Linear Mixed Model (LMM) test assumed that residuals must be normally distributed. All of the residuals, except the ones carried out for *Game duration* and *Comment length*, obtained a p-value greater than 0.05 with the normality test. We obtained normally distributed residuals for *Game duration* and *Comment length* by using neperian logarithm transformation and cubic root transformation respectively.

For the statistical analysis of this variables with LMM, we used $RV = \ln(Comment\ length)$ and $RV = \sqrt[3]{Commentlength}$ in formula (1). For the rest of the variables, *RV* is equal to their value.

Table 6.5 shows the results of the Type III fixed effects test for each of the response variables or transformations, and for each fixed factor of the statistical model used in each case. Factors or combinations of factors that are not present in the statistical model selected to explain the variable are marked with the value NA or are not included in the table. Values indicating significant differences are shaded in grey. According to the results show in Table 6.5, not all the fixed factors included in the statistical models that explain the response variables produce significant changes in them. For example, to explain the variable *Game duration*, the statistical model used on the transformation of the variable ($RV = \sqrt[3]{Commentlength}$) was $RV \sim Tech.+DevGames+GamerP+Tech.*DevGames+(1|Subj.)$ with the fixed factors **Technique**, **Developing Games**, and **Gamer Profile**, and the combination of factor **Technique** and **Developing Games**, but there are significant differences in the response variable only for the factor **Technique** and the combination **Technique** and **Developing Games**. The changes in the *Game duration* due to the **Technique** used to create the boss being played are statistically significant, just as there are significant differences between the differences between the time spent playing each boss (RCT or PCT) as a function of the time spent developing video games (the alternatives of **Developing games**. As shown by the means and standard deviations of the time spent playing each boss as a function of the time spent developing video games (see Table 6.3 first three rows of third column), subjects who spend less time developing software played more time with the PCT boss and less time with the PCG boss than the time that subjects who spend more time developing video games spent playing with the same bosses.

Answer to RQ1. For all the response variables related to the quality perceived by subjects, except for *Immersiveness*, the differences due to **Technique** were statistically significant with p values of less than 0.05. Therefore, we can answer our first research question ***RQ1*** rejecting our first null hypothesis, $H_{0,1}$. The two

techniques compared in the experiment, PCT and PCG, result in bosses with different quality perceived by the subjects, and it can be concluded that the **Technique** has effects on the perceived *Quality* of the game. The effect size and direction of these differences previously described, suggest that the subjects perceive the boss generated by PCT to be of superior quality in comparison to the one generated with PCG.

Answer to RQ2. With regard to the second research question, ***RQ2***, the answer is that the null hypothesis $H_{0,2}$ cannot be completely rejected. Our results cannot confirm that the **Evaluator's profile**, represented by **Profile**, **Developing Games**, and **Gamer Profile**, has a significant effect on the evaluation of the *Quality* of a game. The results indicated that no significant changes were observed in the majority of the response variables used to evaluate the quality of bosses. The only statistically significant changes were observed in the comments made by the subjects and in the won rate.

Not all of the factors and blocking variables considered in the statistical analysis cause statistically significant differences in the response variables. In fact, for the blocking variables related to the evaluators profile, **Profile**, **Developing Games**, and **Gamer Profile**, no statistically significant differences were confirmed in any of the response variables related to the quality perceived by subjects, with the exception of *Won rate* and *Game duration*. The p-value of less than 0.001 for **Gamer Profile** in *Won rate* confirms the statistical significance that could be inferred in the previous subsection from the large effect size of the differences in the response variable due to **Gamer Profile**. Subjects who were the furthest from the target audience of the game did not win their games, while the closer the Gamer profile was to the target audience, the more the *Won rate* increased. However, there were not significant differences due to **Gamer Profile**, nor due to **Profile** or **Developing games**, in the evaluation of *Boss difficulty*, *Design*, *Fun*, or *Immersiveness*.

However, there are statistically significant changes in the variables related to the subjects' comments due to the blocking variables **Profile** and **Gamer Profile**. The p values of less than 0.05 for *Comment length* and *Comment type* in the last two rows

of the second and fourth columns of Table 6.5, confirm the statistical significance of these differences. Developers and subjects with a gamer profile that is closer to the target audience made statistically significant longer and better quality comments than players or, in particular, subjects further away from the game's target audience.

Table 6.5: Results of the Type III test of fixed effects for each response variable and factor, or factor's interactions. NA=Not Applicable

	Technique (Tech.)	**Profile**	**Developing Games (DevGames)**	**Gamer Profile (GamerP)**	**Group**	**Tech.*Profile**	**Tech.*DevGames**	**Tech.*GamerP**	**Tech.*Group**
ln(*Game Duration*)	F=43.369 ; p=<.001	NA	0.818;p=0.371	F=1.44; p=0.25	NA	NA	F=6.585; p=0.014	NA	NA
Won rate	F=38.542 ; p=<.001	F=1.884; p=0.178	NA	F=26.034; p=<.001	F=3.322; p=0.076	NA	NA	NA	NA
Boss Difficulty	F=30.358; p=<.001	F=1.299; p=0.261	NA	F=2.281; p=0.116	F=0.203; p=0.655	NA	NA	NA	NA
Design	F=16.445; p=<.001	F=0.257; p=0.615	F=0.575; p=0.453	F=0.081;p=0.922	F=4.301 ; p=0.045	NA	NA	NA	NA
Fun	F=8.199; p=0.007	NA	NA	F=0.666;p=0.519	NA	NA	NA	F=0.696; p=0.504	NA
Immersiveness	F=0.702; p=0.407	F=1.064;p=0.309	F=0.004; p=0.952	F=0.534;p=0.59	F=0.145; p=0.706	NA	NA	NA	NA
$\sqrt[3]{CommentLength}$	F=2.108 ; p= 0.154	F=27.315; p=<.001	F=2.104 ;p=0.155	F=3.784 ; p=0.031	NA	NA	NA	NA	NA
Comment Type	F=1.455; p= 0.234	F=18.069;p=<.001	F=3.564 ;p=0.067	F=7.959;p=0.001	F=2.692; p=0.109	NA	NA	NA	NA

6.3 Discussion

In the context of video games, reuse is not perceived as a completely positive practice. In fact, developers fear that reusing might be perceived as repetitive by players. On the other hand, the stochastic nature of PCG is perceived positively as an extension in the range of the creativity space for new content. Our experiment shows that this negative view of reuse is not aligned with the results. On the contrary, our results reinforce the PCT path, which boosts the latent content and leads to better results than PCG. During the focus group, subjects agreed that PCT was a natural evolution of the original content. In contrast, PCG was negatively classified as content that did not appear to have been developed by professional developers.

Previous studies considered only players as the subjects of the experiments. In our experiment, we go one step beyond and analyse the differences between players and developers. For researchers, it can be difficult to find developers to run experiments. However, that could not be the case for development studios. For instance, a large studio can enroll developers from different projects. This is relevant for studios because they put a lot of effort into enrolling players (not developers) for their games. It may seem paradoxical that it is hard to find players, but the experience of testing parts of a game in development is not the same as testing a full game as the developers in the focus group pointed out. Our experiment reveals that there are no relevant differences in terms of statistical values between players

and developers, suggesting that studios can leverage their developers. Furthermore, when it comes to feedback developers provided more beneficial feedback as the focus group acknowledged.

This experiment combines the specific quality aspects of video games ('design', 'difficulty', 'fun', and 'immersiveness') and the rigorousness of more traditional software work. This includes the provisioning of a replication package, something that no previous works did. One may think that the complexity of video games makes it difficult to design packages for replication. Nevertheless, we expect that our work along with the replication package will serve as a basis and inspiration for future researchers of the GSE community.

6.4 Threats to Validity

We use the classification provided by Wohlin *et al.* [237].

6.4.1 Conclusion Validity

We mitigated possible threats due to *low statistical power* by using a confidence interval of 95% for the statistical analysis. We also mitigated the *reliability of measures* by computing the evaluation measures directly from the data sheets automatically generated from the on-line questionnaire answers provided by the participants. Finally, we use an identical procedure in all the sessions of the experiment, to mitigate for possible threats arising from the *reliability of treatment implementation*.

6.4.2 Internal Validity

To mitigate the *instrumentation* threat, we conducted a pilot study to verify the design and the instrumentation of our study. The *interactions with selection* threat may affect the internal validity because there were subjects who had different levels of experience and, in general, different levels of knowledge of the video game domain. To mitigate this threat, the treatments were applied randomly and the statistical analysis includes the analysis of blocking variables related to participants' profile. The effects of the design factors, sequence and period, also have been included in the statistical analysis though the analysis of the factors **Group** (Sequence) and **Technique*Group** (Period). Only the variable *Design* had significant changes due

to the factor **Group**. The effect of this factor is medium with a Cohen *d* value of -0.497 in favor of subjects who play first with the PCG boss and after that with the PCT boss. The subjects in this group (G2, PCG-PCT) demonstrated a greater appreciation for the design of both bosses, both the PCT boss and the PCG boss, than the subjects in the group that carried out the experiment with the other sequence (G1, PCT-PCG). However, both groups value the design of the PCT bosses better than the PCG bosses. The box plots in the bottom center of Fig. 6.5 illustrate the effects of the factor **Group** and its combination with **Technique** in *Design*. The voluntary nature of participation also poses a *selection* threat, which we mitigated by inviting professional developers and students from a course whose content was in line with the experiment activities to avoid issues with student motivation.

6.4.3 Construct Validity

All of the measurements were affected by *Mono-method bias*. To mitigate this threat we mechanized the measures as much as possible by means of correction templates. The experiment may suffer from the *mono-operation bias* threat since we only compare two representative bosses of each technique.In order to mitigate the *author bias* threat, the tasks were extracted from a commercial video game and the bosses were selected by Kromaia's experts as the most representative of those obtained after the application of the two techniques compared. To weaken the *evaluation apprehension* threat, at the beginning of the experiment, the instructor explained to the participants that the experiment was not a test of their abilities, and that neither participation nor results would affect their grades in the course where the experiment took place.

