

## **Co-Designing the Lighting of a Game Level**

**João Miguel Freitas Gonçalves de Oliveira**

Thesis to obtain the Master of Science Degree in  
**Information Systems and Computer Engineering**

Supervisor: Prof. Dr. Carlos António Roque Martinho

### **Examination Committee**

Chairperson: Prof. Rui Filipe Fernandes Prada  
Supervisor: Prof. Dr. Carlos António Roque Martinho  
Member of the Committee: Prof. Helena Sofia Andrade Nunes Pereira Pinto

**October 2019**



# Acknowledgments

First and foremost, I would like to thank Prof. Dr. Carlos Martinho, who even though I was not the easiest person to work with, always gave me the best guidance and support, and who without the completion of this work would not be possible.

I would also like to express my gratitude to the participants of the tests of our evaluation phase, Pedro Cabral, Rui Barata, Daniel Gonçalves and Miguel Francisco.

I would like to thank my family, who never gave up on me, even when I gave up on myself. The amount of support and patience through this process was unprecedented.

Also, thank you to my girlfriend who was my biggest motivation to conclude this chapter of my life. Without her patience and her constant inspiration this thesis would never come to an end.

Thank you to my friends, who always make me laugh and keep me in a good mood.

**Thank you.**



# Abstract

The inspiration for this work comes from the belief that computers can do more than perform computational tasks, but rather tackle a concept that most people think is impossible for a computer to achieve, due to being unique to the human being: Creativity. To demonstrate this, the proposed solution consists of a level design tool with co-creativity at its core, this is achieved by presenting the user with suggestions of possible modifications to their current level. There are two types of suggestions: level layout and lighting. Level layout consists on which tiles a player can walk through or not. And the latter, lighting is how each part of the level is illuminated to provide the player with the best experience possible. These suggestions are generated by genetic algorithms with parameters that can be adjusted through an user-intuitive interface, either it being more evolved towards level layout or the lighting setup.

## Keywords

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



# Resumo

A inspiração para esta dissertação vem da crença que os computadores podem ser usados para mais do que meras tarefas computacionais. Podem também ser usados para tarefas mais complexas como a criatividade. De forma a demonstrar este conceito, a nossa solução consiste na criação de uma aplicação focada no design de níveis para videogames. Esta aplicação irá funcionar em paralelo com um editor de níveis. Através de algoritmos genéticos irá gerar sugestões de possíveis alterações do nível, em termos da sua disposição e iluminação. As sugestões irão ser representadas na interface da aplicação. Nesta interface o utilizador pode também personalizar os parâmetros dos algoritmos genéticos de forma a gerar sugestões mais do seu agrado.

## Palavras Chave

Desenho de níveis, co-criatividade computacional, geração procedimental de conteúdo, algoritmos genéticos;





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	3
1.2	Problem . . . . .	3
1.3	Hypothesis . . . . .	4
1.4	Objectives . . . . .	5
1.5	Contributions . . . . .	6
1.6	Document structure . . . . .	6
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	Creativity . . . . .	9
2.2	Procedural Content Generation . . . . .	11
2.3	Discussion . . . . .	12
<b>3</b>	<b>Related Work</b>	<b>13</b>
3.1	Computational Creativity . . . . .	15
3.2	Genetic Algorithms . . . . .	17
3.3	Legend of Grimrock 2 . . . . .	18
3.4	Lighting in level design . . . . .	20
3.5	Discussion . . . . .	22
<b>4</b>	<b>Previous Work</b>	<b>23</b>
4.1	Creativity Computational Model . . . . .	25
4.2	User Interface . . . . .	26
4.3	Chromosome Representation . . . . .	26
4.3.1	Pedro Lucas's Representation . . . . .	27
4.3.2	Gonçalo Delgado's Representation . . . . .	28
4.4	Gonçalo Delgado's Breadth-First Graph Search . . . . .	30
4.5	Execution Flow . . . . .	31
4.6	Algorithms . . . . .	32
4.7	Discussion . . . . .	36

<b>5</b>	<b>Solution Model</b>	<b>37</b>
5.1	Computer colleague paradigm . . . . .	39
5.2	Solution Model . . . . .	40
5.2.1	Interface . . . . .	40
5.2.2	Behavior . . . . .	42
<b>6</b>	<b>Solution Implementation</b>	<b>44</b>
6.1	Solution Implementation . . . . .	45
6.1.1	Lighting Concept . . . . .	45
6.1.2	Editor integration . . . . .	46
6.1.3	Genetic Algorithm Implementation . . . . .	47
6.1.4	Editor Buddy Lighting Mode execution flow . . . . .	48
6.1.5	Algorithms . . . . .	50
6.1.6	Design decisions . . . . .	53
<b>7</b>	<b>Evaluation</b>	<b>57</b>
7.1	Solution evaluation . . . . .	59
7.2	Methods . . . . .	60
7.3	Results . . . . .	63
7.4	Study Conclusions . . . . .	71
7.5	Recommendations . . . . .	72
<b>8</b>	<b>Conclusion</b>	<b>73</b>
8.1	Solution performance . . . . .	75
8.2	Future Work . . . . .	75
8.3	Final Remarks . . . . .	76
<b>A</b>	<b>Questionnaire Appendix</b>	<b>79</b>

# List of Figures

2.1	Comparison between Vertical and Lateral Thinking. . . . .	10
3.1	Genetic Algorithms examples of the Crossover and Mutation phases. . . . .	18
3.2	Example of one level Legend of Grimrock 2, where a player is facing a charging monster. . . . .	18
3.3	Example of the Dungeon Editor of Legend of Grimrock 2, as can be seen the designer has 32 by 32 grid where he can place any item from list on the left. On the right side there is a preview window of the current level.. . . .	19
4.1	Editor Buddy's expert interface. . . . .	27
4.2	Editor Buddy's Puzzle Mode interface. . . . .	28
4.3	2D map to chromosome transformation. . . . .	29
4.4	Example of custom crossover applied to a pair of chromosomes. . . . .	29
4.5	Example of custom crossover applied to a pair of chromosomes. . . . .	29
4.6	Editor Buddy Puzzle Mode example map and its corresponding chromosome representation (cropped). . . . .	30
4.7	Editor Buddy's execution flow. . . . .	31
4.8	Editor Buddy's Puzzle Mode execution flow. . . . .	32
5.1	Editor Buddy Lighting Mode interface. . . . .	41
5.2	Diagram of the interaction of the solution with a level editor. . . . .	43
6.1	Example of the Editor Buddy Lighting Mode 2-Dimension (2D) canvas when displaying the generated heatmap. . . . .	46
6.2	Example of level and its chromosome representation, where the light grey tiles represent the walkable tiles of the level and the triangle shaped objects the torches. . . . .	48
6.3	Editor Buddy Lighting Mode algorithm execution flow diagram and its element captions. . . . .	49
7.1	Editor Buddy Lighting Mode user testing User Interface (UI). . . . .	60
7.2	The level participants were asked to illuminate in task 1 and 2. . . . .	61

7.3	Tricksters Lair level from LoG2 that participants were asked to illuminate in task 3. . . . .	61
7.4	Setup used in the user testing. . . . .	62
7.5	Chart regarding the questionnaire's question 3. . . . .	64
7.6	Innovation algorithm generated suggestions - expectation vs usefulness . . . . .	65
7.7	Objective algorithm generated suggestions - expectation vs usefulness . . . . .	66
7.8	User Map algorithm generated suggestions - expectation vs usefulness . . . . .	66

# List of Tables

6.1	Simple test with two Generational Limits and how they affect the algorithm's execution time.	55
7.1	Questionnaire usability results . . . . .	65
7.2	Screen recordings task 2 and 3 results. . . . .	67
7.3	Comparison between the screen recordings task 3 results with and without one of the participants. . . . .	67

# Acronyms

<b>EB</b>	Editor Buddy
<b>EBLM</b>	Editor Buddy Lighting Mode
<b>EBPM</b>	Editor Buddy Puzzle Mode
<b>UI</b>	User Interface
<b>AI</b>	Artificial Intelligence
<b>LoG2</b>	Legend of Grimrock 2
<b>PCG</b>	Procedural Content Generation
<b>GA</b>	Genetic Algorithms
<b>2D</b>	2-Dimension
<b>1D</b>	1-Dimension
<b>IST</b>	Instituto Superior Técnico
<b>GAF</b>	Genetic Algorithms Framework
<b>GUI</b>	Graphical User Interface

# 1

## Introduction

### Contents

---

1.1 Motivation . . . . .	3
1.2 Problem . . . . .	3
1.3 Hypothesis . . . . .	4
1.4 Objectives . . . . .	5
1.5 Contributions . . . . .	6
1.6 Document structure . . . . .	6

---





## 1.1 Motivation

Creativity is an aspect of the human mind which is yet to be fully comprehended. It is a unique quality that every human being possesses. However not every person thinks the same way and some people are naturally more creative and others need a stimulus to awaken their creativity.

