TÉCNICO LISBOA

Co-creativity in Videogame Level Design

Pedro Miguel Sanches Torres Lucas

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Carlos António Roque Martinho

Examination Committee

Chairperson: Prof. Nuno João Neves Mamede
Supervisor: Prof. Carlos António Roque Martinho
Member of the Committee: Prof. Daniel Jorge Viegas Gonçalves

May 2016

“Never tell me the odds.” - Solo, H

To my parents, for their unconditional love and support, who gave me an education and every opportunity I had. To my sister, who I grew up with, for being there on every step of the way.

Acknowledgements

First and foremost I would like to thank Prof. Dr. Carlos Martinho, whose passion for his work and tireless efforts on supporting the development and writing of this thesis were essential for both its completion and its quality. On a more personal note, as a fellow videogame enthusiast, thank you for helping light the first of, hopefully, many bonfires.

I would like to thank the participants of our formal evaluation sessions: Pedro Mestre game designer at Bica Studios, Nuno Monteiro for his availability and their remaining team; André Silva and Safal Kapoor, game designers at Miniclip; André Lima, Francisco Nina and Gonçalo Delgado, students at IST-Taguspark. Their contributions helped us validate the current approach and identify future improvements for the Editor Buddy. Also, for their help during preliminary evaluations, I thank my friends André Assunção, António Dias, David Ferreira, Hugo Cabrita and Ricardo Imperial and Ricardo Silva for their time and patience.

I would like to outline and thank the contributions of our related work authors, particularly Yannakakis, G. Liapis, A. and Togelius, J. for their work on the Sentient Sketchbook, which provided a solid reference when validating our distinct approach; John Newcombe, creator of the Genetic Algorithm Framework, for providing a mature framework when implementing our solution.

I would like thank my closest colleagues and friends during these last couple of years: João Higino, Nuno Franganito, Pedro Oliveira and last, but not least, Pedro Pereira who proved to be a constant motivation and friendly rival pushing me to try my best during the development of the Editor Buddy and the writing of this thesis.

Finally, I would like to thank my family, my parents and my sister as well as my girlfriend, Sofia Gomes, whose support, care and motivation helped me abstract from the more stressful times of this process and overcome many other dificulties.

Abstract

Motivation for this work comes from the belief that there is some untapped potential in having a computer work as a *colleague* with the videogame level designer as a source of creative stimulus, instead of simply being a tool, in order to achieve overall more creative results than those obtained from solo development. The proposed solution consists of a co-creative level design tool, focused on fostering creativity by allowing human and computer to work together in producing content using the Legend of Grimrock 2 Level Editor, exploring the digital "peer" paradigm. Its interface can be used by the designer to preview generated suggestions and orient its behavior. Suggestions are generated and iteratively evolved by two genetic algorithms and can be guided by the designer on different domains: *innovation*, *objective* and *user map*. Innovation seeks to generate content different from the designer's level, objective seeks to generate content respecting specific layout guidelines and user map seeks to generate content strongly based on the designer's level. Results showed this approach to be promising since it takes into account the smaller nuances of the co-creative interaction as a source of a positive influence. Outlined improvements such as a better way to support designer-specific patterns, selection context or on-demand suggestion generation helped set a direction for future work. We concluded that through an intuitive interface, flexible and adjustable behavior and an agile interaction process, we were able to provide some interesting contributions to the quality of the co-creative level design process.

Keywords: Level-design, computer co-creativity, procedural content generation, genetic algorithms

Resumo

A motivação para este trabalho provém da ideia que existe potencial inexplorado no uso do computador como colega de trabalho para *level designers* de videojogos como fonte de estimulo criativo, não apenas como uma ferramenta, de forma a alcançar resultados, mais criativos do que aqueles criados de forma individual. A solução proposta consiste numa ferramenta co-criativa para desenho de níveis, com o objectivo de promover a criatividade através da parceria entre humano e computador na produção de conteúdo usando o editor de níveis do jogo Legend of Grimrock 2, explorando o paradigma do "parceiro" digital. A interface da solução pode ser usada para visualizar conteúdo gerado bem como para orientar o seu comportamento. As sugestões são geradas e iterativamente evoluídas por dois algoritmos genéticos e podem ser guiadas pelo *designer* nos domínios: *innovation*, *objective* e *user map*. *Innovation* gera conteúdo distinto do do *designer*, *objective* gera contúdo de acordo com certas características topologicas e *user map* gera conteúdo fortemente baseado no do *designer*. Os resultados mostram que esta abordagem pode ser promissora uma vez que toma em consideração os pequenos detalhes da interacção co-criativa como fonte de potencial estimulo criativo. Melhorias apontadas incluem uma forma de gerar padrões feitos pelo designer, contexto da selecção e uma forma de gerar sugestões *on-demand* ajudaram a definir uma direcção para trabalho futuro. Concluímos que graças a uma interface intuitiva, comportamento ajustável e flexível e um processo de interacção ágil, conseguimos proporcionar contribuições interessantes para a qualidade do processo co-criativo de *level design*.

Palavras-Chave: Desenho de níveis, co-creatividade computacional, geração procedimental de conteúdo, algoritmos genéticos

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Creativity is innate to us all. As part of our minds, it is just as inscrutable. The fact it can be influenced by countless factors, such as experiences and personalities, different amongst individuals, makes the task of detailing this process inconceivably complex.

Because the creative process is, most of the times, an individual exercise, it is invariably biased. This can become monotonous and sometimes dreary if an individual is not exposed to sufficient external stimulus. This tends to lead to a metaphorical dead-end, where one is unable to deviate from his own line of thought. Regardless of its source, the smallest amount of entropy, caused to this static thinking process, could release a flood of ideas, like water trapped in a dam, breathing new-found inspiration into the whole creative process.

The motivation for this work comes from the belief that there is some untapped potential in having a computer work as a *colleague* with the videogame level designer as a source of creative stimulus, instead of simply being a tool, in order to achieve overall more creative results than those obtained from solo development in a more time efficient manner.

People most often think of computers as a way to automate tedious tasks. In the context of level design, game engines such as Unity3D [1] or Unreal Engine [2] provide an example of this standpoint, by offering facilitating mechanisms to generate terrain, create lighting and other visual or audio effects with ease, but no ways to help improve user creativity. Some videogames support the creation of user content such as the design of levels, such as Legend of Grimrock 2 [3] or the more successful case of Minecraft [4].

We believe level design in such cases is a good example of an activity which would benefit from a computer-assisted co-creative tool. In this particular situation, new players who wish to create their own levels quickly, and without much hassle, could benefit from having some kind of help in the creative front, while more experienced users could also use a hand at improving what they are already capable of doing. There is currently a respectable amount of work and research on the topic of computational creativity and we believe these findings have some useful insight on enriching the creative process of videogame level designers.

[1] Unity Technologies, 2005
[2] Epic Games 1998
[3] Almost Human Ltd., 2014
[4] Mojang, 2011

1.2 Problem

In the particular case of level design, computers are often used to provide stimuli to user creativity, usually in the form of well defined alternative content. However, this approach may very well overlook potentially useful details in-between. That is, by omitting the intermediate stages of the alternative content generation process from the user we are, perhaps, discarding equally valuable content. Existing programs such as the Sentient Sketchbook [1] provide an interesting way to explore completely playable variations of maps for strategy games but provide minimal interaction. Tanagra [2], another co-creative tool for designing levels for platformer games, offers a reactive process which helps the user complete his level, based on what he has created, but still acts in a more independent way when generating content.

At the time of this writing, computers still lack the ability to fully understand or perfectly deduce the intentions of the human designer, something only made harder thanks to human factors, such as personality or a "bad" mood which may end up influencing these aspects, regardless of the individual. This means that, in spite of having solutions where content is generated in a more independent way which present only optimal results, these methods could prove more promising if carefully accompanied by the designer in a more step-by-step approach, further oriented by him.

In sum, the human creative process, when performed in the absence of any stimulus, invariably evolves towards stagnation, which is to say, similar styles of thought are recurrent. Current solutions present a correct, self-contained way of providing these stimulus, assuming it is the designer's sole interest to reach an end, regardless of the means. This comprehends there is nothing to be gained from the co-creative process itself, whereas we believe there is something to be gained from exploring such an activity with a computer agent, other than the outcome itself. Similar solutions which seem to disregard the analysis of the human-computer interaction, at the more intimate level, can possibly fall short of their full potential.

1.3 Hypothesis

Considering the potential of existing co-creative solutions, which are based on the preexisting notion that one or more definitive and feasible alternatives to any user generated content is the best way to assist him in the creation of new content, there is still a broad array of intermediate solutions left for the user to consider.

In this sense, we consider the benefits of fostering the user's creativity, during the creation of videogame levels, by exploring his interaction with a software tool which presents iteratively evolved suggestions. Because we are unable to accurately define the needs of the designer, qualitative or quantitatively, we consider it a necessity to have a way to guide the behavior of such a tool in order to dampen eventual discrepancies between user expectation and content generated through these means. Although sometimes content generated this way seems sub-optimal, it may hold potentially useful value, nonetheless.

By exposing the user to these less refined suggestions and inviting him to take an active part in exploring and improving them, we hope to understand what benefits can be drawn from using a more interaction-focused co-creative solution as opposed to existing ones which favor solely the display of generated content as a way to enhance creativity.

In the end, we propose a creativity-enhancing co-creative solution which focuses mainly on exploring the interaction with the designer by providing iteratively evolved, interactable, visual suggestions further guidable through interface controls to adapt them to the needs of the user.

1.4 Objectives

The final goal of this work is the development of a software tool which seeks to foster creativity in level designers by offering a digital "peer" to interact with, during level design activities and in the end evaluate its *utility* and *usability* as well as the *user experience*.

This tool is aimed at easing more troublesome stages of video game level design such as deciding the initial layout or alternative ones, thus improving the designer's efficiency by exploring the potential of co-creativity between the designer and the AI.

The first step was the implementation of the software tool itself which serves as an external application to the Legend of Grimrock 2 Editor. The second step of this work was the evaluation of the final solution. This preliminary evaluation would help us identify the strong and weak points in our implementation and decide on a better direction for future work.

In the in the end, we draw some conclusions as to whether this tool hinders or improves the performance of the end user, having ideally two distinct type of users as test subjects: professional game and level designers and inexperienced users.

1.5 Contributions

Below, we present a list of contributions this work will present to the scientific community.

1. State of the art co-creative PCG techniques in videogame level design
2. Computational model for generating and evolving content alongside the level designer on specific domains
3. Software implementation integrated with a videogame level editor
4. Preliminary qualitative evaluation on the usability and utility of the implemented computational model with video game level designers

In the end, we focus on providing a self-critical reflection in order to help identify the weak and strong points in this work and what can be done in future work to help correct and improve them.

1.6 Document structure

In our first chapter we began by introducing our motivation and objectives for this work. On the second chapter we start by presenting the results from our research phase, the background section, where relevant subjects are introduced and analyzed.

The following chapter addresses the related work where we describe existing solutions, state of the art and scientific work which supports our theories. This is done by discussing them in-line by topic. In the end of this chapter, we briefly discuss what these finding meant to the development of our solution.

The following two chapters pertain to the presented solution, the first one details our solution model, an overview of what it consists and how it fits the theoretical background. The second one, refers to the solution implementation, how it integrates with existing software and details regarding its more technical aspects and decisions.

In the evaluation chapter we present the several stages of evaluation, a couple of earlier preliminary evaluations and the formal evaluation conducted on the final version of our tool. We describe our study, participants, used methods and interpret their consequent results.

In the final chapter we present our conclusions in regards to the developed work and how our evaluation helped on identifying areas for future work.

Chapter 2

Background

In this chapter we address the more theoretical aspect of our work as well as some relevant concepts to it, in a background perspective. The first topic being creativity, what do we understand by creativity, relevant concepts, how they relate with each other and what does creativity mean in computer terms. Procedural Content Generation is our second background topic, we address what is meant by PCG, how can we see it as a way to help improve creativity and its relevant aspects.

2.1 Creativity

"Creativity is the ability to come up with ideas or artifacts that are new, surprising, and valuable." – Boden, M. A. [3]

Creativity is inherent to each and every one of us, present in various aspects of our lives. It is, therefore, not a skill to be acquired (although it can be trained) but one of many facets of the human mind. In this sense, rather than asking if someone can be creative, we should be asking just how much creative can someone be. The concept of "new" can be quite subjective, depending on whether we are referring to an individual alone or the entire human history. Chances are, when an individual has an idea, that conception is not at all new to the history of Mankind. Margaret Boden [3] distinguishes these definitions as "P-creative" for *psychological* creativity and "H-creative" for *historical* creativity. In the interest of this work, we turn our attention to the former, the psychological creativity.

Although creativity is still a very mystic and shrouded concept, and impossibly hard to define intelligibly in scientific terms, there must be a way to start discussing it. Science encourages us to believe it must follow some kind of order, as in all things. Margaret Boden [3] defines a structured way of thinking called *conceptual spaces*, which are influenced by the surrounding, culture, individuals and other external factors that contribute to the formulation of a line of thought, accepted within a certain social group.

Human creativity can be categorized into three different styles, ordered by the degree on which they attempt to break preconceptions within their own conceptual space:

- **Combinational creativity**, which consists of simple combinations of existing preconceptions, inside a conceptual space, in order to produce a new idea
- **Exploratory creativity**, which explores preconceptions inside the conceptual space
- **Transformational creativity**, where the alteration of the conceptual space itself leads to new forms of reasoning and, therefore, ideas otherwise unthinkable

Associated with idea generation is the surprise, reflecting the amount of novelty the idea conveys, which directly relates to the type of creativity, where the largest degree of surprise derives from *transformational* creativity.

Lateral and Vertical thinking

Closely related to creativity is the notion of lateral thinking. Lateral thinking is defined as the act of problem solving through an *indirect* and *creative* approach. Edward de Bono's introduction to lateral thinking in [4] proposes lateral thinking is closely related to creativity in a way that creativity is often merely the description of a *result*, whereas lateral thinking is the description of a *process*. Moreover, he states the purpose of lateral thinking is to update currently established ideas and preconceptions. The way it attempts this is by challenging old ideas, whether with a different idea or with new information. In its essence, it means bringing about conflict in order to question existing assumptions not just for the sake of it.

Vertical thinking, as the most frequent style of thinking, is a very strict and well defined process, a sequence of steps where each step depends on the validity of the previous one. It always chooses the most valuable alternative, above all others, and always follows a path as long as there is a more valuable alternative down that path.

As stated by Edward de Bono in [4], "Vertical thinking moves only if there is a direction in which to move, lateral thinking moves in order to generate a direction.". Lateral thinking, therefore, is all about generating conflict, for the sake of generating new paths or alternatives. Figure 2.1 presents an illustration of how these two techniques differ.

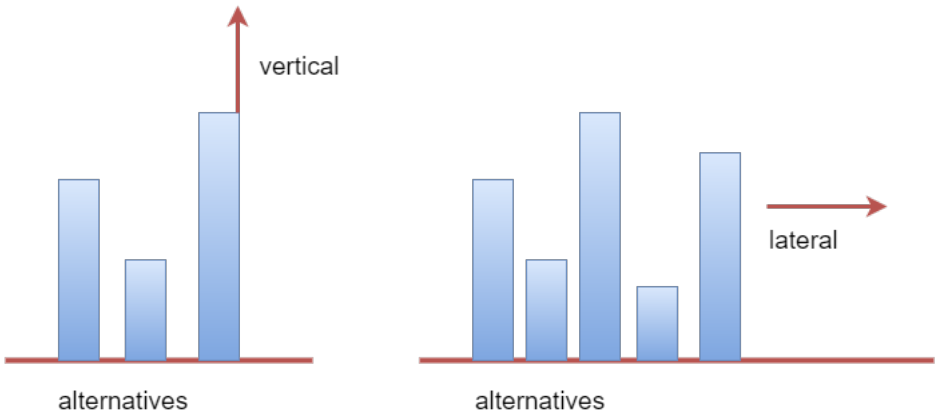


Figure 2.1: Difference between lateral and vertical thinking

We can identify the outlines of some techniques and behaviors related with lateral thinking which can be explored during the design activity. De Bono presents seven types of lateral thinking techniques:

1. **Alternatives**, using concepts themselves as starting point in order to find less obvious alternatives
2. **Focus**, when and how to change the focus of our thinking
3. **Challenge**, tear down the boundaries of traditional thinking making way for different ideas even though the may be less than ideal
4. **Random Entry**, using unconnected input to open up new lines of thinking
5. **Provocation and Movement**, generating provocative statements and build new ideas on top of them

6. **Harvesting**, capture all creative output at the end of a creative session, instead of just that which seems most practical and conventional

7. **Treatment of Ideas**, treating ideas to fit into a specific situation

In the end, however, the vertical and lateral styles of thinking are so distinct we would miss out on relevant information if we discarded on of them in detriment of the other. Both of them are valid, important processes and both are necessary. In this work we would like to explore the potential of the computer in applying some of these techniques and evaluate just how useful they are in nurturing the human-computer co-creative process as a whole.

Computational creativity

What of the computer? How can a machine be programmed to act creatively, when even humans have not found a way to accurately express how the creative process works, let alone model it. Quoting Margaret Boden [3], there are those who discard computer creativity as valid and argue this is a consequence of human action, since humans are the ones who programmed it in the first place. Several other arguments are presented such as the computer's inability to neither appreciate nor produce a work of art, because it lacks conscience and therefore desires and preferences. Such feats are only at the reach of humans, since a work of art is thought to portrait experiences and feelings, automatically excluding machines.

Boden addresses this topic analytically [5] by suggesting that, at the light of the definition of *combinational* creativity, computers are, at the very least, capable of putting together two "ideas" (structures) in attempt to generate a new one. Would these ideas have any valuable meaning? That's what sets aside humans from computers. Ideas generated this way would have to be evaluated, as well as the two previous ones, to extract some subtle connection between them that would prove the added value resultant from that fusion. Such a task would require a vast amount of information, similar to what a human adult would have access to, including a life-time of experiences. There are, however, very capable computer models that present interesting results using combinational creativity. Such is the case of JAPE [6], the riddle generator, which besides using combinations relies on strict rules of structure to satisfy the requirements to produce clever puns.

Exploratory creativity too can be modeled by AI, if the rules that define the boundaries of the relevant thinking style can be clearly stated. Easier said than done, since humans themselves have spent large years in identifying different styles. Nevertheless, *exploratory* creativity is regarded as more successfully accomplished by computers than *combinatorial* creativity, namely in the level of quality of their solutions, comparable to those of a competent human professional. Boden mentions a couple of fields where such examples can be found, like stereochemistry, physics or music. She draws our attention, however, to a particularly interesting program devoted to the category of visual art called AARON [7], created by Harold Cohen. Also, the work of Pat Langley et al.[8] tackles the decade old argument of AI as a science rather than just engineering with several scientific examples he calls of BACON family. Under development since the 1970's, this set of examples focused on the "discovery" of several laws of classical physics such as Boyle's law or Ohm's law as well as basic principles of chemistry. Although these discoveries were obviously *P-creative*, latter members of the set were oriented towards genuine discovery, or *H-creativity*.

On the level of *transformational* creativity, there are a few AI-programs that happen to transform their conceptual space, resulting in rule reformulation in order to promote the appearance of varied ideas.

