Co-creativity in Videogame Puzzle Creation

Gonçalo Nuno Dias dos Santos Delgado

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Dr. Carlos António Roque Martinho

Examination Committee

Chairperson: Prof. Dr. Paolo Romano
Supervisor: Prof. Dr. Prof. Carlos António Roque Martinho
Member of the Committee: Prof. Dr. Francisco João Duarte Cordeiro Correia dos Santos

May 2018
“Everything is going to be OK!” - Vaz, A. T.
To my parents, who gave love and support, through the bad times, and gave me this chance of education and to try to follow my dreams. To my brother, who's following my footsteps by being in a Computer Science course, hopefully some of my work here can help him in the near future.
Acknowledgements

There are three people I would like to thank, since they helped and encouraged me to continue another semester in order to finish this thesis. First and foremost I would like to thank Prof. Dr. Carlos Martinho, for influencing me with the passion for his work and for giving ideas to surpass the various obstacles along the path of this study, for keeping me interested in attending the classes and enjoying the master’s course when I first started. On a more personal note, thank you for your reasoning with me to not give up and our various conversations about games in general. Hopefully this is the first step to, sooner or later, be able to follow my dream.

The second person is my great friend Rui Mendes, thank you for pushing me in a positive way and for motivating me to keep going with this project and, and overall, for always being around giving good advices.

Finally, a big thank you note for my best friend and girlfriend, Teresa Vaz, for all the support and for cheering me up when I was down. I’m thankful for having you believing that this day would come and, as you said, everything would be OK.

I would like to thank the participants of our formal evaluation sessions: Ricardo Rodrigues and Diogo Rato, researchers at IST-Taguspark; Bruno Almeida and Tiago Baltazar Master students at IST-Taguspark; Filipe Silva an IST-Taguspark Alumnus and friend from my previous university and Pedro Lucas, an IST-Taguspark Alumnus and programmer at Marmalade Game Studio, creator of the first version of the Editor Buddy, the work that served as a base for this project.

Also, for their help during the preliminary evaluations, I thank my friends Rui Mendes and Ricardo Korn Moreira for their time and patience.

I would like to give a very special thanks to my cousin, Filipa Moniz, for the enormous help during the writing process of this thesis, reviewing it and helping me with suggestions. A big thanks my best friend Alexandre Maia for keeping motivated and being around overall.

I would like to outline and thank the contributions of our related work authors, particularly, once again, Pedro Lucas, for his work on the Editor Buddy and technical hindsight about the tool, which provided a solid reference for this project; John Newcombe, creator of the Genetic Algorithm Framework, for providing a mature framework when implementing our solution.

My final highlight and big thank you is to, my parents and my brother, whose support, care and motivation helped during the difficult moments faced during this project, and for believing in me until the end.
Abstract

This work proposes a solution to improve the cooperation between humans and computer Artificial Intelligence (AI), as a *colleague*, in the creation of puzzles for video game levels. With this interaction we hope to give the designer a source of creative stimulus, in order to achieve overall more creative results than those obtained if said designer was working alone. The proposed solution consists of a co-creative puzzle creation tool, focused on improving 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 three genetic algorithms and can be guided by the designer on different domains: *objective*, *innovation*, *user map*; all then combined in a fourth one that re-evaluates the best suggestions of each previous algorithm, again on the three domains, with different weights based on the users configuration, to choose the best suggestion overall. Results showed a positive influence on the puzzle creation because our approach takes into account the smaller nuances of the co-creative interaction. Outlined improvements such as a better way to support designer-specific interaction patterns, improved algorithm behaviour and integration with past tools set the direction for future work. We concluded that through an intuitive interface, flexible and adjustable behavior, we were able to provide some positive contributions to the quality of the co-creative puzzle creation process.

**Keywords:** Level-design, computer co-creativity, procedural content generation, genetic algorithms, puzzle creation
Resumo

Este trabalho propõe uma solução para melhorar a cooperação entre humanos e computadores, como colegas, na criação de puzzles para níveis de jogos de computador. Com esta interação esperamos dar ao designer uma fonte de estímulo de criatividade, para atingir resultados, no geral mais criativos, que aqueles que o designer obteria se estivesse a trabalhar sozinho. A solução proposta consiste numa ferramenta de co-criatividade na criação de puzzles, através do trabalho conjunto entre humanos e computador na criação de conteúdo, usando o editor de níveis do Legend of Grimrock 2 e explorando o paradigma do “parceiro” digital. A interface da ferramenta pode ser usada pelo designer para prever as sugestões geradas e orientar o seu comportamento. As sugestões são geradas e iterativamente evoluídas por três algoritmos genéticos e que pode ser guiada pelo designer em três domínios: objetivo, inovação e orientado ao mapa do utilizador; que são combinados num quarto algoritmo que reavalia as melhores sugestões dos algoritmos anteriores, outra vez nos mesmo três domínios com diferentes pesos de acordo com as configurações do utilizador e escolhe a melhor sugestão, no geral, para apresentar ao utilizador. Os resultados obtidos mostraram uma influência positiva na criação de puzzles uma vez que a nossa ferramenta toma em consideração os pequenos detalhes da interação co-criativa como fonte de potencial estímulo criativo. Melhorias apontadas incluem melhor suporte de padrões específicos de interação dos utilizadores, melhorias no comportamento dos algoritmos e integração com ferramentas já existentes. Concluimos que com uma interface intuitiva, comportamento flexível e ajustável, conseguimos criar uma contribuição positiva para a qualidade do processo de co-criação de puzzles.

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

List of Tables xi
List of Figures xiii

1 Introduction 3

1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
1.2 Problem . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.3 Hypothesis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.4 Objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.5 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
1.6 Document structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2 Background 7

2.1 Level Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
2.2 Creativity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
2.3 PCG . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.4 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

3 Related Work 13

3.1 Creativity-enhancing computer models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
3.2 Computer Co-creativity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
3.3 Evaluating Computer Creativity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
3.4 Search-based creativity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
3.5 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

4 Solution Model 21

4.1 Computer colleague paradigm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
4.2 Solution Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
   4.2.1 Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
   4.2.2 Behavior . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
   4.2.3 Editor integration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
   4.3 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

5 Solution Implementation 29

5.1 Legend of Grimrock 2 Level Editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
5.2 Solution Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
   5.2.1 Genetic Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
   5.2.2 Problem specific constraints . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
   5.2.3 Editor Buddy Puzzle Mode execution flow . . . . . . . . . . . . . . . . . . . . . . . . 33
List of Tables

6.1 Frequency of answers to usability questionnaire . . . . . . . . . . . . . . . . . . 46
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Brown's graph representation of a dungeon from Legend of Zelda: The Minish Cap.</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>Stimulating user creativity through diagrams</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Evolutionary Dungeon Designer interface</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Example interaction of the solution with a level editor</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>Editor Buddy Puzzle Mode Interface</td>
<td>23</td>
</tr>
<tr>
<td>4.3</td>
<td>Sections' Graph of a created map using it's Optimal and Complete form.</td>
<td>24</td>
</tr>
<tr>
<td>4.4</td>
<td>Representation of corridors in the 2D canvas</td>
<td>25</td>
</tr>
<tr>
<td>4.5</td>
<td>Representation of the Gate/Pressure Plate(Key) connection of the 2D canvas</td>
<td>25</td>
</tr>
<tr>
<td>4.6</td>
<td>Modifications made to the original Trickster's Lair map layout</td>
<td>27</td>
</tr>
<tr>
<td>5.1</td>
<td>Legend of Grimrock 2 in-game screenshot and Level Editor</td>
<td>30</td>
</tr>
<tr>
<td>5.2</td>
<td>Relation between a chromosome and the generated map</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>Order in which the nodes are expanded on a Breadth-First Graph Search</td>
<td>34</td>
</tr>
<tr>
<td>5.4</td>
<td>Editor Buddy Puzzle Mode algorithm execution flow diagram</td>
<td>35</td>
</tr>
<tr>
<td>5.5</td>
<td>Flow diagram's element captions</td>
<td>35</td>
</tr>
<tr>
<td>6.1</td>
<td>Interface from the first preliminary test</td>
<td>42</td>
</tr>
<tr>
<td>6.2</td>
<td>Final interface</td>
<td>42</td>
</tr>
<tr>
<td>6.3</td>
<td>Setup used in evaluation tasks</td>
<td>45</td>
</tr>
<tr>
<td>6.4</td>
<td>Objective slider expectation vs usefulness</td>
<td>47</td>
</tr>
<tr>
<td>6.5</td>
<td>Length slider expectation vs usefulness</td>
<td>47</td>
</tr>
<tr>
<td>6.6</td>
<td>Backtrack slider expectation vs usefulness</td>
<td>47</td>
</tr>
<tr>
<td>6.7</td>
<td>Preference slider expectation vs usefulness</td>
<td>47</td>
</tr>
<tr>
<td>6.8</td>
<td>Innovation slider expectation vs usefulness</td>
<td>48</td>
</tr>
<tr>
<td>6.9</td>
<td>User Map slider expectation vs usefulness</td>
<td>48</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

The main motivation for this study is the fact we see level design as a vital part of the game creation and, therefore, worthy of our dedication to the matter. It is a critical process, and often a solitary one, that can lead to process bumps, such as frustrating situations, odd or boring puzzles. This may conduct the players to quit before finishing the game, which is what we want to avoid. The designers must try to keep the player in the flow state, where the games are neither too difficult nor too easy, but provide enough challenge to keep the player motivated. To face this problem we aim that using computational creativity can have a huge potential to help with the creative process. Our goal is to make a tool that helps in the creative process and not to automatize the work of designers.

An example of challenges with puzzle creation, is the very difficult and infamous Goat Puzzle regarding the point-and-click game Broken Sword: The Shadow of the Templars\(^1\). Here, it is introduced a new mechanic mid-game where it demands precise timing for the player and, as it was halfway through the game, the player was not expecting a new approach to puzzle solving, resulting in confusion. For similar reasons, creating puzzles can be a tough task to balance in order to keep the player in the flow state.

This can become a harder problem for inexperienced or non-professional level designers, either by creating mods or by getting their feet wet\(^2\) in level editors of games such as Legend of the Grimrock\(^3\), they can struggle when trying to create levels that are interesting since it is a process that is still traditionally validated through play-testing, a process that takes time and resources to test the game. This is one of the reasons we are keen to yield a tool to ease the creation process of levels, more specifically the puzzles, by having the user interacting with the computer like a digital “assistant”, so both can work together towards a common goal, resulting in an interesting puzzle, from the gameplay perspective.

We have chosen Legend of the Grimrock 2 as the target game for our tool, since the game itself has a level editor and good mod support. Another point, is that it is a dungeon crawler, so it has maze-like dungeon layouts and it can be a good layout type for puzzles.

We aim, with the help of computers, to contribute positively for the inspiration of inexperienced level designers. Then the level designer, can follow the suggestions of our program and continue to iterate on the level, using his ideas to improve what the program creates, thus the final product be a co-creation of the Artificial Intelligence and the level designer.

\(^1\)Revolution Software, 1996
\(^2\)Almost Human Ltd., 2014
\(^3\)
1.2 Problem

Can an AI and a person work together to make more fun and interesting puzzles in a level instead of having the mentioned person working alone? Is the player experience enhanced by doing these puzzles? Puzzles that captivate the player and that are fun?

The process of designing levels demands creativity, moreover creating puzzles for a level. Creativity is innate to us Humans, the problem is that since the process of generating levels is most of the times a solitary activity the puzzle solution sometimes, may be trivial for the designer, but for the player it does not make any sense letting him stuck on the puzzle and maybe giving up on the game, churn, or force him to read guides, like the Broken Sword Goat Puzzle mentioned before.

The AI creates puzzle solutions, based on rules the designer made, and in return, these very solutions can trigger a flood of ideas for the designer, and then the AI can iterate on that, repeating this process until the puzzle is satisfying to the designer.

This is what we aim to verify with the creation of this study.

We wish to create a tool that helps the user to generate fresh ideas for puzzles while trying to keep the solutions consistent with the other types of puzzles.

1.3 Hypothesis

The human creative process, without stimulation, tends to move to a stagnate state but if an idea works really well, it may “trap” an individual to make similar iterations, even if not perceptible at first glance. But sometimes, in order to break that monotony of ideas, that same individual can try to create an idea too innovative that the solution does not connect with other people. For this reason, the cooperation between Human work and a computer can trigger better results when creating things, as the computer acts as an adviser with unbiased ideas that can inspire the user into creating a better solution than the previous iteration of the solution.

This is the main focus of our proposed solution, a way to inspire the user and allow him to explore different alternatives to the current state of his work.

For that we built a tool that will explore the “peer” paradigm, i.e., we do not want a tool that runs and delivers one possibility and stops, we want the tool taking turns with the designer, back and forth. The tool delivers a draft based on what the designer has and wants to create. Then, the designer modifies the suggestion of the tool to what he want by, maybe changing its parameters of draft generation so the tool adapt and delivers a new draft based on those changes. Then the designer can make new changes if he finds necessary and accept as either the final solution or go back to check a new draft from the tool based on the changes, thus continue this back-and-forth process until he is satisfied.

1.4 Objectives

The main objective of this work is to develop and evaluate the utility of a software tool applied to The Legend of Grimrock 2 game Editor which aims to simulate the interaction between humans in the creation of puzzles for a level of a game. This tool seeks to improve the quality of the said puzzles of non-professional users, by exploring co-creativity between designers and the AI. We want the tool to help the user to create fresh ideas for puzzles while trying to keep the solutions coherent with the other puzzle types and not to take the designers work, by automatizing their job. The tool should share the same activity, creation of puzzles for game levels and according to Lubart[1], this falls under the paradigm of
colleague or peer of the roles that a computer can have on the support of human creativity. We will go further on the other alternatives for this paradigm in the Related Work section.