6.4.4 External Validity

The *interaction of selection and treatment* may pose a threat to our experiment because a different number of participants took part in each alternative of the blocking variables, and players are more represented than developers. The *domain* threat occurs because the experiment has been conducted in a specific domain (video game) and for a very specific type of game, a spacial shooter. Other experiments using

Table 6.6: Overview of related work. Evaluation: generated content (A), variants of the proposed algorithm (VA), generated content compared to a baseline (C). Measures: Design (De), Difficulty (Diff), Fun (F), Immersiveness (I).

Work	Year	Evaluation	Measures	Hypotheses Formulation	Statistical Analysis	Replication Package	Sample
Cardamone *et al.* [158]	2011	VA	De	✗	✗	✗	5 players
Plans *et al.* [31]	2012	A	F	✗	✓	✗	31 players
Adrian *et al.* [192]	2013	VA	De, Diff, F	✗	✗	✗	22 players
Dahlskog *et al.* [197]	2013	VA	De, Diff, F	✗	✗	✗	24 players
Togelius *et al.* [113]	2013	A	De, Diff, F	✓	✓	✗	147 players
Gravina *et al.* [95]	2015	A	F	✗	✗	✗	35 players
Kaidan *et al.* [148]	2015	VA	De	✗	✗	✗	12 players
Olsted *et al.* [108]	2015	VA	De	✗	✗	✗	13 players
Prasetya *et al.* [157]	2016	C	F	✗	✗	✗	33 players
Ferreira *et al.* [6]	2017	VA	De, Diff, F, I	✗	✓	✗	139 players
Charity *et al.* [176]	2020	A	De, Diff	✗	✗	✗	2 players
Lopez-Rodriguez *et al.* [153]	2020	VA	Diff	✗	✗	✗	30 players
Kraner *et al.* [159]	2021	A	De	✗	✗	✗	5 players
Pereira *et al.* [44]	2021	C	Diff, F	✗	✓	✗	16 players
Brown *et al.* [88]	2022	A	De	✗	✗	✗	35 players
Our work	2024	PCGvs PCT	De, Diff, F, I	✓	✓	✓	32 players + 12 developers

different games should be performed in the future to further generalise our findings. We have carefully described our methodology and made a replication package publicly available in order to enable other researchers to replicate, reproduce and extend our study.

6.5 Related work

In this section we describe previous work involving human participants to assess automatically generated video game content, specifically focusing on the empirical elements of their experiments. We refer the reader to previous surveys in the field of automated content generation [2, 1, 25] to learn more about the latest trends and approaches to generate video game content.

Experimentation in Software Engineering is a practice that has been studied for decades [239]. Researchers have adopted established guidelines to be rigorous [237], such as hypotheses formulation, statistical analysis, or including a replication package. However, this has not always been the case for experimentation involving video games engineering, especially in the area of automated content generation., as explained below.

Video game content generation is a large field [37]. The types of generated content are diverse, such as vegetation [30], sound [31], terrain [32], Non-Playable Characters [33], dungeons [34], puzzles [35], or even the rules of a game [36]. However, it is difficult to find experiments with human participants that compare approaches [13]. Table 6.6 summarises this work. We observe that previous evaluations involving human participants mainly explore the quality of the content generated by one proposal [88, 113] or different variants of a same proposal [192, 44]. On the other hand, work such as the ones by Pereira *et al.* [180] and by Prasetya *et al.* [157] compared the content generated by their proposal against a baseline (see *Evaluation* in Table 6.6). Our work is the first that involves human participants to carry out a thorough comparison of two different previously proposed techniques generating content for video games.

In terms of measures, studies have been conducted to examine the distinctive characteristics of video games [238]. We observe that previous studies have investigated player preferences and perceptions regarding various aspects of video games [238]; this accounts for the use of different measures including design [148, 108], difficulty [153, 180], or fun [31, 157]. Another aspect of video games is the user engagement and immersion, which plays crucial roles in shaping the overall gaming experience [247] (see *Measures* in Table 6.6). Our work consider all these measures. Previous work have only asked players to evaluate content, i.e., they have not considered the perception of developers (see *Sample* in Table 6.6). In contrast, we study both the players assessment and the point of view of professional video game developers, and their differences when assessing the quality of the generated content. User empirical studies in PCG often employ a variety of methodologies to explore user experiences. These methodologies include user surveys [6], interviews [159], and usability testing [172]. Each method offers unique insights into different aspects of user interaction, such as user preferences, emotional responses, and usability issues.

Finally, none of the previous works adopt best practices for empirical studies, which are instead widely adopted in general software engineering research. In fact,

73% of the studies have neither hypotheses and validity, statistical analysis, or replication package (see Hypotheses Formulation, Statistical Analysis, and Replication Package columns of Table 6.6). Our work aims to compare the generated content with empirical rigor. To do so, we adopted the commonly followed guidelines for Software Engineering Research [248].

6.6 Conclusion

Until now, the majority of content generation experiments in game software engineering have failed to conform to best practices for Software Engineering research (*e.g.*, hypothesis and validity, statistical analysis, or replication package). Our research integrates the quality measures embraced by the video game community with the well-established practices of empirical software engineering research. Our results challenge the current dogma by highlighting that content reuse provides advantages towards content generation. Additionally, our findings unlock new possibilities for engaging developers in experimental endeavors. Ultimately, our work can encourage for the empirical game software engineering community to align with the established empirical practices in general software engineering research.

Chapter 7

Conclusions

This chapter serves as the concluding section of this PhD thesis, which has introduced our automated software transplantation methodology implemented in IMHOTEP for the generation of video game content.

In Chapter 3, we conducted a comprehensive survey of the state of search-based procedural content generation (SBPCG). Building on two earlier surveys [2, 1], we proposed a refined taxonomy to systematically classify recent research. This effort allowed us to document progress across Game Bits, Game Space, Game Systems, Game Scenarios, and Game Design, while also revealing that many challenges remain unresolved. Issues such as the efficiency of online PCG, the solvability, fairness, and diversity of generated content, the potential of bricolage, and the use of statistical rigor continue to require deeper investigation. By identifying both achievements and gaps, this chapter provides the foundation for introducing and evaluating our own approach, IMHOTEP.

In Chapter 4, we present a novel technique for Procedural Content Transplantation (PCT) that reimagines PCG through the metaphor of software transplantation. Instead of generating content entirely from scratch, developers can now treat game content as organs to be transplanted from a donor into a host. This approach opens the door to questioning how transplantation compares to established PCG practices and whether it can produce higher-quality or more diverse results.

Chapter 5 advances this idea where we demonstrate the large-scale application of IMHOTEP in a commercial video game, successfully performing 645 transplants

across diverse content types. Beyond confirming the feasibility of transplantation, we enhanced the method with a simulation-based objective function that guides the search more effectively than traditional test-based strategies. This improvement invites reflection on whether simulation-based evaluations are better suited to capturing gameplay dynamics and producing superior results. We also highlight the potential of co-evolving both transplants and their simulations, by varying NPCs, items, or scenarios, to better understand the contexts in which transplanted content excels and to support cross-game transplantation.

Finally, in Chapter 6, we introduce a controlled experiment with human participants designed to assess IMHOTEP alongside an existing PCG technique. This study extends the evaluation beyond technical performance to consider perceived quality: does the underlying content generation method influence how players experience the game? Our findings show that content reuse and transplantation can indeed shape perceptions of quality. Moreover, we observed that evaluators with different backgrounds assessed the generated content in distinct ways, underscoring the importance of accounting for diverse perspectives when validating PCG techniques.

Together, these chapters provide a coherent narrative: a broad survey of the field (Contribution 1), the proposal of a novel technique (Contribution 2), its empirical validation in an industrial context (Contribution 3), and its assessment with human participants (Contribution 4). By addressing how transplantation compares to current practices, whether simulation-based objectives improve effectiveness, how generation techniques affect perceived quality, and how evaluator profiles influence judgments, this research advances both the theory and practice of PCG while grounding the field more firmly in rigorous empirical methods.

7.1 Future Work

The findings of this study highlight both the promise and the current limitations of IMHOTEP, pointing to several important opportunities for future research. While the controlled evaluation provided valuable evidence of feasibility and potential impact, broader investigations are required to understand how the approach can scale,

adapt, and integrate into real-world game development environments. The following directions outline key areas where further work could extend and strengthen the contributions of this research.

While this work provides valuable insights through controlled evaluation involving both players and developers, several avenues remain for future exploration. One key limitation is the lack of direct developer feedback on the usability, integration, and long-term adoption of IMHOTEP within real-world workflows. Although the initial findings suggest promise, further studies are needed to assess its acceptance among professional developers, particularly regarding cost, workflow compatibility, and maintainability. Dedicated usability studies, interviews, or field deployments would offer deeper insights into developers' needs and expectations, guiding refinements that better support seamless integration into existing game development pipelines.

Another limitation concerns the evaluation context. The current study focused on a single game, produced by a single company, and centered on a specific boss encounter. While this scope was justified given the complexity of the domain and practical constraints of in-depth experimentation, it limits the generalizability of the findings. Future work should expand evaluations to encompass multiple games, diverse development teams, and a wider range of gameplay scenarios. Such efforts would help validate the adaptability and robustness of the approach across varied development contexts, thereby strengthening its practical relevance and potential impact.

In addition, the rise of Generative AI presents new opportunities and challenges for PCG. Many state-of-the-art generative approaches rely on large training datasets to produce high-quality and diverse outputs, which can be difficult to obtain in the context of bespoke game development. PCT offers a compelling complement by enabling a wide range of content to be generated from relatively limited resources. By systematically reusing and adapting existing assets, it could provide training data for generative models or function as a lightweight alternative when large-scale data collection is infeasible. Exploring this intersection between transplantation

and generative methods represents a promising avenue for broadening the creative capacity of procedural content generation while reducing dependency on massive datasets.