This is the inspiration for our work, to create a tool that works as a colleague providing suggestions and alternatives in order to spark the creativity within a person. More specifically to this work, we believe that our tool will help level designers to achieve a higher satisfaction with the levels they have produced.

Some people consider the process of creating a level an arduous and dull task, and to facilitate it some tools have been created, such as Unreal Engine<sup>1</sup> or GameMaker<sup>2</sup>, by providing the user the means to generate terrain, visual effects, light sources etc. Despite game engines being used to facilitate tedious tasks, none of them are used in the paradigm we want to explore, computer as colleague in the creative process. Although, there are few tools which tackle this concept, the most notorious being the Sentient Sketchbook and Tanagra, which share the same vision as this work. Another work that is worth mentioning is Pedro Lucas's Editor Buddy, which consists in an application that co-creatively builds a level layout with its level designer. This work's motivation is the belief that a computer can work as colleague in level creation and help novice level designers who lack the skills to construct a level from scratch. Or even to simply stimulate creativeness of more advance level designers, making them overcome moments of absence in creativity or simply suggesting paths that did not cross their minds.

Although some tools, have been designed with this concept in mind, none or only a few have the ability to co-creatively build the lighting of a level with its user, which is one of the most important aspects in level design. The illumination of a level has the ability of delivering different feelings and experiences to a player. Our work will be focused on giving a level designer the experience of co-developing the lighting of a level.

## 1.2 Problem

Traditionally computers are not coded to be creative, so how can we take this concept and provide a colleague that can absorb a level designer's idea and present meaningful suggestions to it. Can this Artificial Intelligence (AI) deliver an experience as good or even better than what a user could achieve with another user's interaction? How can a tool stimulate the user to explore different paths?

As mentioned, similar tools already exist. The Sentient Sketchbook by Antonios Liapis [1] offers a simple yet effective approach of displaying the user with multiple alternatives to his/hers design. However, despite being simple to use, the tool lacks interaction and customization to its parameters, which

---

<sup>1</sup> Epic Games 1998

<sup>2</sup> YoYo Games, 1999

in the end affects the user's experience. And Tanagra by G. Smith, J. Whitehead, and M. Mateas [2] is comparable to the Sentient Sketchbook but has a deeper focus on platform games and still lacks options for customization, functioning more individually when generating content.

So how can we create a tool that works closely with the user and gives a valuable contribution to the level's design? This is what we aim to accomplish, to create a tool that co-creatively develops a level with its user by generating innovative, correct layouts and interesting lighting.

Two former students from Instituto Superior Técnico (IST) have each created their own solutions to this computer as colleague paradigm in level design. Pedro Lucas's work focused on building a tool that co-created a level layout with a user. And Gonçalo Delgado developed a tool that receives a fixed level layout and, along side the level designer, generates puzzles that a player must fulfill in order to successfully complete the level. However creating a co-creative experience is not a simple task. Both Lucas and Delgado followed the same approach, they created an interactable user interface with three sliders. Each slider controls the parameters of the algorithms responsible for generating suggestions and each correspond to one of the three core components of co-creative experience (innovation, guidelines and convergence). By providing the users with the capability to adjust the algorithms' parameters, it guides the application's creativity into generating more valuable suggestions.

Our solution resides in taking the concepts Pedro Lucas has constructed and developing our own algorithms that provide suggestions on level illumination.

However, there is the problem of the computational efficiency, since we strive for the best user experience possible, the algorithms must be quick generating suggestions. The magnitude of this problem becomes even greater when we realize that both algorithms cannot run in parallel, since in order to create proper lighting for a certain level layout, the algorithm must already know the layout. Therefore the algorithms must be ran sequentially, first the level layout algorithm that Pedro Lucas developed, then our own lighting algorithm that takes the generated level layout as an input and illuminates the level.

### **1.3 Hypothesis**

Considering that concept of creativity allows for an unlimited number of distinct ways a human mind can think, we needed to find an approach that could be translated into a computational model and that could be applied into level design. Upon reflecting and investigating the method Pedro Lucas took to overcome this challenge, we plan on exploring a similar design as his. Modeling the computational creativity into three concepts in order to generate suggestions. We believe these three concepts are three core elements in the process of co-creating a level and that in absence of one, it would shed the idea of a cooperative experience.

The first core element is to take the current design of the level designer into account. Otherwise the

concept of the co-creative experience is lost, since when interacting with a partner we always have to acknowledge their work.

The second element tackles the concept of creativity, meaning that we need to produce content that is innovative or completely new. Without it, tackling this notion our work would be meaningless.

The third and last concept is the objective, without it the level becomes meaningless and level design would lose its purpose, as a player would not have any goals to achieve.

We plan to take this approach of the three core components of co-creating a level and apply them to development of an application, that is able to generate meaningful creative suggestions regarding the level's lighting.

## 1.4 Objectives

The objective of this work is to build a tool that mimics the interaction of a person working along side the user, to create a level in terms of its layout and the lighting. In simpler terms, to create a tool that works as peer in the process of creating a level. This paradigm of having the computer function as a peer, is explained in more detail in the Related Work section.

The tool will function as a separate application to the level editor of the game Legend of Grimrock 2<sup>3</sup>, yet it will take the user's modifications as an input and properly show suggestions to possible alternatives the user might wish to pursue.

We believe that in order for these suggestions to be appealing to the user, they must be fresh, innovative ideas that follow the criteria of how the user is building his/hers level. To do so, the tool must evaluate the level and generate alternatives that follow the user's preferences, either them being innovative content or suggestions that do not heavily deviate from their current design.

In the end, we draw some conclusions as to whether this tool hinders or improves the end user's performance, we will be performing a series of user tests with subjects that we consider to be the target audience for this application: inexperienced level designers.

To conclude, we explain the evaluation phase which was used to take some conclusions on how valuable this tool is. Since our target audience is people with slight experience in level design, the tests were done with novice level designers, that have basic concepts on level design or might even built some levels.

---

<sup>3</sup> Almost Human Ltd., 2014

## 1.5 Contributions

We believe our work contributes to the scientific community in the following matters:

- In co-creative Procedural Content Generation (PCG) techniques for videogame level design.
- The generation and evolution of content alongside the level designer.
- How two softwares can work side by side with different yet similar goals (our tool and the videogame level editor).
- How the process of illuminating a level can be transformed into a co-creative experience.

We also present a self-critical view of our developed work while also giving advice for anyone who is interested in developing a project in the subject.

## 1.6 Document structure

The document's first chapter starts by introducing our motivation, problem and objectives for this work. Afterwards, on the second chapter, we present a background on topics that will be discussed throughout the document, such as the concept of creativity and Procedural Content Generation. The third chapter, entitled Related Work, describes the current state of the art and other scientific works that contributed to our work. This chapter includes topics such as computational creativity, genetic algorithms and the logic behind videogame level illumination. The fourth chapter, is entirely dedicated to two works by Pedro Lucas and Gonçalo Delgado, that share the same vision of this work and that were used as a foundation for our solution. The next two chapters, describe in detail the implementation of our solution, as well as any constraints and how we faced them during its development. The seventh chapter, is dedicated to the evaluation process, it describes the tests that were performed to evaluate the usefulness of our solution. Lastly, in the eighth chapter, we present our conclusions of all of the developed work as well as providing some areas that it could be improved in a future work.

# 2

## Background

### Contents

---

2.1 Creativity . . . . .	9
2.2 Procedural Content Generation . . . . .	11
2.3 Discussion . . . . .	12

---



In this chapter we will analyze what has been done in the field of computer science that contributes to this document. We will consider what Creativity is and how it can be modeled as a language a computer can comprehend. Then we will investigate Procedural Content Generation as it is relevant in the context of computer creativity and is also vital for our proposed solution. To conclude, there is a brief discussion about the topics and how they are relevant to our work.