Such is the case of Evolutionary Algorithms (EAs), who change rules during their run, choosing the best structure for each generation. The challenge with these types of programs remains the accurate formulation of human values into computer understandable rules which define what *valuable* means, during the program's execution. For the most part, programs mainly rely on user input, such as a visitor in an interactive museum. Boden, however, mentions that sceptics still argue this process of self-mutation seems forced and that its results where synthetically simulated. This was far from the process of organic transformational creativity, they said, in which organisms assimilated elements present in the world and incorporated them into their environment, acknowledging them as a defining part of their conceptual space. However, even if true for simulation Genetic Algorithms (GAs) given their lack of sensors in the real world, the same couldn't be proved for the rest of Artificial Intelligence (AI). Supporting this conclusion, a team of the University of Sussex [9] reported an interesting finding. When using a GA - running in hardware - to evolve oscillator circuits, they observed the final circuit (chosen by the algorithm) acted as a primitive radio antenna which would pick up and modify the signal being sent from a nearby computer. In short, they concluded this unforeseen physical parameter had, in fact, had a crucial impact in the evolution of the GA and, ultimately, its final product. Still, because the number of factors involved may be too great and uncertain, the doubt behind genuine (not merely accidental) transformational creativity still persists.

2.2 PCG

Procedural Content Generation (PCG) comprehends the generation of content through random or pseudo-random processes, resulting in an unpredictable range of results. PCG, has had somewhat of a mixed presence when it comes to games. In one way, there are more than a few games that have sported this kind of facilitating technology, some as early as Rogue (M. Toy and G. Wichman, 1980) or Elite (Acornsoft, 1984) or some more recent titles like Diablo 3 (Blizzard Entertainment, 2012) or Torchlight 2 (Runic, 2012). One reason, if not the main reason, for relying on this technology was the degree of replay value these titles would achieve when offering this feature. As suggested in [1], the meaning of content is subjective to the particular demands of any one game, for instance the Borderlands (Gearbox, 2009) franchise would draw the consumers' attention towards the large variety of weapons in-game as a main selling-point of the game, a result of PCG, nonetheless.

PCG is becoming an increasingly popular shortcut taken by game companies to create relatively good content in a short amount of time. This is especially true for games where content needs to be vast but still fall within certain constraints (weapon generation in the Borderlands franchise, for instance), where it becomes more cost and time efficient to use procedural content generation instead of human designers. Although PCG has been very useful in solving some problems in game development, mainly those associated with content production costs, it is still somewhat far from being an everyday solution for every game. The reason being, it is still not mature enough to replace the careful, custom-tailored experiences game and level designers spend so much effort in producing. Therefore, there is this sense of trade-off when deciding to use this technology. On the one hand, it's a way of crafting "new" gameplay content with virtually no effort, on the other hand, we must sacrifice the more subtle details and nuances when it comes to gameplay experiences, in detriment of obtaining faster results, some would consider inferior. One can indulge in wondering if PCG can have the potential of creating a breed of video games with mechanics centered exclusively on the creation of content in this fashion. This topic will be discussed further when addressing computational creativity. Nevertheless, there is no denying PCG is a rapidly emerging aspect, even more when urged by the rising demand of consumers in the increasingly popular videogame industry. This being said, it does not fall within the goals of this work to create an

automated level generation tool.

Taxonomy

Procedural Content Generation does not limit itself to the visual paradigm of game content, there is Procedural Storytelling, as well as Procedural Audio for instance, and both can be considered valid gameplay content. Focusing in the objectives of our work we will narrow our view towards analyzing the more tangible kind of content, which is related with navigation (map layout).

We now take a look at the taxonomy used by Togelius *et al.*[10], where PCG techniques are evaluated according to different aspects:

- Online vs Offline
- Necessary vs Optional Content
- Random seed vs Parameter vectors
- Stochastic vs Deterministic Generation
- Constructive vs Generate and Test

This alternative taxonomy allows us to classify and place different PCG techniques in a continuum, rather than separating them by behaviors. Let us review these aspects more closely what translate how they affect the procedure of content generation.

The distinction between *online* and *offline* content generation refers to when the generation takes place. Offline content generation occurs in the design phase of the game, where the designer can still fine-tune some aspects of the content generating algorithm before the product is published, whereas online content generation takes place during the gameplay, for instance, when the player switches areas in an open world game, the topology of the terrain is generated in real-time.

The *necessary* vs *optional* content distinction relates to whether the generated content is a main part of the gameplay, or just secondary with little impact to the player experience. A golden rule for necessary content, however, is it must be consistently correct. What this means is, it is bad practice to have content go against what the player is trying to accomplish, which is to progress. The definition of necessary and optional really depends on the game we are addressing, meaning that something that may serve as optional content in one game, could be the necessary content of another. For instance, in games with highly-immersive visuals and realistic graphics, the slightest misshaped tree can break the main objective of the game, which is the sense of immersion itself.

Defining the level of parametrization in PCG is also a relevant aspect, hence the *random seeds* vs *parameter vectors* distinction. This aspect depends on the level of control and customization we want, or rather, need to have when guiding the algorithm towards the type of content we wish to generate. This can range from a simple pseudorandom number, to a multidimensional array of parameters.

Stochastic versus *deterministic* generation is a difference that measures the amount of randomness of a given algorithm. Deterministic algorithms always return the exact same results for the exact same parameters. Stochastic algorithms, found in roguelike games, for instance, do not produce the same output, even if parameters are the same. For those cases, a random seed is used besides the parameters, which is why results vary.

Finally, a more technical distinction between procedures are those who can be called *constructive* and those who can be called *generate-and-test*. For the former, the generation process is simple and consists of following a sequence of operations, which are previously guaranteed to always produce correct content, to generate the final content. An example for this is the use of fractals in order to

generate terrain. The latter comprehends a cycle of content generation and testing until the produced content satisfies the problems' constraints. In case it is not satisfied for a given candidate content, some or all of its content is discarded and regenerated.

2.3 Discussion

In the end of this chapter we understood the essential aspects of creativity as introduced by Margaret Boden [3], what does it mean to be creative, different existing types of creativity in regards to how formulation of new ideas is performed inside conceptual spaces. We also drew our attention to the distinction between lateral and vertical thinking two descriptions of common thinking processes and how lateral thinking is more closely related with the goal of our work. In the light of previous definitions, we explained how existing computers models are capable of different creative processes and presented some examples of particular implementations.

In regards to PCG, we were able to detail what it is, its growing importance in the area of videogames and how this concept can ends up being relevant in our case. Being such a complex technique (or techniques) capable of being so specifically parametrized, we reviewed how a specific syntax used by Togelius *et al.*[10] helped us classifying existing techniques and place them into a spectrum.

Chapter 3

Related Work

In this chapter we address relevant work from which we draw several examples to help support our decisions. Instead of discussing each literature individually, we try to analyze them by topics and discuss these works in-line with what makes them relevant to our research and development, according to each topic.

3.1 Creativity-enhancing computer models

In the work written by Lubart [11] and addressed in [12], it is possible to identify and categorize the role of the computer as a way to support human creativity. The proposed classification is based on four categories, stating computers may facilitate:

1. The management of creative work
2. Communication between individuals collaborating on creative projects
3. The use of creativity enhancement techniques
4. The creative act through integrated human-computer cooperation during idea production

From the mentioned categories, Lubart also presents how they can be translated into four lines of thought about the role of computers in creativity enhancement: computer as a nanny, computer as pen-pal, computer as a coach and computer as a colleague.

In the paradigm of the *nanny* computer, the author refers persistence as one of the most desirable qualities needed for creativity. In this case, the computer takes a more passive role in the creative process and is used instead in the management of the user's productivity, namely his schedule, by detecting periods of procrastination and productivity breaks.

In the *pen-pal* example, the computer assists the user by promoting communication. Communication, mainly in a collaborative creative process with two or more individuals, is a fundamental tool to fulfill the need of idea sharing and synthesis. Moreover, it is all the more practical when we think of rather large groups of people contributing to one project that would be difficult to accommodate in a limited physical space.

In the paradigm where the computer is used as a *coach*, it is proposed a user's cognitive domain and narrowed thinking style is sometimes inconvenient and often hinders his performance for a certain task. For this reason, and thanks to its knowledge in creativity-relevant techniques, the computer assumes the responsibility of presenting a kick starting sentence or topic to the creative process, as well as offering information about existing techniques to stimulate creativity.

The final role of the computer as a creativity enhancer is the most ambitious, the role of a *colleague*. It envisions a real partnership between human and computer, where the creative process would be a shared activity, based on a cycle where the user would have the initiative to start the process followed by a suggestion or a modification, randomly or heuristically, performed by the computer until a meaningful outcome is produced. Even if computers, in this sense, are bad at evaluating the value of an idea, they excel at exhausting a search-space. Because one of the successful tactics in creative thinking is to rely on random or semi-random search methods, computers can help arrive to unconventional, potentially valuable raw ideas. From this point, the user would transform this rough concept and fine-tune it into a coherent idea.

Taking this into account, we analyse what existing solutions have to offer in this field. The Sentient Sketchbook [1] approach mentioned the need for pro-activeness from both parties, the human and the computer, although the degree of contribution from each party does not necessarily need to be the same. As an example, the authors mention the clear distinction in the area of PCG regarding this matter by pointing out professional tools such as *Garden of Eden Creation Kit* [1] or game engines such as the *Unreal Development Kit* [2] in the sense they provide great utility to the designer by aiding him in less creative tasks such as path-finding and interpolations but greatly limit the computer's initiative. In contrast, there are other more content-oriented PCG tools which almost shut down the designer's initiative such as the *SpeedTree* [3] which focus exclusively in generating procedural artifacts, trees in this case. In a clear attempt to explore the possibilities in-between, Sentient Sketchbook generates its suggestions by mutating the designers sketch, exploring those alternatives and present playable solutions to the designer.

Tanagra [2], as another mixed-initiative co-creative design-assisting tool, behaves in a slightly different way. Tanagra focuses on generating content for platformer genre games. By allowing the human designer to specify positions of key-platforms it then generates content around those key-platforms to create the remaining map topology. Tanagra works in an iterative way, meaning it takes into account eventual alterations made by the designer and rapidly regenerates sections of the level, suggested by it, to accommodate those new decisions. In their findings with genetic algorithm performances, Smith et al. underline the requirements for new set of techniques in a mixed-initiative approach such as Tanagra, because such algorithms fell short of their expectations when regards to their real-time performance, although they have proved their worth in offline level generation.

Lubart's work also addresses a crucial point: there is no "average" user, which is to say there is no definitive way of building the perfect creative computer program. There is, instead, a rather large number of different perspectives on how creative computer model should perform. Because personality, thinking style, experience and even the mood heavily impact the creative potential of each individual at a given time, which ends up conditioning how an interactive co-creative session between them and a computer unfolds. Where some would find the nanny model useful, others could potentially find it bothersome and useless. More generally, some could find the whole interactive process with a computer a step backwards from a typical face-to-face discussion, whereas others believe there is something valuable to be reaped when engaging in a creative thinking session with a computer partner. The bottom line is, it's plausible to think of computers as a positive influence when having them for an intervening partner during a creative session instead of using them only as a technical environment, whether for the generation, evaluation or refinement of ideas, or maybe all of the previous reasons.

[1] Bethesda 2009
[2] Epic Games 2009
[3] IDV 2002

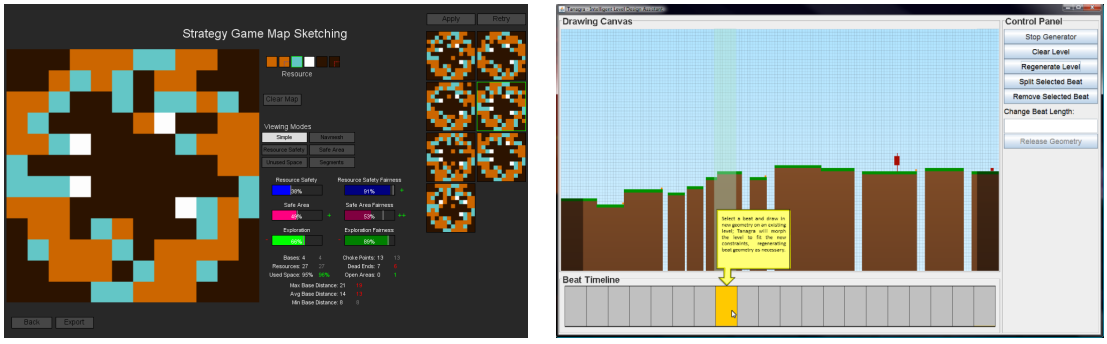
3.2 Computer Co-creativity

In the context of this work we now turn our attention to the more particular subject that is computer co-creativity. The findings of Yannakakis *et al.*, described in [13], present a pertinent analysis on the role of both human and computer during Mixed-Initiative Co-Creativity. This is particularly interesting since it addresses the concept of *lateral thinking* and merges it with that of *diagrammatic reasoning*.

Yannakakis' work refers to lateral thinking as a fundamental contribute to creativity improvement. It also mentions the random stimulus (entry) principle of lateral thinking which relies on an external interaction to agitate the pool of notions and preconception by forcing the assimilation of that external element into the process of idea synthesis. In this sense, the main source of randomness would be the computer agent. These stimuli, however, need not be completely random, as they can be heuristically-driven to better adapt to a given problem.

Diagrammatic reasoning is the second key element in this study and can be defined as the use of visual representations in the reasoning process. It is based on the idea, as literature suggests, that complex information processing tasks are better performed when there is some sort of visual aid or illustration. Much like assembling a chair, one would find such activity trivial if there was some kind of visual guide.

By bringing these two concepts together, the notion of diagrammatic lateral thinking emerges. This fusion is as straightforward as the name suggests: the activity of lateral thinking aided by visual stimulation. Studies on the use of diagrams [14] argues the process of constructing a diagram is, in the end, more important than the result. In this sequence, mixed-initiative co-creativity, occurring in the form of diagrammatic representations, produces several non-linear lateral paths which contribute to the characteristically deep exploration of possibility space that is the core of lateral thinking. Examples of some applications that put this theory to practice are the Sentient Sketchbook [1] and Tanagra [2], where figure 3.1 illustrates their user interface.



(a) Sentient Sketchbook user interface

(b) Tanagra user interface

Figure 3.1: Stimulating user creativity through diagrams

When it comes to computer initiative, in creative terms, its contributions need to be *novel*, but also *valuable*. To do so, the state search space for the computer to explore is constrained in order for the computer to present both useful and novel suggestions during its generative process. Let's remind ourselves, in the light of Boden's work, the specific case of Mixed-Initiative Co-Creativity (MI-CC) realizes mainly exploratory creativity, while it could potentially achieve transformational creativity.

3.3 Evaluating Computer Creativity

A widely important topic is how the evaluation of creativity is actually performed. We draw our attention towards a couple of pertinent views on computer creativity evaluation. David Brown argues in [15] creativity is not within computer models, instead, it is inherent to the generated artifact. In this sense, value comes from the final result of those models as opposed to the process itself. As such, we would need to establish evaluation metrics which allowed us to qualify the value of a certain artifact. This author refers to several factors relevant towards evaluation: First of all, the knowledge, experience, feelings and personal preferences of the evaluator. Also, the type and duration of exposure to the target of evaluation. Last but not least the "norms" for the relevant population to which the evaluator belongs to, expert or novice. This study recommends surprise, described by Boden [3], as a way of evaluating the creativity of artifacts in detriment of the perhaps more common novelty, value or utility.

This evaluation, in the particular case of Mixed-Initiative Co-Creativity [13], is not so straightforward if we consider the fact there is an ongoing interaction between human and computer which is hard to represent in the final outcome. Yannakakis *et al.* better explain this as the difficulty of capturing the impact of pro-activeness of the AI on human creativity and vice-versa. Therefore, Yannakakis considers two types of evaluation when a computational creator is involved: the evaluation of the final (or intermediate) outcome and the evaluation of the co-creative process of outcomes, solutions or items. Evaluating the final outcome is easy through the use of heuristics, but the evaluation of the entire co-creative process is far from trivial as the hard part lies in enumerating all the human creativity sub-processes involved. It is also argued MI-CC supports and fosters the creative process towards a certain outcome in addition to fostering the creative value of that same outcome.

When addressing the evaluation of computer generated content in the context of videogame levels, Liapis, Yannakakis and Togelius in [16] introduce the notion of design patterns namely three, for having high enough generality while being easily quantifiable: *Area Control*, *Exploration* and *Balance*. In the context of their work, they derive two formulas used to evaluate game levels in regards to the aforementioned patterns, each for a different game genre: multiplayer strategy game maps and single-player roguelike dungeons. Amongst other reasons, they also draw our attention the the importance of design patterns as a way to facilitate the study of existing games as well as a way to teach game and level design. In order to help the recognition of design patterns and, therefore, evaluation, they introduce the notion of map sketches, first presented by them in [17] as a way to abstract complex game levels into their basic building blocks, isolating the essential mechanics from contextual and thematic details. Thus, the abstract visualization and compact size of map sketches allow allow for both the human designer and an algorithm (an *artificial designer*) to process and edit map sketches with less effort. Also, at the implementation level of the Sentient Sketchbook, the genotype of individuals is directly encoded as an array of integers, denoting impassable and passable tiles. In this sense, playability is automatically assured using constrained optimization, enforced by their implementation of a FINS algorithm. In their case study, there is mention to popular design paradigms for roguelike dungeons, but we focus our attention on the complexity of the game level, considering the player's starting position as well as the level's exit location.

3.4 Search-based creativity

If we recall Boden's [3] work and Wiggins' [18] consequent formalization of it, we can translate the problem of creativity at the *exploratory* level into a *constrained optimization search* conducted on the conceptual space. This further supports the idea that search algorithms are a valid choice to simulate

the state space exploration represented by the second type of creativity, the exploration of the conceptual spaces. Furthermore, if we consider the aforementioned capability of computers towards transformational creativity we can further narrow down our pool of candidate AI to EAs, more specifically, GAs.

Procedural dungeon generation

In the work of Bidarra *et al.* [19] we find several interesting methods on how to perform procedural generation of dungeons. The authors introduce the concept of dungeons, mentioning them as good candidates to display the capabilities of PCG due to how they combine *pace*, *gameplay* and *game spaces*. By describing common practices and their pros and cons, they provide a valuable reference in regards to comparing existing implementations and how they perform individually and among each other. From the listed methods of procedural dungeon generation, we took a particular interest in those who made use of Genetic Algorithms. We now address these solutions and outline their workings.

The first documented implementation belonged to Hartsook *et al.* [20] and how they present a way to procedurally generate stories, as opposed to conventional 2D or 3D content generation, using a metaphor of islands and bridges to create a space tree. Interestingly enough, genetic algorithms in this case, specifically the mutation and crossover operators, handled the addition and deletion of outer most nodes in the tree. Effectively, this tree would later be translated into a grid in order to be used as basis for a map. In spite of having some potential use in some form of 3D mapping, we were more interested in 2D focused implementations.

The next detailed implementation was by Valtchanov *et al.* [21] and is based on a similar tree-like structure. In their case, they use this structure to represent partial levels, where connections between tree nodes translated into connected rooms in their map representation. In their specific implementation, they used a single fitness function with a strong affinity for maps composed of small, condensed clusters of rooms. They seem to show, however, no explicit gameplay-oriented room placement, which meant solutions tended to converge into these tightly packed room clusters, sooner or later.