In order to do the above mentioned we will provide a general overview of the current state of the art of computational creativity in the context of procedural generated content in games. Then we will develop a computational model for a digital assistant that improves the creative component of puzzle creation in levels and improve the previous work from Lucas [2] to integrate this model and in the end, test the tool with non-professional users.

1.5 Contributions

We would like to share some of the point where we see this study can be fruitful:

1. A way to categorize a map according to the puzzle difficulty/length and backtracking/branching
2. State of the art co-creative PCG techniques in videogame puzzle creation.
3. Computational model for generating and evolving content that helps a level designer in the puzzle creation.
4. Software implementation of said model and integration with a videogame level editor.
5. Qualitative evaluation on the usability and utility of the implemented computational model with non-professional level designers.

1.6 Document structure

In the first chapter we began by introducing our motivation and the objectives for this work. On the second chapter we have the more theoretical part of this work were we present the results of our research phase, this background chapter is where relevant subjects are introduced and analyzed.

The third chapter addresses the related work where we describe existing solutions, state of the art and scientific work that supports our theories. It will be done by discussing them by topic. In the end of the chapter we’ll briefly discuss what these works meant to the development of our solution.

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

After that, we have the evaluation chapter, were we present our evaluation stage, with a 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 regarding to the developed work and how our evaluation helped on identifying areas for future work.
Chapter 2

Background

In this chapter we look at the more theoretical aspect of our work. The first topic is Level Design, how we can evaluate a puzzle in a map in terms of length and backtrack and what topology of map is useful for us to use. After that we will look at Creativity, what do we understand by it, relevant concepts and how they relate with each other and what creativity means in computer terms. Finally our third topic is Procedural Content Generation, we will discuss what is meant by PCG and how we can see it as a way to stimulate creativity.

2.1 Level Design

“Level Design is where the rubber hits the road.” – by Jay Wilbur in conversation with Cliff Bleszinski, as reported in [3]

Level design is one of the key aspects of the game design process, and as Totten, C.W. mentions it in his book [4], “it is also one of the most exhilarating”. Rollings, A. and Adams, E. argue on their book [5] that “there is no standard way to design levels”. So having a key aspect as this and no standard way to evaluate the quality, how can we evaluate it? Totten, C.W. tries to answer this by saying “as you watch people play your level”[4] we must keep these, among other reasons, relevant to our study:

- **Do they understand how to play the level?** Meaning, that teaching is an important mission of level design, and if a player does not understand a puzzle type that we want to repeat during the game, the player may need a better or more transparent introduction to the mechanic.

- **Is the level too hard for the player?** It must be avoided sudden increases in difficulty without proper balancing or player preparation. If a player is stuck in an early puzzle of the game, we may need to place in a later time of the game, or build easier puzzles to prepare the player.

- **Embrace happy accidents.** Sometimes players may resolve puzzles in an unique way that the designers did not expected them to.

For our project we will keep simple individual puzzle pieces, pressure plates and gates, that the player by looking at it can understand how to interact, but create a more complex puzzle made of simpler smaller ones. The difficulty for a player is a subject of matter, but we want to achieve with the level of puzzle difficulty, an easier one that has overall less steps to complete than a harder one. The happy accidents, we are not expecting to see any, but if they occur, it can be a great way to motivate creativity. A misplaced puzzle piece, that can be accessed too early and leads the player to skip most of the puzzle objectives that he needs to complete can generate a new idea with this new puzzle solution.
In terms of layout, we will use Legend of Grimrock 2 in game layouts, since it is a dungeon crawler, the predominant game space structure is a Maze. Totten, C.W. describes mazes[4] not as unicursal labyrinths, but as “branching spatial puzzles where occupants and players must find their way through an elaborate structure of walls and pathways with multiple dead ends to find an exit point”. He also points that due to their branching nature, mazes are multicursal, by having more than one defining path. Due to this nature, it has potential dead ends that implies a risk-reward structure, where the player must choose from different uncertain options and hope to choose the most advantageous one. This type of game space structure is a good choice for puzzle creation, because the player must explore the maze to solve the puzzles, and face the uncertainty in order to proceed with the game.

Another interesting idea for our approach of level design, came from an Youtube series, called Boss Keys, from Brown, M., a game journalist that wrote for various important gaming news websites. In this video[6], he explains The Legend of Zelda: The Minish Cap's dungeon design and to do so, he converts the dungeons layouts and puzzles into graphs. We can use a similar approach to our algorithm, convert the level layout into a graph, since it will be a fixed layout it will be easier than a random generated one.

![Image](image.png)

(a) Caption of the graph, a square represents a lock and the diamond represents the key/item/switch to open the lock of the same color and image.

(b) The Legend of Zelda: The Minish Cap's Deepwood Shrine's graph made by Mark Brown.

Figure 2.1: Brown's graph representation of a dungeon from Legend of Zelda: The Minish Cap.

2.2 Creativity

"Creativity is a puzzle, a paradox, some say a mystery. Artists and scientists rarely know how their original ideas come about. They mention intuition, but cannot say how it works.” – Boden, M. A. [7]

The first question we must ask is “what is creativity?” According to Boden [7], creativity is “the ability to come up with ideas or artifacts that are new surprising and valuable”. “Ideas” include concepts, theories, etc, things in a “mental” level, “Artifacts” are the manifestation in concrete things of these ideas, life paintings, sculptures, among others. On that note, with these examples, creativity exists virtually on every aspect of life, it is not a special ability, but another aspect of human intelligence. This means every human being can be creative. Boden [7] distinguishes creativity in “psychological” creativity and “historical” creativity (P-creativity and H-creativity respectively). P-creativity involves “coming up with a surprising, valuable idea that is new to the person who comes up with it. Does not matter if another person had that idea before. On the other hand, H-creativity means an idea “no one else had it before: it has arisen for the first time in human history”

For this work, we will focus on the psychological creativity, because due to the nature of our project, a set layout and a reduced, compared to the Legend of Grimrock 2 Editor, set of items, we do not have

---

1Nintendo, 2004
the objective of creating something really unique on the context of all the possibilities of the game, it can be something that happens, but we want to focus on inspiring a non-professional level designer to create something innovative for him, even though someone else had a similar or equal solution.

Boden [7] categorizes creativity in 3 forms:

- **Combinational creativity**, that involves unfamiliar combinations of familiar ideas;
- **Exploratory creativity**, are new ideas from a known conceptual space;
- **Transformational creativity**, that is the alteration of the conceptual space itself, giving it a new perspective and leading to new ideas that were unthinkable.

Our work will lean on the exploratory creativity: the Legend of the Grimrock 2 Editor has a finite number of objects for puzzle creation, and it is up to the designer how to combine them to create the puzzle(s).

**Lateral and Vertical thinking.**

de Bono, E. introduces lateral thinking[8] as a close relation with creativity in a way that creativity is most of the time merely the description of a result, and lateral thinking is the description of a process. He also states the intention of lateral thinking is to update currently established ideas and preconceptions. It attempts by challenging old ideas, weather with a different idea or with new information. In sum, by bringing conflict in order to question the existing assumptions. Vertical thinking is a very strict and well defined process where each of its steps are dependent on the validity of the previous one. It always chooses the most valuable alternative, over all others, and always follows a path as long as exists a more valuable alternative down that path. De Bono states in [8] that “Vertical thinking moves only if there is a direction in which to move, lateral thinking moves in order to generate a direction”. Lateral thinking it is all about generating conflict, in order to generate new paths of alternatives.

De Bono presents[8] seven types of lateral thinking techniques:

1. **Alternatives**, using concepts as starting point in order to find obscure alternatives
2. **Focus**, when and how to change the focus of our thinking
3. **Challenge**, break the boundaries of traditional thinking making way for different ideas even if not ideal
4. **Random Entry**, using unconnected input to open up new ways 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
7. **Treatment of Ideas**, treating ideas to fit into a specific situation

**Computational creativity.**

“[Computational creativity is] the philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative.” – Colton, S. and Wiggins, G. A. [9]
As Ventura, D [10] suggests “if games are the “killer app” for computational creativity, then perhaps computational creativity is the future of games.” based on Liapis, A et al. [11] work. So by “embracing computational creativity, game developers can build computational collaborators that take real creative responsibility as a member of a team”, he goes even further by saying it could build a fully autonomous content, but that is drifting away from the scope of our work as we want a “peer” solution.

The solution produced using computational creativity cannot be classified as “good enough” or the “best solution”, instead we qualify as novelty and utility or value[10]:

- **Novelty**: the quality of being new, original or unusual; this is relative to the population of artifacts in the domain in question and can be applied on the personal or historical sense.

- **Utility or value**: the importance, worth or usefulness of something; this would be typically ascribed by practitioners of the domain in question.

For our project, we want the AI to produce novelty content, having some value even though it can be considered not the best solution, but it can inspire an Human colleague to improve on that.

**Creativity-enhancing computer models**

Lubart[1] describes creativity-enhancing computer models and, is addressed in this work[12], they categorize the role of the computer as a way to support human creativity in four categories, stating that it 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 this categories Lubart [1] translated into four lines of thought a computer’s role in creativity enhancement:

- **Colleague paradigm**: or a “peer”, where the creative process would be a shared activity between the human and the computer.

- **Nanny paradigm**: in this paradigm, the computer takes a more passive role in the creative process and is used as a manager of the user’s productivity by detecting periods of procrastination and productivity breaks.

- **Pen-pal paradigm**: the computer assists the user by promoting communication, mainly in a collaborative creative process with two or more individuals. This communication is a fundamental tool to fulfill the need of idea sharing and synthesis.

- **Coach paradigm**: in this paradigm it is proposed an user’s cognitive domain and narrowed thinking style, that can be inconvenient and can hinder his performance for a certain task. For this reason, the computer can assume the responsibility of kick starting a sentence of topic to the creative process, as well as offering information about existing techniques to stimulate creativity, thanks to its knowledge in creativity-relevant techniques.
2.3 PCG

Procedural Content Generation (PCG) is the generation of content using random or pseudo-random algorithms, resulting in unpredictable results. Various games use this method of generating content such as Rogue\textsuperscript{2}, Elite\textsuperscript{3} and in some more recent cases like Terraria\textsuperscript{4}, Path of Exile\textsuperscript{5} and No Man’s Sky\textsuperscript{6} to name a few. The main reason to use this technology is the amount of replay value that it can offer, without requiring a designer to create levels and content by hand. Related to this technology Karavolos, Daniel et al. [13] wrote about a “search-based generative method which created game level by evolving the intended sequence of player actions”. This is one of the objectives of our project, to create a “sequence of player actions” or in other words, create puzzles for a player to resolve a sequence of actions. We will not use the same game, the game Dwarf Quest\textsuperscript{7} but we want to have a similar result applied to Legend of Grimrock 2 in the puzzle generation. In 2011 Togelius et al. wrote an article\textsuperscript{14} about the Procedural Content Generation. They defend that PCG techniques are evaluated according to these aspects:

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

Let us look in detail to each aspect:

**Online vs Offline:** This first distinction is whether the generation of content is performed during runtime of the game (online) or during its development (offline).

**Necessary vs Optional Content:** This aspect is related to if the generated content is necessary for the completion of the game, like a key for a door for example, or if it is optional and the player can skip this content, like a weapon for example, the player can choose not to trade weapons.

**Random seed vs Parameter vectors:** The algorithm might take only a seed to its random number generation or take multidimensional vector of real-value parameters.

**Stochastic vs Deterministic Generation:** This distinction is whether the algorithm using the same parameters produces always the same result (deterministic) or if from another hand the results are different even when using the same parameters (Stochastic).

**Constructive vs Generate and Test:** The final distinction is whether the algorithm generates the content once, and stops, after verifying if it is correct of “good enough” while it is being constructed. Or if from another hand, after creating the content, it tests it according to some criteria and if the test fails, all or some of the candidate content is discarded and regenerated, until the content is good enough.

According with these aspects, our PCG will be offline (created when using the Legend of Grimrock 2 editor), a mix of necessary and optional content (depending of the algorithm, one may take into account the necessary pieces while others only focus on more or less equal to the designer’s creation, generating optional content), parameters vectors (we have parameters like length and backtrack of the puzzles), stochastic (even with the same parameters we want different puzzle options) and finally it will be constructive.

\textsuperscript{2}M. Toy and G. Wichman, 1980
\textsuperscript{3}Acornsoft, 1984
\textsuperscript{4}Re-Logic, 2011
\textsuperscript{5}Grinding Gear Games, 2013
\textsuperscript{6}Hello Games, 2016
\textsuperscript{7}Wild Card Games, 2013
2.4 Discussion

On this section we’ve reviewed some of the work that we have based our project in. Related to our work we tried a similar approach to Brown’s video[6] and translate the dungeon to a graph and implement that view on the GAF. On the Procedural Content Generation aspect as discussed before our tool is offline, a mix of necessary and optional content, parameters vectors, stochastic and constructive. Finally on the Creative side of our work, we will focus on the Exploratory creativity, but we have some space for Combinational creativity, since the user/tool can re-arrange the puzzle order to create a new puzzle overall. Our study will focus also on the Lateral thinking line of thought, we want our work to stimulate Alternatives, from the user using the tool from an empty layout to give the first solution and for the user to try to find alternatives for that first solution. We want the user through the tool, to find that perhaps a particular length or backtrack setting of the map is not the most accurate for what he hopes to achieve and by experimentation from using it to generate solutions and iterating them, he decides to change the Focus completely and change the tool settings to a different set of solutions. Through the generation of different solutions, sometimes, the generated result seems random to the user, but to the algorithm it makes sense. With this Random Entries, perhaps they can spark new ideas to the user and even change Focus with them. Through Provocation and Movement we want that the tool generates provocative solutions, totally different from the user’s solution, and if he wants to, building over them, or it can be an exaggeration of the users’ settings. With all the created solutions, the user can Harvest all of them and maybe come up with a solution of his own. And finally, with the Treatment of Ideas the user can have a more restrictive approach, and try to fit the generated outcome into what he wants by having what he created have more weight on the generation of solutions and so the tool will work with the previous objects that the user deployed on the level with bigger priority and re-arrange them and even add new objects to suit that the user wants to create.
Chapter 3

Related Work

In this chapter we will talk about relevant work from which we took inspiration to help support our decisions. We will analyze them by topics and discuss why they are relevant to our research and development.