Finally, the challenge of scaling evaluations, especially those that involve subjective judgments of content quality, suggests opportunities for alternative methodologies. Crowdsourcing offers a compelling avenue, for instance through the release of free demo versions featuring automatically generated content. This would allow the collection of large-scale player feedback in naturalistic settings, capturing diverse perspectives and engagement patterns. Beyond validating perceived quality, such large-scale evaluations could also reveal edge cases, inform design improvements, and ultimately enhance the robustness of the content generation system.

Appendix A

Summary of surveyed papers

Ref	Year	Venue	Encoding	Objective Function	Content
GAME BITS					
[85]	2018	EvoApplications	Indirect	Direct - Theory Driven	Textures
[86]	2018	EvoApplications	Indirect	Direct - Theory Driven	Textures
[87]	2020	Multimedia Tools and Applications	Indirect	Direct - Theory Driven	Textures
[88]	2023	ToG	Indirect	Direct - Theory Driven	Textures
[31]	2012	T-CIAIG	Indirect	Interactive - Implicit	Sound
[94]	2013	PCGames	Indirect	Interactive - Implicit	Weapons
[95]	2015	CIG	Indirect	Simulation - Static	Weapons
[3]	2016	GEM	Indirect	Direct - Theory Driven	Weapons
[30]	2021	CISTI	Indirect	Direct - Theory Driven	Vegetation
GAME SPACE					
[4]	2012	Soft Computing	Indirect	Direct - Theory Driven	Terrains
[100]	2012	CIG	Indirect	Direct - Theory Driven	Terrains
[101]	2016	EvoCOP	Indirect	Direct - Theory Driven	Terrains
[102]	2011	EvoCOP	Direct and Indirect	Simulation - Static and Direct - Theory Driven	Shooter Maps

[107]	2014	CIG	Direct	Simulation - Static	Shooter Maps
[108]	2015	CEC	Direct	Interactive - Explicit	Shooter Maps
[109]	2017	CIG	Direct	Simulation - Static	Shooter Maps
[110]	2018	TOG	Direct	Simulation - Static and Interactive - Explicit	Shooter Maps
[114]	2012	GAME-ON	Indirect	Simulation - Static	Strategic Maps
[122]	2012	EvoCOP	Indirect	Direct - Theory Driven	Strategic Maps
[113]	2013	Genet. Program. Evolvable Mach.	Indirect	Direct - Theory Driven	Strategic Maps
[249]	2013	GECCO	Direct	Direct - Theory Driven	Strategic Maps
[115]	2013	EvoCOP	Indirect	Simulation - Static	Strategic Maps
[116]	2013	LSSC	Indirect	Simulation - Static	Strategic Maps
[118]	2013	SEED	Indirect	Direct - Theory Driven	Strategic Maps
[117]	2014	Natural Computing	Indirect	Simulation - Static	Strategic Maps
[119]	2014	CEC	Indirect	Direct - Theory Driven	Strategic Maps
[120]	2014	Entertainment Computing	Indirect	Direct - Theory Driven	Strategic Maps
[124]	2015	CEC	Indirect	Direct - Theory Driven	Strategic Maps

[250]	2015	CEC	Indirect	Direct - Theory Driven	Strategic Maps
[123]	2017	CoSECivi	Indirect	Direct - Theory Driven	Strategic Maps
[125]	2018	CIG	Indirect	Direct - Theory Driven	Strategic Maps
[199]	2020	CoG	Indirect	Simulation - Static	Strategic Maps
[127]	2021	TETCI	Indirect	Direct - Theory Driven	Strategic Maps
			GAME SYSTEM		
[131]	2012	SBGames	Direct	Simulation - Static	Entity Behaviour
[128]	2013	ECAL	Indirect	Direct - Theory Driven	Entity Behaviour
[132]	2014	SBGames	Direct	Simulation - Static	Entity Behaviour
[129]	2017	GHITALY	Indirect	Direct - Theory Driven	Entity Behaviour
[133]	2017	Soft Computing	Direct	Simulation - Static	Entity Behaviour
[134]	2020	CEC	Indirect	Direct - Theory Driven	Entity Behaviour
[43]	2021	Multimed. Tools Appl.	Indirect	Interactive - Implicit	Entity Behaviour
[45]	2021	JSS	Indirect	Simulation - Static	Entity Behaviour
[44]	2021	SBGames	Indirect	Direct - Theory Driven	Entity Behaviour
[5]	2022	GECCO	Indirect	Direct - Theory Driven	Entity Behaviour

[33]	2022	CoG	Indirect	Direct - Theory Driven	Entity Behaviour
[130]	2023	Multimed. Tools. Appl.	Indirect	Simulation - Static	Entity Behaviour
GAME SCENARIOS					
[136]	2011	T-CIAIG	Direct and Indirect	Direct - Theory Driven and Simulation - Static	Mazes
[137]	2011	Computational Intelligence Magazine	Indirect and Direct	Direct - Theory Driven	Mazes
[138]	2011	CEC	Direct and Indirect	Direct - Theory Driven	Mazes
[139]	2011	CIG	Direct and Indirect	Direct - Theory Driven	Mazes
[140]	2012	CIG	Direct and Indirect	Direct - Theory Driven	Mazes
[144]	2015	ACALCI	Direct and Indirect	Direct - Theory Driven	Mazes
[142]	2015	CGAMES	Direct	Direct - Theory Driven	Mazes
[145]	2016	Connection Science	Direct and Indirect	Direct - Theory Driven	Mazes
[141]	2016	CEC	Direct	Direct - Theory Driven	Mazes
[143]	2018	IIAI-AAI	Direct	Direct - Theory Driven	Mazes
[17]	2013	CIG	Indirect	Direct - Theory Driven	Physics
[146]	2014	ACE	Direct	Simulation -Static	Physics

[147]	2014	CIG	Direct	Simulation - Static	Physics
[148]	2015	GCCE	Indirect	Interactive - Implicit	Physics
[152]	2016	EvoCOP	Indirect	Direct - Theory Driven and Simulation - Static	Physics
[6]	2017	T-CIAIG	Direct	Simulation - Static	Physics
[149]	2019	EvoApplications	Indirect	Direct - Theory Driven and Simulation - Static	Physics
[150]	2019	IJCCI	Indirect	Simulation - Static	Physics
[151]	2019	GECCO	Direct	Simulation - Static	Physics
[153]	2020	OLA	Direct	Direct - Theory driven and Simulation - Static	Physics
[156]	2011	T-CIAIG	Indirect	Simulation - Static	Tracks
[158]	2011	GECCO	Indirect	Interactive - Explicit	Tracks
[7]	2015	Applied Soft Computing	Indirect	Interactive - Explicit	Tracks
[157]	2016	IJEEI	Indirect	Direct - Theory Driven	Tracks
[162]	2012	PCGames PCG Workshop	Indirect	Simulation - Static	Rooms
[166]	2015	CEEC	Indirect	Simulation - Static	Rooms
[167]	2019	CoG	Direct	Simulation - Static	Rooms
[171]	2019	ICGA	Indirect	Direct - Theory Driven	Rooms
[168]	2020	Applied Soft Computing	Direct	Simulation - Static	Rooms
[174]	2020	FDG	Direct	Simulation - Static	Rooms
[174]	2020	CoG	Direct	Simulation - Static	Rooms

[175]	2020	ToG	Direct	Interactive - Explicit	Rooms
[177]	2020	AIIDE	Indirect	Direct - Theory Driven	Rooms
[160]	2021	ISSSR	Direct	Simulation - Static	Rooms
[159]	2021	NT	Indirect	Direct - Theory Driven	Rooms
[172]	2021	ToG	Direct	Direct - Data Driven	Rooms
[126]	2022	SBGames	Indirect	Direct - Theory Driven	Rooms
[161]	2022	FDG	Direct	Simulation - Static	Rooms
[169]	2022	Appliad Science	Direct	Direct - Theory Driven	Rooms
[170]	2022	SIC	Direct	Direct - Theory Driven	Rooms
[178]	2023	ToG	Indirect	Simulation - Static	Rooms
[8]	2012	ICPS	Indirect	Direct - Theory Driven	Dungeon
[181]	2016	EvoCOP	Indirect	Direct - Theory Driven	Dungeon
[188]	2017	CEEC	Indirect	Direct - Theory Driven	Dungeon
[186]	2017	GECCO	Indirect	Direct - Theory Driven	Dungeon
[189]	2018	Computers	Indirect	Direct - Theory Driven	Dungeon
[179]	2018	Computation CEC	Indirect	Direct - Theory Driven	Dungeon
[182]	2018	CIG	Direct	Direct - Theory Driven	Dungeon
[183]	2018	SBGames	Indirect	Direct - Theory Driven	Dungeon

[185]	2018	ToG	Direct	Direct - Theory Driven	Dungeon
[184]	2020	Applied Intelligence	Direct	Direct - Theory Driven	Dungeon
[180]	2021	Expert Syst. Appl.	Indirect	Simulation - Static	Dungeon
[187]	2022	Applied Soft Computing	Direct	Direct - Theory Driven	Dungeon
[196]	2011	ACE	Direct	Direct -Theory Driven	Timeline
[192]	2013	CIG	Indirect	Direct - Theory Driven	Timeline
[197]	2013	DPG	Indirect	Direct - Theory Driven	Timeline
[198]	2014	EvoCOP	Indirect	Direct - Theory Driven	Timeline
[191]	2015	EvoCOP	Indirect	Simulation - Static	Timeline
[193]	2017	CSIEC	Indirect	Direct - Theory	Timeline
[201]	2018	FDG	Indirect	Direct - Theory and Simulation - Static	Timeline
[194]	2018	EECSI	Indirect	Direct - Theory	Timeline
[202]	2019	GECCO	Indirect	Simulation - Static	Timeline
[203]	2019	CoG	Indirect	Simulation - Static	Timeline
[195]	2020	JPCS	Indirect	Simulation - Static	Timeline
[80]	2020	GECCO	Direct	Simulation - Static	Timeline
[200]	2022	Genet. Program. Evolvable Mach.	Indirect	Direct - Theory Driven	Timeline
[206]	2021	GI	Indirect	Direct - Theory Driven	Stories
[207]	2021	FDG	Indirect	Direct - Theory Driven	Stories

[210]	2022	FDG	Indirect	Direct - Theory Driven	Stories
[209]	2022	FDG	Indirect	Direct - Theory Driven	Stories
[208]	2022	Entertain. Comput.	Indirect	Direct - Theory Driven	Stories
GAME DESIGN					
[212]	2014	IJAIT	Indirect	Simulation - Static	System Design
[9]	2015		Indirect	Simulation - Static	System Design
[213]	2016	EvoCOP Arxiv	Indirect	Simulation - Static	System Design
[214]	2023	CoG	Direct	Simulation - Static	System Design
[217]	2012	EvoCOP	Indirect	Direct - Theory	Camera Control

Bibliography

[1] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications and Applications*, 9(1):1–22, 2013.