The third mentioned implementation is that of Ashlock *et al.* [22] in which he lists four distinct ways to explore the generation of maze-like dungeons, in regards to genetic algorithms: direct binary, direct colored, indirect positive and indirect negative. Bidarra *et al.* draw out attention towards the direct mazes for being the most interesting. By being made up of a grid, crossover and mutation operators would perform genetic recombination by flipping cells into becoming a wall or an accessible area, withing the context of their maps. An interesting aspect of this implementation is the existence of four different fitness functions, which can be used, one at a time, to produce different results each time the program is used. We found this aspect to be potentially useful for creative purposes.

All of the previous implementations make use of conventional genetic algorithm parameters, such as number of generations, initial population size and the fitness function description. In spite of showing promising potential in supporting several types of content to be generated, it could prove hard for designers or general users with no programming or mathematics background to parametrize. This led us to the most promising implementation listed by Bidarra *et al.*, none other than the Sentient Sketchbook. They underline how it handles the emergent problem of genetic algorithm parametrization by exploring a more gameplay-oriented fitness evaluation. During our research phase, this seemed the most promising type of approach, as such, we now focus on the more technical details of this implementation, and where they are placed, in terms of their creative process.

Objective-driven and novelty search

The work of Liapis and Yannakakis [1], more explicitly their findings in experimenting with Fi2pop [23] and FINS [24], showed to produce good results in the paradigm of exploratory and transformational creativity, considering the extensiveness of the state space. We are most interested in a couple of details on how these algorithms behave, such as how infeasible individuals are handled between generations as well as objective-driven and novelty search.

With standard Genetic Algorithms, when the selection phase is reached, the most promising offspring is usually kept for the breeding of the following generation and the remaining offspring is discarded. Instead of immediately discarding the infeasible offspring, a variation of the Standard Genetic Algorithm - the Feasible Infeasible 2 Population GA - maintains two groups of individuals: one for feasible and another for infeasible individuals. The theory behind this rests in the potential for valuable, previously unforeseen, feasible solutions within unexplored infeasible individuals. As such, individuals in the feasible population evolve towards maximizing their current fitness function, whereas individuals in the infeasible population evolve towards minimizing their distance from the feasibility border, attempting to become feasible and earning a place amongst feasible individuals. This two population approach, vulgarly Fi2pop, values individuals for their unforeseen potential over their punctual infeasibility.

Novelty search is also motivating in the paradigm of this work. Recalling Lubart [11], we wish to place the computer in the role of a colleague, something that would challenge us and feed us interesting suggestions and collaborate with us towards a common goal. This is particularly interesting when considering how novelty search complements the objective-driven implementations while behaving "in a more satisfactory way in domains where fitness functions are deceptive, subjective and hard to quantify" [25]. This type of algorithms, instead of following a complex fitness function, strives to diversify the solutions in a population. It does this by selecting individuals according to their novelty score, which basically translates to the amount of difference from other individuals in the population. In their work in [26] Liapis et al. attempt to conciliate novelty with objective-driven searches and search for both good and diverse game levels. Although the notion of "novel" and "diverse" can be quite subjective, they present three different measures to novelty, in the context roguelike genre games, amongst others: *Tile-based* diversity, *Objective-based* or *quality-based* diversity and *Visual impression* diversity.

Yet another key factor in the implementation of the Sentient Sketchbook is how user decisions affect the algorithm in the long term. Liapis *et al.*[27] introduce us on how to accommodate the specific style of a designer into the algorithm, and how to extract it. To understand how the Sentient Sketchbook accommodates the style of the designer we first need to take into account how it handles multi-dimensional content evaluation. The final fitness function is the sum of several weighted fitness functions, which correspond the several relevant dimensions within the context of their work. Thus, accommodating the style of the designer is easily accomplished by changing the weights of the dimensions in the final fitness function. In this sense, weights are increased towards specific dimensions if the user has selected a computer generated suggestion heavily based on those dimensions, at the same time weights of the remaining dimensions are toned down. This practice is based on choice-based interactive evolution (CIE) and supports the idea that a designer's style carries relevant traits. Extracting the designer's style can occur in one of two ways: *modeling style* and *modeling process*. The former learns only when a user performs the selection of a suggestion, storing the weights in a database. The latter learns on each user action, but does not store the updated weights. Both have advantages and drawbacks but both of them share the belief that choice-based interactive evolution draws fitness scores from a weighted sum of relevant metrics, where the weights vary according to the user.

3.5 Discussion

In our first section we addressed an inevitable topic for this work, how are we effectively using the computer for our creative purposes. For this project, we intended to create an experience where the user would work closely with the AI during the whole level design activity, much like the last category identified by Lubart [11]. This way, by having the user adapt the behavior of the AI to their own needs, this could invalidate the need for a perfect creative computer program, something which Lubart seemed to dismiss as unlikely, either way. We mention how existing solutions, described in the related work section such as Tanagra and the Sentient Sketchbook, behave in this sense. Tanagra is a more interaction-centric experience where the generated content is presented in a quick, reactionary fashion. The Sentient Sketchbook behaves in a more independent way, where for each sketch update the designer performs, several playable alternatives are presented after some time. In our case, we wondered about the benefits of having somewhat of a combination of these approaches, in the sense of having both an agile interaction cycle between designer and computer as well as content variety and alternative exploration. That meant adapting the content generation process to a more step-by-step model where the algorithm gradually improves its solution as opposed to a more self-contained process where for a given input, the algorithm outputs a definitive solution. In this sense, we envisioned a turn-taking cooperative level design activity, between human and computer, where both parties would simultaneously explore their ideas while able to examine each others' and be able to incorporate those ideas into their work.

The role of the computer, in our particular case, aims to be that of the colleague, as described by Lubart [11], by fostering creative outcomes through "semi-random search mechanisms to generate novel, unconventional ideas". These ideas, these stimuli, don't necessarily need to be a blind attempt at finding a meaningful solution, like we mentioned before. Actually, considering the nature of this project, there are several ways the computer can provide "novel ideas" without being completely dependant on randomness. Referring to what Liapis *et al.*[16], in level design, there are certain patterns which are universally recognized as meaningful, like in the domain of Exploration, such as tight maze-like layouts or divisions connected by paths in our particular case. We strive to generate similar visual patterns in order to poke at the designer's interest and creativity, much like described in the theory of diagrammatic lateral thinking. At this point, we are still only considering the more passive type of computer colleague. However, to what extent, we may ask, is it worth having a colleague you can simply ignore in detriment of one who is actively contributing to the entropy of the cooperative activity and, ultimately, its outcome? We can argue the element of frustration plays an important role and that at the end of the day, its the designer's call to bring closure to the activity. During development, we have indulged in questioning the benefits and hindrances of having such an active colleague and, although it was doable, we decided to establish a better grounded theory before exploring that alternative.

Evaluation is a necessary part of this interaction, and we need to distinguish two moments of evaluation. Evaluating the quality of the work resultant from the co-creative activity and evaluating the quality of the co-creative activity itself. In our particular case, we are equally concerned on generating good content for the benefit of the designer (and the player), as well as providing an consistently fruitful and engaging interaction. Although problem specific constraints play an important part on our implementation and its usage, there was a time where we had to commit to a trade-off between responsiveness and quality of generated content. We do not necessarily mean a decrease in quality in the sense of its intrinsic, potential value, but rather a lack of immediate heuristic value. Sure, there can be dozen playable alternatives to a designer's level, however, those alternatives may come with a significant delay. At which point we should wonder: to what extent does an heuristically superior, but significantly late, solution prevail over one of its more rudimentary, but more timely, counterparts? Moreover, it is not solely the quality of the generated content, but also the quality of idea communication that we try to balance.

In terms of evaluation, all these aspects play a relevant part, as we cannot tip the scales in favor of one party without expecting the whole dynamics of the interaction to shift, even if only slightly.

Although the findings of Smith *et al.* in Tanagra [2] go against the usage of genetic algorithms, for valid reasons in the topic of creativity-enhancing computer models, we concede, we could not help but think the potential within evolutionary algorithms could be worth the investment we would make, in the short to medium term. The several, albeit brief, overviews we did of the works mentioned in [19] helped us understand just what has been explored in procedural dungeon generation for games. In these overviews we tried to convey the strong points of each implementation, as each one had an interesting approach to the challenge of dungeon generation. In some way or another, we ended up being drawn by one or another particularly interesting aspect from some of these implementations. We would like to outline, however, the use of several fitness functions in [22] which served different purposes as a way to provide content variation, as well as their insightful views on maze representations.

The work of Liapis and Yannakakis on the Sentient Sketchbook [1] serves as a good point of comparison to what we tried to accomplish and initially helped validate our first implemented features, like the interface and the behavior. We never intended to replicate the Sentient Sketchbook in any of these aspects, nor any other mentioned work, however, we can relate to its base premise of presenting visually suggestive content hoping it results in better creative responses from the designer. This is were we focused on the findings of Liapis *et al.* regarding constrained novelty search and the aspiration to find both “good” and “diverse” game levels, whatever their connotation may be to the designer, mainly focusing on *Tile-based* diversity. We adopted a different approach concerning both algorithm behaviors, but still the outcome we expected was similar, although occurring at a different rate. Having two algorithm implementations each exploring significantly contrasting search-space areas is what motivated our approach, but instead of simply having all those valid alternatives be presented at the end of a long run, much like the Sentient Sketchbook, we experimented with merging content from both algorithms using experimental techniques. In the end, behavior on our application is handled by novelty and objective driven algorithms who work together to generate a pool of varied potentially creative content. Although the algorithms we use evaluate their individuals by means of a weighted fitness function, based on more than one attribute, and it seemed like fit candidates for a similar approach regarding the extraction of the designer’s style, this ended up falling outside of the scope of this work.

Chapter 4

Solution Model

In this chapter we detail our solution model and start by referring the importance of the computer colleague paradigm we are focused on conveying and what that means to the model itself. Secondly, we describe our model by components and present an overview of their elements and what purpose do they serve. Additionally, we briefly address how our solution is integrated with the level editor we chose, which will be formally introduced in the next chapter.

4.1 Computer colleague paradigm

There is a particular paradigm we are trying to portray with our solution and it is crucial to emphasize it so we are able to understand both the relation between content it generates and respective interface configuration and its iterative level design cycle. To better impersonate a digital "peer", or at least try to produce similar behaviors, our solution acts according to three different domains: *Innovation*, *Objective* and *User Map*. Because more than one domain may be taken into account at a given time, we are bound to find discrepancies between the suggestions it presents and user expectations. These may sometimes be perceived as unwanted, however, we must keep in mind the ultimate purpose of this work is to explore the ability to foster creativity, in this case by allowing multiple, potentially useful suggestions to emerge. Much like a session of brainstorming, sometimes the wrong idea at the right time makes all the difference in the world.

That being said, the designer is still given all the means to guide or orient the type of content generated by the program (much like asking a human partner for a suggestion on a more specific topic) rather than directly controlling the programs execution in a more traditionally predictable way. For instance, should the level designer feel more receptive to a wider array of suggestions, with potentially district traits, he can make these domain guidelines more lax. If, on the other hand, he feels like exploring only a more specific type of suggestions then these guidelines need to be set up in a more strict way. Say, for instance, allowing only suggestions from the Innovation domain. In the end, it's still up to the designer to choose whether a particular suggestion in worth exporting or even consider. If not, disregarding the displayed suggestion is always a possibility. Figure 4.1 represents an example interaction between a level editor, where a designer performs any creative actions, and our solution. In theory, this concept is generally applicable to other level editors, other than the one we chose.

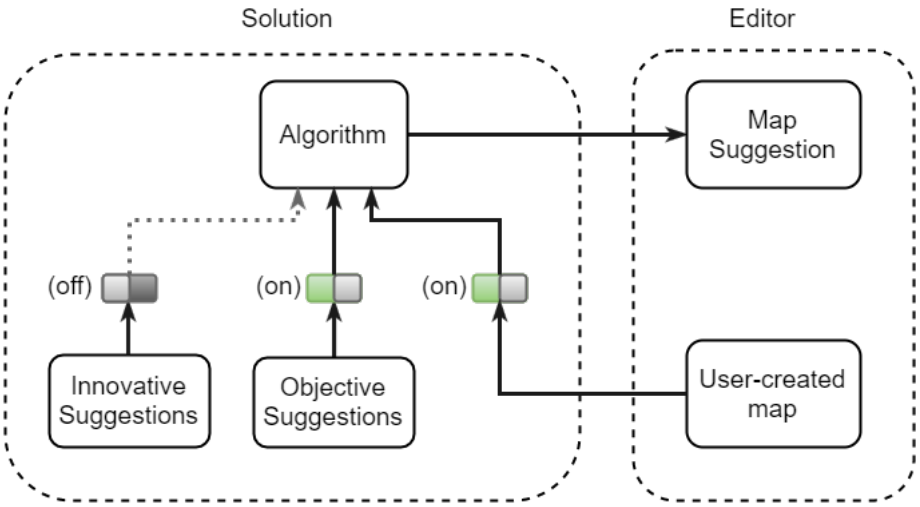


Figure 4.1: Example interaction of the solution with a level editor

4.2 Solution Model

In this section we present an overview of the implemented solution, describing it in a more general way where we identify its main features, how they relate with each other, the designer, and with level editor we chose.

The developed solution, named Editor Buddy, consists of a GUI based application, where Figure 4.2 represents its final version. Its primary goal is to foster creativity during a level design activity by presenting visual hints and suggestions iteratively, alongside the designer's work. Its interface serves two purposes: it provides the level designer with visual information as a way to stimulate his creativity and a way to guide the application's behavior towards generating more or less specific content. The Editor Buddy behavior is defined by two genetic algorithms as well as input from the designer's current work. Generated suggestions are then displayed to the designer in the interface using a 2D preview. This behavior can be configured by the designer using the available interface controls. The designer can also easily export the displayed suggestion, or parts of it, to his current level.

4.2.1 Interface

The Editor Buddy User Interface (UI) consists of these main components:

- Controls for defining the application's behavior. These include a standard and expert modes which can be toggled by the designer
- A 2D canvas where the program's generated content is displayed
- The canvas can also be used to perform a selection of the suggestion the designer wishes to export
- A set of buttons which can be used to interact with the generated suggestion
- Highlight setting which help in identifying the delta between the suggestion and the designer's level

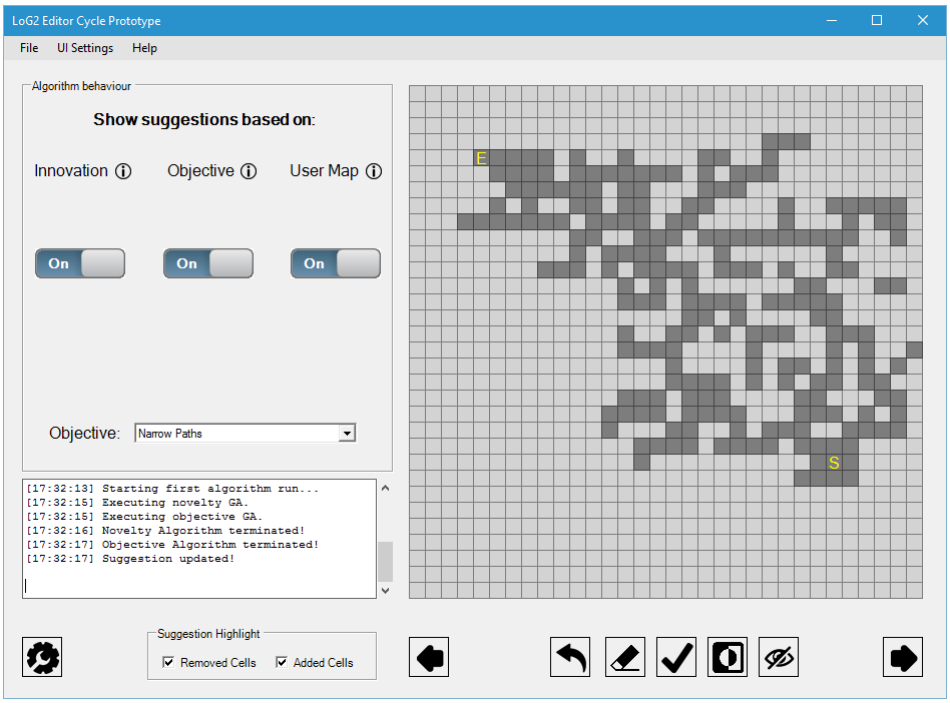


Figure 4.2: Final Editor Buddy interface

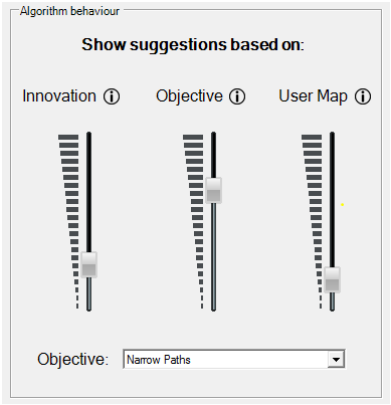


Figure 4.3: Editor Buddy expert mode controls

Controls are shown in the form of toggle switches as the standard mode. Alternatively, there is an expert mode, illustrated by figure 4.3, where switches are replaced with sliders, to provide a way to fine-tune the Editor Buddy behavior. Specific to the Objective domain, the designer can choose a particular behavior using the appropriate combo-box element named “Objectives”.

Suggestions are displayed in the 2D preview canvas. Walkable tiles are displayed in dark grey, while non-walkable tiles are displayed in light grey. Additionally, the Editor Buddy can detect where the start and exit points of a level are and identify them with the character “S” for the starting location and the character “E” for the exit.

Initially, suggestions remain hidden and the designer can expose the hidden part within an area by drawing a selection over them in the preview canvas. To do so, we can start or increment a selection using the left mouse button while dragging over the preview canvas. Similarly, to deselect we use the right mouse button. This selection can also be cleared using the clear selection button or inverted using the invert button. However, the designer can peek at the hidden part of the suggestion by pressing the visibility button, which shows a faded version of the entire suggestion.

After the selection is created, the designer can swiftly export the content inside the selection by using the export button to apply it to his level. These changes can be undone using the revert button. The left and right arrow buttons, which sit in opposite sides of the button row, navigate through the suggestion historic, allowing the designer to revisit previously displayed suggestions.

The interface also contains two check-boxes used to highlight the changes proposed by the suggestion. Map tiles which the suggestion removes are displayed in red, and those which the suggestion adds are displayed in blue.

4.2.2 Behavior

It was our main concern, for this project, that the Editor Buddy acted as a digital "peer" during the whole level design activity, iteratively suggesting eventual improvements or alternatives to the designer's work. It attempts to do so by using two GAs and the state of the designer's level layout at a given time. The algorithms are executed on separate threads and evolve randomly generated populations of individuals for two distinct purposes. The first one being innovation, where the fittest individuals are those which are most distinct from the designer's level. The second one being a particular objective such as narrow halls or small rooms, as well as a generic objective, valid for both algorithms, defined within the context of the tasks performed during the evaluation phase which is the creation of a valid path between start and end points. Additionally, a third element is taken into account when generating suggestions, the level created by the designer.

These suggestions, however, are tightly related to predefined task objectives. Meaning the generation of content related to design patterns other than the ones implemented for the purpose of this work (narrow halls or rooms) is unlikely at the very least. We chose to do so in order to quickly assert the feasibility of having the computer and the human designer cooperate towards these predetermined goals which, in one way or the other, converge towards similar map layouts. Also, because we are only focused on the layout itself, and not the part it will play in later stages of development, for instance when adding monsters or puzzles to the map, we are not immediately bound by those specific constraints related with said particular stages.