## 2.1 Creativity

“Creativity is defined as the tendency to generate or recognize ideas, alternatives, or possibilities that may be useful in solving problems, communicating with others, and entertaining ourselves and others.”  
- Human Motivation, 3rd ed., by Robert E. Franken. [3]

Creativity is a quality every human being possesses, some more than others. It is an innate skill of the human mind, that is not fully comprehended and therefore continues to be studied. However with those studies, definitions of creativity become clearer and Margaret Boden [4] perceptions of how creativity can be described are relevant to our work. Margaret Boden categorizes creativity into two terms, P-Creativity and H-Creativity.

Psychological-Creativity or P-Creativity in short, is described as a valuable idea that is completely new to the person who came up with it, even if such idea has previously occurred to someone else. In contrast, Historical-Creativity or H-Creativity is described in the same way as P-Creativity, however it has to be completely new, meaning that no one in history has ever thought about it before. In Margaret Boden's words:

“.. has arisen for the first time in human history” [4].

As previously described, our work is the creation of a tool that co-creatively creates level designs with the user and there is a limit to the different possible variations of how a level can be created in Legend of Grimrock 2. This and the fact that the tool absorbs the current level design produced by its user, the logical type of creativity that we will focus for the scope of this work is P-Creativity.

Margaret Boden work also categorizes creativity into three different approaches:

- **Combinational creativity:** “Produces unfamiliar combinations of familiar ideas, and it works by making associations between ideas that were previously only indirectly linked.” [5]
- **Exploratory creativity:** Consists on exploring the preconceptions in a conceptual space to generate new ideas.

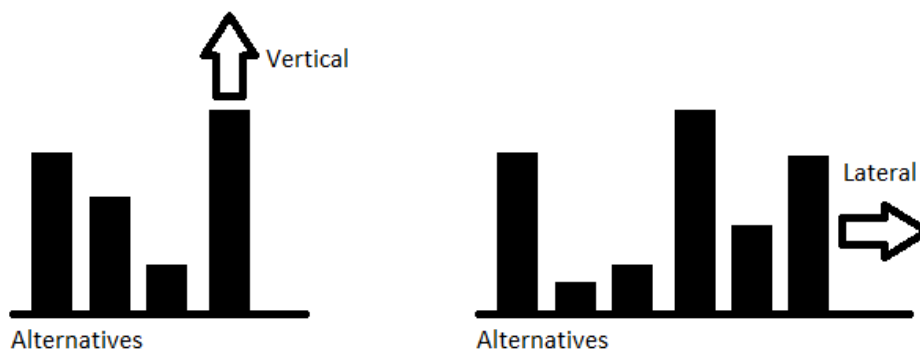
- **Transformational creativity:** Is the alteration of the conceptual, leading to the generation of unthinkable ideas.

The idea of generating lighting within a predetermined level layout makes use of Exploratory creativity, as the conceptual space is already known and our interest is to explore the preconceptions in it.

## Lateral vs Vertical thinking

“Vertical thinking is selective, lateral thinking is generative.” - Lateral Thinking by Edward de Bono. [6]

The term Lateral thinking was first promulgated by Edward de Bono. As he describes, Lateral thinking is the process of generating as many alternative ideas as one possible can. Lateral thinking can be seen as producing new ideas by constantly clashing with existing ones. Whereas Vertical thinking is the most typically used by the human mind, where one individual selects the best possible approach to a certain problem and tries to refine it.



**Figure 2.1:** Comparison between Vertical and Lateral Thinking.

“With vertical thinking one is trying to select the best approach but with lateral thinking one is generating different approaches for the sake of generating them.” - Lateral Thinking by Edward de Bono. [6]

Although Vertical Thinking might seem the most logical approach to follow, where we take the lighting a level designer has created and try to improve it, we believe that this approach does not deliver a co-creative experience, as it lacks the previously mentioned innovation component. Therefore we decided to follow the concept of Lateral Thinking, where the suggestion to be display to the user will be created by generating multiple different ideas. This will be explained in more detail at a further section, however we considered important in explaining how Edward de Bono’s work contributed to our future tool’s behavior.



## 2.2 Procedural Content Generation

Procedural Content Generation is a method of generating content through random or pseudo-random algorithms. The main reason for building PCG into the core of a game, is the level of replay value it offers to the user, as the game randomly generates its content, offering its users with constantly diverse experiences between levels. PCG has become somewhat popular nowadays, with a wide range of games using it as a core mechanic, such as the widely popular *The Binding of Isaac*<sup>1</sup> or more recently the game *Dead Cells*<sup>2</sup>, however the idea of PCG is quite old, dating back to 1980 in the game *Rogue*<sup>3</sup>.

Procedural Content Generation is also an effective approach in reducing the production costs, which is quite clear in the case of level design generation. For example, instead of having employees developing fifty levels manually, PCG is a compelling way to avoid this burden, as one algorithm can generate a multitude of levels all by itself. However, as good as it sounds, not many game developers utilize it, one reason being the struggle to generate this content at a reasonable performance cost. As Togelius [7] states, PCG can be performed in two ways: Online and Offline. Online is the generation content in real-time as the player is advancing through the levels, which comes at high expense of computational performance. And Offline, all the computation is executed beforehand in the game's development phase, which alleviates the performance bottleneck from the end-user but may affect the game's replayability.

Although the main reason for PCG not being a standard in all modern games is its difficulty to generate well crafted experiences. When producing content, level designers can tweak certain aspects of a level or include details that add to a more complete experience.

The use of PCG by game developers is drastically rising, most noticeably in indie games, however this trade-off between replay value and custom-tailored level designs, is why most of the game developers for AAA games (games with the highest development budgets and levels of promotion) shy away from PCG.

To our work, Procedural Content Generation is crucial as the tool requires the generation of content by itself. One way of implementing PCG is by using Genetic algorithms which will be examined in the next section.

---

<sup>1</sup> *The Binding of Isaac*, PC, 2011, Florian Himsl, Edmund McMillen

<sup>2</sup> *Motion Twin*, 2018

<sup>3</sup> M. Toy and G. Wichman, 1980

## 2.3 Discussion

This section provided some insight on the current understandings of human creativity and their different types - Combinational, Exploratory and Transformational. We also investigated Vertical and Lateral thinking, the two styles of thinking proposed by Edward de Bono.

In terms of Procedural Content Generation, we have discussed online vs offline and as our tool's mission is to provide the user with meaningful suggestions to its current design, the only logical option is to resort to online PCG.

Finally in terms of Creativity, we will mainly focus on Exploratory creativity. As for the type of thinking, we feel the most appropriate is the Lateral thinking line of thought, since we desire to search as many different alternatives as possible to a certain solution rather than trying to take the best one and explore it to the fullest.

# 3

## Related Work

### Contents

---

3.1 Computational Creativity . . . . .	15
3.2 Genetic Algorithms . . . . .	17
3.3 Legend of Grimrock 2 . . . . .	18
3.4 Lighting in level design . . . . .	20
3.5 Discussion . . . . .	22

---



In this chapter we discuss relevant work from which we retrieved examples to guide our design decisions. We will briefly explain the topics, while also discussing why they are relevant for our work, as well as how intend to apply the concepts of each topic to the development of our solution. We start by tackling the concept of transforming creativity into a computational model. Afterwards we explain the theory behind Genetic Algorithms and how they work. Next, we present the videogame we chose for our solution to work side-by-side. To conclude, we explore the concept of Lighting in level design and have a discussion about the topics of this section.

### 3.1 Computational Creativity

How can a computer be programmed to be creative? Computers were built to perform sequences of arithmetic or logical operations, so how can they be instructed to act as creative mind? Margaret Boden states that some people say it is impossible, since it was a human who programmed the computer to be creative in the first place, is this creativity really genuine?

Boden tries to address this by going back to her three different types of creativity:

In terms of Combinational creativity, computers are very capable of taking two different ideas and combining them together into a new one, however how can we determine if this new idea is indeed correct or if it offers any valuable meaning? Computer generated ideas need evaluation, which raises a distinct problem. How can a computer evaluate its own creative outputs? This requires an immense amount of information in order to check if the idea is brand new or if it is even valid. Databases containing a vast amount of information already exist, consider how Google and Wikipedia store their information. Although absorbing all of their data to evaluate the value of one idea leads to unreasonable load in computational power. Boden gives an explain of a riddle generator named JAPE [8] and as she states:

“The prime reason that JAPE’s jokes are not hilarious is that its associations are very limited, and also rather boring, when compared with ours.” [5]

Exploratory creativity AI can too be modeled in computers, in fact it is regarded as more successful than Combinational creativity due to the quality of its solutions being more closely matched to the ones achieved by humans. Boden gives an example of a work in this field, AARON [9], created by Harold Cohen which is a computer program that generates original artistic images.