3.1 Creativity-enhancing computer models

From the work written by Lubart [1] and mentioned in [12] it's possible to categorize the computer role as way to support human creativity. The classification is translated into four lines of thought about the role of the computers in creativity enhancement:

1. Nanny
2. Pen-pal
3. Coach
4. Colleague

In the nanny paradigm, Lubart refers perseverance as one of the most frequently cited attribute for creativity. In this case, the computer has a more passive role in the creative process, being used in the management of the user's creative work, like reminding the user to take breaks from work to prevent fatigue that can hinder creativity. As a pen-pal, the computer assists the user by promoting communication between individuals. Communication in a collaborative creative process is a key tool to fulfill the need of idea sharing. For example online chat and conference software that let people from across the planet contribute with ideas for a project in real time. The third paradigm is of the computer as a coach. In this case the computer can help an individual to have different ways of coming up with creative ideas. If the said individual is interested in trying to use a certain cognitive process such as divergent thinking, for example the computer can provide tutorials and exercises to use it. Finally we have the most ambitious role: a computer that acts as a colleague. This situation envisions a real partnership between a human and a computer, where both would share the creative process. The user would have the initiative to start the process and the computer would make a suggestion or modification, randomly or heuristically, until a meaningful outcome is produced. Computers can not produce creative ideas as good as a Humans plus, a human is still needed to refine those ideas, they excel at exhausting the search-space. Due to this fact, a computer can help to achieve unconventional and potential valuable raw ideas that the user can fine-tune into a coherent idea.
We have analyzed the existing solutions that take this into account. The Sentient Sketchbook \cite{15} needs a degree of pro-active action from both, the computer and the human, even though it does not necessarily need to be on the same level. For example, the authors mention clear differences in the PCG area by saying that professional tools such as Garden of Eden Creation Kit\footnote{Bethesda 2009} or game engines like the Unreal Development Kit\footnote{Epic Games 2009} provide a great utility to the designer by allowing him to speed up the game development tasks, they are limited only to interpolation, path finding and rendering, and the human initiative is the solo driver in the creative process. On the other hand, PCG content-oriented tools like SpeedTree\footnote{IDV 2002} can create large amount of game content, in this case trees, but the human’s initiative is limited to choosing parameters for the generation algorithms and editing generated artifacts in the end, with no interaction during the generation process. Sentient Sketchbook tries to explore that space in between the two approaches to PCG. It generates suggestions by mutating the designer’s sketch and exploring those alternatives to present playable solutions to the designer.

Another tool that we looked at is Tanagra \cite{17}, another mixed-initiative co-creative design-assisting tool. It has a different behaviour, it focuses on generating content for platformer genre games. The designer specifies the position of key-platforms and the tool creates the remaining map topology. It works in a iterative way, meaning it goes back and forth between the designer and the computer, it takes eventual alterations by the designer and rapidly generates sections of the level, to accommodate those new decisions. Smith et al. say that a mixed-initiative approach to level generation requires a new set of techniques more suited to a real-time nature of a design tool. Evolutionary algorithms have been successful for offline level generation but they rely on an unbounded search process that makes them too slow for a real-time tool.

Lastly, the tool that directly inspired this study, Luca’s Editor Buddy \cite{2}. This tool needs a pro-active interaction from both parties, the computer and the designer, where this last one sets the configuration that he wants to the level layout and when there is a change on the Legend of Grimrock 2 Editor, it triggers an evolutionary run on the tool’s algorithms. The Editor Buddy only calculates a new layout when there is a change on the editor, by the designer interaction directly with it or by using the tool to export a partial or total solution. There is also the option of a more passive tool behavior where the user can deactivates it running automatically when there is a change on the editor, but only after the user clicks a button.

Lubart’s addresses a crucial point by stating that there is no “average” user. There is a large number of different perspectives on how a creative computer model should perform and every user has different personalities, thinking styles, experiences or even different moods that can impact the creative potential of each user and can affect an interactive co-creative session between them and a computer unfolds. Some people may prefer a paradigm over another or even feel that is a step backwards from a real interaction between another human being and have a real discussion.

### 3.2 Computer Co-creativity

In this section we will focus on the subject of computer co-creativity. Yannakakis et al. analyze in their paper \cite{16} the role of both humans and computer during a Mixed-Initiative Co-Creativity. They draw direct connections with lateral thinking , diagrammatic reasoning and Mixed-Initiative Co-Creativity (MI-CC) Yannakakis’ work refers lateral thinking as fundamental contribute to boost creativity. The lateral thinking principle “random stimulus(entry)”, that relies on the introduction of an external element with the objective of disrupting preconceived ideas and habitual patterns of thought by forcing the user to
integrate and/or exploit that external element in the creation of an idea or solution. In this case, the main source of random external elements would be the computer AI, that would not generate completely random stimuli necessarily but ones that can be heuristically-driven to better adapt a on a particular task/problem.

The other important element of this study is diagrammatic reasoning that is the use of visual representations in the reasoning process, a cognitive procedure which is enabled during game level design for example. As literature suggest it is based on the idea that complex information processing tasks performance is improved by the use of some kind of visual or illustration like diagrams. For example assembling an object, one would find easier it is has a step by step assembling visual representation. By combining these two concepts, we have the notion of diagrammatic lateral thinking. This combination means that the activity of lateral thinking is aided by visual stimulation.

Vile and Polovina’s study [18] on the use of diagrams says that the process of constructing a diagram may be, in the end more important than it is result, by stating that “Conceptual graphs are an inexact, incomplete model of social and cultural contexts, these graphs may provide a vehicle, or toolkit that mediates internalization, hence generate thought.”, from this perspective a conceptual diagram can never exactly represent a concept, but in the process of constructing it helps the constructors of said diagram to get a better “idea” of the concept and its context.

Here is the example in the figure 3.1 of 3 applications that put this theory to practice: Sentient Sketchbook [15], Tanagra [17] and Editor Buddy [2]

![Sentient Sketchbook user interface](image1)

(a) Sentient Sketchbook user interface

![Tanagra user interface](image2)

(b) Tanagra user interface

![Editor Buddy user interface](image3)

(c) Editor Buddy user interface

Figure 3.1: Stimulating user creativity through diagrams
3.3 Evaluating Computer Creativity

Another very important topic is how creativity evaluation is actually performed. Brown, D. says in [19] that there is no such thing as a creative computational system, only that the artifacts that are produced can be evaluated as creative or not. It means that the creative value comes from the final result of a computer design system as opposed to the process itself. For that reason we need to establish evaluation metrics that allow us to qualify the creative value of a certain artifact. Brown refers various factors relevant to the evaluation process such as:

- The knowledge, experiences, context, feelings and preferences of the evaluator;
- The type and degree of exposure to the thing being evaluated;
- The “norms” for the relevant population to which the evaluator belongs to (e.g., novices, experts).

The author recommends the use of surprise, described by Boden [7], as a way to evaluate the creativity of artifacts instead of novelty, value or utility.

In the particular case of Mixed-Initiative Co-Creativity [16], this evaluation it is not trivial on an ongoing interaction between human and computer because it is hard to represent it in the final outcome. Yannakakis et al. explains that is difficult to capture the impact of pro-activeness of the computational activity on the human creativity and vice-versa. Due to this, they consider two types of evaluation when a computational creator is involved: the evaluation of the final (or intermediate) outcome, through the use of a number of heuristics for the task at hand (such as novelty and usefulness) or through crowdsourced estimates of creativity from a human audience; and the evaluation of the co-creative process for the generation of outcomes, solutions or items. This evaluation is less straightforward because the exact human creativity processes are either completely unknown or only partially known. It requires that we identify milestones within the co-creative process through heuristics of novelty, value, surprise or other relevant heuristics. Or, some type of meta-level mechanism of the quality of the process. Or, finally, an evaluation based on a temporal model of the co-creative process. They argue that MI-CC supports and fosters the creative process towards a certain outcome in addition to fostering the creative value of that same outcome.

Relative to the evaluation of computer generated content in the context of videogame levels, Liapis, Yannakakis and Togelius in [20] introduce three generic metrics for levels in games: Area Control, Exploration and Balance. In the context of their work, they derive two formulas used to evaluate game levels according to the aforementioned metrics, each for a different game genre: multiplayer strategy game maps and single-player roguelike dungeons. Amongst other reasons, they talk about the importance of design patterns as a way to help in the study of existing games as well to teach game and level design. As a way to help the recognition of design patterns and, consequent, evaluation, they introduce the notion of map sketches, that they have presented them in [21] as a way to abstract complex game levels into their basic building blocks, which isolate the essential game mechanics from thematic context and details. This abstract visualization and compact size of map sketches allow both the human designer and an algorithm (an artificial designer) to process and edit these map sketches with less effort.

3.4 Search-based creativity

We can translate the problem of creativity at the exploratory level into a constrained optimization search conducted on the conceptual space, according to Boden’s [7] work and Wiggins’ [22] consequent formalization of it. This 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 space.
Procedural dungeon generation

In the work of Bidarra et al. [23] there is several interesting methods on how to perform procedural generation of dungeons. They say that due to how dungeons combine pace, gameplay and game spaces dungeon levels are good candidates to display the capabilities of PCG.

The authors describe common practices in procedural dungeon generation and their pros and cons, they serve as a valuable reference in regards how they compare with each other. From the listed methods of procedural dungeon generation, we took particular interest in those that use Genetic Algorithms. Here is a brief description of each solution.

The first one belongs to Hartsook et al. [24] and how they present a way to procedurally generate a role playing game world based on a story, as opposed to conventional 2D or 3D content generation, using a metaphor of islands and bridges to create a space tree. The genetic algorithms in this case, specifically the mutation and crossover operators, handled the addition and deletion of nodes and edges in the tree. This tree would later be translated into a grid in order to be used as basis for a map.

The next detailed implementation was by Valtchanov et al. [25] 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. They use a fitness function that has a strong preference for maps composed of small, tightly packed clusters of rooms which are interconnected by hallway paths. These generated maps have no explicit gameplay-oriented motivation, because of this, these maps tend to converge into tightly packed room clusters, sooner or later.

The third mentioned implementation is that of Ashlock et al. [26] in which he lists four distinct ways to explore the generation of maze-like dungeons with genetic algorithms:

- Direct binary: gene with bits, representing a wall or accessible area;
- Direct colored: gene with letters that represent colors;
- Indirect positive: chromosome represents structures to be placed;
- Indirect negative: chromosome represents structures to be removed.

Bidarra et al. draw out attention towards the direct mazes for being the most interesting. These are made up of a grid, where the operators of crossover and mutations are used to flip cells into becoming a wall or a accessible area.

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. Unfortunately even though they show promising potential in supporting several types of content to be generated, they require the user to find or creation of a suitable fitness function, and that can be a hard task, especially for designers with no background in programming or mathematics.

Bidarra et al. refer the Sentient Sketchbook[15], they underline how it handles the emergent problem of genetic algorithm parametrization by exploring a more gameplay-oriented fitness evaluation.

Lucas’ Editor Buddy [2] has a grid-like approach like Ashlock et al. work but he has a more gameplay-oriented focus like the Sentient Sketchbook, meaning that the genetic algorithm operators of mutation and crossover flip the cells accessibility (walls or walk-able paths) but using it the users does not need to parameterize a fitness function, only to select the objective for the algorithm (rooms or corridors) and select the weight of each of the three algorithm (objective-driven, innovation-driven or current user map driven).

The last work worth to be looked at is the one from Baldwin et al. [27], the Evolutionary Dungeon Designer (EDD). They use Fi2pop (Feasible-Infeasible Two-Population) to operate over a previously gen-
erated map to produce optimal suggestions for the distribution of all map elements for each room. This includes the placement of treasures, enemies, doors and walls.

These rooms are evolved from a randomized starting population based on the user configuration for the tool. The user can select the difficulty and quantity of treasures and enemies, among other things. They have encoded the population individuals in the following way: A two-dimensional \( m \times n \) array of integers, this integers range from 0 to 5, corresponding to the 6 tile types:

- Floor
- Wall
- Enemy
- Treasure
- Entrance
- Door

Each room has between one and four doors, which one is the entrance. For that room to be playable, or valid, it must exist at least one treasure, one enemy and a path between the entrance and all other doors, treasure and enemies. The algorithm must also detect fundamental design patterns [28]: micro-patterns corridor, connector and room (referred as chamber in the work to avoid confusion).

**Objective-driven and novelty search**

The work of Liapis et al. [15] discuss theirs findings in experimenting with Fi2pop [29] and FINS [30], both produced good results in the paradigm of exploratory and transformational creativity, considering the extensive search space. We are most interested on how these algorithms behave, for example in regard on how infeasible individuals are handled between generations as well as objective-driven and novelty search.

With a standard Genetic Algorithms, when the parent selection phase is reached, the most fitted offsprings are usually kept for the breeding of the next generation and the remaining ones are discarded.
Instead of immediately discarding the infeasible offsprings, the Feasible Infeasible 2 Population GA - a variation of the Standard Genetic Algorithm - maintains two groups of individuals, one for the feasible and another for infeasible individuals. The theory behind this lies in the potential for valuable, previously unforeseen, feasible solutions within unexplored infeasible individuals. Because of this, individuals in the feasible population evolve towards maximizing their fitness function, whereas the ones in the infeasible population evolve towards minimizing their distance from the feasibility border, attempting to become feasible and earn a place amongst the feasible individuals.