Essentially, we can ensure the unique work of each algorithm by allowing them to evolve their population individually but we also experiment with breeding individuals from both populations in order to spawn potentially more well rounded individuals, should the designer need them. We achieve this by introducing a percentage of chromosomes representing the designer's level and a percentage of chromosomes from the innovation into the objective algorithm prior to the start of a new algorithm run.

The designer can take advantage of this flexibility to inhibit some of the application's behaviors if, for instance, he wishes to fine tune its level. He can lead the Buddy into generating more restricted type of suggestion by toggling the available switches (or sliders, for the expert mode) labeled Innovation, Objective and User Map. While enabled, these controls allow the application to take into account the type of content it generates for the corresponding label. For example, by setting the Innovation switch on and the Objective and User Map off, the Editor Buddy suggestions are chosen only amongst those in the innovation algorithm's population.

Suggestions displayed by the Buddy are direct response to a designer's interaction with the level editor used. Each one triggers a new evolutionary run in the algorithms which in turn results in an update of the displayed suggestion. This way we further emphasize the aforementioned turn-taking approach we wished to bring to the level design activity. On each iteration, the algorithms gradually evolve the previous population, effectively refining its suggestion alongside the designer. This cycle results in a cooperative level design activity in which suggestions generated by the Buddy may influence the decisions of the designer which in turn may end up provoking a change in the type of suggestion

displayed. We will detail this interactive cycle further below.

4.2.3 Editor integration

The developed application is not a standalone program but, instead, works closely with the level editor we chose, to be formally introduced in the next chapter.

Each new level project needs to be first created using the level editor by specifying a *name*, *creator* and a *project location*. After a project has been created, the Editor Buddy can then be launched and the designer can open the directory of the newly created project in order to start exploring suggestions while working on his own content.

All level creation and editing made by the designer is done in the level editor side. The only exception being when and if the designer wishes to replace his work, or a part of it, with the Editor Buddy's suggestion, using the available interface tools. The Editor Buddy handles the automatic saving of the level, so when there is anything new on the designer's end, the Editor Buddy's algorithms begin a new iteration, taking into account the current configuration. When exporting changes to the designer's level, the procedure is somewhat the inverse. We take the selected suggestion's 2D layout and write it in the appropriate format so the Editor is able to load it. The Editor Buddy also handles the automatic reload of the updated map file, so the level editor view reflects the performed changes.

Summary

By the end of this chapter, it is our goal to have provided a clear overview of the the proposed solution model, including its different components and their respective elements and functions. Regarding the applicability of this model, presented in our diagram, we compromised with a specific level editor, however, we theorized it can be used with similar others.

Chapter 5

Solution Implementation

In this chapter we detail how our solution is implemented, starting by formally presenting the Legend of Grimrock 2 (LoG2) game and its Level Editor. Secondly, we present the implementation of our solution, where we point out problem specific constraints regarding our particular approach, the genetic algorithm framework we used in our implementation and required modifications performed by us. We also detail how the execution cycle of our solution unfolds as well as the more technical aspects of this cycle. Next we present the used algorithms as well as their specific parametrization and a brief explanation as to why they were chosen. In the end of this chapter we include a section dedicated to describing several design choices to help substantiate decisions we took during the development stage.

5.1 Legend of Grimrock 2 Level Editor

Legend of Grimrock 2 [1] is a dungeon crawling role-playing game where players attempts to progress through monster ridden and puzzle filled dungeons and mazes. Additionally, Legend of Grimrock 2 includes a level editor, which players or level designers can use to create their own content.

In the Editor, the user is given a 2D canvas which can have a maximum size of 32 by 32 tiles. He's also given three main tools to edit the layout of his level as well as its content. Arguably the most important one is the tile-editing tool. With it, designers can easily shape up the layout of their level thanks to a wide array of floor and wall tiles which, themselves, can be more or less thematic, such as forest or beach floor and wall tiles, in addition to the more common stone floors. In the context of our work, designers exclusively worked with the tile-editing tool, in order to assemble their level's layout.

The remaining tools, a pointer and object-editing tool, are used to complement the 2D tile based layout, previously created using the tile-editing tool. By adding different map elements such as door, levers, teleporters or even monsters to their level, designers can quickly build a sizeable and rich adventure with these capable yet simple tools. However, this step of procedural object placement fell out of the scope of our work and mention it in our future work section.

Other interesting features of the LoG2 level editor include a way for the designer to preview how his level plays out inside the editor itself, without needing to open it in the game. Also, by using the described tools, the designer can create several layers of levels with different sets and themes and, when he's done, he can export his creations and share them with other users.

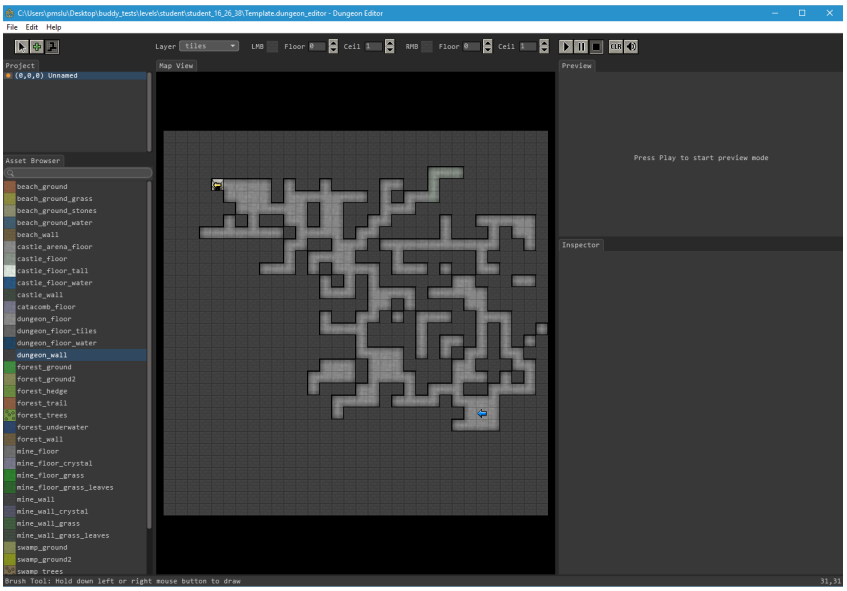
In regards to this work, a crucial feature of the level editor is how it handles the user projects. User generated content is simply stored in plain text files, which immediately prevents a whole lot of problems when integrating it with other software for the goal of content generation. Essentially, this means we can

[1] Almost Human Ltd., 2014

write content, in a LoG2 syntax-friendly manner, to a particular file and immediately reload our project to reveal it in the editor. Similarly, any change performed to the our project and consequently save it will be immediately accessible in the plain text file. Nevertheless, we did need to learn how to parse and interpret these project files which did not prove to be difficult. Figure 5.1 illustrates a screenshot of both Legend of Grimrock 2, figure 5.1a, and its Level Editor, figure 5.1b.

Finally, we can understand why Legend of Grimrock 2 and its level editor seemed like a interesting choice for this work, given it visual representation and creative potential, in addition to being *mod* friendly, facilitating the task of developing an add-on solution for it.

(a) Legend of Grimrock 2 game



(b) Legend of Grimrock 2 Level Editor

Figure 5.1: Legend of Grimrock 2 in-game screenshot and Level Editor

5.2 Solution Implementation

This section addresses the more technical aspects of the proposed solution, design decisions, limitations and respective workarounds regarding the Editor Buddy behavior, its interface and its interaction cycle with the designer.

5.2.1 Genetic Algorithms

Before we get into further details, let us first to go over a few notions of how genetic algorithms work. Evolutionary algorithms, more specifically genetic algorithms, are based on the survival of the fittest theory. This means only individuals with the right traits will be able thrive within a population, carrying on and generating offspring which can have the potential to be even fitter than their predecessors. Now, in living organisms, this goes down to the genetic level, where individual traits are represented in the form of chromosomes. An organism's full hereditary information and how that translates into the visible traits of that organism is called genotype to phenotype mapping.

The way GAs implement these concepts, so they can be interpretable by computer programs simulating this method, is by translating potential solution traits into a data structures, the chromosome of an individual. This individual then reproduces with many others, for several generations, in the hope of generating offspring who can provide a solution for a given problem. Termination for an algorithmic run can be predefined by, for instance, a fixed number of generations or can be set to occur under more specific circumstances, such as after reaching average population fitness or only when the optimal solution has been found.

Regarding the process itself, in order to perform genetic recombination, as well as simulating naturally occurring processes such as mutation, genetic algorithms rely on generic operators such as crossover, mutation and elitism.

The crossover operator handles the recombination of genetic material between two previously selected parent chromosomes. Offspring generally carries out with approximately half of the genetic material of each parent. Frequent ways of performing this operation include a single-point or a two-point crossover. Mutation comprehends and simulates naturally occurring phenomena in which a minimal change in an individual's genetic information can result in a larger impact when translated into, in this case, their potential problem solving traits. In an algorithmic standpoint, tries to avoid the population getting stuck on local maxima. Frequent types of mutation include flip bit mutation where a single gene in inverted or the swap mutate where two genes are swapped between their places. Elitism is an operator which ensures the most fit individual, or more than one, are carried over, copied, with no change from mutation or crossover. Parametrization needed for Elitism is only the percentage of individuals who are kept unmodified. These operators perform the required genetic material exchange which, depending on how they are parametrized, can generate more different, interesting individuals from their parents.

Arguably the most important feature of genetic algorithms is the fitness function. Whereas operators simply state how and how much genetic material is exchanged, the fitness function is what evaluates which individuals are valuable and which are not. In a sense, the fitness function is what dictates where and how the population evolves, and a good fitness function can be the difference between a poor performing genetic algorithm run and a good one.

As a fundamental part of our solution, genetic algorithms guarantee the generation of suggestions, innovative and objective oriented each with its own fitness function and parametrization, as well as based on user level layout, by evolving individuals which represent the layout of levels.

5.2.2 Problem specific constraints

Map to chromosome representation

Remembering the work of Liapis *et al.*[17] on map sketches as a way to abstract from complex game details, let us assume we abide by the map sketch philosophy, albeit with a couple adaptations. Given the nature of the game we based our work on, we required a way to represent a 2-Dimension (2D) complex data structure into a 1D data structure, our chromosome. The game maps from the game LoG2 consist of a 2D grid of with a maximum side of 32 tiles where the user can specify the type of floor to use for each one. Tiles can manly consist of two types: **Floor** tiles which can be walked on and **Wall** tiles which can't be walked on. The Editor canvas is where the designer builds his level layout using **floor** and **wall** type tiles. The Editor Buddy is capable of handling all types of **floor** and **wall** tiles the designer chooses to use in his level. However, in the context of this work, the Editor Buddy specifically uses the **"dungeon_floor"** type for walkable tiles and the **"dungeon_wall"** type for non-walkable tiles. Another interesting subject is the way we translate an individual's genetic information into a 2D map and vice-versa in our particular case. Technically, it's a simple matter of representing a 2D map as a 1D data structure. We do so by concatenating all map lines, like illustrated by figure 5.2. Similarly, when decoding an individual's genetic information, we only need to take an χ amount of genes, where χ is the width of the canvas, and assemble them line by line in a 2D array.

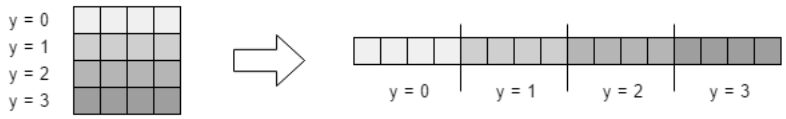


Figure 5.2: 2D map to chromosome translation used by the Editor Buddy algorithm

Custom GA operators

In this sub-section we turn our attention towards the framework used for our algorithms as well as the modifications needed to adapt the default implementation to accommodate our particular needs.

We briefly addressed above how we need to translate the layout of generated maps into chromosomes so they are able to be used in the genetic algorithm runs. Now, assuming we require our genes (map tiles) to hold meaningful information, instead of just using a simple binary representation of a map, i.e. 1 for "floor" or walkable tiles, 0 for "wall" or non-walkable tiles, we chose to encode each chromosome gene as a class object value. In other words, there is more relevant information that a gene can hold for a specific tile of a map than just its type.

This way, in spite of going against the basic principle of map sketching, we provide a more future proof implementation since information such as puzzle elements or Non-Playable Characters (NPCs) and monsters can be stored individually for each gene (a map tile), although at this time we worked solely on tile placement. Still, we required a way to store useful information in the context of this level design activity, such as start and end point information or tile type.

The genetic algorithm library used for the Editor Buddy's genetic algorithm implementation is the Genetic Algorithm Framework (GAF) written by John Newcombe [28]. This framework was adopted because it provided a mature and stable genetic algorithm implementation while offering all the features we required and also enough flexibility to modify some particular aspects. Such modifications were performed at the genetic operator level, namely for the crossover and mutation operator. Before we look into these with more detail, it is important to introduce some notions which led to these customizations.

Crossover operator

We have already addressed above the need to transcribe a 2D map into a 1D structure in order to make it possible for the Genetic Algorithm Framework to handle it as a chromosome. It was our understanding conventional crossover methods were perhaps inadequate to perform the kind of meaningful genetic combinations required in the context of 2D level design. In other words, regular single-point or two-point crossover did not produce the kind of localized variation we wished to see in the context of 2D level design.

With this in mind, we decided to try a *modified crossover* operator which would help us keep, or emphasize rather, 2D visual context. This was achieved simply by translating a given section of the 2D grid into a 1D pattern using a 2D shape, as shown in figure 5.3. This pattern is then used in the genetic recombination process as the crossover points between two chromosomes, as shown in figure 5.4.

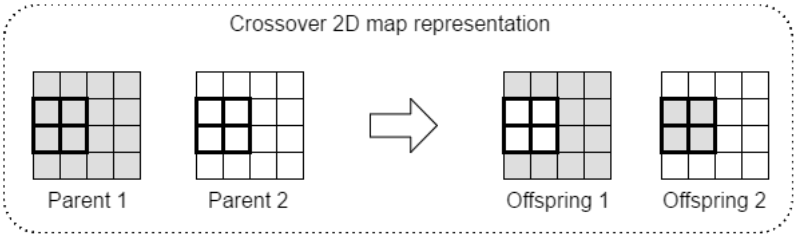


Figure 5.3: Example of custom crossover applied to a 2D map

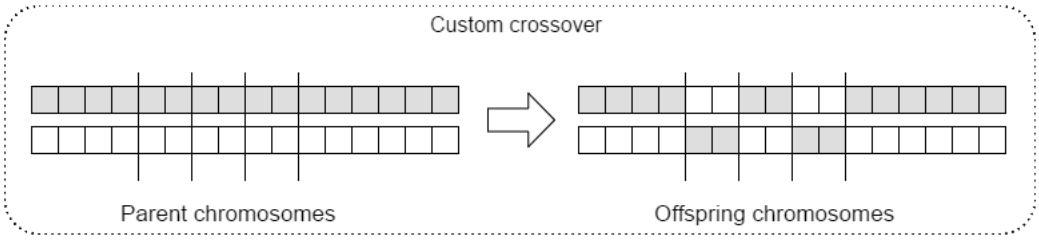


Figure 5.4: Example of custom crossover applied to a pair of chromosomes

We experimented with different sizes and 2D shapes for use in this crossover method, such as **2x2**, **3x3** and **4x4** squares, a **3x3** circle and a dispersion pattern, which basically picks random tiles within a certain area. We did not conduct, however, any dedicated study to precisely determine which combination was best. Still, we have performed several tests to try and understand which one resulted in the most interesting performance. In this sense, we chose to use the **3x3** square pattern. Regardless of shape details, this allowed for potentially valuable genetic traits to be preserved, although at a highly localized level. As a side note, section dimensions shown in figure 5.3 are merely for demonstration purposes. We confirm the actual size of the section is a **3x3** square, just as mentioned above.

Mutation operator

On the Mutation operator level, there were a couple limitations with the provided implementation from GAF [2]. The main limitation was the fact the provided swap mutate [3] operator performed the mutation at the object level, meaning every trait of one gene would be swapped with that of another. This was partially unwanted as it became evident when the location of start and end points began shifting between evolutionary runs when they were not supposed to for the duration of the activity. The solution was the

[2] These issues arose due to the particular needs of this work as opposed to poor implementation decisions in the framework.
[3] Swaps the position of two genes in the same chromosome (individual).

implementation of a custom mutation operator which would only swap tile type data between genes while leaving all other gene data unadulterated.

One operator, however, remained untouched which was the Elitism operator. Elitism allows for a small percentage of individuals to carry over the next generation without any modification. Since its behavior was correct in the context of our problem, there was no need for further modification.

5.2.3 Editor Buddy execution flow

In this subsection we detail how the Editor Buddy life cycle unfolds. We address when and how it begins, relevant information regarding algorithm interaction and how the displayed suggestion is chosen.

The following diagram, illustrated by figure 5.5, represents the Editor Buddy execution flow. Algorithm execution is comprised of two different stages: An initialization phase, where algorithm populations are first generated and a post-initialization phase, or suggestion phase, where the previous algorithm populations are iteratively evolved and from which provide the suggestions presented to the designer. For each modification the designer makes to his level's layout, creative or destructive, triggers a new evolutionary run in the Editor Buddy algorithms, guided by the interface configuration.

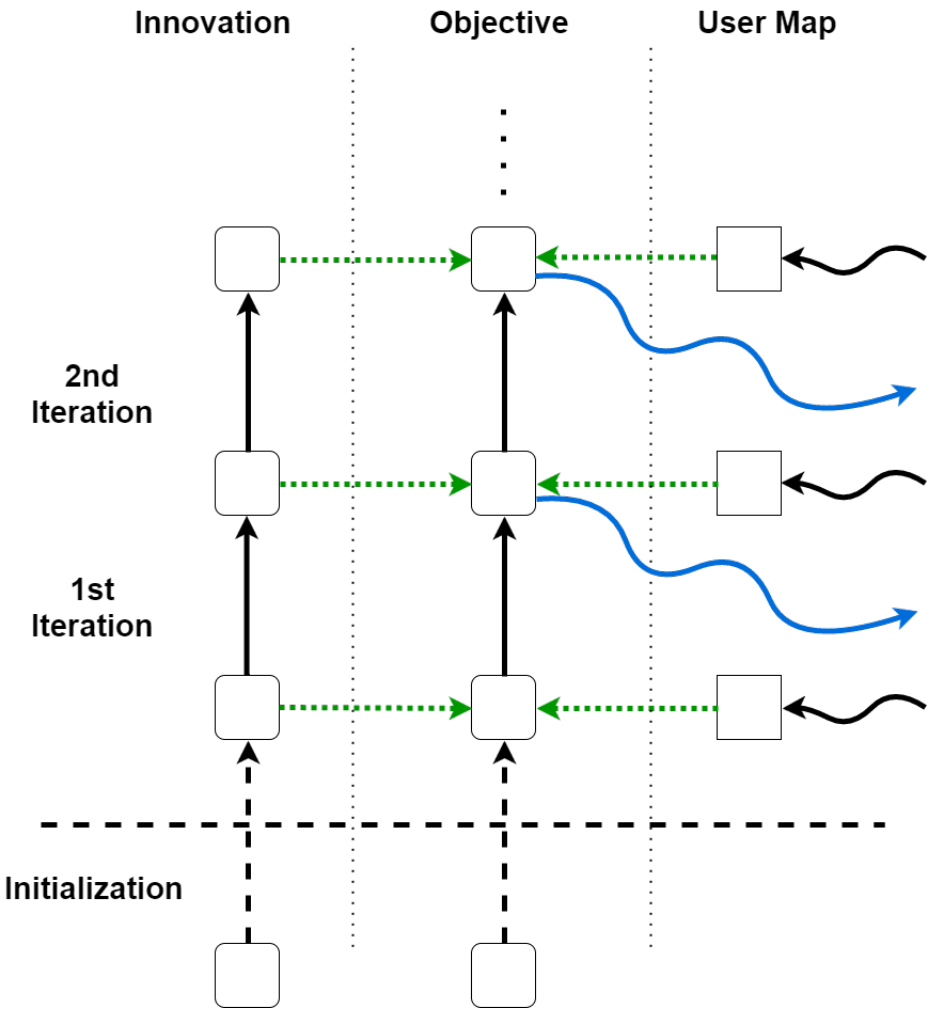


Figure 5.5: Editor Buddy algorithm execution flow diagram

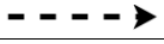
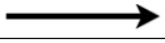
Element	Description
	Individual chromosome
	Population of inviduals
	Initialization run
	Algorithm run
	Chromosome transfer
	Designer input
	Displayed suggestion

Figure 5.6: Flow diagram's element captions

Initialization phase. The initialization phase consists of both algorithm's initial population generation. It occurs initially when the Editor Buddy is first launched as well as when the designer changes the behavior switch/slider configuration or even the current objective on the objective combo-box. Further below we detail why this is done.