Lastly for Transformational creativity a program needs to able to transform the conceptual space in order to encourage the generation of new ideas. Evolutionary algorithms are a case of Transformational creativity, who constantly change the rules as a part of their mutation phase. However, they suffer from the same problem we mention in Combinational creativity, how can we evaluate if the output is a valuable idea? Evolutionary algorithms use fitness functions whose job is to determine the quality of

each individual, thus ensuring they are worthy of proceeding to the next phase. Nonetheless, it becomes a challenge when the topic in question is related to creativity and the generation of ideas.

Solutions is a term we have consistently spoken of, although how can we qualify it? Upon reading Margaret Boden [4] she states: "Because creativity by definition involves not only novelty but value." According to her definitions:

- **Novelty:** The quality of being new, original or unusual. It quantifies how unique the solution found is.
- **Utility or value:** The importance, worth or usefulness of something. Meaning that it evaluates how useful the solution found is.

In the interest of our work, we desire both to be achieved. We intent to produce solutions with novelty, while making sure they are in fact valuable to what the user is trying to accomplish.

## Computers as co-creative partners

As briefly described before our project is to build a tool that works along side the user, but how can a computer work as partner? Lubart [10] addresses this concept and categorizes it into four distinct approaches:

- **Computer as nanny:** The computer takes a more passive role. It pressures the user to work in order to complete its deadlines while detecting periods of procrastination.
- **Computer as pen-pal:** It requires multiple individuals and the computers job is to promote the communication between them. It allows the sharing of ideas, to improve the creative process.
- **Computer as coach:** The computer is used as jump-start to the creative process by providing information in a different approach that humans would not necessarily think of.
- **Computer as colleague:** It is the most ambitious vision, it involves having the computer working as peer in the creative process.

In the interest of our work, to have a co-creative process between human and computer, the tool will work accordingly to the computer as colleague paradigm.

## 3.2 Genetic Algorithms

Inspired by Charles Darwin's theory of natural evolution [11], where the fittest individuals are selected and used for reproduction, in order to produce the offspring of the next generation, Genetic Algorithms (GA) (a subclass of Evolutionary Algorithms) are used to generate high-quality solutions to search and optimization problems.

The natural selection process starts by selecting from the population the fittest individuals. They reproduce and generate multiple offsprings that inherits their characteristics, which will be later on passed to future generations. By constantly matching the most fitted individuals, it ensures that their offspring will be parents with higher chances of survival. This process is iterated until a solution to the problem is found, this solution will consist of the fittest individuals. Genetic Algorithms have five phases:

1. **Initial Population:** The process starts with a set of individuals, called Population. Each individual is a solution to the problem in question and they possess a set of parameters/variables known as Genes, which are combined into a string, more commonly referred as a Chromosome.
2. **Fitness Function:** It calculates the quality of each individual when compared to the others in the population. This fitness score is then used to determine which individuals are the most appropriate for reproduction.
3. **Selection:** It selects the fittest individuals to go the next phase. Individuals are selected in pairs (Parents) based on their fitness scores. The individuals with a higher fitness score, are more likely to advance to the next phase.
4. **Crossover:** Is the most crucial phase of the GAs. For each pair of parents a crossover point is randomly chosen within their genes. The offsprings are then generated by exchanging the genes of the parents until the crossover point is reached.
5. **Mutation:** When a new offspring is created, there is low probability that some of its genes may be subjected to mutation. This implies that some of its bits can be flipped. Mutation occurs to ensure diversity within the population thus preventing premature convergence.

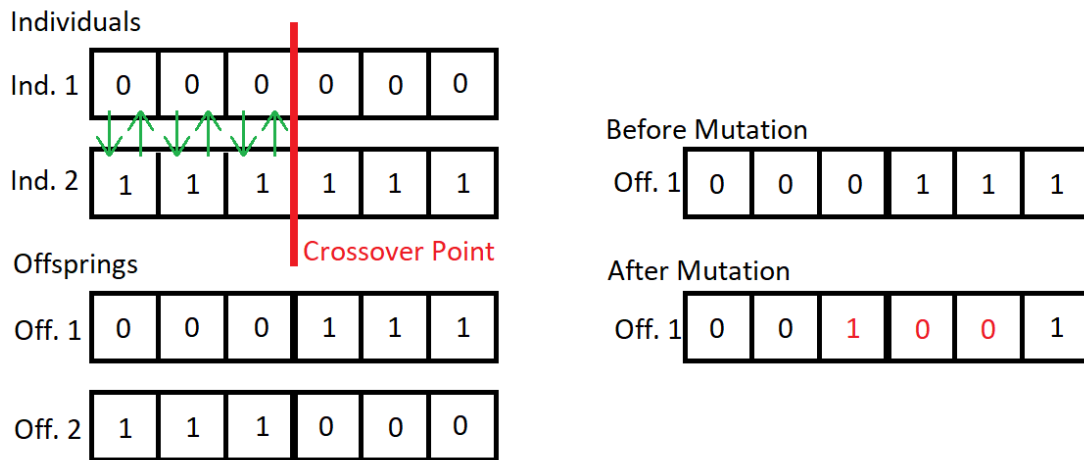


Figure 3.1: Genetic Algorithms examples of the Crossover and Mutation phases.

### 3.3 Legend of Grimrock 2

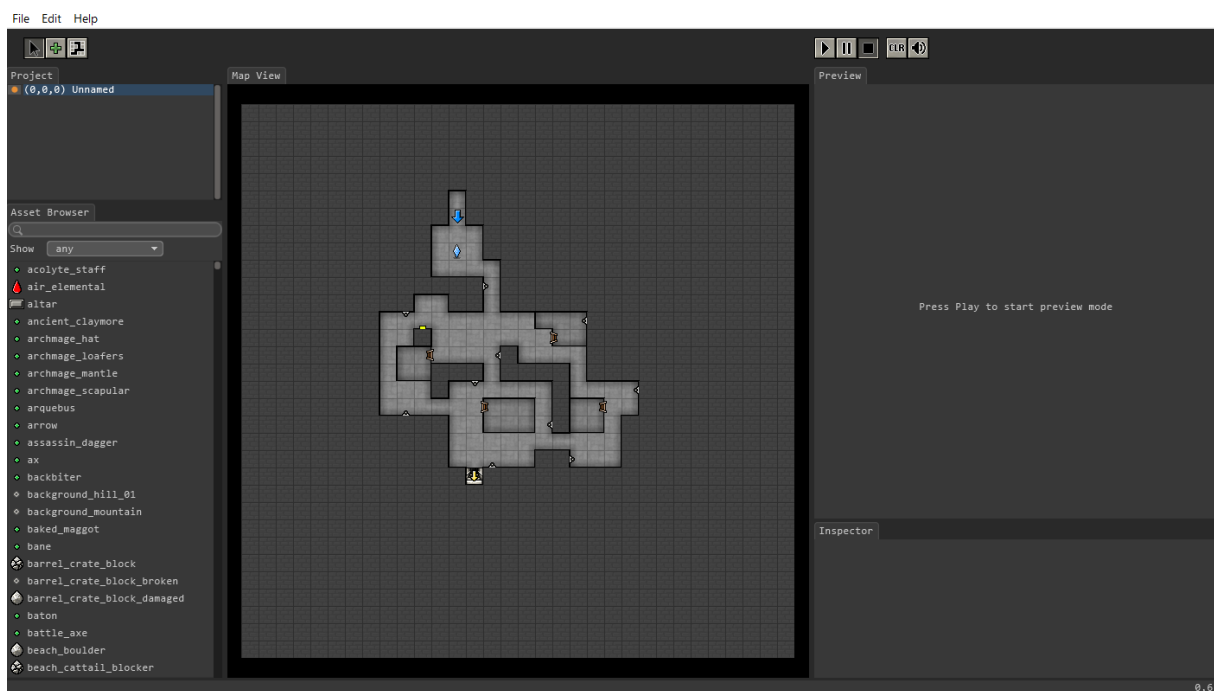
Legend of Grimrock 2 (LoG2) (Figure 3.2) is a tile-based real-time dungeon crawler videogame that was developed and published by Almost Human, where a player can control four unique characters. Each character has its own traits and abilities, with each one having both strengths and weaknesses. The goal of the game is to successfully advance through the different maze-like dungeons, which are filled with monsters and bosses that a player must defeat. There are also puzzles within the level which the player must decipher in order to advance to new sections of the dungeon.