[2] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186, 2011.

[3] Daniele Gravina, Antonios Liapis, and Georgios N Yannakakis. Constrained surprise search for content generation. In *Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2016.

[4] Miguel Frade, Francisco Fernandez de Vega, and Carlos Cotta. Automatic evolution of programs for procedural generation of terrains for video games. *Soft Computing*, 16(11):1893–1914, 2012.

[5] Roberto Gallotta, Kai Arulkumaran, and LB Soros. Evolving spaceships with a hybrid l-system constrained optimisation evolutionary algorithm. In *GECCO '22: Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 711–714, 2022.

[6] Lucas Nascimento Ferreira and Claudio Fabiano Motta Toledo. Tanager: A generator of feasible and engaging levels for angry birds. *IEEE Transactions on Games*, 10(3):304–316, 2017.

[7] Luigi Cardamone, Pier Luca Lanzi, and Daniele Loiacono. Trackgen: An interactive track generator for torcs and speed-dreams. *Applied Soft Computing*, 28:550–558, 2015.

[8] Valtchan Valtchanov and Joseph Alexander Brown. Evolving dungeon crawler levels with relative placement. In *Fifth International C* Conference on Computer Science and Software Engineering*, pages 27–35, 2012.

[9] Jakub Kowalski and Marek Szykuła. Procedural content generation for gdl descriptions of simplified boardgames. *arXiv:1508.00212*, 2015.

[10] Mike McShaffry. *Game coding complete*. Cengage Learning, 2009.

[11] Luca Pascarella, Fabio Palomba, Massimiliano Di Penta, and Alberto Bacchelli. How is video game development different from software development in open source? In *Proceedings of the 15th International Conference on Mining Software Repositories*, pages 392–402, 2018.

[12] Emerson Murphy-Hill, Thomas Zimmermann, and Nachiappan Nagappan. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In *36th International Conference on Software Engineering*, ICSE 2014, page 1–11, New York, NY, USA, 2014. Association for Computing Machinery.

[13] Apostolos Ampatzoglou and Ioannis Stamelos. Software engineering research for computer games: A systematic review. *Information and Software Technology*, 52(9):888–901, 2010.

[14] `http://www.gamesoftwareengineering.com`.

[15] Benjamin Engelstätter and Michael R Ward. Video games become more mainstream. *Entertainment Computing*, 42:100494, 2022.

[16] Colin Charles Mathews and Nia Wearn. How are modern video games marketed? *The computer Games Journal*, 5(1-2):23–37, 2016.

[17] Mohammad Shaker, Mhd Hasan Sarhan, Ola Al Naameh, Noor Shaker, and Julian Togelius. Automatic generation and analysis of physics-based puzzle games. In *Conference on Computational Inteligence in Games*, pages 1–8. IEEE, 2013.

[18] Melina Farshbafnadi, Sepideh Razi, and Nima Rezaei. Chapter 7 - transplantation. In Nima Rezaei, editor, *Clinical Immunology*, pages 599–674. Academic Press, 2023.

[19] Earl T Barr, Mark Harman, Yue Jia, Alexandru Marginean, and Justyna Petke. Automated software transplantation. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, pages 257–269, 2015.

[20] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically finding patches using genetic programming. In *2009 IEEE 31st International Conference on Software Engineering*, pages 364–374. IEEE, 2009.

[21] Stelios Sidiroglou-Douskos, Eric Lahtinen, and Martin Rinard. Automatic error elimination by multi-application code transfer. 2014.

[22] Tianyi Zhang and Miryung Kim. Automated transplantation and differential testing for clones. In *2017 IEEE/ACM 39th International Conference on Software Engineering*, pages 665–676. IEEE, 2017.

[23] Wei Yang, Deguang Kong, Tao Xie, and Carl A Gunter. Malware detection in adversarial settings: Exploiting feature evolutions and confusions in android apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 288–302, 2017.

[24] Stelios Sidiroglou-Douskos, Eric Lahtinen, Anthony Eden, Fan Long, and Martin Rinard. Codecarboncopy. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 95–105, 2017.

[25] Mar Zamorano, Carlos Cetina, and Federica Sarro. The quest for content: A survey of search-based procedural content generation for video games. *arXiv preprint arXiv:2311.04710*, 2023.

[26] Mar Zamorano, África Domingo, Carlos Cetina, and Federica Sarro. Game software engineering: A controlled experiment comparing automated content generation techniques. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 302–313, 2024.

[27] María Del Mar Zamorano López, Daniel Blasco, Carlos Cetina, and Federica Sarro. Video game procedural content generation through software transplantation. In *International Conference on Software Engineering: Software Engineering in Practice*. IEEE/ACM, 2025.

[28] Dominik Sobania, Alina Geiger, James Callan, Alexander Brownlee, Carol Hanna, Rebecca Moussa, Mar Zamorano López, Justyna Petke, and Federica Sarro. Evaluating explanations for software patches generated by large language models. In *International Symposium on Search Based Software Engineering*, pages 147–152. Springer, 2023.

[29] Guillermo Iglesias, Mar Zamorano, and Federica Sarro. Search-based negative prompt optimisation for text-to-image generation. In *14th International Conference on Artificial Intelligence in Music, Sound, Art and Design*, 2025.

[30] Carlos Mora, Sandra Jardim, and Jorge Valente. Flora generation and evolution algorithm for virtual environments. In *16th Iberian Conference on Information Systems and Technologies*, pages 1–6. IEEE, 2021.

[31] David Plans and Davide Morelli. Experience-driven procedural music generation for games. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3):192–198, 2012.

[32] Miguel Frade, Francisco Fernandéz de Vega, and Carlos Cotta. Breeding terrains with genetic terrain programming: the evolution of terrain generators. *International Journal of Computer Games Technology*, 2009, 2009.

[33] Breno MF Viana, Leonardo T Pereira, and Claudio FM Toledo. Illuminating the space of enemies through map-elites. In *IEEE Conference on Games*, pages 17–24. IEEE, 2022.

[34] Breno MF Viana and Selan R dos Santos. A survey of procedural dungeon generation. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment*, pages 29–38. IEEE, 2019.

[35] Edirlei Soares de Lima, Bruno Feijó, and Antonio L Furtado. Procedural generation of quests for games using genetic algorithms and automated planning. In *Brazilian Symposium on Computer Games and Digital Entertainment*, pages 144–153, 2019.

[36] Cameron Bolitho Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.

[37] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*, volume 2. Springer, 2018.

[38] Nicolas A. Barriga. A Short Introduction to Procedural Content Generation Algorithms for Videogames. *International Journal on Artificial Intelligence Tools*, 28(2):1–11, 2019.

[39] Jonas Freiknecht and Wolfgang Effelsberg. A survey on the procedural generation of virtual worlds. *Multimodal Technologies and Interaction*, 1(4):27, 2017.

[40] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgard, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games*, 10(3):257–270, 2018.

[41] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis, and Julian Togelius. Deep learning for procedural content generation. *Neural Computing and Applications*, 33(1):19–37, 2021.

[42] Konstantinos Souchleris, George K Sidiropoulos, and George A Papakostas. Reinforcement learning in game industry—review, prospects and challenges. *Applied Sciences*, 13(4):2443, 2023.

[43] Laura Anna Ripamonti, Federico Distefano, Marco Trubian, Dario Maggiorini, and Davide Gadia. Dragon: diversity regulated adaptive generator online. *Multimedia Tools and Applications*, 80(26):34933–34969, 2021.

[44] Leonardo T Pereira, Breno MF Viana, and Claudio FM Toledo. Procedural enemy generation through parallel evolutionary algorithm. In *20th Brazilian Symposium on Computer Games and Digital Entertainment*, pages 126–135. IEEE, 2021.

[45] Daniel Blasco, Jaime Font, Mar Zamorano, and Carlos Cetina. An evolutionary approach for generating software models: The case of kromaia in game software engineering. *Journal of Systems and Software*, 171:110804, 2021.

[46] Aristid Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 1968.

[47] Michele Brocchini, Marco Mameli, Emanuele Balloni, Laura Della Sciucca, Luca Rossi, Marina Paolanti, Emanuele Frontoni, and Primo Zingaretti. Monster: A deep learning-based system for the automatic generation of gaming assets. In *International Conference on Image Analysis and Processing*, pages 280–290. Springer, 2022.