User Input. User input, identified as the User Map element in our flow diagram, is the third key-element of our solution. However, it is independent from both algorithms. Its development over time is sole responsibility of the designer, whereas the Editor Buddy merely interprets it and reacts accordingly, when configured to do so. The only exception being when the designer actively uses the export button. Even then, those decisions can always be reverted using the revert button. A new interaction cycle begins when the designer performs some type of constructive or destructive action on his level. This means the new user map is passed along to the Editor Buddy to generate the next suggestion.

Configuration and Population / Chromosome transfer. After the designer's action triggers a new algorithm execution, the first thing the Editor Buddy performs is a check on the state of the behavior switches or sliders. According to how they are configured, the pool of potential suggestions will be composed of different types and/or amounts of individuals. Having the Innovation or Objective switches or sliders turned on means we have our two algorithms working at the same time. This is why we are exposing the Innovation algorithm's population at the end of each run. A portion of that population is placed in the initial population of the Objective algorithm. Therefore, it's the Objective algorithm who is ultimately responsible for selecting which individual to display. If the Innovation switch is enabled, in the Editor Buddy's UI, a percentage of the Innovation population (top % individuals) is selected to be part of the Objective algorithm's initial population for the next iteration. Similarly, if the User Map switch is enabled, a percentage of chromosome clones representing the designer's level is introduced into the Objective population for the next iteration.

Algorithm execution. After the algorithm populations are transferred according to the Editor Buddy configuration, both the Innovation and Objective algorithms are executed independently. Innovation evolves towards the most distinct population from the user sketch and Objective evolves towards generating individuals who respect task constraints selected in the objectives combo-box.

Displayed suggestion. When the algorithm execution is complete, the new best suggestion is then displayed back to the designer in the Editor Buddy canvas. Regarding to which individual is displayed to

the designer, it's always the one who better adheres to the current global task objectives. For instance, even if the Innovation switch is the only one on, the individual who best respects the currently selected goal, be it narrow halls or rooms, is selected amongst innovative-only individuals. This way we ensure suggestions are, up to a certain extent, coherent with the objectives of any given task as long as those constraints are formalized in the implementation.

Algorithm re-initialization. One important aspect to note when changing the application's behavior switch configuration is the fact the algorithms populations are reset in order to ensure the correct pool of potential suggestions. This means a new initialization run has to take place before any updated suggestions are generated. Take the following example: The designer was using all switches turned on for a few iterations and then decides to turn off Innovation and User Map. With no algorithm reset, there was a chance the next suggestion displayed was still a slightly modified version of the user map caused by a lingering chromosome from previous populations.

5.2.4 Algorithms

This subsection provides further details regarding individual algorithm implementation and their configuration parameters as well as any relevant substantiation as to why those decisions were taken.

Innovation algorithm

The Innovation algorithm evolves towards generating innovative content, the more distinct from the user's the better. We chose to include this behavior in our solution to ensure distinct content was always a possibility. This way, users who sought mandatory novel content could always be sure to find it with the contributions from this algorithm. At the map level, innovation means tile placement which goes against what the designer sketch reveals. These novel alternatives respect only the constraint of keeping the start and end points with a walkable tile type.

Population and chromosomes. In the initialization phase the initial population contains about 30 randomly generated individuals. Each chromosome contains, on average, 500 walkable type tiles which is half of the total amount of genes for each chromosome. Higher or lower tile type density at the gene level would, in theory, result in much poorer variety of game level topography after the genetic mapping. This provides a generous margin for genetic variation, since there is a relatively high chance that genetic recombination could result in a potentially meaningful outcome despite its small intervention. These observations, however, were not extensively tested, as it was not the main focus of our work but still we have experimented with several values for these parameters and found the aforementioned value as a good starting point for a future analysis. After the initialization procedure, on each consequent iteration, the previous population is evolved towards the maximum amount of innovation which is the opposite of the users' current design.

Objectives. Innovation in the standpoint of this algorithm means the layout most distinct from the designer's is the most valued. Because we are only focusing on tile-based diversity, it is relatively easy to evaluate which individual is the fittest in terms of novelty. At this stage playability constraints still play a role since we must at, the very least, guarantee start and end points have walkable tiles at their positions. Other than that, their value is later reviewed when placed on the Objective algorithm's population to be evaluated according to the current task's objectives.

Operators. Crossover parameters include 80% crossover probability using the implemented modified **3x3** 2D square shape custom crossover. For parent selection we chose to use Fitness Proportionate Selection [29], also known as roulette wheel selection. This selection type directly associates an individual's fitness to the probability of its selection. For our algorithm's replacement policy we chose

generational replacement [29] where the best offspring replaces its least fit parent. For our Mutation operator and its parameters, we use the aforementioned custom mutation operator using swap mutation which simply swaps relevant gene information between two genes. Mutation has a 15% chance of occurring for this algorithm. Elitism parameters include a 5% elitism percentage and prioritizes the fittest individuals. This means the top 5% fittest individuals for this population are copied to the next generation without being modified. These parameter values have not been target of an in-depth evaluation, nor was it in the scope of this work, however, we have performed a few tests regarding algorithm performance with several combinations of these values and have drawn a preliminary conclusion that these values led to a balanced performance of the algorithm.

Termination. The evolutionary run terminates after about 80 generations (this value will be discussed below), by the end of which the final population is kept in memory for the next iterations. A clone of this population is also exported to the Objective algorithm instance.

Objective algorithm

The Objective algorithm evolves towards a generating suggestions which contain evident design patterns, coherent with the objectives implemented and included in the objectives combo-box. Tasks in the evaluation phase were also designed with these objectives in mind. This way, we needed to have an algorithm which generated content towards respecting certain restrictions and, in the end, was visually interesting for the task at hand. For this, we chose the following parameters to configure the objective algorithm performance.

Population and chromosomes. In the initialization phase, similarly to the innovation algorithm, a 30 individual population is randomly generated with an average of 500 walkable tiles. These values were the same as detailed above, and the logic behind their choice is also the same. After the initialization stage, each consequent iteration evolves the previous population towards a specific goal as well as another sub-goal which the designer can easily set and alter at any given time. The primary goal of this algorithm is the selection of individuals that contain at least one walkable path between a start and an end point, both of which are predetermined and part of the initial template for both evaluation tasks.

Objective goals. The meaning of goals, here, translates into what features of an individual are better rated by the **fitness functions**. For this algorithm in particular, primarily, we valued the creation of at least one valid path from start to end points, in the context of the evaluation tasks, as well as one of the following implemented sub-goals:

- the emergence of narrow halls
- the emergence of ample spaces or rooms

Evolving towards the first sub-goal means the fitness function gives preference to individuals whose genetic material represents map layouts whose tiles have only one or two walkable neighbors. Evolving towards the second sub-goal means the fitness function gives preference to individuals whose genetic material translate into map layouts whose tiles have more than two (in this case three or four) walkable neighbor tiles.

Effectively, the fitness value calculated by this algorithm for each individual is the sum weighted fitness functions for each of the aforementioned goals: a viable path and one of the existing sub-goals, narrow paths or rooms. Other features contribute to the fitness of an individual, such as the average number of walkable tiles in a suggestion. This is more useful at later stages of the interaction because if the same solution keeps being evolved endlessly, particularly in the generation of rooms, we noticed and unwanted over-saturation of walkable cells. In this sense we define a target amount of walkable cells and penalize solutions which go over the predetermined amount.

As it stands, the creation of a valid path has a bigger weight than the generation of the remaining sub-goals. These goals and sub-goals, described in the tasks performed during the evaluation phase, were chosen to ensure, up to a certain extent, some type of convergence between the Editor Buddy behavior (generated content) and work of the designer, resulting in a minimum amount of cooperation towards common goals.

Operators. Crossover parameters include 80% crossover probability using the implemented modified **3x3** 2D square shape custom crossover. For parent selection we chose to use Fitness Proportionate Selection, similar to the innovation algorithm, and as for our replacement strategy we also chose generational replacement for the objective algorithm. For our Mutation operator and its parameters, we use the aforementioned custom mutation operator using swap mutation which simply swaps relevant gene information between two genes, just like in the previous algorithm. Mutation, in this case, has a 10% chance of occurring for this algorithm. Elitism parameters include a 5% elitism percentage and prioritizes the fittest individuals. This means the top 5% fittest individuals for this population are copied to the next generation without being modified. These parameter values have not been target of an in-depth evaluation, nor was it in the scope of this work, however, we have performed a few tests regarding algorithm performance with several combinations of these values and have drawn a preliminary conclusion that these values led to a balanced performance of the algorithm.

Termination. The evolutionary run of the objective algorithm terminates after about 80 generations, same as the innovation algorithm. Both the objective and innovation algorithm runs have the same, low, target value for its execution. In the following section we address the reason for this curiously low number as a way to provide timely feedback for the designer and explore our options.

5.2.5 Design decisions

There were several significant details present in the final implementation which had to be decided during the development stages. In one way or another, these were some of the main design decisions we had to make a commitment to and led to the final implementation of the Editor Buddy. In this subsection, we mention and discuss a couple of these decisions and what they meant in the context of this work and its outcome.

Global task objectives. In this work, we soon decided that our application would have to consider and respect the work produced by the human designer when proposing any type of content. However, we wondered how to produce content outside the scope of what the designer was doing. At a more abstract level, that meant defining a set of guidelines that would result in a more flexible behavior for the Editor Buddy. Effectively this meant we could explore additional solutions, beyond the limitations imposed by the designer's layout but still ensuring we did not generate content aimlessly.

In the innovation algorithm standpoint this makes less sense, since our suggestion is ultimately coming from our objective algorithm, like we state previously in this chapter. The objective algorithm is then the main intervenient in this process. It helps orient individuals from the innovation population, its own population and the input from the designer's work. Our reasoning behind this was that co-creative activities between humans often go through a phase where both parties share each others current goals and may end up making a commitment between both points of view. In this sense, we saw this as a way to remove the need for such an event and chose to provide the both the computer and the human designer with a set of topological features which both would have to respect. This guarantees, up to a certain extent, both parties' creations converged toward similar outcomes, increasing their chances to reuse each others' feedback as useful creative stimulus for their consequent creations.

Generated content context. Closely related to the previous subject of global objectives, there is also

an important implementation decision concerning content generation. Because we outline topological guidelines for the Editor Buddy to abide by when generating its suggestions, it does so by considering the whole possible map area. It does this in order to guarantee the suggestion makes sense, as a whole, to the designer, at the same time it obeys the global objectives.

A disadvantage of this may become evident as soon as the designer performs a selection of the suggestion, exposing it, and comes across a partial suggestion which may or may not feel adequate in the context of that selection in particular.

Suggestion quality vs execution time. An important subject in our work is how we balance the trade-off between algorithm execution time and quality of generated content. This balance was be fundamental the end experience the Editor Buddy would provide. In the context of a human-to-human co-creative activity, we generally expect to get feedback or comments relatively quickly. In this sense, there was only a reasonably limited amount of time to perform the execution of two evolutionary algorithms which, Smith *et al.* [2] underline, are great for generation of offline content, but may behave not that well in situations which require almost a real-time response. We already addressed why we chose this alternative in spite of these observations.

During our development phase, this was one of the details we had to experiment with. The value used in both our GAs was found to provide the most efficient commitment between execution times and quality of results. A higher value did not exactly translate into better results, due to diminishing returns and a lower value would produce results with not enough quality for what we were expecting. Although results are not optimal, their refinement takes place in the following iterations of both the algorithms. This, perhaps, better simulates what a real co-creative activity between humans unfolds and works well in the paradigm we are trying to portray. Nevertheless, because created content is consistently reevaluated and improved, it comes a time where it starts being increasingly difficult to achieve better results in the algorithm's standpoint. By this time we hope the intermediate human-computer interactions which lead towards optimal results are just as valuable for the designer.

Mutation rate. Regarding the abnormally high mutation rate on the objective algorithm in particular, we explain why and how it fits in the context of this work. All decisions made during algorithm optimization were always done so with the cooperative level design activity in mind which is something that, typically, consists of a slow paced, backtrack-filled process. However, interactions between human participants usually take little time to occur: idea exchange, minor adjustments, reverting a modification, all those have a relatively short response time. In the genetic algorithm standpoint this meant shorter evolutionary runs to provide similar reaction times. Otherwise the designer-AI interaction would become dull and, ultimately, the pace of the activity itself would be affected. These short evolutionary sprints, however, result in sub-optimal solutions, like mentioned above. In the end, these sprints performed better with higher mutation rate than that of longer, more crossover-dependent, traditional runs. Furthermore, this gradual optimization of previous solutions better reflects the actual process of creative tasks in humans, resulting in a refinement process alongside the work of the human designer.

Summary

By the end of this chapter we hope we were able to provide a deeper understanding of the several intervening parties, during the use of the proposed solution together with the Legend of Grimrock 2 level editor. On the LoG2's level editor behalf we presented its main features, design tools and how straightforward and flexible it is. Regarding the proposed solution, we attempted to provide our standpoint and motivation towards choosing evolutionary algorithms and their parametrization and, despite its

initial coarseness, how we feel content generated and evolved this way fits in the paradigm of the current work, as well as the remaining design choices.

Chapter 6

Evaluation

In this chapter we present our evaluation procedures, starting by the description of our preliminary evaluations, conducted during the development stages, and insights gained from these activities which resulted in changes to the interface and behavior of our solution. Secondly, and most importantly, we introduce and describe in greater detail how the evaluation of the final version of the proposed solution was conducted, which methods were used, presentation of results and our interpretation of them. Finally, a conclusion on the evaluation process regarding our key-findings and eventual improvements to the current implementation.

6.1 Preliminary evaluations

To validate and dismiss certain features of our first implementations we chose to perform a couple of informal preliminary evaluations. In this section we detail which key-changes were made and why, regarding the Editor Buddy user interface and its behavior during the development stage, after performing these two preliminary evaluations.

First preliminary evaluation

On the first preliminary evaluation the Editor Buddy interface resembled figure 6.1. At this stage the chosen controls were Knob controls. The reason behind this was to provide a more analog feel to the interface, even though user would be interacting with a 2D interface element. Selections were performed on the canvas using a modifier key. *Ctrl+Left Mouse button* would increment the selection and *Shift+Left Mouse button* would remove selection. To export the selection, a context menu would have to be accessed by *right-clicking* the canvas and selecting export. The suggestion displayed in the canvas, at this stage, would remain always visible. Regarding the Buddy behavior, testers noted the results of the Objective algorithm were lacking, which consequently led to a revision of its fitness function.

Changes

Knob controls were replaced afterwards when the entirety of test participants failed to acknowledge the ability to interact with these controls since they found these controls visually non-intuitive. The majority of participants found this method to be cumbersome and a new way to create the selection was implemented using the left and right mouse buttons to add or remove the selection, respectively. Because participants found the whole suggestion display a bit overwhelming this was changed into a hidden suggestion at first to avoid over-influencing the user. A new set of buttons was created to replace

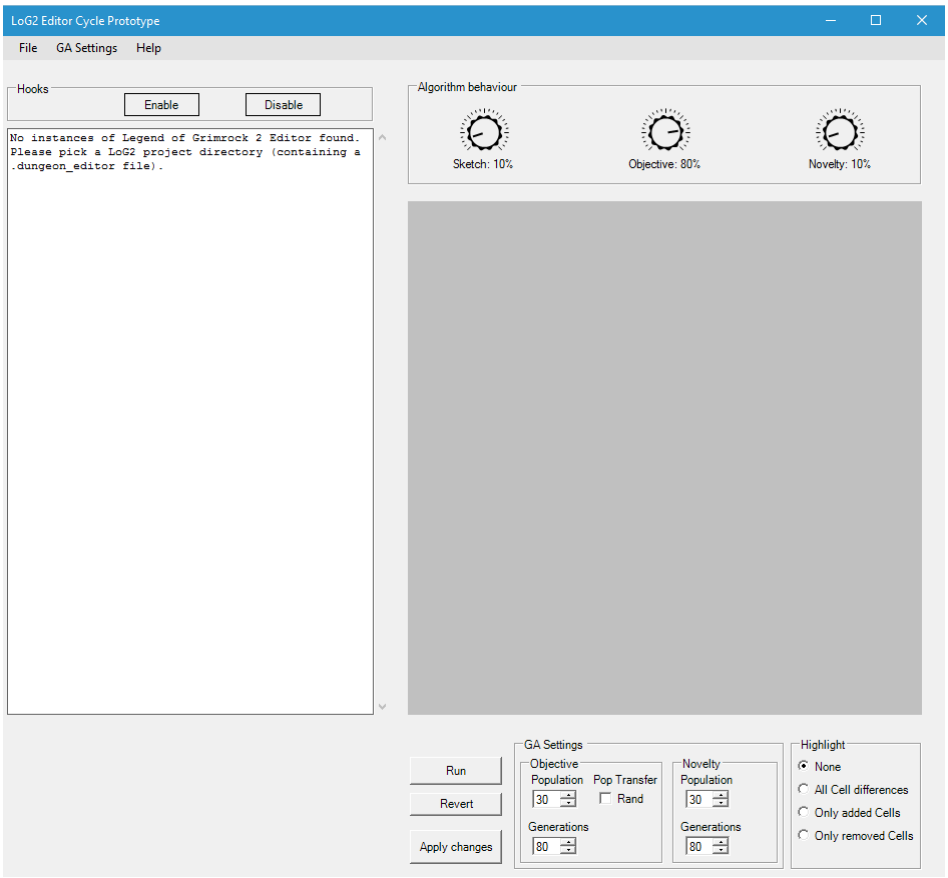


Figure 6.1: Interface used in the first preliminary evaluation

the many functions of the context menu such as the export, inverse and clear functions. Additionally, the revert and visibility buttons were added as well as a couple of buttons which allowed the user to go through the suggestion history.

Second preliminary evaluation

After changes from the first preliminary evaluation were implemented, the Editor Buddy interface resembled figure 6.2.