Figure 3.2: Example of one level Legend of Grimrock 2, where a player is facing a charging monster.



However, for our work the most relevant feature of the game, is its built in level editor, called Dungeon Editor (Figure 3.3), which features an intuitive layout with a 32 by 32 square grid that represents the tiles in which a designer can place walkable terrain, walls, puzzle items, enemies or light sources. The Dungeon Editor, on the left side of application, provides the complete list of items the designer can place on the level's grid. On the right side of the tool there is one of the most interesting features, a window that allows the user to preview the level without the need of opening the game. Under the preview window, there is a window for LUA scripting that can be used to change the assets parameters.



**Figure 3.3:** Example of the Dungeon Editor of Legend of Grimrock 2, as can be seen the designer has 32 by 32 grid where he can place any item from list on the left. On the right side there is a preview window of the current level..

A crucial feature of the level editor to this work, is how it saves the project files. The user created levels are stored in plain text files, which prevents issues with the level editor integration. Parsing the project files is a simple task, as the Legend of Grimrock 2 developers created an easy syntax to represent every element of the level, each component is clearly labeled, such as the level layout and the placement of in-game items. Essentially, it means, that we can parse the file responsible for the level and extract all the needed information, while also saving it in the same manner. Another great feature when saving the project file, is that the LoG2 Dungeon Editor automatically reloads the project, meaning that the applied suggestions are instantly displayed to the level designer without the need of extra actions.

All these details make **LoG 2! (LoG 2!)** and its level editor a perfect candidate for our work, combining with the fact that both Lucas and Delgado had already tested it in a similar concept we are exploring.

### 3.4 Lighting in level design

“Lighting is one of the designer’s greatest tools to establish mood.” - Beginning Game Level Design, John Feil and Marc Scattergood [12]

The art of level illumination can go by unnoticed in the process of creating a level. However, it is one of the most important elements to give character to a level and it is not all about its placement. Details such as the intensity, color, movement and shadows, all affect the player’s experience. The correct use of lighting can transmit different emotions to a player, such as fear or wonder, for example darker levels usually transmit the sense of anxiety and insecurity. However, lighting a level is a difficult task to do right, adding or removing one light source can have a tremendous impact on the overall level experience.



**Figure 3.4:** A screenshot of the game Bioshock Infinite<sup>1</sup>, that uses light to give the statue a godly status.

John Feil and Marc Scattergood divide lights into two groups: static lights and dynamic lights:

- **Static lights** are pre-rendered into the level, meaning that their are actually a part of the environment. The use of static lights is very common, since they are computationally inexpensive and usually the best way to illuminate most of the level’s game objects. However, they come at the cost of realism.
- **Dynamic lights** are generated in real time, meaning the game engine has to render every light, reflection and shadows in every frame. At the cost of performance, dynamic lights are much more

---

<sup>1</sup>Irrational Games and 2K Games, 2013

natural looking than static lights, since the shadows and light effects are generated in real time and evolve with the movement of the scene.

Level designers usually try to strike a balance in use of the two, so that the level's performance is acceptable while also maintaining visual fidelity.

In term of light sources, game engines usually provide three types:

- **Point or ambient** are light sources that emit light in a 360-degree field. These are usually used to provide the general illumination of a level, they are commonly known as fill lights.
- **Spotlights** are strong directional lights that usually have a discernible start and end. Typically are use to highlight certain areas or objects as they draw the player's attention to the spotlight's end.
- **Directed lights** are directional lights, just like spotlights but not as intense. They are still strong enough to illuminate small objects or specific areas and they are usually used to complement already existing lights.

To our work, as we will focus on the creation of lighting for dungeon levels, we will mainly be using the torches as our light sources. Torches in Legend of Grimrock 2 are Point lights that emit light in 360 degree field and can reach a maximum of 5 tiles. They are Dynamic lights, meaning that the illumination they provide is generated in real-time as a player is advancing through the level.

## Light usage

As mentioned, illumination can have a tremendous impact on the player experience, so light usage should be considered as one of the most important aspects when building a level. One very common technique that level designers use, is the use of lighting to guide players towards the level's ending or to important items in the level. Players naturally tend to travel through or towards the most illuminated sections of a level, as they provide better knowledge of the player's surroundings while also giving a sense of security. However, level designers often do the opposite, hiding the level's objective or important items in the darkest sectors of the level, in order to challenge the players into the unknown. The approach a level designer takes into lighting a level, heavily depends on the game genre or its current point in the game's plot. We have identified three more technical aspects a level designer must keep in mind when placing the light objects:

The first step a level designer has to think of when developing the lighting in a level, is the positioning of the light sources. The placement of the light objects has to be well thought out, as they need to blend in with the environment.

The second step is to think of how the illumination contributes to player's experience. Lighting is a fantastic approach to set the player's mood, for example cold colors, such as blue transmit the feeling of a cold environment.

Another aspect to keep in mind, is the positioning of the light source. Although, the level designer has already chosen where the light source is, the way that it is positioned affects the player's feelings, specially when taking into account how the shadows are cast. For example, a light source placed at low level in front of a monster will cast a bigger shadow, thus making the enemy scarier.

The last thing to consider is light intensity. A high light intensity can wash out the color of a scene, while low intensity can shroud the scene in gloom. Also another detail to keep in mind is, the more focused the light is, for example spotlights, the higher its intensity will be. In games, level designers usually use lights of medium intensity.

In terms of light usage for our work, since we will be using Legend of Grimrock 2 torches and we will mainly focused on their placement, since their light intensity and color are already pre-defined. The torches have a fixed intensity that reaches a maximum of five tiles and their color is mostly yellow and red as they simulate fire.

### **3.5 Discussion**

In this chapter we started by studying how can we transform creativity in a computational mode, as well as presenting four different roles a computer can fulfill as co-creative partner.

As for Genetic Algorithms, we studied their value in search and optimization problems. GAs are responsible for the Procedural Content Generation. They are in charge of generating the suggestions in our proposed solution and to do so, we will modify the existing frameworks in order to be in line with our objectives of generating suggestions for both level layout and level illumination.

Afterwards we presented the videogame Legend of Grimrock 2 and its Dungeon Editor and also described their role in this work.

Lastly we explored the concept of lighting in level design, presenting the types of light sources that are used in the illumination of levels. We also briefly described the process level designers go through when lighting a level, as well as discussed how the LoG2 light sources work.

# 4

## Previous Work

### Contents

---

4.1 Creativity Computational Model . . . . .	25
4.2 User Interface . . . . .	26
4.3 Chromosome Representation . . . . .	26
4.4 Gonçalo Delgado's Breadth-First Graph Search . . . . .	30
4.5 Execution Flow . . . . .	31
4.6 Algorithms . . . . .	32
4.7 Discussion . . . . .	36

---



Pedro Lucas and Gonçalo Delgado have both developed works in the theme of Level Content Co-creation, which will be the foundation for our work.

Lucas created a tool called Editor Buddy (EB) [13], which allows for the creation of a level layout along-side a human using the Legend of Grimrock 2 video game. The Editor Buddy takes as an input the level the user has created and displays suggestions.

Delgado's work was inspired by Lucas's Editor Buddy and shares most of the principles found on his tool. However, Delgado's tool, called Editor Buddy Puzzle Mode (EBPM) [14], instead of co-creating the level's layout, focuses on puzzle generation. The generated suggestions represent changes to the puzzles found on the user's level, such as pressure plate and gate locations, as well as their connections. However, as it was not the focus of his work, Delgado did not use the level layout generation that Lucas developed, instead he decided to use a fixed level layout and construct the puzzle generation around it.

## 4.1 Creativity Computational Model

In our Hypothesis section we discussed how we could take the concept of co-creativity in level design and translate it into a computer model. Lucas design to overcome this challenge, was to break down co-creativity in level design into the three core components we have mentioned.

For each of the three elements, Lucas developed a genetic algorithm with the purpose of generating a suggestion that could fulfill the cores of co-creativity.