[48] Ruizhe Li, Masanori Nakayama, and Issei Fujishiro. Automatic generation of 3d natural anime-like non-player characters with machine learning. In *2020 International Conference on Cyberworlds*, pages 110–116. IEEE, 2020.

[49] Mark Harman and Bryan F Jones. Search-based software engineering. *Information and software Technology*, 43(14):833–839, 2001.

[50] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, 45(1):1–61, 2012.

[51] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 2022.

[52] Jan Kruse, Andy M Connor, and Stefan Marks. Evaluation of a multi-agent "human-in-the-loop" game design system. *ACM Transactions on Interactive Intelligent Systems*, 12(3):1–26, 2022.

[53] Craig Miles, Arun Lakhotia, and Andrew Walenstein. In situ reuse of logically extracted functional components. *Journal in Computer Virology*, 8:73–84, 2012.

[54] Justyna Petke, Mark Harman, William B Langdon, and Westley Weimer. Using genetic improvement and code transplants to specialise a c++ program to a problem class. In *European Conference on Genetic Programming*, pages 137–149. Springer, 2014.

[55] Alexandru Marginean. *Automated Software Transplantation*. PhD thesis, UCL (University College London), 2021.

[56] Yonghwi Kwon, Weihang Wang, Yunhui Zheng, Xiangyu Zhang, and Dongyan Xu. Cpr: cross platform binary code reuse via platform independent trace program. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 158–169, 2017.

[57] Stelios Sidiroglou-Douskos, Eli Davis, and Martin Rinard. Horizontal code transfer via program fracture and recombination. 2015.

[58] Josip Maras, Maja Štula, and Ivica Crnković. Towards specifying pragmatic software reuse. In *Proceedings of the 2015 European Conference on Software Architecture Workshops*, pages 1–4, 2015.

[59] Yuepeng Wang, Yu Feng, Ruben Martins, Arati Kaushik, Isil Dillig, and Steven P Reiss. Hunter: next-generation code reuse for java. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 1028–1032, 2016.

[60] Miltiadis Allamanis and Marc Brockschmidt. Smartpaste: Learning to adapt source code. *arXiv preprint arXiv:1705.07867*, 2017.

[61] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[62] Yanxin Lu, Swarat Chaudhuri, Chris Jermaine, and David Melski. Program splicing. In *Proceedings of the 40th International Conference on Software Engineering*, pages 338–349, 2018.

[63] Yonghwi Kwon, Xiangyu Zhang, and Dongyan Xu. Pietrace: Platform independent executable trace. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering*, pages 48–58. IEEE, 2013.

[64] Jorge Chueca, Javier Verón, Jaime Font, Francisca Pérez, and Carlos Cetina. The consolidation of game software engineering: A systematic literature review of software engineering for industry-scale computer games. *Information and Software Technology*, page 107330, 2023.

[65] CryEngine. Cryengine. `https://www.cryengine.com`. Accessed: 01/02/24.

[66] Unreal Blueprint. Unreal blueprint. `https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/`. Accessed: 01/02/24.

[67] Unity Scripting. Unity scripting. `https://unity.com/features/unity-visual-scripting`. Accessed: 01/02/24.

[68] Meng Zhu and Alf Inge Wang. Model-driven game development: A literature review. *ACM Computing Surveys*, 52(6):1–32, 2019.

[69] Emanuel Montero Reyno and José Á Carsí Cubel. Automatic prototyping in model-driven game development. *Computers in Entertainment*, 7(2):1–9, 2009.

[70] Edward Rolando Núñez-Valdez, Vicente García-Díaz, Juan Manuel Cueva Lovelle, Yago Sáez Achaerandio, and Rubén González-Crespo. A model-driven approach to generate and deploy videogames on multiple platforms. *Journal of Ambient Intelligence and Humanized Computing*, 8(3):435–447, 2017.

[71] Edward Rolando Nuñez-Valdez, Oscar Sanjuan, Begoña Cristina Pelayo Garcia-Bustelo, Juan Manuel Cueva-Lovelle, and Guillermo Infante Hernandez. Gade4all: developing multi-platform videogames based on domain specific languages and model driven engineering. *International Journal of Interactive Multimedia and Artificial Intelligence*, 2013.

[72] Muhammad Usman, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan. A product-line model-driven engineering approach for generating feature-based mobile applications. *Journal of Systems and Software*, 123:1–32, 2017.

[73] Jaime Solís-Martínez, Jordán Pascual Espada, Natalia García-Menéndez, B. Cristina Pelayo García-Bustelo, and Juan Manuel Cueva Lovelle. VGPM: using business process modeling for videogame modeling and code generation in multiple platforms. *Computer Standards & Interfaces*, 42:42–52, 2015.

[74] James R Williams, Simon Poulding, Louis M Rose, Richard F Paige, and Fiona AC Polack. Identifying desirable game character behaviours through

the application of evolutionary algorithms to model-driven engineering metamodels. In *International Symposium on Search Based Software Engineering*, pages 112–126. Springer, 2011.

[75] Heather J Goldsby and Betty HC Cheng. Automatically generating behavioral models of adaptive systems to address uncertainty. In *International Conference on Model Driven Engineering Languages and Systems*, pages 568–583. Springer, 2008.

[76] SlashData. State of the developer nation 23rd edition. [Online; accessed 18-December-2023].

[77] África Domingo, Jorge Echeverría, Oscar Pastor, and Carlos Cetina. Evaluating the benefits of model-driven development: empirical evaluation paper. In *International Conference on Advanced Information Systems Engineering*, pages 353–367. Springer, 2020.

[78] Speed Tree. Speed tree. `https://store.speedtree.com`. Accessed: 01/02/24.

[79] Erika Puiutta and Eric MSP Veith. Explainable reinforcement learning: A survey. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 77–95. Springer, 2020.

[80] Oliver Withington. Illuminating super mario bros: quality-diversity within platformer level generation. In *GECCO '20: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 223–224, 2020.

[81] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering*, 43(9):817–847, 2016.

[82] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *18th International Conference on*

Evaluation and Assessment in Software Engineering, EASE '14, pages 1–10. ACM, 2014.

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

[84] Karl Sims. Artificial evolution for computer graphics. In *18th Conference on Computer Graphics and Interactive techniques*, pages 319–328, 1991.

[85] Jakub Kowalski, Antonios Liapis, and Łukasz Żarczyński. Mapping chess aesthetics onto procedurally generated chess-like games. In *International Conference on the Applications of Evolutionary Computation*, pages 325–341. Springer, 2018.

[86] Antonios Liapis. Recomposing the pokémon color palette. In *International Conference on the Applications of Evolutionary Computation*, pages 308–324. Springer, 2018.

[87] Alessio Bernardi, Davide Gadia, Dario Maggiorini, Claudio Enrico Palazzi, and Laura Anna Ripamonti. Procedural generation of materials for real-time rendering. *Multimedia Tools and Applications*, pages 1–22, 2020.

[88] Joseph Alexander Brown and Marco Scirea. Evolving woodland camouflage. *IEEE Transactions on Games*, 2022.

[89] Hanna Järveläinen. Algorithmic musical composition. In *Seminar on Content Creation Art@ Science*. University of Technology, Laboratory of Acoustics and Audio Signal . . . , 2000.

[90] Karen Collins. An introduction to procedural music in video games. *Contemporary Music Review*, 28(1):5–15, 2009.

[91] Erin J Hastings, Ratan K Guha, and Kenneth O Stanley. Evolving content in the galactic arms race video game. In *IEEE Symposium on Computational Intelligence and Games*, pages 241–248. IEEE, 2009.

[92] Erin Jonathan Hastings, Ratan K Guha, and Kenneth O Stanley. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245–263, 2009.

[93] Erin J Hastings and Kenneth O Stanley. Interactive genetic engineering of evolved video game content. In *Workshop on Procedural Content Generation in Games*, pages 1–4, 2010.

[94] Eric McDuffee and Alex Pantaleev. Team blockhead wars: Generating fps weapons in a multiplayer environment. In *Second Workshop on Procedural Content Generation in Games*, 2013.

[95] Daniele Gravina and Daniele Loiacono. Procedural weapons generation for unreal tournament iii. In *Games Entertainment Media Conference*, pages 1–8. IEEE, 2015.

[96] Marcus Toftedahl and Henrik Engström. A taxonomy of game engines and the tools that drive the industry. In *Proceedings of DiGRA 2019 Conference: Game, Play and the Emerging Ludo-Mix*, 2019.

[97] Sebastian Risi, Joel Lehman, David B D'Ambrosio, Ryan Hall, and Kenneth O Stanley. Petalz: Search-based procedural content generation for the casual gamer. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):244–255, 2015.

[98] Miguel Frade, Francisco Fernandez de Vega, and Carlos Cotta. Evolution of artificial terrains for video games based on accessibility. In *European Conference on the Applications of Evolutionary Computation*, pages 90–99. Springer, 2010.

[99] Miguel Frade, F Fernandez de Vega, and Carlos Cotta. Evolution of artificial terrains for video games based on obstacles edge length. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[100] Miguel Frade, F Fernandez de Vega, and Carlos Cotta. Aesthetic terrain programs database for creativity assessment. In *Conference on Computational Intelligence and Games*, pages 350–354. IEEE, 2012.

[101] Andrew Pech, Chiou-Peng Lam, Philip Hingston, and Martin Masek. Using isovists to evolve terrains with gameplay elements. In *European Conference on the Applications of Evolutionary Computation*, pages 636–652. Springer, 2016.

[102] Luigi Cardamone, Georgios N Yannakakis, Julian Togelius, and Pier Luca Lanzi. Evolving interesting maps for a first person shooter. In *European Conference on the Applications of Evolutionary Computation*, pages 63–72. Springer, 2011.

[103] Nicholas Cole, Sushil J Louis, and Chris Miles. Using a genetic algorithm to tune first-person shooter bots. In *2004 Congress on Evolutionary Computation*, volume 1, pages 139–145. IEEE, 2004.