Before the second preliminary evaluation, the Knob controls were replaced by a different type of sliders. Furthermore, they were moved to the left side of the canvas and its size was increased in order to show higher visual relevance and avoid the same issues as before. To ensure a better understanding of the Objective algorithm, a new “ComboBox” item was added just below the slider controls. This contained a short description of the current objective being followed by this algorithm. At this time, only one objective was implemented since there was only one task planned to be performed on the final evaluation. A new tile highlight option was added, to allow the designers to turn that visual hint on or off.

Changes

After this second preliminary evaluation, a smaller number of adjustments was required. We decided to simplify the slider controls and start by presenting toggle switches as our default mode in the final version. Optionally, an expert mode would be available for users who felt the need for finer tuning of algorithm behavior using the slider controls. The main reason for this change was the struggle when first contacting with the controls. Initially, participants would still have a hard time associating behavioral changes to minor control adjustments. By presenting an “on or off” mode first, this provided an easier

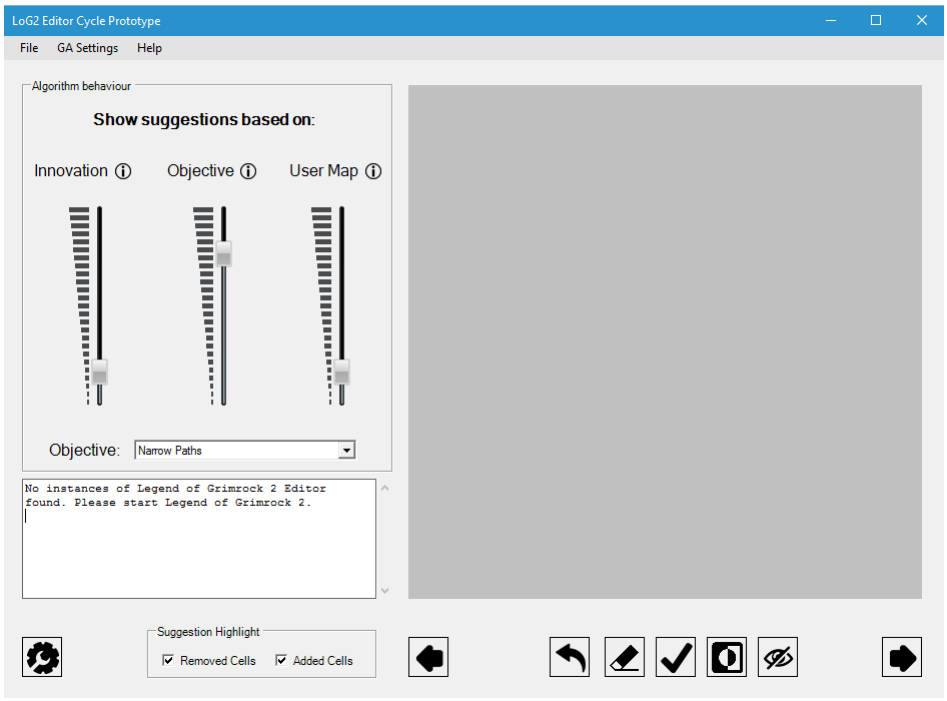


Figure 6.2: Interface used in the second preliminary evaluation

introduction to this "cause-effect" when using the sliders in expert mode. Additionally, a second objective was listed in the Objective "ComboBox", the generation of rooms within a map. This was needed because we decided to create an additional task for the final evaluation with a different goal layout. This way, we are able to convey the Editor Buddy's ability to evolve towards more than one level design goal using its Objective algorithm.

6.2 Solution evaluation

Study goals

The main goal of this study was the evaluation of the Editor Buddy utility as well as its efficiency. We define utility, in this case, as the ability to contribute, direct or indirectly, with useful content. It remains, however, at the designer's discretion the definition of usefulness. This brings us to the second topic of evaluation, the application's ability to be configured in such a way as to produce coherent and useful content, according to the needs or desires of the designer at a given time. In other words, evaluate if the UI controls are flexible enough to provide a configuration which generates adequate content at all times, as well as evaluating the ease of use of such task.

Study Description

The target of evaluation was the developed software application in the scope of this MSc thesis. This study was conducted by myself on more than one occasion and with different participants. The first and second moments of evaluation took place at IST-Taguspark on April 22nd and 27th. The third moment of evaluation took place at Bica Studios on May 5th. For this study, we looked to find participants who had distinct professional experience and computer science students who had little to no experience in level design, but were at least familiar with concepts such as artificial intelligence, user interface and what

level design in games is. Professional participants had significant backgrounds, which is something that proved useful in identifying design flaws in the Editor Buddy, from a professional standpoint.

6.3 Methods

The description of the methods used in this study were based on the methodology and guidelines described in “Qualitative Data Analysis: A Methods Sourcebook” [30].

Study design

The type of study conducted was a **collective-case study** performed on a small group of participants for relative short periods of time. First, participants were shown a video introducing the most important features of both the LoG2 Editor and the Editor Buddy, after which they were able to experiment with these tools and ask questions regarding their usage in a tutorial stage. After this initial contact, participants were then asked to execute the two official tasks, using the Editor Buddy application to support them during level creation using the LoG2 Editor. Their objective was to design two playable levels with distinct layouts, given a few constraints regarding the sub-goals of each task. Participants were given a template level containing a few walkable tiles and predefined start (blue arrow icon) and end points (yellow arrow icon), illustrated by figure 6.3. Additionally, participants were asked to assume these sessions took place during the development stages of a videogame in a fictitious company, simply to provide an interesting and contextually coherent way to motivate them as well as abstract them from the evaluation itself.

The first task goals were:

- The resulting level was required to have **at least one valid path** from the starting point to the end point. Without this, the level would not be playable
- The level layout must contain **narrow halls**, like a maze, since monsters with a charge attack will be placed in the level, later during development

The second task goals were:

- The resulting level was required to have **at least one valid path** from the starting point to the end point. Without this, the level would not be playable
- Their level layout needed to be altered to have **rooms** connected by halls, because new monsters would be placed on the level and they were too big to fit narrow halls and would, instead, reside inside these rooms

In both tasks participants were also asked to create interesting levels, not just a straight line from start to finish, for example a couple of detours from the path connecting the start and end points.

They would be finished when they demonstrated enough satisfaction towards their final product and at least the primary requirements were met. However, the second task was immediately introduced right before the participants were able to completely finish the first one, **without** starting from a new template level. This was intentional on our part so we could evaluate on which occasion the Editor Buddy worked best, if starting a level from a blank template or from existing content.

A time limit was not contemplated, however, duration of these activities was expected to fall within reasonable values, if not, we would have to interrupt it. We first expected a session length of 15-20 minutes for each participant subject on average, plus another 10 to 15 minutes for data collection.

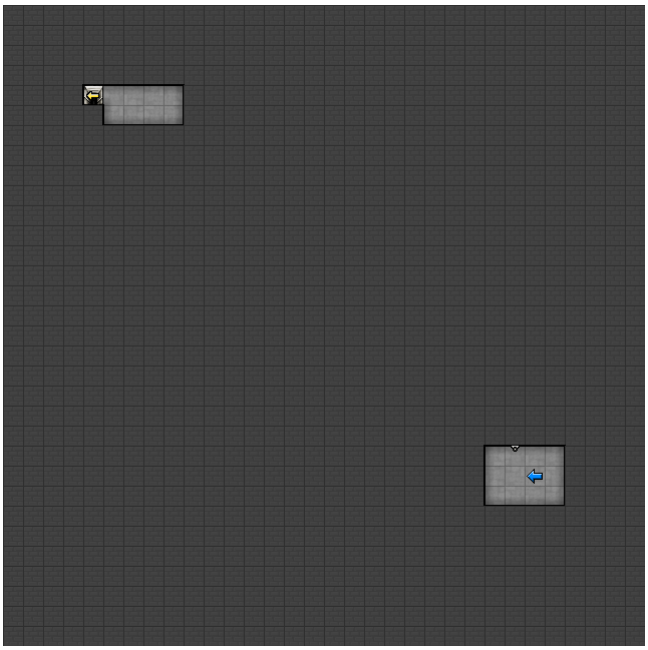


Figure 6.3: Template used in evaluation tasks

Sampling method

Two types of participants were involved in this study: professionals and non-professionals. Sampling method was mainly a purposeful sampling, where on one side we had professionals or expert individuals and on the other side we had non-professionals or inexperienced individuals. This allowed for identification of prominently lacking or redundant or obsolete features. The professionals were level and game designers at Miniclip and Bica Studios and were asked to participate in the co-creative level design tasks mentioned above. Both the professionals and the non-professional participants had no previous contact with any version of the Editor Buddy.

Data collection methods

Data collection methods used included a **questionnaire**, **participant observation** and a **structured interview**. For quantitative data collection, we used a linear scale questionnaire in order to roughly draw a measure of the level of usability for the Legend of Grimrock 2 Editor and the Editor Buddy interfaces, as well as the amount of user expectation versus generated content discrepancy. This way we can quickly pick out eventual inaccuracies when analyzing results from observation and interviews and cross-reference it with these results. Questionnaires were handed to participants before the start of the official tasks and after they could experiment with both the Editor Buddy in a tutorial task.

Observation was conducted in the form of participant observation and screen recording during the entire test. All participants were notified of this intention and granted us permission to do so. We chose screen recording because we were interested in the participant's actions during level design activities including their interaction with both the in-game Editor and the Editor Buddy instances as well as any type input in-between, which may or may not have been disregarded according to its relevance.

After participants were finished with these tasks, a structured interview was conducted. The goal for these interviews was to draw out patterns from common concepts and insights regarding the personal experience of each participant with both the interface interaction and the generated suggestions. Interviews were recorded with previous consent from each participant.

The testing setup consisted of a single computer running an instance of the Legend of Grimrock 2 In-game Editor alongside an instance of the Editor Buddy, illustrated by figure 6.4.

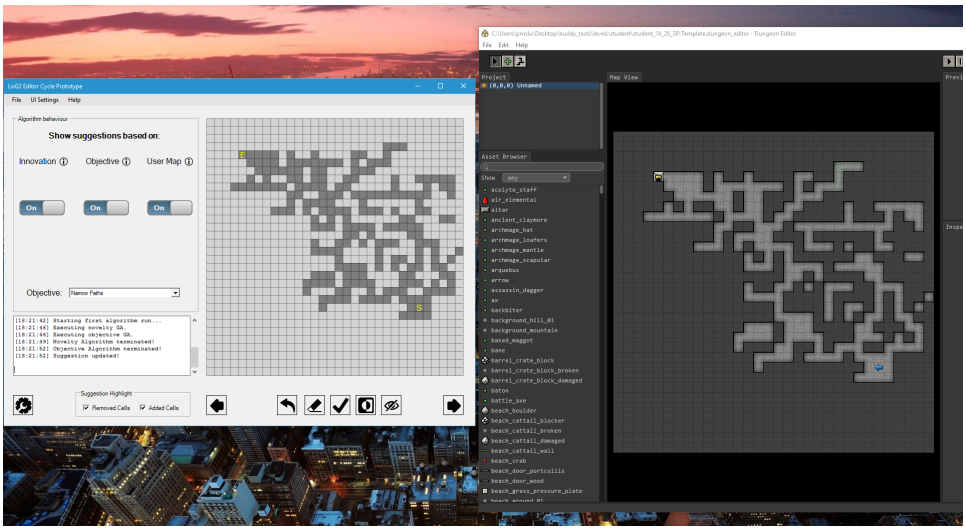


Figure 6.4: Setup used in evaluation tasks

Data analysis methods

Because we used questionnaires as well as participant observation and interviews, we analyzed them accordingly using distinct methods.

Data collected from questionnaires was better analyzed after displaying it in charts. Although we had relatively few participants to be able to reliably pick out emergent patterns within displayed results, any discrepancy we detected in other methods could possibly be spotted this way, whether for the interface or the behaviors.

For the data collected from participant observation, through screen recording and presential observation, there were specifically useful strategies which were followed to better analyze this particular method of data collection.

- Description of **key events**, not necessarily by order of occurrence, but mainly by **order of importance**. In this case, what kind of interaction was performed on the application, regarding its configuration or radical modifications of the level layout
- **Chronology**, description of participant's **actions over time**
- **Description of the participants**, how well are they related to the area of this work, what is their experience with similar tools
- **Setting description**, the location and means where the observation took place, as well as other relevant relation between places and evident behavior
- Description of **important processes** such as control interaction or decision-making
- Description of **key-issues** which led to a change of behavior on the participants behalf

Interview results were better evaluated using a **content analysis** approach. Interview transcripts were analysed and relevant excerpts were coded appropriately. Relevant sentences or words included references to actions, their own or the Editor Buddy's, abstract concepts regarding the given task or their

context, the game itself, their intentions and need or wishes, among others. Coded sentences and excerpts were then grouped depending on what they transmit and these groups were then categorized to sum up type of information they conveyed in the transcript. These categories were then put in a hierarchy to better reflect their interaction. In the end we look for preconceived hypothesis regarding the creative process and plausible designer/AI interaction development and try to identify those patterns in the previously conducted analysis.

6.4 Results

In this section we present the data collected in the previous steps regarding the questionnaires, observations and interviews. This is followed by an interpretation of that data in the light of previously established theories and concepts. In the end we discuss a couple of methodological difficulties which may have ended up affecting the quality of our results.

Presentation

Questionnaires

From the questionnaires, which can be consulted in Appendix A, we were able to gather the following data.

Evaluation sessions were performed with six participants, young adults, with ages ranging from 21 to 35 years old. All of the participants were also male, for no particular reason. In our test sessions we were able to gather three participants which were professional level or game designers and the remaining three were individuals inexperienced with either activity.

Regarding the usability of the Editor and the Editor Buddy, the majority of participants found them easy to use and their integration was intuitive and satisfactory. Table 6.1 shows results from the usability section of the questionnaire, questions 4 through 7, where:

- Question 4: It was easy to edit the layout of my level using the Legend of Grimrock 2 Editor
- Question 5: It was easy to configure the Buddy behavior using the available interface controls
- Question 6: It was easy to make and edit a selection of the map suggestion made by the Buddy
- Question 7: There were no communication issues between the Editor and the Buddy

LoG2 Editor and Editor Buddy usability				
Scale	Question 4	Question 5	Question 6	Question 7
Totally disagree	0	0	0	0
Somewhat disagree	0	0	0	0
Neither agree nor disagree	0	0	1	1
Somewhat agree	2	2	1	2
Totally agree	4	4	4	3

Table 6.1: Questionnaire usability results

Regarding the behavior of the Editor Buddy, the following charts, figures 6.5 through 6.7, compare the usefulness of content generated by each behavior switch or slider to the results expected by the

designer from those individual controls. These results pertain the algorithm behavior section from the questionnaire in Appendix A

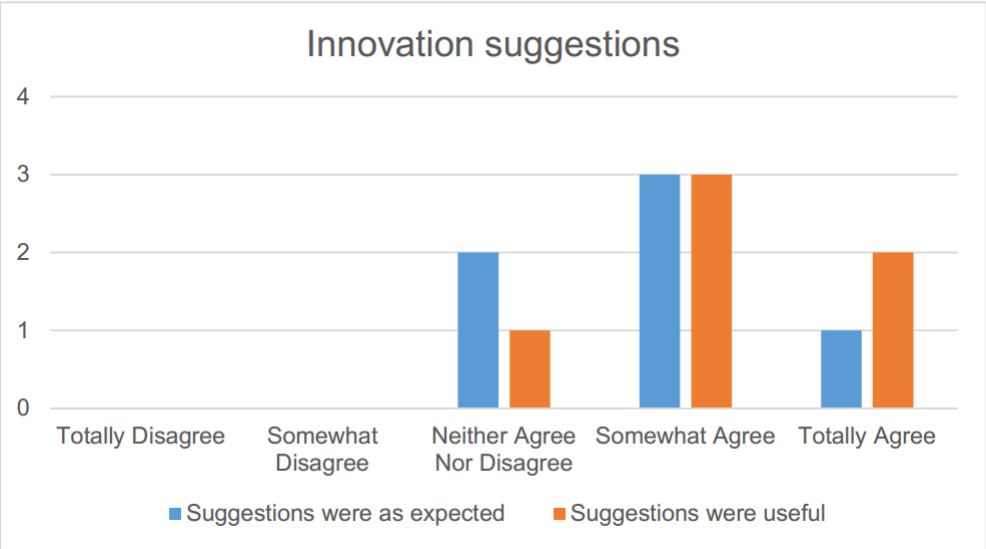


Figure 6.5: Innovation control expectation vs usefulness

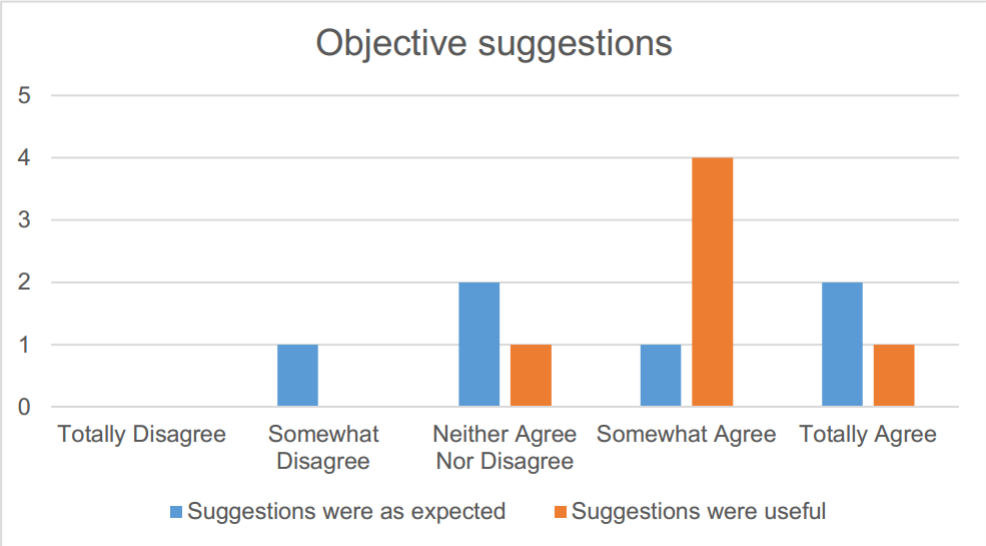


Figure 6.6: Objective control expectation vs usefulness

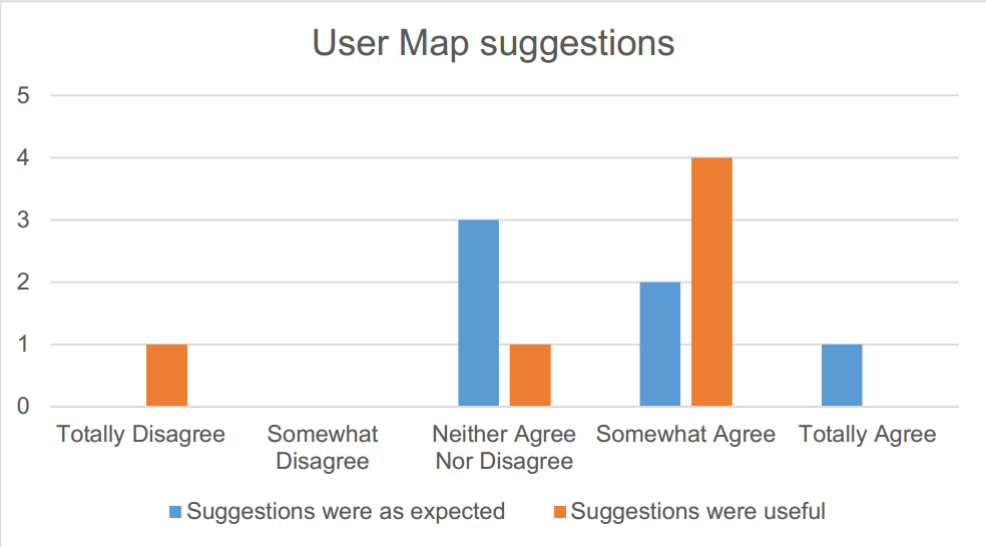


Figure 6.7: User Map control expectation vs usefulness

Observations

From the observations we have been able to identify a number of patterns regarding how participants interacted with the LoG2 Editor and the Editor Buddy. Amongst the topics we mentioned as relevant for this particular activity, below we present the most pertinent observations.