- **Convergence algorithm:** Focuses on the current co-proposal. It takes the level being developed by the level designer and generates similar suggestions. Its purpose is to always take the designer's level in consideration, so that the concept of a cooperative experience is not shattered.
- **Innovation algorithm:** Focuses on exploring new directions. Meaning it is responsible for bringing novelty into the level. It generates suggestions as distinct as possible from the level designer's level. Without it there would not be a creative component.
- **Guidelines algorithm:** Focuses on the generation of content that respects specific design goals. It ensures that the generated suggestion is aligned with the level designer's goals. These goals depend on the task or concept of the project. In Lucas's case of level layout, there are two objectives, the generation of suggestions that either focused on narrow halls or on rooms.

The influence each of the algorithms has in the final suggestion to be displayed to the level designer, can be regulated by three sliders present in the application's user interface.

Delgado's approach was the same as Lucas, however since his work was targeted at the puzzle component of level design, he adjusted the Objective algorithm's goals. Both Lucas's and Delgado's implementation of the algorithms will be further explained in a future section of the document.

## 4.2 User Interface

Both works have a similar application User Interface (UI) with these main components:

- A 2-Dimension (2D) canvas where the program's generated content is displayed;
- A set of buttons which can be used to interact with the generated suggestion, such as a revert button and export button (it applies the suggestion to the designer's level);
- Three sliders that represent the three core elements to a co-creative experience as we explained in the previous section. These sliders can be adjusted by the level designer, in order for the application to generate suggestions more according to his/hers intentions.
- A box that displays the tool's activity/logs;

The figures 4.1 and ?? show both of the tool's interfaces, in which these similarities can be clearly observed. However, some key differences are also noticeable, such as the two extra sliders in Delgado's work, which a user can set appropriately depending on his/hers preference of level Backtrack or level Length. Delgado also added four bars, each one displaying one of the algorithms' progress. Lucas' Editor Buddy has the additional functionality of allowing a user to choose between basic and expert mode. In basic mode, the interface is simpler and does not contain the sliders, allowing the user to simply enable or disable the algorithms.

In terms of Algorithms both used Genetic Algorithms, more specifically they used the Genetic Algorithms Framework (GAF) written by John Newcombe [15]. In addition Delgado also wrote a Breadth-First Graph Search algorithm, which will be explained further down in this section.

The path we plan on exploring is to use Pedro Lucas's work and make it cooperate it with a different concept, lighting in level design. We will utilize Lucas's algorithms and, alongside our developed algorithms, produce meaningful suggestions, in terms of level layout and lighting. If a user were to introduce changes to his current level, the algorithms will take it as an input, communicate and generate suggestions in compliance to what the user is trying to achieve.

The next part of this section will be a detailed comparison between both implementations, starting with how each of them developed their own chromosome representation.

## 4.3 Chromosome Representation

In this subsection, we will explain how Lucas and Delgado developed their chromosome representation. Each work has a different objective, in the case of Lucas's is level layout generation and Delgado's is puzzle element generation, which leads to reasonably distinct representations.



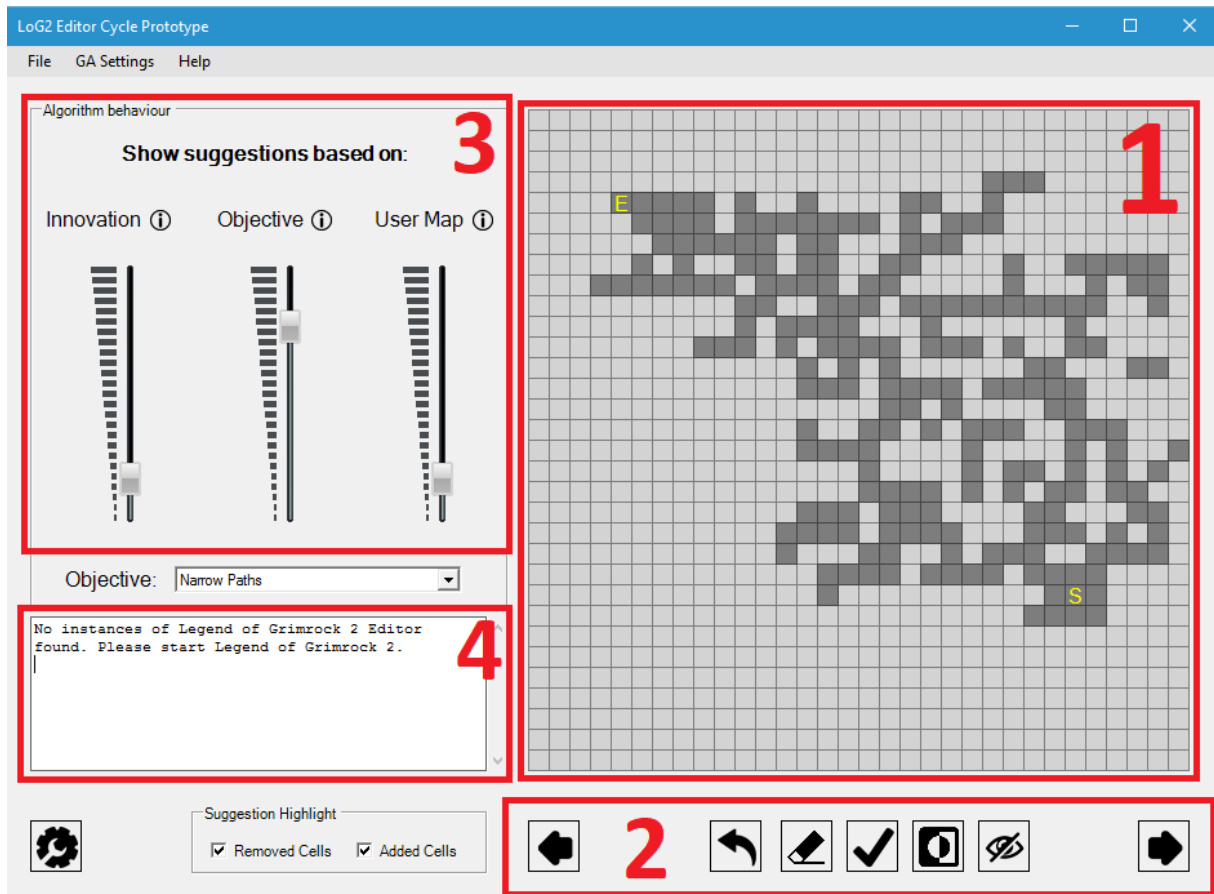


Figure 4.1: Editor Buddy's expert interface.

Although we will not be using Delgado's work it is still valuable to examine his algorithms and gather knowledge, with it we hope to gain a better grasp on genetic algorithm construction.

### 4.3.1 Pedro Lucas's Representation

As mentioned, Pedro Lucas used the Genetic Algorithms Framework in the development of his tool. GAs are typically built for simple 1-Dimension (1D) structures as the population is commonly represented by strings of bits, however for level layout design Lucas needed to take this concept and apply it to a 2-Dimension world space. Legend of Grimrock 2 maps are defined as a 32x32 tile grid and in the case of Lucas's work, each tile is represented by a gene, which can be either floor tiles or wall tiles. In order to transform the chromosomes to represent sections of the map, he took the approach of concatenating all map lines into a single string, which is illustrated by figure 4.3.