[104] Bernard Gorman, Christian Thurau, Christian Bauckhage, and Mark Humphrys. Believability testing and bayesian imitation in interactive computer games. In *International Conference on Simulation of Adaptive Behavior*, pages 655–666. Springer, 2006.

[105] Niels Van Hoorn, Julian Togelius, and Jurgen Schmidhuber. Hierarchical controller learning in a first-person shooter. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 294–301. IEEE, 2009.

[106] Kenneth Hullett and Jim Whitehead. Design patterns in fps levels. In *Fifth International Conference on the Foundations of Digital Games*, pages 78–85, 2010.

[107] Pier Luca Lanzi, Daniele Loiacono, and Riccardo Stucchi. Evolving maps for match balancing in first person shooters. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

[108] Peter Thorup Ølsted, Benjamin Ma, and Sebastian Risi. Interactive evolution of levels for a competitive multiplayer fps. In *IEEE Congress on Evolutionary Computation*, pages 1527–1534. IEEE, 2015.

[109] Daniele Loiacono and Luca Arnaboldi. Fight or flight: Evolving maps for cube 2 to foster a fleeing behavior. In *2017 IEEE Conference on Computational Intelligence and Games*, pages 199–206. IEEE, 2017.

[110] Daniele Loiacono and Luca Arnaboldi. Multiobjective evolutionary map design for cube 2: Sauerbraten. *IEEE Transactions on Games*, 11(1):36–47, 2018.

[111] Julian Togelius, Mike Preuss, and Georgios N Yannakakis. Towards multiobjective procedural map generation. In *Workshop on Procedural Content Generation in Games*, pages 1–8, 2010.

[112] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, and Georgios N Yannakakis. Multiobjective exploration of the starcraft map space. In *IEEE Conference on Computational Intelligence and Games*, pages 265–272. IEEE, 2010.

[113] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, Georgios N Yannakakis, and Corrado Grappiolo. Controllable procedural map generation via multiobjective evolution. *Genetic Programming and Evolvable Machines*, 14(2):245–277, 2013.

[114] Raúl Lara-Cabrera, Carlos Cotta, and Antonio J Fernández-Leiva. Procedural map generation for a rts game. In *13th International GAME-ON Conference on Intelligent Games and Simulation*, pages 53–58. Malaga (Spain): Eurosis, 2012.

[115] Raúl Lara-Cabrera, Carlos Cotta, and Antonio J Fernández-Leiva. A procedural balanced map generator with self-adaptive complexity for the real-time strategy game planet wars. In *European Conference on the Applications of Evolutionary Computation*, pages 274–283. Springer, 2013.

[116] Raúl Lara-Cabrera, Carlos Cotta, and Antonio J Fernández-Leiva. Using self-adaptive evolutionary algorithms to evolve dynamism-oriented maps for a real time strategy game. In *International Conference on Large-Scale Scientific Computing*, pages 256–263. Springer, 2013.

[117] Raúl Lara-Cabrera, Carlos Cotta, and Antonio J Fernández-Leiva. On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms. *Natural Computing*, 13(2):157–168, 2014.

[118] Raúl Lara-Cabrera, Carlos Cotta, Antonio José Fernández-Leiva, et al. Evolving aesthetic maps for a real time strategy game. 2013.

[119] Raúl Lara-Cabrera, Carlos Cotta, and Antonio J Fernändez-Leiva. A self-adaptive evolutionary approach to the evolution of aesthetic maps for a rts game. In *Congress on Evolutionary Computation*, pages 298–304. IEEE, 2014.

[120] Raúl Lara-Cabrera, Carlos Cotta, and AJ Fernández-Leiva. Geometrical vs topological measures for the evolution of aesthetic maps in a rts game. *Entertainment Computing*, 5(4):251–258, 2014.

[121] Lawrence Johnson, Georgios N Yannakakis, and Julian Togelius. Cellular automata for real-time generation of infinite cave levels. In *Workshop on Procedural Content Generation in Games*, pages 1–4, 2010.

[122] Tobias Mahlmann, Julian Togelius, and Georgios N Yannakakis. Spicing up map generation. In *European Conference on the Applications of Evolutionary Computation*, pages 224–233. Springer, 2012.

[123] Raúl Lara-Cabrera, Víctor Rodríguez-Fernández, Javier Paz-Sedano, and David Camacho. Procedural generation of balanced levels for a 3d paintball game. In *Congreso de la Sociedad Española para las Ciencias del Videojuego*, pages 43–55, 2017.

[124] Gabriella AB Barros and Julian Togelius. Balanced civilization map generation based on open data. In *IEEE Congress on Evolutionary Computation*, pages 1482–1489. IEEE, 2015.

[125] Jakub Kowalski, Radosław Miernik, Piotr Pytlik, Maciej Pawlikowski, Krzysztof Piecuch, and Jakub Sekowski. Strategic features and terrain generation for balanced heroes of might and magic iii maps. In *Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2018.

[126] Artur OR Franco, Wellington Franco, José GR Maia, and Miguel Franklin. Generating rooms using generative grammars and genetic algorithms. In *21st Brazilian Symposium on Computer Games and Digital Entertainment*, pages 1–6. IEEE, 2022.

[127] Lianbo Ma, Shi Cheng, Mingli Shi, and Yinan Guo. Angle-based multi-objective evolutionary algorithm based on pruning-power indicator for game map generation. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):341–354, 2021.

[128] Andrea Guarneri, Dario Maggiorini, Laura Ripamonti, and Marco Trubian. Golem: generator of life embedded into mmos. In *12th European Conference on Artificial Life*, pages 585–592. MIT press, 2013.

[129] Daniele Norton, Laura Anna Ripamonti, Mario Ornaghi, Davide Gadia, and Dario Maggiorini. Monsters of darwin: A strategic game based on artificial intelligence and genetic algorithms. In *CEUR Workshop Proceedings*, volume 1956. CEUR-WS, 2017.

[130] Daniel Blasco, Jaime Font, Francisca Pérez, and Carlos Cetina. Procedural content improvement of game bosses with an evolutionary algorithm. *Multimedia Tools and Applications*, 82(7):10277–10309, 2023.

[131] André Siqueira Ruela and Frederico Gadelha Guimaraes. Evolving battle formations in massively multiplayer online strategy games. In *Brazilian Symposium on Games and Digital Entertainment*, pages 49–55, 2012.

[132] André Siqueira Ruela and Frederico Gadelha Guimaraes. Coevolutionary procedural generation of battle formations in massively multiplayer online strategy games. In *2014 Brazilian Symposium on Computer Games and Digital Entertainment*, pages 89–98. IEEE, 2014.

[133] André Siqueira Ruela and Frederico Gadelha Guimarães. Procedural generation of non-player characters in massively multiplayer online strategy games. *Soft Computing*, 21(23):7005–7020, 2017.

[134] Joseph Alexander Brown, Daniel Ashlock, Sheridan Houghten, and Angelo Romualdo. Evolutionary graph compression and diffusion methods for city discovery in role playing games. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2020.

[135] Daniel Ashlock. Automatic generation of game elements via evolution. In *IEEE Conference on Computational Intelligence and Games*, pages 289–296. IEEE, 2010.

[136] Daniel Ashlock, Colin Lee, and Cameron McGuinness. Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260–273, 2011.

[137] Daniel Ashlock, Colin Lee, and Cameron McGuinness. Simultaneous dual level creation for games. *IEEE Computational Intelligence Magazine*, 6(2):26–37, 2011.

[138] Cameron McGuinness and Daniel Ashlock. Decomposing the level generation problem with tiles. In *Congress of Evolutionary Computation*, pages 849–856. IEEE, 2011.

[139] Cameron McGuinness and Daniel Ashlock. Incorporating required structure into tiles. In *Conference on Computational Intelligence and Games*, pages 16–23. IEEE, 2011.

[140] Cameron McGuinness. Statistical analyses of representation choice in level generation. In *Conference on Computational Intelligence and Games*, pages 312–319. IEEE, 2012.

[141] Cameron McGuinness. Multiple pass monte carlo tree search. In *Congress on Evolutionary Computation*, pages 1555–1561. IEEE, 2016.

[142] Paul Hyunjin Kim and Roger Crawfis. The quest for the perfect perfect-maze. In *2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games*, pages 65–72. IEEE, 2015.

[143] Paul Hyunjin Kim and Roger Crawfis. Intelligent maze generation based on topological constraints. In *7th International Congress on Advanced Applied Informatics*, pages 867–872. IEEE, 2018.

[144] Andrew Pech, Philip Hingston, Martin Masek, and Chiou Peng Lam. Evolving cellular automata for maze generation. In *Australasian Conference on Artificial Life and Computational Intelligence*, pages 112–124. Springer, 2015.

[145] Andrew Pech, Martin Masek, Chiou-Peng Lam, and Philip Hingston. Game level layout generation using evolved cellular automata. *Connection Science*, 28(1):63–82, 2016.

[146] Lucas Ferreira and Claudio Toledo. Generating levels for physics-based puzzle games with estimation of distribution algorithms. In *11th Conference on Advances in Computer Entertainment Technology*, pages 1–6, 2014.

[147] Lucas Ferreira and Claudio Toledo. A search-based approach for generating angry birds levels. In *Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

[148] Misaki Kaidan, Chun Yin Chu, Tomohiro Harada, and Ruck Thawonmas. Procedural generation of angry birds levels that adapt to the player's skills using genetic algorithm. In *4th Global Conference on Consumer Electronics*, pages 535–536. IEEE, 2015.

[149] Laura Calle, Juan J Merelo, Antonio Mora-García, and José-Mario García-Valdez. Free form evolution for angry birds level generation. In *International Conference on the Applications of Evolutionary Computation*, pages 125–140. Springer, 2019.