Regarding the **participant description**, participants belonged to two distinct main groups: professionals and non-professionals. Professional participants were either game or level designers and had a broad experience with the activity of level design. Non-professionals were computer science students at IST-Taguspark, more specifically from the game course, and at the same time, they were familiar with such areas as AI and videogames.

Regarding the **setting description**, these observations took place in a controlled environment, a meeting room, where the participants could take their time in performing the tasks and enlighten any particular doubts with any of the software used or the evaluation activity itself. The object of observation was the participant's interaction with both the LoG2 Editor and Editor Buddy, when asked to perform the official evaluation tasks, using screen capture. These tasks were performed in a single computer where the LoG2 Editor and the Editor Buddy were placed side-by-side.

Key-events are, in this case, not necessarily ordered by time of occurrence, but ordered by relevance. That being said, we were able to observe:

- A general increase of interaction with the behavior sliders, immediately after the expert mode was introduced
- There was an increase in interaction with the Buddy canvas and suggestion selection in all cases after expert mode introduction
- Exporting suggestion selections almost always meant a consequent refinement of the that content in the designer's level
- When asked to work towards a different objective with the Buddy, the vast majority immediately changed the Editor Buddy objective using the appropriate UI element

Key-issues generally detected in the observations were:

- When facing participants with poor suggestions from the Buddy that rarely translated into an attempt to reconfigure its behavior using the standard mode
- When trying to preview the entire suggestion, the majority of participants would perform a selection of the whole canvas instead of using the dedicated visibility button
- Participants almost never interacted with the historic buttons
- After being introduced to the expert mode (sliders), participants didn't change back to the standard mode (switches), despite being told its use was not mandatory

Important processes such as decision making or control interaction were different for each participant. Essentially, we can describe a few of these patterns among the participants.

- The participants who started interacting with the buddy early on, effectively performed selections, exported and improved these changes frequently until they were finished
- Participants which had a predetermined idea for their level, the majority being professionals, wouldn't interact with the Buddy until after they were finished with a first version of it

- Generally, the participant's interaction with the behavior switches and/or sliders increased over time
- Participants would perform selections over the canvas regularly, even if they did not export anything
- Participants would not export a selection very often, however none of the participants used the revert button to undo this step
- Even though it there was no time limit, duration of each task was very even amongst participant, which averaged about 10 to 15 minutes

Interviews

Results collected during the interviews were, as expected, more diverse and provided a more in-depth response to the fundamental questions which motivated these testing sessions. As such, it's slightly more difficult do convey, in its entirety, the whole spectrum of experiences and preferences of the participants but like in the above results we will try to focus on the more pertinent subjects while trying to include different opinions. As a reminder, the structure of the interview used in these tests can be found in Appendix B.

The Editor Buddy was widely accepted as being and overall positive influence during both the evaluation tasks. The editor buddy interface was found to be user-friendly and intuitive.

Non-professionals preferred to use the Editor Buddy during the first task where they were asked to create content from a minimal template. Professionals, however, generally preferred to use the Editor Buddy during the second task where they were asked to modify their work towards a different objective instead of creating content from zero, with one exception being a level designer who was interested in receiving much more different alternatives.

Recommended configurations were somewhat dependant on which task we were referring to, however, there is consensus towards using the Objective switch turned on (or slider on high) for the duration of both tasks. Innovation and User Map switches and sliders were the controls which had most disparity concerning their usage and preference but both were related to which task and which stage of design participants were focusing on at the time. There where some who recommended the use of Innovation or User Map controls in early stages in order to generate a large quantity of content quickly and there were others who recommended their use in later stages of the level design, where the exploration of more detailed changes preferred. Still, in both cases there where exception where some participants preferred a lot more suggestion innovation in the end, and a lot less in the beginning, and the same for user map.

Inversely, but still dependant on the stage of their level's design and the task at hand, some participants did not recommend the use of Innovation when they wanted to slightly modify their work while others did not recommend the use of User Map when they wanted more different alternatives. Like the recommended configurations, there were some exceptions, where less used configurations by some were more used by others and vice-versa.

Suggested improvements to the interface included:

- A change in the color scheme of the suggestion highlight
- A way to evidentiate viable paths from start to finish
- A more detailed description of the behavior switches/sliders tooltips

- A way to display real-time suggestion highlight, where more recent suggestions would display more vibrant colors and older suggestions a more faded color
- An initial configuration procedure to help set up the optimal interface and behavior according to the type of user
- Grouping interface buttons to optimize intuitiveness, such as historic buttons and selection-editing buttons

Suggestions towards improving the Editor Buddy behavior included:

- A button to enable the start of a new Buddy iteration in order to receive the next suggestion
- A way to improve the quality of the displayed suggestion, in exchange for a more late feedback, which is to say, adjust the trade-off between suggestion complexity and response time
- A way for the Buddy to identify patterns in the designer sketch and to enable the generation of those patterns such as rooms and branching to complement the main design of the level
- A new slider to control map difficulty, similar to the complexity slider, related to the size of the map
- Take into account the context of the performed selection, and not just the whole map area. Additionally a new control to adjust the granularity of suggestion generated, whether it is on the context of the whole map or the selection

When inquired about the type of presence the Editor Buddy should have, whether it should remain a more passive intervenient or whether it could be interesting to have the Buddy apply its suggestions directly in the designer's level, the response was unanimously negative. Some did not completely discard the more active behavior completely, suggesting it could be kept as an alternative mode, one that the user would have to actively turn on and off.

Finally, the expert mode (sliders) were well received overall and gave participants a new-found motivation to engage in interaction with the behavior controls. An exception being one case where a professional felt it was an excessive addition to the interface and was not particularly useful in their case.

After collecting qualitative data from the interviews, we performed a preliminary treatment of this information, called coding, as part of our result analysis. This process consisted of labeling paragraphs or expressions which convey similar ideas or contain similar terms or expressions. Table 6.2 represents the coding process performed by us on the feedback we obtained from the interviews, where sub-categories were attributed to labels.

After going through the interview transcripts and coding pertinent sentences this way as well as relating them to several sub-categories, were able to synthesise three main, more broad, categories amongst the identified sub-categories:

- Activity objective
- User receptiveness
- Program adaptability

And figure 6.8 illustrates how they relate to each other, hierarchically.

Label	Sub-category
Innovation Randomness Active / passive behavior	Behavior
Guided suggestions Usefulness Alternative suggestions Complementary content Objective agnostic content Alternative search	Content
Difficulty Complexity Linearity	Constraints
Idea definition Creative process	Ideas
Intuitiveness Appealing sliders / switches	Interface
Selection context Global context Design patterns Main path	Layout
Create from scratch Refinement at the end Initial layout	User action
Predefined user goals Designer responsibility	User objective

Table 6.2: Coding table - labels and sub-categories

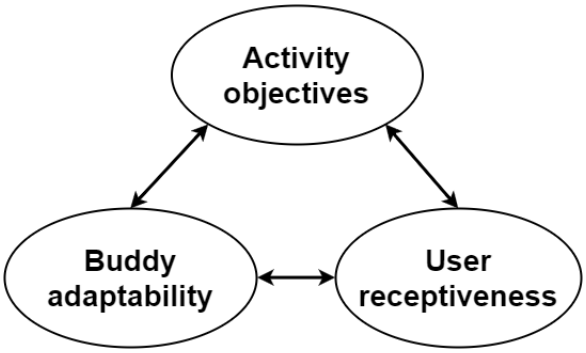


Figure 6.8: Content analysis hierarchic relationship diagram

These categories represent a generalization of similar and recurrent observations, sentences or expressions collected from participants during interviews, a result of content analysis process [30]. In this sense, we describe how we believe the identified categories are relevant.

To explain what “Activity objectives” mean in this analysis, we need to remember we relied in having predefined goals throughout our development stages so we could make sure that the Buddy behaved in a coherent way outside of suggestions around the User Map domain. This became evident in our evaluation tasks, so that participants could feel they were working towards a common goal. Often during interviews participants found the Buddy worked best for them when working to achieve particular goals, reassuring us the global objectives of the activity play a big role in the interaction.

The “User receptiveness” category reflects the multiple reactions participants expressed towards generated content and their expectations. After being given a suggestion, participants would comment on how it would be interesting for them or not so much. Additionally, there were those who chose to avoid suggestions altogether for some time. This suggested that the receptivity of each participant was a matter beyond what could be controlled by simply setting global design objectives.

The last category, “Buddy adaptability”, is directly related with the previous one. In case they were not pleased with what they were being suggested, they often mentioned how and why they would guide the Buddy into generating content towards their personal preferences at the time, which reflects the need for flexibility on behalf of the Editor Buddy.

In the end the test sessions lasted about one hour, on average, including the interviews and questionnaire answering, with plenty of flexibility left, allowing for a laid back test environment where participants were always able to pause and pose any doubts during the procedure or simply to speak their mind.

Interpretation

In this subsection we first use the previously synthesised categories and look to validate or dismiss any of our previous theories behind the usage of the Editor Buddy, formed by our personal experience during development as well as preliminary evaluation phases. Additionally we attempt to substantiate any remaining doubt or support any statement with results from the remaining data collection methods we used, questionnaires and observations. After that, we will try to adhere to the order of questions in the interview document and present our interpretations accordingly, along with any additional relevant information from other methods.

Let’s start by reminding ourselves that the topic of creativity is hugely subjective and we are merely scratching the surface of what we believe could prove to be a good way to improve the creative process in the minute portion of creativity that is level design.

From the beginning of development we though about setting global design objectives for both the Editor Buddy and the designer. It became clear that doing so ended up helping us distinguish participants into two different types based on how they interacted with the Editor Buddy. Although a common objective had been established, it was plausible to consider not all participants had decided on how to start designing their level. Nevertheless, there were those who were receptive enough to explore suggestions from the Editor Buddy right from the start and those who only referred to it only when their expectations for their level was somewhat achieved. Still it was evident that the Editor Buddy was able to be configured in a way to suit both types of users and also that both types of user were able to find a way to adapt the Buddy to their immediate needs.

Amongst other things, the amount of receptiveness each participant showed, or how it evolved, helped us confirm why we needed a way to orient the Editor Buddy behavior. We started by setting all behavior switches on as default for the Editor Buddy during these evaluations. On the one hand there where those did not wish to interact with the Buddy or look at a suggestion until they reached a certain stage of their design, on the other hand there may have been those who where not sure what they were trying to accomplish and in turn were pleased with the potential variety of suggestions proposed right from the start and simply did not change that default configuration. One important note is the distinction between the type of user and the degree of user receptiveness. By type of user we mean whether participants were professionals or non-professionals and by user receptiveness we mean how much were they inclined to receive feedback, novel or not. These concepts may seem quite similar, and the truth is at times they proved to be firmly related since the user mindset ultimately influences his receptiveness, such is the case of professional level designers who typically seem inexorable towards predetermined goals of their own. We did not immediately consider this, but later on we wondered just how much professional participants demonstrated less inclination towards receiving and interacting with suggestions, because they felt like they were being evaluated themselves in how well they could accomplish the task, instead of us being evaluating our solution. Nevertheless, one of our initial design goals was to keep the suggestion pool diverse, allowing potentially useful suggestions to emerge if the designer kept all the behavior switches on, meaning user receptiveness towards several potential solutions was great. Furthermore, results from the questionnaire regarding the Editor Buddy behavior tell us that in spite of finding results generated by the individual behavior switches/sliders less than expected, these were useful nonetheless, with the exception of a few punctual cases. One of these is directly related with the negative evaluation evident in figure 6.7, where the sole purpose of this participant was the search for novel or objective-oriented content, meaning user map based suggestions were clearly unwanted, thusly rendering the user map control useless for his needs.

We noted in our observations the amount of interaction with the Editor Buddy behavior controls generally increased over time. This means there is at least some need for flexibility since neither party can guess what the other is thinking and as soon as the creative process starts grinding to a halt, either one of two things can justify the need for such flexibility: either the designer is not content with the result of his level and seeks further new, not yet reached, alternatives, regardless of the amount of change or he seeks to validate the quality of his level by discarding those yet unexplored, potentially useful, emergent suggestions. Additionally, an overall increase in interaction piqued our interest during observations after we introduced the expert mode where the switches were replaced with slider controls. This could be due to the nature of the element itself, which provides a higher degree of control in contrast to only *on* or *off* positions of switches, or because switches were not enough to answer the need of a more deep control over the Buddy behavior. We purposefully included both modes and in the final version and waited until the second task to introduce the expert mode. Thus, this immediate adherence to this mode can mean there are more unforeseen advantages in using it than the simplified, switch-based version.

A primary goal for these test sessions was to measure the difficulty in adapting the Editor Buddy to

generate content towards what was both the desires of the designer and the task objectives. Perhaps if we had focused less on establishing a common goal for both the designer and the Buddy and instead focused solely on what the designer created as guidelines for the Buddy to interpret and simply complement, it could completely eliminate any eventual configuration issues by rendering it unneeded. In contrast, having a way to inhibit or control the emergence of certain aspect in the suggestions broadens the spectrum of utility for this tool. We were happy to verify that there were participants who could relate better with one of both cases. Some preferred to use it as a way to receive inspiration through novelty and alternatives, others as a way to refine their current work, and those who perceived it as useful on both accounts. It was our intention to provide enough flexibility to the Editor Buddy so we could encompass both styles of design. Nevertheless, it was clear after these test sessions, professional users require a more custom-tailored approach in regards to both generated content and degree of control. We were suggested, by one of the professional level designers, the Editor Buddy could include a certain configuration procedure when initializing it, before you can immediately proceed to use it. This could be interesting to explore, since it could help better encompass the two distinct types of users we detected by providing a more appropriate co-operative design experience, directly related to the user's level of expertise. This initial configuration would influence the amount of options to include, the objectives of that co-creative interactive in particular and other settings which could potentially contribute to improve the behavior of the Editor Buddy.

We found during our observations that when facing participants with poor suggestions from the Buddy, that rarely translated into an immediate attempt to reconfigure its behavior early on. We say poor, because there were a lot situations where participants did not immediately export selections and only kept re-making them. We are merely speculating, but in their experience it could be an attempt to adapt the suggestion they were being given to their work by changing the visible portion of the suggestion or because they did not wish to reconfigure the Buddy but wanted another different suggestion. A couple of participants discussed this issue in particular and offered some insight on how to improve this. By allowing the designer to request the Editor Buddy to re-generate a new suggestion or at least for the selected area through the use of a button, perhaps, the designer could avoid any alteration to his level as well as any disturbance to the Editor Buddy configuration and still be able to receive as many alternative suggestions as he wanted, on-demand. Still regarding selections on the Buddy canvas, we noted some participants tried using the Buddy to explore an area of the map they were not exploring at that time, effectively working in parallel with it, which was something we remembered to look for during evaluations to confirm one of our usage patterns.

Recommended and not recommended behavior control configurations also did not reach a consensus amongst participants. We were, however, able to indirectly pick out some more frequent patterns when participants agreed that User Map domain was a better candidate than Objective and Innovation domains in later stages of the design when a smaller amount of changes, more based on their work, was preferred. In contrast, users agreed that Innovation and Objective domains were far more useful when creating a large amount of alternative content in earlier stages of the design. Again, there were a couple of exceptions, but we cannot be fully sure whether this was because of specific circumstances or exceptionally divergent logical or creative processes given we are dealing with different individuals, each with their own lifetime experience. It is hard to state from such a small sample but we like to believe there is some correlation between some of our initial theories on these patterns, although we would not go as far as to state we were able to validate them. When starting with a blank canvas, participants often like more innovative, complete suggestions and then require a more fine-tune approach when their level was close to completion, regardless of which task they were working on. One discrepancy we found with questionnaire results was that of a professional level designer, where suggestions using solely the User Map switch or slider were not useful, whereas these results were neither expected not unexpected.

Coincidentally, this was the same user who mentioned, personally, he frequently needed to explore new alternatives for his levels and in that sense, the small amount of entropy the User Map control introduced was clearly not enough. Inversely, he found the Innovation control much more useful and expected.

A few improvements to the interface were suggested, the majority coming from the professional participants. Participants in general agreed with the overall interface design, and provided a few minor and other more in-depth improvements, followed by some behavior changes. Minor improvements included changes which would not significantly change its functionality such as a different color for tile highlight or a color scheme to better convey the idea of their function. Regarding the interface layout, one of the professional level designers drew our attention towards the proximity of the left historic button and the revert button and noted the potential for the user to be confused since they look similar. To fix this, he suggested grouping buttons by functionality in order to optimize intuitiveness in the UI. We found the suggestion regarding the valid path preview interesting both because we did not remember that during development and because it was mentioned by more than one participant. One element of the Buddy interface participants did not interact with, or very little, were the historic buttons. This can mean one of two things, either their goal was to create content quickly and, in this sense, they focused only on the most recent the suggestions because revisiting older suggestions were not worth the trouble or because they simply overlooked it.

Suggestions towards improving the Buddy behavior were more diverse and pertinent, like we were expecting. There was one which meant implementing an additional button in order to trigger a new iteration on the Editor Buddy so a new suggestion is proposed, like we mentioned above in this section. Its functionality could help fill the gap between moments where the designer neither wishes to further alter his level layout in order to trigger a new suggestion nor he wishes to alter the Editor Buddy configuration to provoke a change in suggestion pool. Perhaps this is related to the frequent re-selection on the Buddy canvas, in an attempt to make the best out of the presented suggestion.

Another pertinent observation from one of the professional participants was having a way to control the content quality vs reaction time trade-off we were making, in a sense that the Editor Buddy would suggest potentially more correct or refined solutions at the expense of taking more time to perform those generations. We did not initially consider the option to expose that control to the designer, however we have performed some preliminary tests when choosing the best values for this trade-off. We have already mentioned those decisions in the Algorithm Implementation section, and found that the diminishing returns on the size and number of generations in an algorithm run were not worth the linear increase in execution times. Still, conducting further tests to better adjust these values may prove useful before exposing that additional control to the users.

One of the professional participants had a different view from the rest when using the Editor Buddy. He believed the Editor Buddy would be able to clearly recognize the main path in his sketch, in the context of the current tasks. After which, the Buddy would suggest similar design patterns with that particular path in mind as a way to complement it, without compromising any part of his main layout. As it stands, the Editor Buddy values the existence of a valid path above other goals. Our User Map domain ensures that much of he generated content will be based on the designer's work, but will still attempt to break preconceptions within that solution by providing valid alternatives nevertheless. However, this aforementioned, improved behavior could provide a potentially better alternative to users which already had a strong sense of objective envisioned for their level's layout. One possible approach for this would be understanding exactly what the main path is in a designer's level, guarantee that path stays unaltered, at least in a way that does not compromise relevant metrics such as number of steps or average traverse time. The next step would be identifying the several design patters inside the designer's layout such as rooms or hallways and build a graph with these patters as we keep finding new ones. This could mean the Buddy could experiment with patterns themselves and present slightly different, novel ones.