Due to this approach in the translation from 2D layout to a 1D structure, Lucas could not use the traditional crossover methods as they would produce flawed solutions, sections of the level layout were now separated, to put it into context, the tile 1x1 (line and column identifiers) and the tile 2x1 are vertically

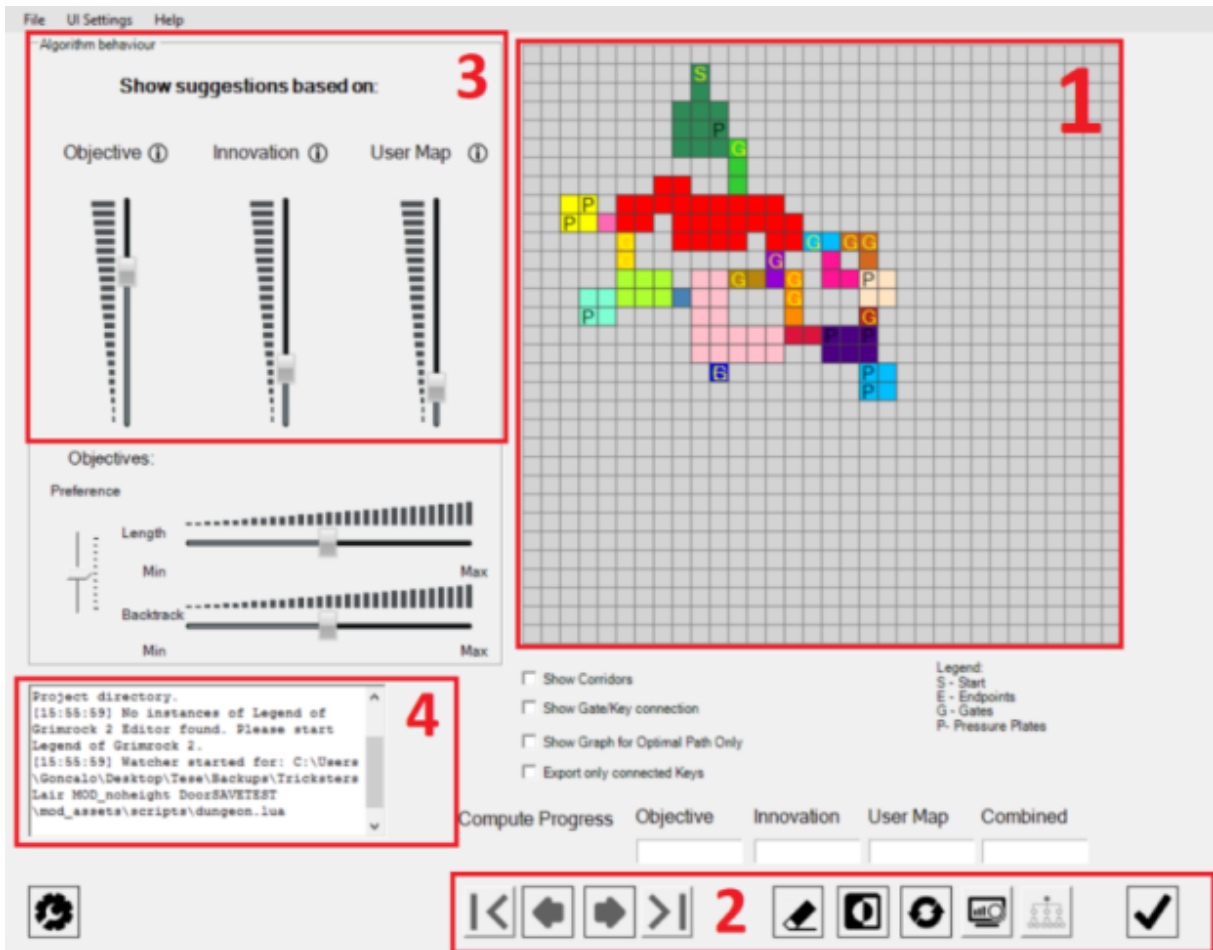
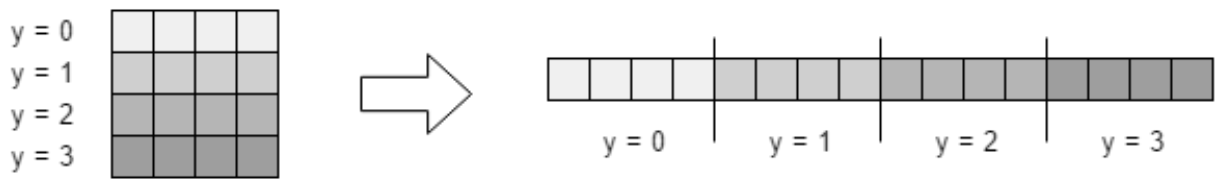


Figure 4.2: Editor Buddy's Puzzle Mode interface.

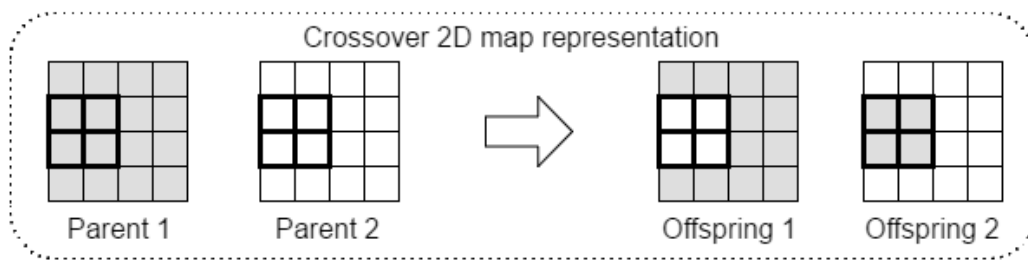
right next to one another, however since Lucas's approach to concatenate each line map, this two tiles are now 31 genes apart, which when applying conventional crossover operators, generates inadequate results. So in order to fix it, he modified the crossover operator to perform the genetic combinations in a 2D level design context. This was achieved by simply translating a given section of the 2D grid into 1D patterns, as can be seen in figure 4.4. This patterns are constituted by 2D shapes and then used in the genetic recombination process as the crossover points between two chromosomes, as shown in figure 4.5.

### 4.3.2 Gonçalo Delgado's Representation

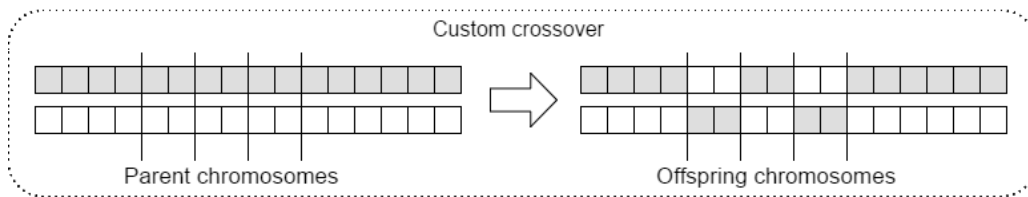
Delgado's approach is quite different from Lucas's, as his work is focused solely on puzzle generation within a fixed level layout, he decided to divide his map in two types of sections - rooms and corridors. For the chosen level layout, there are twenty three sections with a maximum of two puzzle items per section. The genes have two meanings, if the section is a room then the gene will determine how



**Figure 4.3:** 2D map to chromosome transformation.



**Figure 4.4:** Example of custom crossover applied to a pair of chromosomes.



**Figure 4.5:** Example of custom crossover applied to a pair of chromosomes.

many pressure plates exist in the room and if the section is a corridor it sets the number of gates within it. Figure 4.6 represents a level layout divided by sections and its corresponding chromosome transformation. However, as the chromosome does not contain the position of the puzzle element in the section, Delgado stores it separately, to ensure the gates and pressure plates are not constantly changing position in every generated suggestion.

Upon experimenting with different ideas on how to represent a chromosome, he would often see cases of an overload of puzzle items in big sections. With this approach he was able to have a better control of the number of maximum puzzle items per section.

The initial population is initialized pseudo-randomly, however it raises the problem of balancing the number of pressure plates and gates, as they have to be the same. To fix this problem, he developed a system, that checks all the genes and if necessary, flips the bits so that the number of pressure plates and gates match.

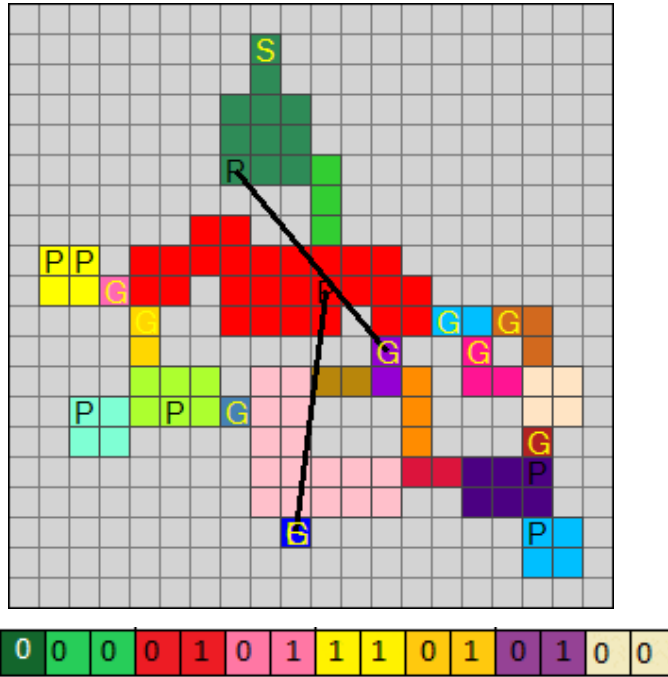


Figure 4.6: Editor Buddy Puzzle Mode example map and its corresponding chromosome representation (cropped).