Novelty search is also a motivating paradigm of this work. We want to place the computer in the role of a colleague, something that can challenge us and feed us interesting suggestions and collaborate with us towards a common goal. This is interesting when considering how novelty search complements the objective-driven implementation by showing “great potential in domains where a fitness function is deceptive, difficult to quantify, or subjective” [31]. Novelty search algorithms, instead of evolving towards complex fitness functions, they evolve towards diversifying the solutions in a populations, by selecting an individual according to its novelty score, which is the average distance between the individual and its closest neighbors. In their work [32] Preuss et al. attempt to conciliate novelty search with objective-driven search and search for both good and diverse game levels. Even though the definition of “diverse” or “novelty” can be quite subjective, they present three different measures to novelty, in the context of roguelike, first-person shooters, strategy games and platform genre games:

- **Tile-based** diversity: difference is measured in the most low-level way, as the number of map tiles that differ from one map sketch to another.
- **Objective-Based** or quality-based diversity: difference is measured in terms of the same quality metrics they use for searching for good maps.
- **Visual impression** diversity: visual features are extracted from the maps through metrics related to balance or grouping, and measure diversity along those metrics.

### 3.5 Discussion

In the first section of this chapter we address an important topic of our work, how are we effectively using computers for creative purposes. For this work, we wanted to create an experience where the user would work alongside with the AI during the whole puzzle creation activity, much like the colleague category identified by Lubart [1]. So by having the user adapt the behaviour of the AI to their own objectives, this could invalidate the necessity for a perfect creative computer program, something that Lubart seems to defend as unlikely: “[...]no single HCI solution can be expected to be optimal in all circumstances.” We described how current solutions, such as Tanagra and the Sentient Sketchbook, behave in this sense. The Sentient Sketchbook works in a more independent way, where each sketch the designer makes, the tool presents several playable alternatives after some time. Tanagra is a more interaction-centric experience where it generates content in a more quick and reactionary way. Lucas tried to combine both previous solutions, for an agile interaction cycle between the designer and the computer as well variety of content, by adapting the content generation process to a step-by-step model where the algorithm gradually improves it’s solution.

We took a different approach to the one of Lucas, we wanted the user to be pro-active on when our tool would make suggestions, because our work is focused in puzzles for a level and we think on the different smaller puzzles of a map as a whole puzzle itself, so an algorithm that would react to each interaction of the user with the editor would go against that, only when the user finishes and is satisfied with his solution, then the tool would run and make its suggestion.
We still want to maintain the back and forth interaction envisioned by Lucas, because of this we aimed for the colleague computer role, as described by Lubart[1], to foster creative outcomes through “semi-random search mechanisms to generate novel, unconventional ideas”. We did not try to generate new puzzle ideas on pure randomness, but use the micro-patterns described here[28], in our case door and key (Pressure Plate in the context of our work) to find an accurate solution in terms of puzzle length and backtrack to inspire the designer.

Evaluation is an important part of this interaction, it can be divided in two moments of evaluation, the quality of the work generated from the co-creative process and the quality of this process itself. We are equally concerned on generating good content for the benefit of the designer (and the player), but also providing an consistently fruitful and interesting interaction between them. As said before, we opted to a more active role on the user-side, to trigger the computer generation process, because of said idea of the small puzzles of a level being treated as a whole, so mid-user interactions suggestions would not fit well in our view, but there is the performance concerned, the default settings of our tool generates a puzzle in about 5 seconds max, it is not good a real-time interaction, but we think it is good for suggesting solutions after the user interaction with the game editor is done. So with this more “user active” approach we can explore more results in the algorithm level to expand the range of our metrics for a puzzle, length and backtrack, to give a more accurate result according to the designers choice.

The work of Lucas with the Editor Buddy[2] was the main inspiration of this work. We followed his view of the interface and the behaviour and adapted to our needs. This work can be seen as an extension to his previous work, were he focused on the layout of a dungeon, our work focuses on the puzzle of said dungeon, we believe both works can complement each other. Also for this work we also focused on the findings of Liapis et al. regarding constrained novelty search and the aspiration to find both “good” and “diverse” game puzzles, and whatever both connotations mean for the designer.

In the end, the behaviour of our application is handled not only by novelty and objective driven algorithms, like Lucas, but also a third one based on the user current solution where we tried to expand the possible space of solutions by having a third evolutionary algorithm.
Chapter 4

Solution Model

In this chapter we will go into detail in our solution model and we will begin to explain the importance of the computer colleague paradigm that we are focused on replicating and what that means to our model itself. After that, we describe our model by components and present an overview of their elements and what purpose do they serve. Finally, we address how our solution is integrated with the level editor we chose, which we will formally present it in the next chapter.

4.1 Computer colleague paradigm

There is a particular paradigm that we are trying to represent with our solution and it is important to emphasize it so we are able to comprehend the relation between content it generates, it interface configuration and its iterative puzzle design cycle. To better impersonate a digital "peer", or to at least try to have similar behaviors, our solutions works on three different domains:

- **Objective**;
- **Innovation**;
- **User Map**.

Since multiple domains can be used at the same time, we expect to find discrepancies between suggestions generated and what the user expects to be generated. These suggestions may be perceived as unwanted or wrong, however, the purpose of this work it is not to give an absolute answer to what the user seeks, but to explore the ability to foster creativity, by allowing potential multiple useful suggestions to appear. Even with these “wrong” suggestions we hope to disrupt potential preconceived ideas or objectives in the user’s mind and perhaps trail a more interesting path.

In the end, the designer will still be given all the means to guide or orient the type of content generated by the program (like asking a human partner to give suggestions more focused on a specific topic). For example, if a level designer feels more receptive to a wider array of suggestions with potential opposing traits, he can make these domains guidelines more lax, on the other hand, if he wants to explore a particular type of suggestions that he can be more strict on how he sets up these domain guidelines. He can either accept the suggestion and export it to work on it (or even consider it a good solution and accept as the final suggestion) or discard and generate another suggestion. Figure 4.1 represents an example of an interaction between a level editor, where a designer creates a puzzle, and our solution. In theory, this concept is generally applicable to other level editors that support puzzles (using the door and key micro-patterns[28]), other than the one we chose.
4.2 Solution Model

The developed solution, a new mode for the previously created GUI based application by Lucas\cite{2}, named Puzzle Mode (Editor Buddy Puzzle Mode), where Figure 4.2 represents its interface. The application's primary goal is to foster creativity during a level puzzle design activity by presenting visual suggestions according to the designers settings. The interface serves two purposes: it provides the level designer with visual information, in the form of the accessibility Graph names (Map) Sections’ Graph and the Map representation with optional toggles to show Pressure Plate and Gate connections, and a way to guide the application’s behavior towards generating a specific content. The Editor Buddy Puzzle Mode(EB Puzzle Mode) behavior is defined by four genetic algorithms, that work in a three plus one (3+1) manner, that we will go into a detailed explanation in the next chapter. Generated suggestions are displayed to the designer in the interface using a 2D preview of the map with the puzzle elements and the corresponding Sections’ Graph associated to the map. The designer can also easily export the displayed suggestions or parts of it, to his current level.

4.2.1 Interface

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

- Controls for defining the application’s behavior. The Objective, Innovation and User Map that control its respective algorithms;

- An Objectives’ section where the user defines the Objective Algorithm behaviour in terms of searching of a puzzle that creates a long or short puzzle, with more or less backtrack and the preference between either setting;

- 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 run the algorithms, interact with the generated suggestion, analyze a map and re-open the section’s graph window;
• Four checkboxes of optional settings;
• A logger that shows extra information from the map to the user;
• Four progress bars to inform the current processing state for each algorithm.

Figure 4.2: Editor Buddy Puzzle Mode Interface
Controls are shown in the form of sliders for the user fine-tune the EB Puzzle Mode behavior. Suggestions are displayed in the 2D preview canvas. Each map section or corridor has a distinct color related to it, that corresponds to the same color node of the Sections’ Graph. The light grey tiles are non-walkable ones. 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, and also represents with a “G” for the gates and “P” for the pressure plates of the map. The left and right arrow buttons, navigate through the suggestion historic, allowing the designer to revisit previously displayed suggestions or imported maps from the games editor that were analyzed and, a first and current or last suggestion arrow.

The user can make a selection on the 2D canvas with the left mouse and deselect with the right mouse button to select the area that he wants to export. This selection can also be cleared using the clear selection button or inverted using the invert button.

The four checkboxes control optional settings:

- The “Show Corridor” shows in black the corridors of the map, shown in Figure 4.4 that are the relevant zones to place a gate for our algorithm;
- “Show Gate/Key connection” shows with a black line like in the Figure 4.5;
- “Show Graph for Optimal Path Only” toggles between the Optimal Path Graph (Figure 4.3a) if checked and the Complete Graph of the level (Figure 4.3b);
- “Export only connected Keys” when checked exports only the Pressure Plates(Keys) that are connected to a gate;
4.2.2 Behavior

It was important to us that for this project, the Editor Buddy (Puzzle Mode) acted as a digital “peer” (or colleague) during the puzzle design activity, working alongside the designer suggesting eventual improvements or alternatives to his work. It tried to do that using four Genetic Algorithms (GAs) and taking into account the state the designer’s layout at a given time in three of them. These algorithms are executed on separate threads and on three of them evolve 3 pseudo-random populations of individuals, and the fourth one receives a percentage of the top individuals of each previous computed algorithms, hence the three plus one (3+1). The first three algorithms are Objective, where the fittest individuals are the ones that are closest to a particular objective(such as more or less length and/or more or less backtrack). The second of these three algorithms is Innovation, where the fittest individuals are those that have a most distinct number of map elements in each room/corridor from the designer’s puzzle. And the last one is User Map, that is the opposite of Innovation, that means the fittest individuals are the ones that have a more equal number of map elements collocation in each room/corridor from the designer’s puzzle. Additionally there is the said extra algorithm, called Combined Algorithm, that blends
a percentage of each fittest population and evolves them towards a combination of the previous three algorithms fitness functions, which weight for them is set by the user’s settings on each of the sliders for the previous three algorithms.

In all of the four algorithms, only the Objective component (either the algorithm itself or the fitness function on the Combined algorithm) need to be a feasible puzzle, meaning the user must reach the exit from the start, Innovation and User Map component only look to a more distinct or equal number of map elements per section, respectively, independently if the map is or is not feasible.

Each run is independent, meaning no individual from a previous run is used on the new run, seeing that there is a reuse of the three first algorithms individuals on the combined one, we decided to make each run disconnected from the past one and, since we have a historic of solutions that, the user can navigate through and choose a previous one that he wishes to work one made sense for us to make each run completely unrelated to previous. Also, we can have multiple possibilities for the same objective that are distinct from each other, possibly, and so the user, again using the historic navigation, he can run the algorithm some times and choose the option that he prefers for example.

The user can take advantage of the EB Puzzle Mode flexibility, meaning he can inhibit some of the tool’s behaviors if, for example, he wants only innovative suggestions towards a certain objective, he can put the User Map slider into zero and tune only Objective and Innovation according to his idea, or in other hand only want puzzles that are related to his elements disposition, regardless of being feasible or not and turn off objective and innovation and only have user Map with a value higher than zero.

The EB Puzzle Mode displays in his 2D canvas each element addition that the user makes on the editor, but it only runs the algorithm, or analyzes the map for the most efficient connection of pressure plates and gates if the users chose to do so. Unlike Lucas our approach is more user active than his, because of our view of puzzle design, that the map puzzle is a combination of smaller ones that makes since when the smaller ones are created, so only when the designer finishes doing his idea of the overall puzzle he can than run the EB Puzzle Mode's algorithms for suggestions or ask for a suggestion for a gate/pressure plate connection. There is also the previous reason of the performance, each run on the default settings takes about 5 seconds that is not that good for real time interaction. Even with this more user active approach we think that Luca’s philosophy of turn-taking between the EB Puzzle Mode and the human user is not lost, but for our context, puzzle creation, we think this more user active role makes sense.

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 and also create a map with a specific cell layout. This layout is based on the Trickster’s Lair map from the Legend of Grimrock 2[^1], see Figure 4.6 for the specific modifications. 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 the puzzle elements addition 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 EB Puzzle Mode’s suggestion, using the available interface tools. The EB Puzzle Mode handles the automatic saving of the level, so when there is anything new on the designer’s end, the EB Puzzle Mode’s only shows on the 2D canvas the added puzzle elements, this can be useful to see if a gate is

[^1]: Almost Human Ltd., 2014
placed in a corridor and the pressure plate in a room, no gate/pressure plates are updated though, the designer must actively analyze the map for the EB Puzzle Mode to create the most efficient connections. When exporting changes to the designer’s level, the procedure is 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 EB Puzzle Mode also handles the automatic reload of the updated map file, so the level editor view reflects the performed changes.

4.3 Summary

By the end of this chapter, our objective was to provide a clear overview of our proposed solution model, including its different components and their respective elements and functions. Regarding the applica-
bility of this model presented in our diagram, we compromised with a specific level editor, however, in theory it can be used with similar others if they have puzzle elements.
Chapter 5

Solution Implementation

In this chapter we go into detail on how our solution is implemented. We start by presenting the Legend of Grimrock 2 (LoG2) game and its Level Editor and after, we follow with presenting the implementation of our solution, where we talk about problem specific constraints regarding our approach, the genetic algorithm framework that we used in our implementation and the algorithm we developed to categorize a map according to its puzzle. We also detail how the execution cycle of our solution works as well as the more technical aspects of this cycle. After that we present the used algorithms and their specific parametrization, explaining why they were chosen. The last section is dedicated to describing several design choices to help support the decisions we took during the development stage.