All consequent suggestions, made by the Buddy, would then be focused around presenting interesting placement of the designer's own patterns around the core layout of his level. This approach, however would mean a more drastic change to the Editor Buddy behavior since it would need to shift and adapt its focus towards the goals of the designer in addition to the existing predefined goals. Still, we can take away a valuable conclusion from this observation in regards to professional, or maybe just more experienced users with a stronger sense of objective, which is the fact that at some point, novel or user-based content is not enough by itself and we require a more in-depth generation of customized user-influenced content.

Many participants drew our attention towards a non-existent way to control what was referred to, by them, as complexity in these testing sessions. By complexity participants meant the amount of branching a certain suggestion would contain from its main path or the amount of room dispersion, in case of the second task. This can be related with yet another suggestion made by participants, which is the way the Editor Buddy takes into account the created selection when generating its suggestion. Actually, the Editor Buddy does not take into account the size and shape of the selection made by users. Instead, it generates its content for the entire map area to better achieve the global task objectives. We did this to ensure suggestions made sense as a whole and that so that they could evolve towards playable solutions. A disadvantage of this is sometimes participants complained some areas exposed by them in the suggestion would seem out of context. This was because the remaining part of the suggestion was hidden. Revealing the hidden portion of the Buddy suggestion would help them understand this, but the need for more contextually focused content was still present. Buddy suggestions initially start hidden to avoid over-influencing the user, while still giving them the option to "peek" at the rest of the solution, while the selection tool would be used to export interesting segments and as a way to focus on revealing only that segment.

Finally, regarding the presence of the Editor Buddy during the co-creative activity, participants agreed that a more passive behavior was preferred over one that would perform changes directly in their work. We can completely relate to this, since it was our guess this was perhaps a step over the fine line that is the dynamic balance between the designer's open-mindedness and the AI's permanent quest for optimum results. Still, we were merely probing for possibilities regarding the direction of future work.

Difficulties

Throughout the test sessions there were a couple of details which may have compromised the quality of the results collected using the observations and the interviews.

Participant observations invariably took place during the whole process of evaluation, whether it was the participant's initiative or our own, there were in fact moments where the discussion of the Editor Buddy usage or behavior led to the discovery of interesting feedback and ideas. In this sense, some discussions may not have been as productive as others, partially because each participant's personality may result in a more or less active exchange of ideas or because they did not immediately feel comfortable in sharing their thoughts. This difference between sessions may have led to different lines of thought which in turn could have influenced some answers during the interviews.

Additionally, there was an issue during one of the screen recordings where the program did not record that session. This had to be mended by writing down what had been observed by myself as well as any relevant spoken information.

As for difficulties in the interviews, there were none in particular at this stage. The only potentially negative aspect we would point out is the fact there were other occupants in the room, conducting similar test sessions not related with our own, although far enough not to disturb any activity conducted in this

study. Nevertheless, these interviews were recorded to make for an easier analysis of the transcript, and the background noise resultant from adjacent conversation sometimes made it difficult to clearly make out some dialogue segments of these recordings.

Not as much of as issue, but more as a curiosity is how these sessions ended. In spite of them ending at the participant's will, we cannot help but wonder if the learning curve was the same for all of those involved or if there were cases where this curve was a bit too steep and they required more time to familiarize themselves with the Editor Buddy. Despite receiving a positive review from participants, we still give it the benefit of doubt, because one session may not have been enough to spot a deeper issue or limitations with the Editor Buddy or its interaction with the LoG2 Editor.

6.5 Study conclusions

Key-findings

Thanks to the feedback of participants, this study allowed us to identify several important aspects and limitations in the developed software. The first would be directly related with the distinct types of users we invited for these test sessions, which is the fact that the mindset of the user and level of experience ultimately influences the usage and utility of the Editor Buddy. Also, for users who already have a clear idea of the layout they wish to create, the need for innovation or alternatives is greatly diminished. Inversely, designers who carried no preconceptions regarding their level's layout could enjoy a more broad array of options from the Editor Buddy and, possibly, a richer creative experience. However, such tools are still dependant on much more than just professional experience. Personality traits and life experience mean as much, if not more, to the creative process and, ultimately, to the potential of such tools.

Besides the aforementioned distinction, there was a different kind of need in professional participants. The fact some expected a more contextualized feedback from the Editor Buddy, in the sense that it should be able to pick out patterns from the designer's level and generated novelty in a more coherent way, made us realise the main focus for those types of participants was the context of their own work. This helped us better understand the individual needs of professional level designers what improvements could be made in that direction.

We wish to underline the fact that an interaction-focused co-creative activity seemed to work well in most of the cases. However, particular cases where the less refined suggestions proposed by the Buddy seemed a bit too rough tell us that perhaps it could benefit from having a way to control this parameter. This would mean a more pronounced trade-off between quality of content and response time, where the amount of time dispended would not linearly translate into better results. Still, the option to do so would be present.

In the vast majority of test sessions there seemed to be a common misconception amongst participants, which its that the Editor Buddy would take their selection into account when generating its next suggestion. The fact participants regarded the Buddy's canvas beyond its most simple use and envisioned a way to work more intimately with the Editor Buddy surprised us. This made us reconsider the importance of such actions and would like to point this out as one of the most interesting improvements for future work.

Logical steps

Following steps would include the implementation of the more straightforward suggestions, which showed

to be a consensual improvement over the current implementation, for example the interface improvements.

For the remaining suggestions, those which required a more careful approach, we would need to perform more in-depth preliminary evaluations to assess implications and costs as well as clearly define which areas are worth being improved and how.

Implications

In the end, this study did not dismiss the main purpose of this tool, if anything it helped us validate it was a step in the right direction and discover some aspects of our particular implementation which could benefit from the insight gained from evaluation sessions. These improvements would serve to better adapt to the need of the more particular user which is level designer and, if possible, appeal to a larger universe of more casual users.

6.6 Recommendations

Recommendations on how to improve similar studies would be, for instance, to perform a recording of the whole activity, should the participant agree to. This is because during the interaction with the tool participants would pose questions, which was perfectly acceptable, however even the smallest bit of dialogue could hold a valuable concept. We tried to write down as much as we could, but cannot be sure we didn't miss anything. In this sense, it is recommended to start collecting interaction data as soon as they are given the chance to experiment with the tool.

An interesting followup for these studies would be putting participants in charge of explaining and demonstrating the use of the Editor Buddy to new participants. This would help understand just how much did the first participants grasp from their experience with the Editor Buddy. Additionally it would give them a chance to validate their previous experiences like their recommended switch/slider combinations, as well as give them a chance the change their mind. At the same time, we would learn, by observation, if there was anything lacking to the current style of evaluation.

Chapter 7

Conclusion

In this section we refer to the several steps of this project and draw out meaningful conclusions as to whether the developed solution had a positive influence over participants or not. We also refer the several possibilities for future work, either those suggested during the evaluation sessions or those which we did not implement during development because of time constraints or simply because it fell out of the scope of this project but were interesting nonetheless.

7.1 Solution performance

We were pleased to observe the overall the performance of the developed solution was positive and it proved to be flexible enough to adapt to several types of users, not taking away any credit to those who were able to make the best out of what they were given, of course. This, in fact, proved to be one of the most important observation we made.

Because our approach was significantly different from the ones we mentioned in our related work section, it provided a twist to existing processes where every step needs to be perfectly correct and everything needs to have an immediate logical explanation. In our particular case, we mean that suggestions which were not immediately playable sometimes motivated participants to effectively export that suggestion, or a part of it, and improve on that idea using their own view. Nonetheless, these suggestions would keep on being evolved by the Editor Buddy. In this sense, we focused more on providing a way for users to interact more precisely with suggestions they were given, as well as watch the Buddy suggestion evolve side-by-side with theirs. Thus, allowing participants to explore every potentially creativity-enhancing aspect of the co-creative level design activity, starting with a blank canvas all the way to a playable solution.

The fact we received so many design suggestions, which we discuss just ahead, taught us there really are no perfect solutions and finding a middle term, where we attempt to please both experienced and inexperienced users, proved to be both a challenge and a good way to determine the difference in needs and behavior. Still, there are multiple factors we cannot control which end up influencing the outcome of such evaluations or real-life applications, even more so when we are talking about something as volatile as creativity or human beings themselves.

In the end, we believe we were able to portray this paradigm of a digital “peer” and we hope it serves as an interesting contribution to the field of human-computer co-creative science.

7.2 Future work

Throughout the development process up until the evaluation phase, we encountered several alternatives or ideas to improve certain aspects of the Editor Buddy. These were the most pertinent ones we kept for future work reference:

The implementation of detection and generation of user design patterns could prove as a very interesting starting point when working towards a more user-centric approach. Related with this observation, we can also look with greater detail into the work of Liapis et al. in their Sentient Sketchbook paper [1], more specifically, their work on accommodating the designer's style into the suggestion of the Editor Buddy.

Another interesting change to the Editor Buddy behavior would be the creation of contextual content depending on the selection made in the Buddy canvas. For example, creating a narrow selection would translate into generation of halls or more straight segments. We found this interesting and cannot help but wonder there is unexplored potential behind contextual generation of content and it might be worth considering for future content.

A big step after the layout generation is polished enough is the procedural generation of monsters and puzzles, as well as lighting in darker dungeons. In this sense, a more thematic, gameplay related approach may be interesting where the Buddy would help populate the newly created level layout according to a certain theme, such as a run-down castle dungeon infested with mutant insects or a sunny beach with the occasional crab and mermaid.

A simple, yet tricky to implement, improvement would be to allow the Editor Buddy to support the parsing and exporting different types of tiles. Currently the Editor Buddy only exports "dungeon_floor" and "dungeon_wall" type tiles, like we mentioned in the implementation section, due to issues with the LoG2 Editor project file parsing. This may seem a small improvement, however, its essential if we were to implement the aforementioned paragraph on thematic levels.

Another interesting request we had in the evaluation phase was a way to evidentiate the valid paths for a given suggestion proposed by the Buddy. This would make it easier for designers to quickly make out which way the Buddy considered interesting and valid. In spite of being simple and relatively easy to implement, it can potentially speed up the design process.

Regarding future evaluation sessions, besides evaluating the human-computer interaction, we would like to compare the quality of levels created with and without the help of the Editor Buddy. In other words, after integrating a couple of new features and polishing existing ones, we could perform tasks with two different groups, one using the LoG2 Editor and the Editor Buddy to create content and another one using solely the LoG2 Editor. At the end of those tasks, a third group would evaluate the creative value of both levels and could eventually play them to find out which results work best in the player's point of view. Another evaluation we could perform would be providing participants with the Editor Buddy to perform a task and ask them to perform another task afterwards without the help of the Editor Buddy and see if they missed it. Inversely we could ask participants to perform a task using only the LoG2 Editor and and another task afterwards where we introduced the Editor Buddy and see if they found it useful in stimulating their creativity.

A more ambitious goal would be adapting the Editor Buddy to other existing level editing tools. Apart from the Legend of Grimrock 2, this would be a huge challenge, because the Editor Buddy is so deeply integrated with the LoG2 Editor. Because the LoG2 Editor works on 2D basis, we would require at least

those tools to work in a similar fashion, otherwise it would need to be redesigned to generate content for 3D game levels.

7.3 Final remarks

To what concerns us, the whole process of research and development was truly an enriching process, personal and professionally. From understanding the state of the art in computer co-creativity, to being able to develop a solution which seeks to expand and improve on those concepts, brought about great satisfaction. We can only hope our contribution is valuable enough to, at least, pique the interest of those looking to explore the niche of human-computer co-creativity in videogame level design.

Bibliography

[1] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-aided game level authoring," *Proceedings of the 8th International Conference on the Foundations of Digital Games (FDG 2013)*, pp. 213–220, 2013.

[2] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.

[3] M. a. Boden, *The creative mind: myths and mechanisms*. Routledge, 2003.

[4] E. De Bono, *Lateral thinking: a textbook of creativity*. Pelican books, Penguin Books, 1977.

[5] M. A. Boden, "Computer models of creativity," *Psychologist*, vol. 13, no. 2, pp. 72–76, 2000.

[6] G. Ritchie, "The JAPE riddle generator : technical specification," *Informatics Research Report EDI-INF-RR-0158*, no. February, 2003.

[7] H. Cohen, "The Further Exploits of AARON, Painter," *Special edition of Stanford Humanities Review, 4:2 (1995), pp. 141–160*, 1994.

[8] P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Zytkow, *Scientific Discovery Computational Explorations of the Creative Processes*, vol. 34. The MIT Press, 1987.

[9] J. Bird and P. Layzell, "The evolved radio and its implications for modelling the evolution of novel sensors," *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, vol. 2, pp. 1836–1841, 2002.

[10] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[11] T. Lubart, "How can computers be partners in the creative process: Classification and commentary on the Special Issue," *International Journal of Human Computer Studies*, vol. 63, no. 4-5 SPEC. ISS., pp. 365–369, 2005.

[12] A. Kantosalo, J. M. Toivanen, P. Xiao, and H. Toivonen, "From Isolation to Involvement : Adapting Machine Creativity Software to Support Human-Computer Co-Creation," *Procedings of the Fifth International Conference on Computational Creativity Ljubljana, Slovenia, 9th – 13th June 2014*, 2014.

[13] G. N. Yannakakis and C. Alexopoulos, "Mixed-Initiative Co-Creativity," *Procedings of the Foundation of Digital Games 2014*, 2014.

[14] A. Vile and S. Polovina, “Thinking of or thinking through diagrams? The case of conceptual graphs,” *Thinking with Diagrams Conference, The University of Wales, Aberystwyth*, pp. 22—-23, 1998.

[15] D. C. Brown, “Creativity, surprise & design: An introduction and investigation,” *The 2nd international conference on design creativity (. . .*, no. September, pp. 1–8, 2012.

[16] A. Liapis, G. N. Yannakakis, and J. Togelius, “Towards a generic method of evaluating game levels,” in *In Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference*, 2013.

[17] A. Liapis, G. N. Yannakakis, and J. Togelius, *Applications of Evolutionary Computation: 16th European Conference, EvoApplications 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, ch. Generating Map Sketches for Strategy Games, pp. 264–273. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[18] G. a. Wiggins, “A preliminary framework for description, analysis and comparison of creative systems,” *Knowledge-Based Systems*, vol. 19, no. 7, pp. 449–458, 2006.

[19] R. v. d. Linden, R. Lopes, and R. Bidarra, “Procedural generation of dungeons,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, pp. 78 – 89, mar 2014. doi: 10.1109/TCIAIG.2013.2290371.

[20] K. Hartsook, A. Zook, S. Das, and M. O. Riedl, “Toward supporting stories with procedurally generated game worlds.,” in *CIG* (S.-B. Cho, S. M. Lucas, and P. Hingston, eds.), pp. 297–304, IEEE, 2011.

[21] V. Valtchanov and J. A. Brown, “Evolving dungeon crawler levels with relative placement,” in *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, C3S2E ’12, (New York, NY, USA), pp. 27–35, ACM, 2012.

[22] D. Ashlock, C. Lee, and C. McGuinness, “Search-based procedural generation of maze-like levels,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, pp. 260–273, Sept 2011.

[23] S. O. Kimbrough, M. Lu, G. J. Koehler, and D. H. Wood, “Introducing a Feasible-Infeasible Two-Population (FI-2Pop) Genetic Algorithm for Constrained Optimization : Distance Tracing and No Free Lunch,” *European Journal of Operational Research (2008)*, vol. 190, no. 2, pp. 310–327, 2008.

[24] A. Liapis, G. N. Yannakakis, and J. Togelius, “Enhancements to constrained novelty search: Two-population novelty search for generating game content,” *Proc. of of the International Conference on Genetic and Evolutionary Computation (GECCO’13)*, pp. 343–350, 2013.

[25] A. Liapis, G. N. Yannakakis, and J. Togelius, “Constrained novelty search: a study on game content generation.,” *Evolutionary computation*, vol. 23, no. 1, pp. 101–129, 2015.

[26] M. Preuss, A. Liapis, and J. Togelius, “Searching for good and diverse game levels,” in *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8, Aug 2014.

[27] A. Liapis, G. Yannakakis, and J. Togelius, “Designer Modeling for Sentient Sketchbook,” *IEEE Conference on Computational Intelligence and Games (CIG)*, 2014.

[28] John Newcombe, “Genetic algorithm framework.”

[29] A. P. Engelbrecht, *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd ed., 2007.

[30] M. Miles, A. Huberman, and J. Saldaña, *Qualitative Data Analysis: A Methods Sourcebook*. SAGE Publications, 2013.

Appendix A

Questionnaire appendix

Editor Buddy Test Form

The primary goal of this form is to measure the utility of the Editor Buddy application and its efficiency at providing a creativity-enhancing experience during a cooperative level design activity. Additionally, it will help understand how well the current user interface and algorithm behavior can impersonate the digital partner paradigm we are trying to convey.

1. **Age**

2. **Sex**
   Mark only one oval.

   - M
   - F

3. **How familiar are you with videogames?**

Interface usability

On a scale of 1 - 5 where,
1 - Totally disagree
2 - Somewhat disagree
3 - Neither agree nor disagree
4 - Somewhat agree
5 - Totally agree
please grade the following statements according to your experience with the Editor Buddy interface and the Legend of Grimrock 2 Editor interface.

4. **It was easy to edit the layout of my level using the Legend of Grimrock 2 Editor**
   Mark only one oval.

	1	2	3	4	5	
Totally disagree						Totally agree

5. **It was easy to configure the Buddy behavior using the available interface controls**
   Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

6. **It was easy to make and edit a selection of the map suggestion made by the Buddy**
   Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

7. **There were no communication issues between the Editor and the Buddy**
   Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

Algorithm behavior

On a scale of 1 - 5 where,
1 - Totally disagree
2 - Somewhat disagree
3 - Neither agree nor disagree
4 - Somewhat agree
5 - Totally agree
please grade the following statements according to your experience with the suggestions generated by the Editor Buddy algorithms.

8. **Suggestions generated using the Innovation controls were in line with my expectations**
   Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

9. **Suggestions generated using the Innovation controls were useful**
   Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

10. **Suggestions generated using the Objective controls were in line with my expectations**
Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

11. **Suggestions generated using the Objective controls were useful**
Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

12. **Suggestions generated using the User Map controls were in line with my expectations**
Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

13. **Suggestions generated using the User Map controls were useful**
Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

Appendix B

Interview appendix

Interview

- Overall, did you find the Editor Buddy was useful?
- In which case did it work best? On the first task where you had to start from scratch, or on task 2 where you had to rework your current level?
- What combination or combinations of behavior switches/sliders would you recommend a friend using this tool for the first time?
- What would be the ones you wouldn’t recommend?
- Was there a right time to use a particular switch or slider all the way up? (Innovation/Objective at the start, User map at the end?)
- If you could make any change to the interface, what would it be? Why?
- If you could make any change to the behavior what would it be? Why?
- If you could have a mode where the Buddy’s suggestions would be directly applied to your work without the need for your approval, how would you feel about that? (Lock area)
- In case you have used the expert mode, what are your thoughts about it? In case you haven’t used it, why haven’t you used it?

Any other questions?