[150] Laura Calle, Juan Julián Merelo Guervós, Mario García Valdez, and Antonio Mora García. Speeding up evaluation of structures for the angry birds game. In *11th International Joint Conference on Computational Intelligence*, pages 237–244, 2019.

[151] Laura Calle, Juan-Julián Merelo-Guervós, Antonio Mora-García, and Mario García Valdez. Improved free form evolution for angry birds structures. In *GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 133–134, 2019.

[152] Raúl Lara-Cabrera, Alejandro Gutierrez-Alcoba, and Antonio J Fernández-Leiva. A spatially-structured pcg method for content diversity in a physics-based simulation game. In *European Conference on the Applications of Evolutionary Computation*, pages 653–668. Springer, 2016.

[153] Carlos López-Rodríguez, Antonio J Fernández-Leiva, Raúl Lara-Cabrera, Antonio M Mora, and Pablo García-Sánchez. Checking the difficulty of evolutionary-generated maps in a n-body inspired mobile game. In *International Conference on Optimization and Learning*, pages 206–215. Springer, 2020.

[154] Julian Togelius, Renzo De Nardi, and Simon M Lucas. Making racing fun through player modeling and track evolution. 2006.

[155] Julian Togelius, Renzo De Nardi, and Simon M Lucas. Towards automatic personalised content creation for racing games. In *IEEE Symposium on Computational Intelligence and Games*, pages 252–259. IEEE, 2007.

[156] Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. Automatic track generation for high-end racing games using evolutionary computation. *IEEE*

Transactions on Computational Intelligence and AI in Games, 3(3):245–259, 2011.

[157] Hafizh Adi Prasetya and Nur Ulfa Maulidevi. Search-based procedural content generation for race tracks in video games. *International Journal on Electrical Engineering & Informatics*, 8(4), 2016.

[158] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *13th Conference on Genetic and Evolutionary Computation*, pages 395–402, 2011.

[159] Vid Kraner, Iztok Fister Jr, and Lucija Brezočnik. Procedural content generation of custom tower defense game using genetic algorithms. In *International Conference "New Technologies, Development and Applications"*, pages 493–503. Springer, 2021.

[160] Zixuan Deng and Yanping Xiang. Roads to what we want: A game generator based on reverse design. In *7th International Symposium on System and Software Reliability*, pages 198–205. IEEE, 2021.

[161] Michael Cerny Green, Ahmed Khalifa, and Julian Togelius. Persona-driven dominant/submissive map (pdsm) generation for tutorials. In *17th International Conference on the Foundations of Digital Games*, pages 1–10, 2022.

[162] Julian Togelius, Tróndur Justinussen, and Anders Hartzen. Compositional procedural content generation. In *The Third Workshop on Procedural Content Generation in Games*, pages 1–4, 2012.

[163] Adam M Smith and Michael Mateas. Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In *2010 IEEE Conference on Computational Intelligence and Games*, pages 273–280. IEEE, 2010.

[164] Tom Schaul. A video game description language for model-based or interactive learning. In *Conference on Computational Intelligence in Games*, pages 1–8. IEEE, 2013.

[165] Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon M Lucas, Adrien Couëtoux, Jerry Lee, Chong-U Lim, and Tommy Thompson. The 2014 general video game playing competition. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):229–243, 2015.

[166] Xenija Neufeld, Sanaz Mostaghim, and Diego Perez-Liebana. Procedural level generation with answer set programming for general video game playing. In *2015 7th Computer Science and Electronic Engineering Conference*, pages 207–212. IEEE, 2015.

[167] Olve Drageset, Mark HM Winands, Raluca D Gaina, and Diego Perez-Liebana. Optimising level generators for general video game ai. In *IEEE Conference on Games*, pages 1–8. IEEE, 2019.

[168] Adeel Zafar, Hasan Mujtaba, and Mirza Omer Beg. Search-based procedural content generation for gvg-lg. *Applied Soft Computing*, 86:105909, 2020.

[169] Aurimas Petrovas and Romualdas Bausys. Procedural video game scene generation by genetic and neutrosophic waspas algorithms. *Applied Sciences*, 12(2):772, 2022.

[170] Aurimas Petrovas, Romualdas Baušys, Edmundas Kazimieras Zavadskas, and Florentin Smarandache. Generation of creative game scene patterns by the neutrosophic genetic cocoso method. *Studies in Informatics and Control*, 31(4):5–11, 2022.

[171] Adeel Zafar, Hasan Mujtaba, Mirza Tauseef Baig, and Mirza Omer Beg. Using patterns as objectives for general video game level generation. *ICGA Journal*, 41(2):66–77, 2019.

[172] Sean Walton, Alma Rahat, and James Stovold. Evaluating mixed-initiative procedural level design tools using a triple-blind mixed-method user study. *IEEE Transactions on Games*, 2021.

[173] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv:1504.04909*, 2015.

[174] Megan Charity, Michael Cerny Green, Ahmed Khalifa, and Julian Togelius. Mech-elites: Illuminating the mechanic space of gvg-ai. In *International Conference on the Foundations of Digital Games*, pages 1–10, 2020.

[175] Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. Interactive constrained map-elites: Analysis and evaluation of the expressiveness of the feature dimensions. *IEEE Transactions on Games*, 14(2):202–211, 2020.

[176] Megan Charity, Ahmed Khalifa, and Julian Togelius. Baba is y'all: Collaborative mixed-initiative level design. In *IEEE Conference on Games*, pages 542–549. IEEE, 2020.

[177] Debosmita Bhaumik, Ahmed Khalifa, Michael Green, and Julian Togelius. Tree search versus optimization approaches for map generation. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 24–30, 2020.

[178] Raphael Bailly and Guillaume Levieux. Genetic-wfc: Extending wave function collapse with genetic search. *IEEE Transactions on Games*, 15(1):36–45, 2023.

[179] Leonardo T Pereira, Paulo VS Prado, and Claudio Toledo. Evolving dungeon maps with locked door missions. In *Congress on Evolutionary Computation*, pages 1–8. IEEE, 2018.

[180] Leonardo Tortoro Pereira, Paulo Victor de Souza Prado, Rafael Miranda Lopes, and Claudio Fabiano Motta Toledo. Procedural generation of dungeons' maps

and locked-door missions through an evolutionary algorithm validated with players. *Expert Systems with Applications*, 180:115009, 2021.

[181] Jose M Font, Roberto Izquierdo, Daniel Manrique, and Julian Togelius. Constrained level generation through grammar-based evolutionary algorithms. In *European Conference on the Applications of Evolutionary Computation*, pages 558–573. Springer, 2016.

[182] André Siqueira Ruela and Karina Valdivia Delgado. Scale-free evolutionary level generation. In *Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2018.

[183] André Siqueira Ruela and Karina Valdivia Delgado. Evolving lock-and-key puzzles based on nonlinear player progression and level exploration. *Brazilian Symposium on Computer Games and Digital Entertainment*, pages 651–654, 2018.

[184] André Siqueira Ruela, Karina Valdivia Delgado, and João Bernardes. A multi-objective evolutionary approach for the nonlinear scale-free level problem. *Applied Intelligence*, 50(12):4223–4240, 2020.

[185] Alexandre Santos Melotti and Carlos Henrique Valerio de Moraes. Evolving roguelike dungeons with deluged novelty search local competition. *IEEE Transactions on Games*, 11(2):173–182, 2018.

[186] Antonios Liapis. Multi-segment evolution of dungeon game levels. In *GECCO '17: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 203–210, 2017.

[187] Breno MF Viana, Leonardo T Pereira, Claudio FM Toledo, Selan R dos Santos, and Silvia MDM Maia. Feasible–infeasible two-population genetic algorithm to evolve dungeon levels with dependencies in barrier mechanics. *Applied Soft Computing*, 119:108586, 2022.

[188] Joseph Alexander Brown, Bulat Lutfullin, and Pavel Oreshin. Procedural content generation of level layouts for hotline miami. In *2017 9th Computer Science and Electronic Engineering*, pages 106–111. IEEE, 2017.

[189] Joseph Alexander Brown, Bulat Lutfullin, Pavel Oreshin, and Ilya Pyatkin. Levels for hotline miami 2: Wrong number using procedural content generations. *Computers*, 7(2):22, 2018.

[190] Ardiawan Bagus Harisa and Wen-Kai Tai. Pacing-based procedural dungeon level generation: Alternating level creation to meet designer's expectations. *International Journal of Computing and Digital Systems*, 12(1):401–416, 2022.

[191] Mohammad Shaker, Noor Shaker, Julian Togelius, and Mohamed Abou-Zleikha. A progressive approach to content generation. In *European Conference on the Applications of Evolutionary Computation*, pages 381–393. Springer, 2015.

[192] Diaz-Furlong Hector Adrian and Solis-Gonzalez Cosio Ana Luisa. An approach to level design using procedural content generation and difficulty curves. In *Conference on Computational Inteligence in Games*, pages 1–8. IEEE, 2013.

[193] Arman Balali Moghadam and Marjan Kuchaki Rafsanjani. A genetic approach in procedural content generation for platformer games level creation. In *2017 2nd Conference on Swarm Intelligence and Evolutionary Computation*, pages 141–146. IEEE, 2017.

[194] Ali Sofyan Kholimi, Ahmad Hamdani, and Lailatul Husniah. Automatic game world generation for platformer games using genetic algorithm. In *5th International Conference on Electrical Engineering, Computer Science and Informatics*, pages 495–498. IEEE, 2018.

[195] Pratama Wirya Atmaja, Sugiarto, and Eka Prakarsa Mandyartha. Difficulty curve-based procedural generation of scrolling shooter enemy formations.

In *Journal of Physics: Conference Series*, volume 1569, page 022049. IOP Publishing, 2020.

[196] Fausto Mourato, Manuel Próspero dos Santos, and Fernando Birra. Automatic level generation for platform videogames using genetic algorithms. In *8th International Conference on Advances in Computer Entertainment Technology*, pages 1–8, 2011.