5.1 Legend of Grimrock 2 Level Editor

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

In the Editor, the user has access to a 32 by 32 tiles 2D canvas. He has also given three main tools to edit the layout of his level as well as its content. For our work, the most important one is the add object tool. With it the designer can easily add puzzle elements to the map from a selection of various elements with different themes, such as beach theme oriented gates and pressure plates, forest pressure plates, or in the context of our tests, dungeon gates and pressure plates. It can also add monsters to the map, but in the context of our work, designers worked exclusively with the add object tool to add new puzzle elements to the map and with the select entity mode, where they could select previous added puzzle elements and modify its attributes, such as position coordinates, orientation and in case of pressure plates the target gate that they would activate. The remaining tool, the tile-edition tool, is important to create a map layout in the 2D canvas, with an array of different themed tiles. This tool was the corner stone of Lucas work, but for our work it will not be used, since we have a pre-created map based on the Trickster’s Lair from the Legend of Grimrock 2 original map. Other interesting features of the LoG2 level editor includes a way for the designer to preview how his level plays out inside the editor itself, without needing to open it in the main game. Also, the designer by using the described tool, he can create levels with different sets and themes, and when he is done, he can export his creations and share them with other users, for example using Steam Workshop. Relatively to this work,a crucial feature is how the level editor handles the user projects. User generated content is simply stored in a plain Lua text

---

1 Almost Human Ltd., 2014
2 Valve Corporation, 2003
file, which immediately prevents a lot of problems when integrating it with other software. For our case, this means we can write content, in a LoG2 syntax-friendly way, to a Lua scripting file and immediately reload our project to reveal it in the editor. The same way, any changes performed to our project and consequently saved, they will be immediately accessible in the Lua file and interpreted by the EB Puzzle Mode. Regardless, it was necessary to learn how to parse and interpret the puzzle elements related code, since we based our tool on Luca’s Editor Buddy, much of the work on that end was developed, we only added the part for the puzzle elements and connection between them.

Figure 5.1 illustrates a screenshot of both Legend of Grimrock 2, figure 5.1a and its Level Editor, figure 5.2b and Appendix C an example of a Lua file generated by the LoG2 Editor.

In the end, we choose Legend of Grimrock 2 and its level editor because there was already some work developed to work alongside it, Luca’s Editor Buddy and much of the developed process could be accelerated by extending the Editor Buddy to work with puzzles. The game itself has a great visual representation for puzzles and creative potential, in addition to being mod friendly, facilitating the task of developing an add-on solution for it.

5.2 Solution Implementation

In this section we address the more technical aspects of the proposed solution, design decisions, limitations and respective workarounds regarding the Editor Buddy Puzzle Mode behavior, its interface and its interaction cycle with the designer.
5.2.1 Genetic Algorithms

First let us 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, meaning that only individuals with the right traits will be able to survive within a population and generate offspring which have the potential to be even stronger or fitter than their predecessors. In Nature, this happens on the genetic level, where individual traits are represented in the form of chromosomes. An organism complete hereditary information and the translation to the visible traits of said organism is called genotype to phenotype mapping.

Genetic Algorithms (GAs) implement such concepts so they can be interpreted by computer programs, simulating this method, translating potential solution traits into data structures, as the chromosome of an individual, that traditionally is represented as a string of bits. This individual reproduces with other ones, for several generations, in the hope of generating offsprings who can provide a solution for a given problem. Termination for GAs runs can be defined by, for example, a fixed number of generations or under more specific circumstances, such as, after reaching a minimum population fitness or only when the optimal solution has been found.

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

The crossover operator handles the recombination of genetic material between two previously selected parent chromosomes. Normally this operation is performed by using a single-point or a two-point crossover. Mutation simulates the naturally occurring phenomena in which can occur a minimal change in an individual's genetic information, that can result in a larger impact when translated into, in case of GAs, their potential problem solving traits. From the algorithmic standpoint, it tries to avoid the population getting stuck on a local maxima. The frequent types of mutation include flipping bit mutation where a single gene (or bit) is inverted (zero to one and vice-versa) or the swap mutation where two genes are swapped between their places. Elitism is an operator which ensures the most fit individual, or individuals, are carried over, copied to the new population with no changes from mutation or crossover. Elitism only parametrization is the percentage of individuals whose chromosomes are copied to the next population without having mutations or crossovers. These operators perform the required genetic material exchange and, depending on their parameters, can generate more different and interesting individuals from their parents.

The most important feature of genetic algorithms is the fitness function, whereas operators simply say how and how much genetic material from the parents is passed and then mutated, the fitness function is what evaluates which individuals are more or less valuable for the problem solution. Because of this, the fitness function is what directs 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, objective, innovative and user map oriented each with its own fitness function and parametrization, as well a combined function that takes into account the previous three that evolve individuals that represent the number of puzzle elements per map section.

5.2.2 Problem specific constraints

Puzzle element to chromosome representation

In our project, since we had the representation of the map per sections of rooms and corridors for the graph, we decided to use the same approach for an internal graph representation with each graph node
representing a map section. There are twenty three sections on our selected map, at the default value, each pair of genes represents a section, meaning that per section there is a max of 2 puzzle items, if supported by the section, there are sections with only one cell, so the max number of puzzle items for that section is one. The chromosome is binary, meaning that a gene is represented by a pair of bits, either zeros (0) and/or ones (1). As represented on Figure [5.2] a partial example of the chromosome and how it is related to that map. Each “one” in a gene represents one element, and not a binary value for a number, meaning “11” represents two (2) and not three (3) elements, so that in a case of mutation on the most significant bit, with two bits per section, would have a big difference on the number of map elements on that section. The puzzle elements number translated in a gene representing a room represents the number of pressure plates in that room and the puzzle elements number in a corridor represents the number of gates on that corridor. We took this approach per section because after testing with a different approach of each bit representing a map tile, we would see cases of an overload of puzzle items in big sections, by having the genes representing sections we can better control the max number of items per section. We have chosen two per section since it is what makes sense for our map but it can be expanded to \( \chi \) elements in a future work.

Because of our concern of some balance between the pressure plates and the gate number we create our initial population. As said before it is pseudo-random, in the sense that we start the first individual with everything at zero, and each consequent individual has, potentially, an extra two genes with value one, one for pressure plates related genes and another for gates related genes, than the previous one until, in our case, the forty-six (46) genes, potentially, are all “ones” and then start all over again in the next individual with the chromosome with only zeros. We do this so we can have a better initial search space coverage, something that is important in a GA. Since it is using a random function to select the sections, it is not guaranteed that all the ones that an individual can have will appear on the chromosome, in case a section is selected more than “two” times, it will only have two puzzle elements on that section, we say that a chromosome will have a maximum of \( n \) genes at “1”.

**Breadth-First Graph Search adaptation**

To characterize a map in terms of length and backtracking we developed a Breadth-First Graph Search that takes into account, from the starting section, the pressure plates he passes through and adds it into a variable, and when it finds a section with gates, it decreases it from the said variable, as long the variable is equal or bigger than zero, it can go forward. The algorithm can pass through the same section again, to prevent getting stuck back and forth between two sections we added the condition that a section is only viable if the state of the world changes, meaning we store all existing pressure plates and gates on a dictionary and if a value from that dictionary changes, the state of the world changes and it is a viable move. Without these restrictions it behaves like a normal Breadth-First Graph Search, where in each node, it adds the adjacent ones to the bottom of a list(if they meet those two criterias), and goes to the next node on the top of the list.

For example, if the chromosome that we convert into a graph is the one from Figure [5.2] the first section, or the root of the graph is the dark green, it has a pressure plate so add +1 to our control variable (that starts at zero (0)). Our control variable now is at 1, the green section does not have elements so our variable does not change. The red section has a pressure plate, so our control variable becomes 2. The light blue corridor, has a gate, so it contributes with -1 and our variable is at 1, its possible to follow that path, but either the pink or the dark orange have a gate and our variable would be 0 and the player becomes stuck, so this path is not viable. The path through the dark yellow corridor is viable, in the light green room the control variable is 2 again, there are two gates to the exit (the greenish corridor and the dark blue corridor, that it is the exit), but this is not the optimal path, the optimal one is
In this subsection we detail how the Editor Buddy Puzzle mode life cycle unfolds. We address how each of the four algorithms work together to generate a solution.

5.2.3 Editor Buddy Puzzle Mode execution flow

In this subsection we detail how the Editor Buddy Puzzle mode life cycle unfolds. We address how each of the four algorithms work together to generate a solution.

Because we chose a Breadth-First Graph Search approach, we can be sure when the exit node is reached, that is the most efficient way to travel from the start to exit with those puzzle elements disposition (see Figure 5.3) and we can stop the search even if there are nodes to be explored and get the metrics from the map. Said metrics are gathered in places, a variable with the number of sections passed since the start and a dictionary with every section and the number of times each section was visited, with this we can have the length of a map and the max backtracking value respectively and said map can be evaluated according to the objective fitness function.

Figure 5.2: Relation between a chromosome and the generated map through the purple corridor (control variable becomes 1 and then open the exit gate (the exit).
The following diagram, illustrated by figure 5.4 represents the Editor Buddy Puzzle Mode execution flow. Each algorithm execution is independent from the ones before, the only possible connection is if the user export's the map and the Innovation and User Map take into account that new map generated before. The three algorithms, Objective, Innovation and User Map, run at the same time and, when they finish after a certain number of generations (that we will discuss in the next sub-section), a percentage of each population, calculated according to the user input, is transferred to the Combined algorithm that re-evaluates and evolves that new population again according to objective, innovation and user map fitness functions with different weights according to the user input. The Combined algorithm calculates the best individual and shows that suggestion to the user, that can either export or ignore and run again the algorithms with the same or new objective and/or algorithms weight configuration. Each run is triggered by the user by clicking on the Run button.

**Designer’s map input for comparison.** Designer’s map, identified as the “Designer’s map input for comparison” element in our flow diagram, is a key-element for the Innovation and User Map algorithm (and fitness functions) for them to compare to the chromosome based on its puzzle elements. Its development over time is sole responsibility of the designer, whereas the EB Puzzle Mode merely interprets it and calculates both algorithms accordingly, when configured to do so.

**Population transference.** After the designer triggering a new algorithm execution, the first thing the Editor Buddy performs is a check on the state of the behavior of the 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 Objective, Innovation and User Map sliders greater than zero means we have our three algorithms working at the same time. A portion of each population is placed in the initial population (based on the weight of each of the three sliders) of the Combined algorithm. Therefore, it’s the Combined algorithm who is ultimately responsible for selecting which individual to display.

**Algorithm execution.** Objective, Innovation and/or User Map run first in parallel, if their percentage is bigger than zero, otherwise they are turned off for that run. The percentage of each algorithm is calculated by the ratio of each slider (ranges from 0 to 100) and the sum the values of all the three
slider values. Objective evolves towards generating individuals who respect task constraints selected in the objectives combo-box. Innovation evolves towards the most distinct population from the user map, in terms of number of puzzle elements per section and, User Map evolves towards the most equal population from the user map, again, in terms of number of puzzle elements per section. After that the top $\chi$ of each algorithm is transferred to the initial population of the Combined Algorithm, that evolves its individuals towards a combination of the three previous algorithms fitness functions weighted differently according to the user input for them.

Displayed suggestion. When the algorithm execution is complete, the new best suggestion is then displayed back to the designer in the EB Puzzle Mode 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 slider is the only greater than zero, the individual who best respects the currently selected goal, independent of length and backtrack values, 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.

5.2.4 Algorithms

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

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. Each time the algorithm runs, a 50 individual population is pseudo-randomly generated, as described before, the first individual starts with all genes at zero, each subsequent individual has two more genes, potentially, with the value one that the one before, one for room sections, to generate pressure plates, and the other to corridors, for the generation of doors. In our default configuration each pair of genes represents a map section, with our map having twenty-three (23) sections our chromosomes have forty-six (46) genes. Since we use a random function to choose the sections, it is not guaranteed that a chromosome has \( n \) “1” valued genes, we define that said chromosome, will have a maximum \( n \) genes at “1”.

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, the fitness value will be higher than zero if the exit can be reached from the start and with the puzzle elements currently in the map (otherwise the fitness is zero), that value is then calculated with our version of the Breadth-First Graph Search, and evaluates a map in two domains:

- Length
- Backtrack

Length is characterized by the number of map sections, rooms and corridors, that are necessary to pass-through (even if a section has multiple pass-through it is counted to that value), on the most efficient way to finish the map with a certain puzzle elements disposition. Backtrack is characterized by the max number a certain section was visited, on the most efficient path to complete a map with a certain puzzle elements disposition. Each domain is tuned with the sliders from the UI, for more or less of that domain, and also for preference for one or another, that also means if a domain, lets say Backtrack, has all the preference slider to its side, Length is completely discarded from the fitness function, and vice-versa. By default each domain has the same weight on the fitness function.

For example the Length slider is at 50%, Backtrack at 100% and preference with 25% towards Length and 75% towards Backtrack, for a population that the Length range is at 7 for minimum and 21 maximum and Backtrack range from 1 to 4 the algorithm will value more chromosomes that represent a Length of 14 and Backtrack of 4 with the Backtrack fitness portion contribution with 75% for the final fitness, that means that solutions with backtrack closer to 4 are more valuable even if they are more distant from 14 of Length than others.
**Operators.** Crossover parameters include 80% crossover probability using double point crossover from the GAF [33] default operator, as for our replacement strategy we chose generational replacement and for parent selection we chose to use Fitness Proportionate Selection [34], also known as roulette wheel selection. This selection type directly associates an individual’s fitness to the probability of its selection. For our Mutation operator and its parameters, we use the GAF default operator that goes through each gene and, in our case, has a 10% chance of changing the byte value to its counter-part. 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 were chosen after a preliminary study but could be better tuned if there was a more in-depth study of their impact.

**Termination.** The evolutionary run of the objective algorithm terminates after 25 generations, we have this low number because of the algorithm performance, higher values would translate into bigger processing time, but with this value we could cover a good number of possible values for Length and Backtrack.

**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 innovative content could always be sure to find it with the contributions from this algorithm. At the map level, innovation means the number of puzzle elements is more distinct from the user’s map, for each section. The algorithm does not take into account the user's map gate-pressure plate connection.

**Population and chromosomes.** Again we followed the same model of population creation used in the objective algorithm. Our default configuration has 50 individuals with, again, 46 genes chromosomes. Again with our pseudo-random population generation, we tried to cover, initially different puzzle elements densities on the map for then evolve them.

**Objectives.** Innovation in the standpoint of this algorithm means the number of map elements in a section is more distinct from the one in the map created by the designer. It is easy to evaluate that, we convert the map created by the user into a chromosome, with a little caveat, to guarantee compatibility between user map chromosome and our chromosomes, if the user in a section has more than two elements, in the generated chromosome will save as only having two elements. After having the two chromosomes, it is comparing each pair of bits, converting them into a number again, each “1” is translated to one element, meaning 11 is equal to two (2) and not three (3) as one would think in binary, with that in mind our algorithm sees the pair “10” and “01” as equal in value, so that section would contribute to the final fitness with zero. This algorithm does not guarantee that the map is feasible, it only evaluates if the number of puzzle elements per section is more distinct.

**Operators.** For the crossover operator we choose the same as the objective algorithm, 80% using double point crossover, generational replacement and Fitness Proportionate Selection to select the parents. Mutation we used again the GAF default mutation operator with a 10% mutation probability. Again elitism parameters include a 5% elitism percentage and prioritizes the fittest individuals.

**Termination.** Since Innovation algorithm is not so heavy as the objective one, we double the generations and so the algorithm stops after 50 generations.

**User Map algorithm**

The User Map algorithm evolves towards generating content similar to the user’s map on the LoG2 Editor. We have included this behavior, inspired by Lucas’ approach, but decided to make an evolutionary
run for it, not just copying a chromosome that represents the user map and filling with copies on the Combined algorithm. This way, users who seek content related to their could always be sure to find it with the contributions from this algorithm. At the map level, a more user map oriented suggestion means the number of puzzle elements is more equal, for each section, from the user's map. Once again, the algorithm does not take into account the user's gate-pressure plate connections, only the number of puzzle elements per map section, just like the Innovation algorithm.

**Population and chromosomes.** Again we followed the same model of population creation used in the objective and innovation algorithm. Our default configuration has 50 individuals with, again, 46 genes chromosomes. And uses again our pseudo-random population generation.

**Objectives.** An user map oriented suggestion in this algorithm means the number of map elements in a section is more equal from the one in the map created by the designer. Again, just like innovation, it is easy to evaluate this, we convert the map created by the user into a chromosome just like with the Innovation algorithm. After having the two chromosomes, it compares each pair of bits, converting them into a number again, each “1” is translated to one element, meaning 11 is equal to two (2) and not three (3) as one would think in binary, with that in mind our algorithm sees the pair “10” and “01” as equal in value, so that section would contribute to the final fitness with 1, being “23” the number total of sections. This algorithm does not guarantee that the map is feasible, only evaluates if the number of puzzle elements per section is more equal.

**Operators.** For the crossover operator we choose the same as the objective and innovation algorithm, 80% using double point crossover, generational replacement and Fitness Proportionate Selection to select the parents. Mutation we used again the GAF default mutation operator with a 10% mutation probability. Again elitism parameters include a 5% elitism percentage and prioritizes the fittest individuals.

**Termination.** Since, just like Innovation, the User Map algorithm is not so heavy as the objective one, we double the generations and so the algorithm stops after 50 generations.

### Combined algorithm

The Combined algorithm evolves towards generating content that takes into account all three previous algorithms, each generated chromosome is evaluated with two algorithms, Objectives and Innovation, since User Map is the inverse of Innovation, we only need to subtract from “1” the fitness value of Innovation to get the User Map fitness. We have this algorithm as a way to gather the best elements of each of the three previous algorithms, and evolve them towards the best one that will be shown as a suggestion.

**Population and chromosomes.** The initial population consists on the top $\chi$ % of each previous algorithms population, this percentage is calculated according with the sliders from the UI. On our default configuration for each algorithm, the initial population will have 50 individuals with 46 genes chromosomes.

**Objectives.** The objective of this algorithm was to gather the best individuals of each population and re-evaluate and evolve them with the previous fitness functions, the weight for the final fitness of each function is given by the slider configuration of each algorithm. In the end, the best individual is the one who will be the displayed suggestion.

**Operators.** For the crossover operator we choose the same as the previous algorithms, 80% using double point crossover, generational replacement and Fitness Proportionate Selection to select the parents. Mutation we used again the GAF default mutation operator with a 10% mutation probability. Again elitism parameters include a 5% elitism percentage and prioritizes the fittest individuals.

**Termination.** Since this algorithm uses the objective fitness function, that is computationally more
intensive, we have only 25 generations again.

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 EB Puzzle Mode. 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.

**Discard User Gate-Pressure Plates Connection** The drawing of the Gate and Pressure Plates Connection was something that came late in the development as quality of life option for the user to test a generated map easily. During the development the focus was towards the elements placement and when we added the connection feature we did not have the algorithms, Innovation and User map, ready to take that into account and neither how to model into the chromosome said connections. Because of this the algorithm that draws the connections on the EB Puzzle Mode is independent from the GAs and runs after the suggestion to be shown is generated and converted to a map.

**Fixed Map** From the beginning we focused on using a fixed map, originally created by the game developers to minimize variables when user testing and ease of the development of the tool. Because of these everything map related, graphs, internal one and the one displayed, map colours and the sections itself are hard-coded, but every algorithm was made generic, in theory if a different map is generated, has sections associated and the equivalent internal graph is created for it, it would work.

**Suggestion quality vs execution time.** Since we decided a more user active approach than Lucas, that means that the user is the one who starts the run of the algorithms (unlike Lucas, his algorithms run after a change on the editor side), we could sacrifice a more immediate solution for a better solution. At our default values each run takes about 3 to 5 seconds to complete each and we can generate a good solution for the user's settings, and quick enough for him to re-run for another suggestion if he does not like the one presented. As Smith et al. [17] underline, evolutionary algorithms are great for generations of offline content and since we have a more user active interaction it fits well our work.

5.3 Summary

By the end of this chapter we hope we were able to provide a deeper understanding of the diverse components of this project, during the use of the EB Puzzle Mode with the Legend of Grimrock 2 level editor. On the LoG2's level editor side we presented its main features, design tools and how straightforward and flexible it is. Regarding the proposed solution, we provided our standpoint and motivation towards choosing evolutionary algorithms and their parametrization, how we felt the content generated and evolved in 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, to test the robustness and verify potential interface problems that needed to be addressed before the formal evaluation. 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 test the EB Puzzle mode stability and interface decisions, we conducted two informal tests with two users, who have no connection to the game industry or level design but work in software development.

Preliminary evaluations

The first one was purely to play around with the tool and find limitations from the UI and to find bugs. In the second one there were some new additions from the first preliminary evaluation, that we will address next, better stability and there was a first version of the tasks so we could test again UI and stability for the tasks we had in mind.

Changes

The major change in terms of UI was the addition of the progress bars to show to the user at what point of the computation of the algorithms the EB Puzzle Mode was, to give the user feedback that it was working. We have also streamlined the initialization process of the EB Puzzle Mode (Loading a map, changing maps, the need to change modes) to minimize user error, so the Editor Buddy would already start in the Puzzle Mode, and the user had the ability to change maps during the section without restarting the application. It was addressed the possibility for the user to close and re-open the Section’s Graph window if he desired to. From the second preliminary evaluation the biggest change in the UI was re-ordering the buttons disposition to be less confusing for the user. Figure 6.1 shows the first version of the interface used in the first preliminary test. The final interface is represented in Figure 6.2 after the two informal tests.
Figure 6.1: Interface from the first preliminary test.

Figure 6.2: Final interface.
6.2 Solution evaluation

Study goals

The main goal of this study was the evaluation of the EB Puzzle Mode utility as well as its efficiency. We define utility, in this case, as the ability to contribute, direct or indirectly, with useful content, however, it remains at the designer’s discretion the definition of usefulness. 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 24th and 26th. For this study, we looked to find participants who were, the majority, interested in Level Design but they are inexperienced or amateurs in that field.

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” [35].

Study design

We conducted a collective-case study performed on a small group of participants for relative short periods of time.

Sample

Following Nielsen and Landauer recommendation[36], 6 participants were selected for this study, with ages ranging from 23 to 29 years old ($M = 25.3, SD=2.07$, 6 male). We used a purposeful sampling method and selected participants that were inexperienced and amateurs in level design. One of the participants works in the game industry, not in a level design position, and all the participants either are or were Computer Science MSc students.

Procedure

Participants would arrive individually at the laboratory where the study would take place and be introduced to the most important features of the Legend of Grimrock 2 level editor and EB Puzzle mode, through a 7 minute video tutorial. They would then experiment the level editor and freely ask questions regarding their usage to the researcher supervising the study. The whole introduction / tutorial took around 15 minutes. After this initial contact, participants would be asked to create puzzles to a previous created map with certain constraints: They would be using LoG2 editor while interacting with EB Puzzle Mode to support them during the
puzzle creation process; the map layout could not be changed; 1 pressure plate only opens 1 gate; a pressure plate could only be placed in rooms (non-black cells) and gates placed in corridors (black cells), for that they could use the “Show Corridor” checkbox to show the map like the Figure 4.4, and the solution should be as interesting as possible for the player.

The first task had an additional constraint:

It must teach the players that pressure plates can open gates in non-adjacent corridors.

The participants were then told they would have all the time needed until they would feel totally satisfied with their solution. The participants would take about 20 minutes to finish the task and then they were introduced to the second task all similar to the first expect for an additional constraint:

The puzzle must make the player pass multiple times in the same room, where in the future will be placed “lore” related objects and we want the player to find these objects, which room or rooms are at the designer's criteria.

This second task, again, had no time limit, but took around 10 minutes to complete. After the puzzle was finished, we asked the participant to fill a questionnaire and undergo a semi-structured interview (both taking about 20 to 30 minutes). After that we congratulate him for his accomplishment, thanking him for his participation and saying goodbye. The full experiment had an average duration of 1 hour.

Data collection methods

To evaluate EB Puzzle Mode, we used qualitative data gathered through participant questionnaire, participant observation, and a structured interview. A researcher was always present next to the participants, taking notes and answering technical questions when requested to, observation as complemented by screen recording during the entire process for later review their actions.

We used a linear scale questionnaire to get a measure of usability for the EB Puzzle Mode interface. The questionnaire also measured the discrepancy between the participant expectation when tuning the control sliders in a certain configuration when compared to the associated suggested content by the EB Puzzle mode. Finally, the questions incentivised the participant to think about certain issues that would be explored during the following interview.

After participants were finished with the questionnaire, a semi-structured interview was conducted. The goal for the interview was to draw out patterns from common concepts and insights regarding the personal experience of each participant while interacting with EB Puzzle Mode.

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 EB Puzzle Mode, illustrated by figure 6.3

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.
Figure 6.3: Setup used in evaluation tasks
Presentation

Questionnaires
From the questionnaires, which can be consulted in Appendix A, we were able to gather the following data regarding the evaluation of the usability of EB Puzzle mode:

- (U1:) It was easy to configure the Buddy behavior using the available interface controls;
- (U2:) It was easy to make and edit a selection of the puzzle suggestion made by the Buddy;
- (U3:) There were no communication issues between the LoG2 level editor and the Buddy;
- (U4:) The section’s Graph was useful;
- (U5:) It was easy to make the connection between the section’s Graph and the corresponding Map sections.
- (U6:) It was easy to make the connection between the section’s Graph and the generated puzzle topology.

<table>
<thead>
<tr>
<th>Scale</th>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
<th>U5</th>
<th>U6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally disagree</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Somewhat disagree</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Neither agree nor disagree</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Somewhat agree</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Totally agree</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6.1: Frequency of answers to usability questionnaire

From this results we can see the EB Puzzle Mode’s interface made easy to configure its behaviour, the user who evaluate “Neither agree nor disagree” expressed that it took a little while to learn to use the tool. Some users wanted to select part of the suggestion to incorporate into his puzzle on the editor, unfortunately that is not possible hence that one “Somewhat Disagree”. The connection issues were polarizing, half evaluated has none, the other half says it had, we theorize, through the interviews that is because of the inability for the EB Puzzle Mode to take into account the connections that the user makes on the LoG2 editor and also during a test the application crashed for unknown reasons that was never reproduced again that for that particular user may had penalized his evaluation. There was also a little nuisance sometimes the Editor would register the EB Puzzle Mode macro to save(and show the most recent work of the user in the Editor) and when user exported back to the Editor, there was a prompt that the user needed to click “No” and export again, no information was lost only it demanded some extra clicks from the user, and that might have affected negatively the results. About the Section’s Graph, the users that used it thought it was useful, there was users who did not even used because it felt no necessity for its information and the EB Puzzle Mode main window was sufficient. Everyone agreed that was easy to make the connection between the graph’s sections and the corresponding Map sections. Relatively to the connection between the Section’s Graph and the generated puzzle topology majority thought it was easy to make.

Regarding the behavior of the Editor Buddy, the following charts, Figure 6.4 shows how EB Puzzle mode’s suggestions, when using the Objective control slider, were expected and how useful they were perceived by the participants. Figures 6.5 through 6.9 do the same for the Length, Backtrack, Preference,
Innovation and User Map control sliders. These results pertain the algorithm behavior section from the questionnaire in Appendix A.

Figure 6.4: Objective slider expectation vs usefulness.

Figure 6.5: Length slider expectation vs usefulness.

Figure 6.6: Backtrack slider expectation vs usefulness.

Figure 6.7: Preference slider expectation vs usefulness.
Figure 6.8: Innovation slider expectation vs usefulness.