[197] Steve Dahlskog and Julian Togelius. Patterns as objectives for level generation. In *Design Patterns in Games*. ACM Digital Library, 2013.

[198] Steve Dahlskog and Julian Togelius. Procedural content generation using patterns as objectives. In *European Conference on the Applications of Evolutionary Computation*, pages 325–336. Springer, 2014.

[199] Michael Cerny Green, Luvneesh Mugrai, Ahmed Khalifa, and Julian Togelius. Mario level generation from mechanics using scene stitching. In *IEEE Conference on Games*, pages 49–56. IEEE, 2020.

[200] Arash Moradi Karkaj and Shahriar Lotfi. Using estimation of distribution algorithm for procedural content generation in video games. *Genetic Programming and Evolvable Machines*, 23(4):495–533, 2022.

[201] Michael Cerny Green, Ahmed Khalifa, Gabriella AB Barros, Andy Nealen, and Julian Togelius. Generating levels that teach mechanics. In *13th International Conference on the Foundations of Digital Games*, pages 1–8, 2018.

[202] Ahmed Khalifa, Michael Cerny Green, Gabriella Barros, and Julian Togelius. Intentional computational level design. In *GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 796–803, 2019.

[203] Vivek R Warriar, Carmen Ugarte, John R Woodward, and Laurissa Tokarchuk. Playmapper: Illuminating design spaces of platform games. In *IEEE Conference on Games*, pages 1–4. IEEE, 2019.

[204] Davy Smith, Laurissa Tokarchuk, and Geraint Wiggins. Rapid phenotypic landscape exploration through hierarchical spatial partitioning. In *International Conference on Parallel Problem Solving from Nature*, pages 911–920. Springer, 2016.

[205] Stacey Mason, Ceri Stagg, and Noah Wardrip-Fruin. Lume: a system for procedural story generation. In *14th International Conference on the Foundations of Digital Games*, pages 1–9, 2019.

[206] Erik M Fredericks and Byron DeVries. (genetically) improving novelty in procedural story generation. In *IEEE/ACM International Workshop on Genetic Improvement*, pages 39–40. IEEE, 2021.

[207] Alberto Alvarez, Eric Grevillius, Elin Olsson, and Jose Font. Questgram [qg]: Toward a mixed-initiative quest generation tool. In *16th International Conference on the Foundations of Digital Games*, pages 1–10, 2021.

[208] Edirlei Soares de Lima, Bruno Feijó, and Antonio L Furtado. Procedural generation of branching quests for games. *Entertainment Computing*, 43:100491, 2022.

[209] Alberto Alvarez and Jose Font. Tropetwist: Trope-based narrative structure generation. In *17th International Conference on the Foundations of Digital Games*, pages 1–8, 2022.

[210] Alberto Alvarez, Jose Font, and Julian Togelius. Story designer: Towards a mixed-initiative tool to create narrative structures. In *17th International Conference on the Foundations of Digital Games*, pages 1–9, 2022.

[211] Julian Togelius and Jurgen Schmidhuber. An experiment in automatic game design. In *IEEE Symposium On Computational Intelligence and Games*, pages 111–118. IEEE, 2008.

[212] Zahid Halim, Abdul Rauf Baig, and Kashif Zafar. Evolutionary search in the space of rules for creation of new two-player board games. *International Journal on Artificial Intelligence Tools*, 23(02):1350028, 2014.

[213] Jakub Kowalski and Marek Szykuła. Evolving chess-like games using relative algorithm performance profiles. In *European Conference on the Applications of Evolutionary Computation*, pages 574–589. Springer, 2016.

[214] Thomas Volden, Djordje Grbic, and Paolo Burelli. Procedurally generating rules to adapt difficulty for narrative puzzle games. In *2023 IEEE Conference on Games*, pages 1–4. IEEE, 2023.

[215] Paolo Burelli and Georgios N Yannakakis. Combining local and global optimisation for virtual camera control. In *IEEE Conference on Computational Intelligence and Games*, pages 403–410. IEEE, 2010.

[216] Paolo Burelli and Georgios N Yannakakis. Global search for occlusion minimisation in virtual camera control. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[217] Mike Preuss, Paolo Burelli, and Georgios N Yannakakis. Diversified virtual camera composition. In *European Conference on the Applications of Evolutionary Computation*, pages 265–274. Springer, 2012.

[218] Mike Preuss. Niching the cma-es via nearest-better clustering. In *12th Conference on Genetic and Evolutionary Computation*, pages 1711–1718, 2010.

[219] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Procedural content generation through quality diversity. In *IEEE Conference on Games*, pages 1–8. IEEE, 2019.

[220] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M Lucas, Adam Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolu-

tional generative adversarial network. In *GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 221–228, 2018.

[221] Philip Bontrager, Aditi Roy, Julian Togelius, Nasir Memon, and Arun Ross. Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution. In *9th International Conference on Biometrics Theory, Applications and Systems*, pages 1–9. IEEE, 2018.

[222] Matthew C Fontaine, Ruilin Liu, Ahmed Khalifa, Jignesh Modi, Julian Togelius, Amy K Hoover, and Stefanos Nikolaidis. Illuminating mario scenes in the latent space of a generative adversarial network. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 5922–5930, 2021.

[223] Dumitru Dumitrescu, Beatrice Lazzerini, Lakhmi C Jain, and Alexandra Dumitrescu. *Evolutionary computation*. CRC press, 2000.

[224] Øystein Haugen, Andrzej Wasowski, and Krzysztof Czarnecki. Cvl: common variability language. In *Proceedings of the 16th International Software Product Line Conference-Volume 2*, pages 266–267, 2012.

[225] Øystein Haugen, Birger Møller-Pedersen, Jon Oldevik, Gøran K Olsen, and Andreas Svendsen. Adding standardized variability to domain specific languages. In *2008 12th International Software Product Line Conference*, pages 139–148. IEEE, 2008.

[226] Keza MacDonald. Tough love: On dark souls' difficulty. `https://www.eurogamer.net/tough-love-on-dark-souls-difficulty`, 2019. Accessed: 01/02/24.

[227] Zachery Oliver. Dynasty warriors = dumb fun. `https://theologygaming.com/dynasty-warriors-dumb-fun/`, 2013. Accessed: 01/02/24.

[228] Brittany Reid, Christoph Treude, and Markus Wagner. Optimising the fit of stack overflow code snippets into existing code. In *GECCO '20: Proceedings*

of the Genetic and Evolutionary Computation Conference, pages 1945–1953, 2020.

[229] Cameron Browne and Frederic Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.

[230] Andrea Arcuri and Gordon Fraser. Parameter tuning or default values? an empirical investigation in search-based software engineering. *Empirical Software Engineering*, 18:594–623, 2013.

[231] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, pages 50–60, 1947.

[232] Andrea Arcuri and Lionel Briand. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability*, 24(3):219–250, 2014.

[233] Entalto Studios. `https://www.entaltostudios.com/`. Accessed: 01/02/24.

[234] Kraken Empire. `https://www.krakenempire.com/`. Accessed: 01/02/24.

[235] Márcio Barros and Arilo Neto. Threats to validity in search-based software engineering empirical studies. *RelaTe-DIA*, 5, 01 2011.

[236] Walter F Tichy. Should computer scientists experiment more? *Computer*, 31(5):32–40, 1998.

[237] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[238] Ronnie ES Santos, Cleyton VC Magalhães, Luiz Fernando Capretz, Jorge S Correia-Neto, Fabio QB da Silva, and Abdelrahman Saher. Computer games are serious business and so is their quality: particularities of software testing in game development from the perspective of practitioners. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2018.

[239] Victor R. Basili and H. Dieter Rombach. The tame project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 1988.

[240] Sira Vegas, Cecilia Apa, and Natalia Juristo. Crossover designs in software engineering experiments: Benefits and perils. *IEEE Transactions on Software Engineering*, 42(2):120–135, 2015.

[241] Brady T West, Kathleen B Welch, and Andrzej T Galecki. *Linear mixed models: a practical guide using statistical software*. Chapman and Hall/CRC, 2014.

[242] Evrim Itir Karac, Burak Turhan, and Natalia Juristo. A Controlled Experiment with Novice Developers on the Impact of Task Description Granularity on Software Quality in Test-Driven Development. *IEEE Transactions on Software Engineering*, 2019.

[243] África Domingo, Jorge Echeverría, Óscar Pastor, and Carlos Cetina. Comparing uml-based and dsl-based modeling from subjective and objective perspectives. In *International Conference on Advanced Information Systems Engineering*, pages 483–498. Springer, 2021.

[244] Jacob Cohen. Statistical power for the social sciences. *Hillsdale, NJ: Laurence Erlbaum and Associates*, 1988.

[245] Barbara G Tabachnick, Linda S Fidell, and Jodie B Ullman. *Using multivariate statistics*, volume 5. Pearson Boston, MA, 2007.

[246] J. Hair, R. Anderson, B. Black, and B. Babin. *Multivariate Data Analysis*. Pearson Education, 2016.

[247] Charlene Jennett, Anna L Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. Measuring and defining the experience of immersion in games. *International journal of human-computer studies*, 66(9):641–661, 2008.

[248] Paul Ralph, Nauman bin Ali, Sebastian Baltes, Domenico Bianculli, Jessica Diaz, Yvonne Dittrich, Neil Ernst, Michael Felderer, Robert Feldt, Antonio Filieri, et al. Empirical standards for software engineering research. *arXiv preprint arXiv:2010.03525*, 2020.

[249] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Enhancements to constrained novelty search: Two-population novelty search for generating game content. In *15th Conference on Genetic and Evolutionary Computation*, pages 343–350, 2013.

[250] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Constrained novelty search: A study on game content generation. *Evolutionary Computation*, 23(1):101–129, 2015.