Figure 6.9: User Map slider expectation vs usefulness.
This results showed that the *Objective* slider worked as expected and the majority think it was useful. Same line of thought for the *Length* slider, worked as expected and it was useful. *Backtrack* had in general a good expectation and usefulness, except for one participant that felt that a purely backtrack oriented objective made the map length too small on the second task. *Preference* was well received by the participants in terms of generated suggestions and usefulness. *Innovation* and *User Map* had overall good usefulness evaluation but in terms of expectation, some participants expected that the connections they made in the editor were taken into account and sometimes felt a little frustrated by the solutions generated by both algorithms, one of them even said that when putting objective and user map at maximum value they wanted to have the same puzzle elements in the same position that they placed and generate a new connection based on them or even the ability to make his connection fixed, so the algorithm would not change them and calculated the new puzzles around his selection, again our algorithm does not take into account the pressure plate-gate connection, only their position.

**Observations**

From the observations we have been able to identify some key-events, key-issues and two polarizing ways to interact with the EB Puzzle Mode and the LoG2 Editor.

The identified **key-events** were:

- In the second task almost everyone changed backtrack, objective to max and some even preference towards backtrack;
- One participant used the EB Puzzle Mode history for the second task, to make the map on the editor completely empty again so he could build from scratch.
- Most of the participants used the “Show Corridors” option to check if a gate was in a corridor;
- Every participant used the “Show Gate/Key connection” option.

The identified **key-issues** were:

- The aforementioned expectation of the pressure plate-gate connection to be taken into account for *innovation* and *user map* oriented solutions was not fulfilled;
- Some participants wanted for a certain disposition of the puzzle elements multiple puzzle solutions could be generated (different gate-pressure plate connections);
- In one of the tests the application crashed once for unknown reasons that was never replicated again;
- Some participants ignored the Section’s Graph altogether;
- The selection of cells to export partial solution did not have much use.
- The “Export only connected keys” was rarely used.

Finally there was two types of users, some of them worked their puzzle from scratch and then used the EB Puzzle Mode to give suggestions based on their design, while others run multiple times the EB Puzzle Mode to find a suitable base to work their puzzle from that in the editor. There were three participants that created their puzzle from scratch before using the EB Puzzle Mode to show suggestions, and the other three used the EB Puzzle Mode first to generate a suitable first version of the puzzle to work over it and adapt to his vision. With this two behaviors we believe the EB Puzzle Mode can support different ways of interaction and adapt to each user’s preferences.
Semi-structured Interview

The interview was guided by the script that can be found in Appendix B.

The EB Puzzle Mode was widely accepted as being and overall positive influence more specifically in the second task. The EB Puzzle Mode interface was found to be user-friendly and intuitive in general, Innovation and User Map for some was not all that clear.

Most users thought it was more useful on the second task, the generation of a more backtrack oriented map, because they felt it was good to change completely the focus of development.

Overall “Show Corridor” and “Show Gate-Key connection” was very well received and used by the participant, especially the “Show Gate-Key connection” had the overall consensus. A participant said that “Show Corridor” was not useful which surprised us that reaction, we think the participant that expressed this opinion was because he used primarily the EB Puzzle Mode to generate suggestions first and then he would work and adapt them to his vision. In that way the EB Puzzle Mode would always put the puzzle elements on the right section, with small changes within sections or deletion of elements was more difficult to put a puzzle element on the wrong place, and so the “Show Corridor” did not feel that important. Some participants expressed that the Logger window was optional, in their opinion, from the designer’s standpoint that extra information was not necessary. Overall Export only connected keys was deemed unnecessary or situational. The majority wanted the Innovation and User Map algorithms to take into account the gate-pressure plate connections on the editor. There was a participant who even said that Objective was the main slider and Innovation and User Map could disappear. This participant felt the behavior of the EB Puzzle Mode not taking into account the connections that he made for the Innovation and User Map were frustrating that he said that with this behavior of these two algorithms they felt less relevant as the Objective algorithm, and so this last one was the main one and the others could be discarded.

Suggested improvements to the interface included:

- A reset button to set every slider at 50%;
- Change some button icons to other more representative;
- A dedicated box to show current map Length and Backtrack value instead of being on the Logger;
- Show on the map the number of time that each section was visited;
- Show on the map with another color(or even highlighting one section at the time) to show the path that the algorithm took.

Suggestions towards improving the Editor Buddy behavior included:

- Innovation and User Map take gate-pressure plate connections into account;
- Without changing the puzzle element’s disposition show alternative ways to connect gates and pressure plates;
- A way to lock element’s to prevent their deletion in new generations.
Interpretation

From this evaluation we confirmed that the EB Puzzle Mode worked really well to generate objective-oriented content, the participants saw the potential of this tool in the creation of puzzles for game levels. We saw a significant number of comments towards the EB Puzzle Mode taking into account gate-pressure plate connections that users make, unfortunately in our approach it is difficult to map connections into chromosomes and since these connections appeared in later stage of development they were not introduced in the genetic algorithms, but for future work this is something that has to be taken into account. The other behavior suggestions are easier to integrate in the current tool. Overall in the interface and usability level there was a good opinion from the participants, one or another had more problems with them but we think might be explained by the behavior of the EB Puzzle Mode itself in discarding gate-pressure plate connections (on the question “There were no communication issues between the LoG2 level editor and the Buddy”) and the small improvements suggested in the previous section. From the interaction of some participants we standard lateral thinking techniques were observed, in the case of participants that generate first multiple suggestions for a certain tool configuration, choose the one that feels better to them, modify them, and then generate again suggestions, if either Innovation and/or User Map had values other than 0, the modifications would take into account the puzzle elements position for the new suggestion. As said before, Lateral thinking it is all about generating conflict and challenging old ideas, a user express that the EB Puzzle Mode was “too disruptive” on the User Map algorithm, so the new idea conflict was there, maybe a little too much for some users, again by not taking into account gate-pressure plates connection, may have been perceived as much different, because of the new generated connections, but for the algorithm, by looking to only puzzle elements position it was similar, but by changing one or two elements from different sections made the objective algorithm go through another path and the consequent generation of connections was completely different.
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 that an interaction-focused co-creativity seemed to work well in most of the cases, even with two distinct types of user, one more LoG2 focused (that constructed his puzzle from scratch) and another that used the EB Puzzle Mode to generate a suitable first version of the puzzle to work over it, both user type worked back and forth with Buddy and the editor, showing the “peer” or “colleague” paradigm interaction between computer and human, defined by Lubart [1], that we hoped to see. We also noticed that standard lateral thinking techniques were observed during interaction and that the overall performance of the developed solution was positive and was flexible enough to adapt to the aforementioned two types of users that we identified.

We described EB Puzzle Mode simple interface, based on Lucas’ work, that allows to fine tune the Buddy in three dimensions: Objective, following specific design goals; Innovation, wanting to explore new directions, and; User Map, focus on the current co-proposal. The design goals are puzzle suggestions in two domains: length and backtrack. The EB Puzzle Mode’s 2D canvas allows to intuitively and visually communicate its suggestions to support diagrammatic reasoning, a cognitive process inherently present in level design.

The fact we received so many design suggestions, which we will discuss in the next section, taught us there are still some aspects that need to be improved regarding how the EB Puzzle Mode better utilizes the designer’s created puzzles (take into account gate-pressure plate connections) and some interface modifications to adapts for different users.

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 research.

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 integration of our solution with Lucas’ work was something that we ambitioned from the beginning, but for time-related reasons and to have one less variable in the evaluation phase, we decided to have a fixed map layout based on the map Trickster’s Lair from the LoG2 game. This is something that can be done in the future, as mentioned before, all map and graph sections related code are hard-coded for the Trickster’s Lair modified map, but all the algorithms are generic for any map.

Another change that could be implemented in the future, was the one expressed by the participants, the EB Puzzle Mode take into account gate-pressure plate connections made by the user in the LoG2 editor for the Innovation and User Map algorithms. Since the connection came into play in later stages of development these algorithms did not take it into account for the results, but the participants expressed that was better if these algorithms would take the connections made by them. This is a situation that is not trivial to tackle, participants wanted to keep the connections, but they can become irrelevant, having no value whatsoever with other map elements, affect the most efficient way to finish a map, that is our main criteria to stop the objective algorithm, or even the change of places of the gate or the pressure plate, keeping their connection, the new solution, for those elements and their connection, might not make sense.

Another behavior change that some users said was for the displayed solution, show other ways to connect gates and pressure plates even if it was not the most optimal way to finish the map, this already can be supported with small code changes.

A interesting addition for the EB Puzzle Mode is to provide different types of puzzle, it already supports the Environmental Mechanical Puzzles, the pressure plates, but could also support observational challenges, logic puzzles, riddles and hidden treasure puzzles.

Various interface improvements were given in the evaluation chapter, some examples are a dedicated box for the puzzle metrics, length and backtrack, show in the map, with another color for example, the sections that the algorithm passed through, amongst other small things detailed on the chapter.

Regarding future evaluation sessions, besides evaluating the human-computer interaction, we would like to compare the quality of the puzzles created with and without the help of the EB Puzzle Mode. 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 EB Puzzle Mode 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 EB Puzzle Mode 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 EB Puzzle Mode and see if they found it useful in stimulating their creativity.

A more ambitious goal would be adapting the EB Puzzle Mode to other existing level editing tools. Apart from the Legend of Grimrock 2, this would be a huge challenge, because the EB Puzzle Mode is so deeply integrated with the LoG2 Editor. Because the LoG2 Editor works on the 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 the research and understanding and state of the art in computer co-creativity, to being able to develop a solution that integrates into an area of game development, that to me personally is very interesting, **Level Design**, and seek to improve the previous work started by Pedro Lucas, into a new domain of level design, puzzle creation. There were hard times during the development, but then seeing the tool improving rapidly and showing results during the ending phase gave us great satisfaction seeing what we theorize in terms of generated suggestions coming into fruition. We hope our contribution is valuable enough to give interest to those looking to explore the niche of human-computer co-creativity in videogame level design, and create tools that help inexperienced and amateur users a little help when starting to create puzzles themselves, as an amateur level designer in small projects and demos the premise of this thesis interested me right from the start.
Bibliography


Appendix A

Questionnaire appendix
Editor Buddy Puzzle Mode Test Form

The primary goal of this form is to measure the utility of the Editor Buddy application Puzzle mode and its efficiency at providing a creativity-enhancing experience during a cooperative puzzle creation activity for a level. 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

*Required

1. Age *

2. Sex *
   Mark only one oval.
   - Male
   - Female

3. How often do you play video games? *
   Mark only one oval.
   - I don't play video games
   - I play video games occasionally when the opportunity presents itself
   - I make some time in my schedule to play video games

4. Are you familiar with dungeon crawling role-playing games (e.g. Dungeon Master, Eye of the Beholder, Legend of Grimrock, Vapurum)
   Mark only one oval.
   - I don’t play video games
   - I play games but not of the dungeon crawler RPG genre
   - I am familiar with the dungeon crawler RPG genre and played at least one game of the genre
   - The dungeon crawler RPG genre is one of my favorite and I have played several games of this genre

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's Puzzle Mode interface and the Legend of Grimrock 2 Editor interface.
5. It was easy to configure the Buddy behavior using the available interface controls *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. The section’s Graph was useful *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. It was easy to make the connection between the section's Graph and the corresponding Map sections. *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10. It was easy to make the connection between the section's Graph and the generated puzzle topology *
    *Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totally agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
5 - Totally Agree
please grade the following statements according to your experience with the suggestions generated by the Editor Buddy’s Puzzle Mode algorithms

Objective Slider

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   Totally disagree   Totally agree

12. Suggestions generated using the Objective slider were useful *
Mark only one oval.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   Totally disagree   Totally agree

Length slider
13. Suggestions generated using the Length slider were in line with my expectations *
   Mark only one oval.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Totally disagree   |   |   |   |   | Totally agree

14. Suggestions generated using the Length slider were useful *
   Mark only one oval.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Totally disagree   |   |   |   |   | Totally agree

Backtrack Slider
15. Suggestions generated using the Backtrack slider were in line with my expectations *
   Mark only one oval.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   Totally disagree  |  |  |  |  | Totally agree

16. Suggestions generated using the Backtrack slider were useful *
   Mark only one oval.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   Totally disagree  |  |  |  |  | Totally agree

Preference Slider
17. Suggestions generated using the Preference slider more to a side than the other were in line with my expectations *
Mark only one oval.

1  2  3  4  5
Totally disagree   Totally agree

18. Suggestions generated using the Preference slider were useful *
Mark only one oval.

1  2  3  4  5
Totally disagree   Totally agree

Innovation Slider
19. Suggestions generated using the Innovation slider were in line with my expectations *
   Mark only one oval.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Totally agree</td>
</tr>
</tbody>
</table>

20. Suggestions generated using the Innovation slider were useful *
   Mark only one oval.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Totally disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Totally agree</td>
</tr>
</tbody>
</table>

User Map Slider
21. Suggestions generated using the User Map slider were in line with my expectations *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

22. Suggestions generated using the User Map slider were useful *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

Interview appendix
Interview

- Overall, did you find the Editor Buddy 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 option and aid checkboxes would recommend a friend using this tool for the first time?
- What would be the ones you wouldn’t recommend?
- How would you explain someone that isn’t familiar with Editor Buddy, how to use the various sliders.
- Was there a right time to use a particular slider all the way up? (Innovation/Objective at the start, Difficulty and preference to it on the second task).
- 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?
- What were your thoughts about the section’s Graph window? Did you find it useful? In case you haven’t used it, why haven’t you used it?

Any other questions?
Appendix C

LoG2 Lua file example
--- level 1 ---

```plaintext
newMap{
    name = "Unnamed",
    width = 32,
    height = 32,
    levelCoord = {0,0,0},
    ambientTrack = "dungeon",
    tiles = {
        "castle_arena_floor",
        "dungeon_floor",
        "dungeon_wall",
    } 
}
```

loadLayer("tiles", {
    3,3,3,3,3,3,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
    3,3,3,3,3,3,1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
    3,3,3,3,3,3,1,1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
    3,3,3,3,3,3,1,1,1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
spawn("starting_location",9,1,2,0,"starting_location_1")
spawn("torch_holder",8,3,0,0,"torch_holder_1")
torch_holder_1.controller:setHasTorch(true)
spawn("torch_holder",10,3,0,0,"torch_holder_2")
torch_holder_2.controller:setHasTorch(true)
spawn("torch_holder",10,5,2,0,"torch_holder_3")
torch_holder_3.controller:setHasTorch(true)
spawn("torch_holder",8,5,2,0,"torch_holder_4")
torch_holder_4.controller:setHasTorch(true)
spawn("torch_holder",11,7,3,0,"torch_holder_5")
torch_holder_5.controller: setHasTorch(true)
spawn("torch_holder", 9, 8, 0, 0, "torch_holder_6")
torch_holder_6.controller: setHasTorch(true)
spawn("torch_holder", 9, 10, 2, 0, "torch_holder_7")
torch_holder_7.controller: setHasTorch(true)
spawn("torch_holder", 8, 9, 3, 0, "torch_holder_8")
torch_holder_8.controller: setHasTorch(true)
spawn("torch_holder", 6, 8, 0, 0, "torch_holder_9")
torch_holder_9.controller: setHasTorch(true)
spawn("torch_holder", 3, 9, 2, 0, "torch_holder_10")
torch_holder_10.controller: setHasTorch(true)
spawn("torch_holder", 5, 11, 3, 0, "torch_holder_11")
torch_holder_11.controller: setHasTorch(true)
spawn("torch_holder", 4, 13, 0, 0, "torch_holder_12")
torch_holder_12.controller: setHasTorch(true)
spawn("torch_holder", 6, 13, 2, 0, "torch_holder_13")
torch_holder_13.controller: setHasTorch(true)
spawn("torch_holder", 9, 15, 3, 0, "torch_holder_14")
torch_holder_14.controller: setHasTorch(true)
spawn("torch_holder", 12, 16, 2, 0, "torch_holder_15")
torch_holder_15.controller: setHasTorch(true)
spawn("torch_holder", 15, 15, 2, 0, "torch_holder_16")
torch_holder_16.controller: setHasTorch(true)
spawn("torch_holder", 17, 16, 2, 0, "torch_holder_17")
torch_holder_17.controller: setHasTorch(true)
spawn("torch_holder", 18, 18, 2, 0, "torch_holder_18")
torch_holder_18.controller: setHasTorch(true)
spawn("torch_holder", 18, 14, 1, 0, "torch_holder_19")
torch_holder_19.controller: setHasTorch(true)
spawn("torch_holder", 19, 12, 1, 0, "torch_holder_20")
torch_holder_20.controller: setHasTorch(true)
spawn("torch_holder",17,12,0,0,"torch_holder_21")
torch_holder_21.controller:setHasTorch(true)
spawn("torch_holder",18,11,3,0,"torch_holder_22")
torch_holder_22.controller:setHasTorch(true)
spawn("torch_holder",17,10,2,0,"torch_holder_23")
torch_holder_23.controller:setHasTorch(true)
spawn("torch_holder",16,11,1,0,"torch_holder_24")
torch_holder_24.controller:setHasTorch(true)
spawn("torch_holder",14,10,2,0,"torch_holder_25")
torch_holder_25.controller:setHasTorch(true)
spawn("torch_holder",13,11,1,0,"torch_holder_26")
torch_holder_26.controller:setHasTorch(true)
spawn("torch_holder",12,12,2,0,"torch_holder_27")
torch_holder_27.controller:setHasTorch(true)
spawn("torch_holder",14,13,1,0,"torch_holder_28")
torch_holder_28.controller:setHasTorch(true)
spawn("torch_holder",9,12,3,0,"torch_holder_29")
torch_holder_29.controller:setHasTorch(true)
spawn("tomb_wall_01",9,1,1,0,"tomb_wall_01_2")
spawn("tomb_wall_01",9,2,1,0,"tomb_wall_01_4")
spawn("tomb_wall_01",9,1,0,0,"tomb_wall_01_1")
spawn("tomb_wall_01",9,1,3,0,"tomb_wall_01_3")
spawn("tomb_wall_01",9,2,3,0,"tomb_wall_01_5")
spawn("tomb_wall_01",8,3,0,0,"tomb_wall_01_6")
spawn("tomb_wall_01",8,3,3,0,"tomb_wall_01_7")
spawn("tomb_wall_01",8,4,3,0,"tomb_wall_01_8")
spawn("tomb_wall_01",8,5,2,0,"tomb_wall_01_9")
spawn("tomb_wall_01",8,5,3,0,"tomb_wall_01_10")
spawn("tomb_wall_01",9,5,2,0,"tomb_wall_01_11")
spawn("tomb_wall_01",10,5,2,0,"tomb_wall_01_12")
spawn("tomb_wall_01",11,6,3,0,"tomb_wall_01_13")
spawn("tomb_wall_01",4,9,0,0,"tomb_wall_01_50")
spawn("tomb_wall_01",2,8,0,0,"tomb_wall_01_51")
spawn("tomb_wall_01",3,8,0,0,"tomb_wall_01_52")
spawn("tomb_wall_01",10,10,2,0,"tomb_wall_01_53")
spawn("tomb_wall_01",9,10,2,0,"tomb_wall_01_54")
spawn("tomb_wall_01",8,10,2,0,"tomb_wall_01_55")
spawn("tomb_wall_01",12,9,2,0,"tomb_wall_01_56")
spawn("tomb_wall_01",11,10,2,0,"tomb_wall_01_57")
spawn("tomb_wall_01",14,10,2,0,"tomb_wall_01_58")
spawn("tomb_wall_01",15,10,2,0,"tomb_wall_01_59")
spawn("tomb_wall_01",17,10,2,0,"tomb_wall_01_60")
spawn("tomb_wall_01",16,16,2,0,"tomb_wall_01_63")
spawn("tomb_wall_01",17,16,2,0,"tomb_wall_01_64")
spawn("tomb_wall_01",18,18,2,0,"tomb_wall_01_65")
spawn("tomb_wall_01",19,18,2,0,"tomb_wall_01_66")
spawn("tomb_wall_01",13,16,2,0,"tomb_wall_01_67")
spawn("tomb_wall_01",12,16,2,0,"tomb_wall_01_68")
spawn("tomb_wall_01",11,16,2,0,"tomb_wall_01_69")
spawn("tomb_wall_01",9,16,2,0,"tomb_wall_01_70")
spawn("tomb_wall_01",9,16,2,0,"tomb_wall_01_71")
spawn("tomb_wall_01",8,13,2,0,"tomb_wall_01_72")
spawn("tomb_wall_01",7,13,2,0,"tomb_wall_01_73")
spawn("tomb_wall_01",6,13,2,0,"tomb_wall_01_74")
spawn("tomb_wall_01",5,13,2,0,"tomb_wall_01_75")
spawn("tomb_wall_01",4,14,2,0,"tomb_wall_01_76")
spawn("tomb_wall_01",3,14,2,0,"tomb_wall_01_77")
spawn("tomb_wall_01",4,9,2,0,"tomb_wall_01_78")
spawn("tomb_wall_01",7,8,2,0,"tomb_wall_01_80")
spawn("tomb_wall_01",3,9,2,0,"tomb_wall_01_81")
spawn("tomb_wall_01",2,9,2,0,"tomb_wall_01_82")
spawn("tomb_wall_01",2,9,3,0,"tomb_wall_01_83")
spawn("tomb_wall_01",15,15,2,0,"tomb_wall_01_116")
spawn("tomb_wall_01",3,8,1,0,"tomb_wall_01_117")
spawn("tomb_wall_01",5,8,3,0,"tomb_wall_01_118")
spawn("tomb_wall_01",19,13,2,0,"tomb_wall_01_121")
spawn("tomb_wall_01",14,12,1,0,"tomb_wall_01_122")
spawn("tomb_wall_01",14,13,1,0,"tomb_wall_01_123")
spawn("tomb_wall_01",14,14,1,0,"tomb_wall_01_124")
spawn("tomb_wall_01",10,17,1,0,"tomb_wall_01_125")
spawn("tomb_wall_01",10,17,2,0,"tomb_wall_01_126")
spawn("tomb_wall_01",10,17,3,0,"tomb_wall_01_127")
spawn("torch_holder",9,1,0,0,"torch_holder_30")
torch_holder_30.controller:setHasTorch(true)
spawn("tomb_wall_01",7,7,0,0,"tomb_wall_01_30")
spawn("tomb_wall_01",13,10,3,0,"tomb_wall_01_31")
spawn("tomb_wall_01",11,10,1,0,"tomb_wall_01_61")
spawn("tomb_wall_01",13,11,1,0,"tomb_wall_01_62")
spawn("tomb_wall_01",4,14,1,0,"tomb_wall_01_128")
spawn("tomb_wall_01",8,7,1,0,"tomb_wall_01_129")
spawn("tomb_wall_01",5,10,1,0,"tomb_wall_01_40")
spawn("tomb_wall_01",7,12,1,0,"tomb_wall_01_41")
spawn("dungeon_stairs_up",10,17,2,0,"dungeon_stairs_up_1")
spawn("tomb_wall_01",6,9,1,0,"tomb_wall_01_24")
spawn("tomb_wall_01",5,11,1,0,"tomb_wall_01_79")
spawn("tomb_wall_01",6,9,2,0,"tomb_wall_01_94")
spawn("tomb_wall_01",6,12,0,0,"tomb_wall_01_119")
spawn("tomb_wall_01",7,12,0,0,"tomb_wall_01_120")
spawn("tomb_wall_01",11,15,0,0,"tomb_wall_01_130")
spawn("tomb_wall_01",12,15,0,0,"tomb_wall_01_131")
spawn("tomb_wall_01",13,13,1,0,"tomb_wall_01_132")
spawn("tomb_wall_01",13,15,0,0,"tomb_wall_01_133")
spawn("tomb_wall_01",13,14,1,0,"tomb_wall_01_134")
spawn("tomb_wall_01",13,12,2,0,"tomb_wall_01_135")
spawn("tomb_wall_01",12,12,2,0,"tomb_wall_01_136")
spawn("tomb_wall_01",11,12,2,0,"tomb_wall_01_137")
spawn("tomb_wall_01",10,14,1,0,"tomb_wall_01_138")
spawn("tomb_wall_01",10,13,1,0,"tomb_wall_01_139")
spawn("tomb_wall_01",16,15,0,0,"tomb_wall_01_140")
spawn("tomb_wall_01",17,15,0,0,"tomb_wall_01_141")
spawn("tomb_wall_01",17,12,2,0,"tomb_wall_01_142")
spawn("tomb_wall_01",16,12,2,0,"tomb_wall_01_143")
spawn("tomb_wall_01",18,13,3,0,"tomb_wall_01_144")
spawn("tomb_wall_01",18,14,3,0,"tomb_wall_01_145")
spawn("tomb_wall_01",18,13,3,0,"tomb_wall_01_146")
spawn("tomb_wall_01",14,9,1,0,"tomb_wall_01_147")
spawn("tomb_wall_01",13,8,1,0,"tomb_wall_01_148")
spawn("tomb_wall_01",14,9,0,0,"tomb_wall_01_149")
spawn("tomb_wall_01",8,10,3,0,"tomb_wall_01_150")
spawn("dungeon_iron_gate",13,11,0,0,"dungeon_iron_gate_1")
dungeon_iron_gate_1.door:setDoorState("open")
spawn("dungeon_iron_gate",13,15,1,0,"dungeon_iron_gate_4")
spawn("dungeon_iron_gate",9,13,3,0,"dungeon_iron_gate_6")
spawn("dungeon_iron_gate",14,14,0,0,"dungeon_iron_gate_7")
spawn("dungeon_iron_gate",15,15,3,0,"dungeon_iron_gate_8")
spawn("dungeon_iron_gate",18,16,2,0,"dungeon_iron_gate_9")
spawn("dungeon_iron_gate",18,15,0,0,"dungeon_iron_gate_10")
spawn("dungeon_iron_gate",17,12,3,0,"dungeon_iron_gate_11")
spawn("dungeon_iron_gate",15,10,1,0,"dungeon_iron_gate_12")
spawn("dungeon_iron_gate",5,11,0,0,"dungeon_iron_gate_2")
spawn("dungeon_iron_gate",13,12,0,0,"dungeon_iron_gate_3")
spawn("dungeon_pressure_plate",9,4,2,0,"dungeon_pressure_plate_1")
spawn("dungeon_pressure_plate",6,13,3,0,"dungeon_pressure_plate_2")
spawn("dungeon_pressure_plate",10,15,3,0,"dungeon_pressure_plate_3")
dungeon_pressure_plate_1.floortrigger:setTriggeredByParty(true)
dungeon_pressure_plate_1.floortrigger:setTriggeredByMonster(true)
dungeon_pressure_plate_1.floortrigger:setTriggeredByItem(true)
dungeon_pressure_plate_1.floortrigger:setTriggeredByDigging(false)
dungeon_pressure_plate_1.floortrigger:setDisableSelf(false)
dungeon_pressure_plate_1.floortrigger:addConnector("onActivate", "dungeon_iron_gate_1", "open")

dungeon_pressure_plate_2.floortrigger:setTriggeredByParty(true)
dungeon_pressure_plate_2.floortrigger:setTriggeredByMonster(true)
dungeon_pressure_plate_2.floortrigger:setTriggeredByItem(true)
dungeon_pressure_plate_2.floortrigger:setTriggeredByDigging(false)
dungeon_pressure_plate_2.floortrigger:setDisableSelf(false)

dungeon_pressure_plate_3.floortrigger:setTriggeredByParty(true)
dungeon_pressure_plate_3.floortrigger:setTriggeredByMonster(true)
dungeon_pressure_plate_3.floortrigger:setTriggeredByItem(true)
dungeon_pressure_plate_3.floortrigger:setTriggeredByDigging(false)
dungeon_pressure_plate_3.floortrigger:setDisableSelf(false)