#### 4.4 Gonçalo Delgado’s Breadth-First Graph Search

Due to the level content being generated randomly, there is a high probability that the level constructed by the algorithm is impossible to be successfully completed. To address this problem, Delgado resorted to a Breadth-First Graph Search.

A Breadth-First Graph Search is an algorithm that starts on a selected point and searches all possible paths to the final point, returning the shortest one. In the context of Delgado’s work the algorithm begins its execution at the start point of the player and terminates at exit of the level. If the search returns a viable path it means that it is possible for the level to be successfully completed.

However, Delgado had to adapt this algorithm to work with pressure plates and gates, otherwise it would get stuck going back and forth between sections. To prevent it Delgado added the condition that a section is only viable if the state of the world changes. This condition is ensured by a control variable that keeps track of all the pressure plates and gates reached.

This Breadth-First Graph Search algorithm is also used to classify the fitness of the generated level in terms of its Length and Backtrack. By analyzing the shortest path provided by the Breadth-First Graph Search, Delgado can determine the backtrack and length required in order to complete the level, and give a fitness level considering how the sliders are set in the user’s interface.

## 4.5 Execution Flow

Both the applications developed by Lucas and Delgado feature a similar execution flow, with the three independent algorithms for the level generation - Objective, Innovation and User Map. However Delgado's work includes a fourth algorithm called Combined algorithm.

Lucas Editor Buddy execution flow, illustrated by figure 4.7, consists of two phases: The Initialization phase, which is responsible for the generation of the initial populations and the post-initialization phase whose task is to generate the suggestions. It is worth mentioning that the Objective algorithm has an additional role, it is also responsible for the selection of which suggestion to display. As can be seen in figure 4.7, the Objective algorithm receives chromosomes of the Innovation and User Map algorithm's population. According to the values of the sliders in the UI, a percentage of the algorithm's most fitted chromosomes are placed in the Objective algorithm's initial population, thus making it the algorithm responsible for selecting which individual to display.

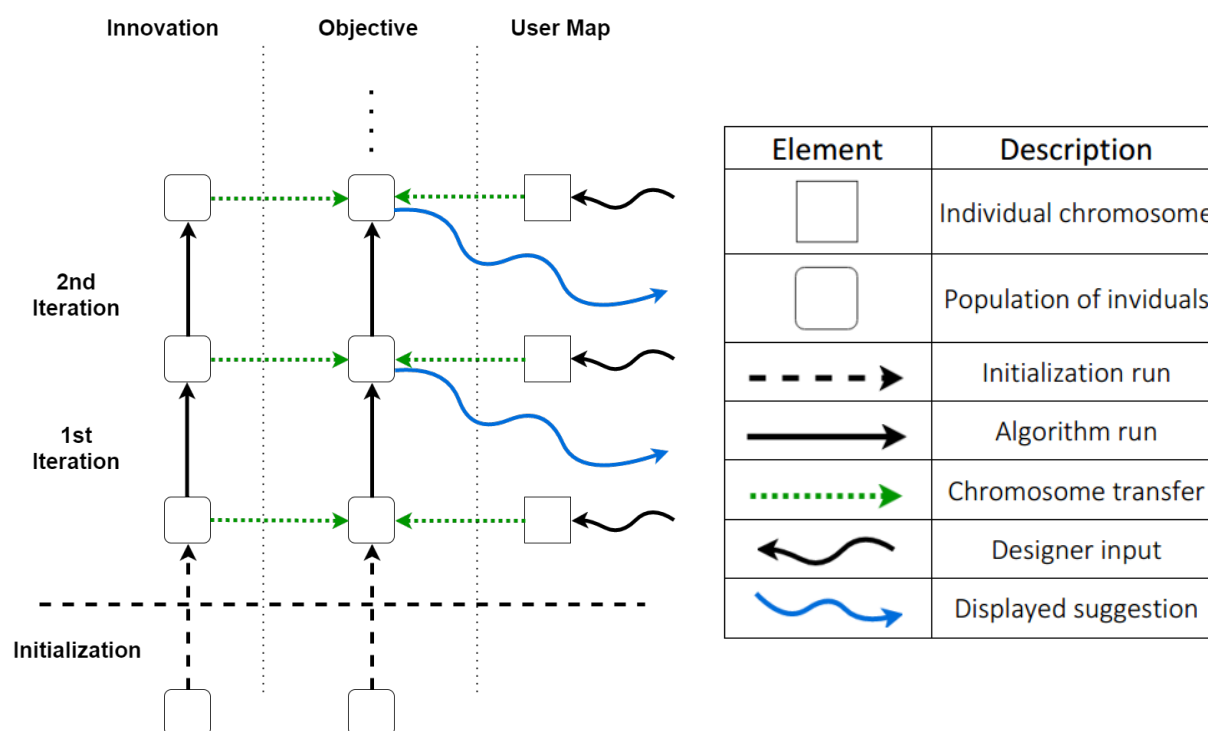


Figure 4.7: Editor Buddy's execution flow.

Delgado's Editor Buddy Puzzle Mode adds complexity to Lucas's version in terms of algorithm quantity. His approach, illustrated by figure 4.8, consists of three core algorithms and one additional algorithm responsible for the combination of the core algorithms outputs. Delgado's Chromosome initial population starts with every gene at zero with the possibility of having two extra genes with the value set to one. As opposed to Lucas's approach, Delgado's Combined algorithm was created with the sole purpose of

selecting a suggestion, although both approaches function in a similar fashion. According to the values of the sliders, a portion of each of the other algorithms' population is placed in the initial population of the Combined algorithm.

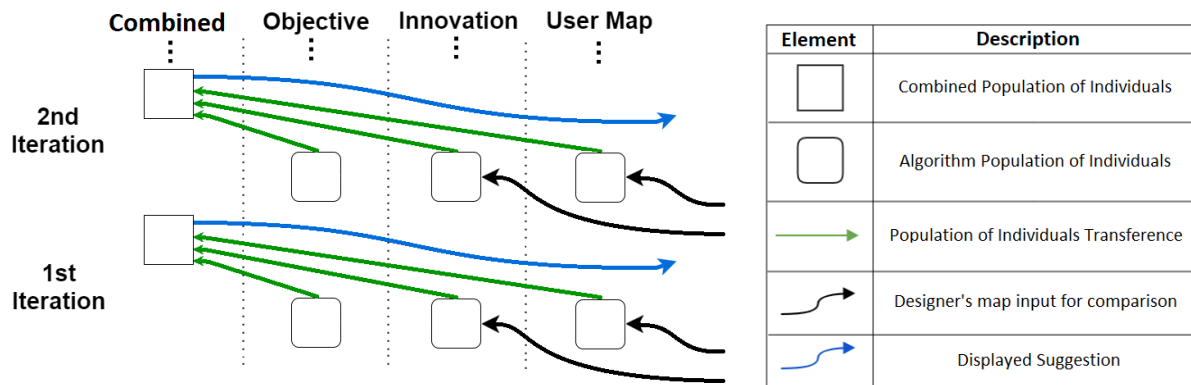


Figure 4.8: Editor Buddy's Puzzle Mode execution flow.

## 4.6 Algorithms

In this section, we will explain the three algorithms present in both works and analyze their similarities and differences, additionally we will also study the Combined algorithm from Delgado's Editor Buddy Puzzle Mode.

### Innovation Algorithms

The Innovation algorithm evolves towards generating innovative content, that seeks to provide content as distinct as possible from the user's. This algorithm targets more novelty rather than utility or value.

- **Population and chromosomes:** Lucas's initialization phase generates a 30 individual population with an average of 500 walkable tiles which is half of the total amount of genes for each chromosome. Delgado's population consists of 50 pseudo-randomly generated individuals with 46 gene chromosomes. This 46 gene chromosome represents the 23 rooms or corridors in the level Delgado chose. Each room or corridor is translated into 2 genes.
- **Objectives:** This algorithm seeks the same core element in both works, as an important part of a creative process is the search of multiple solutions (Innovation). This is a suitable example of the previously explained concept of Lateral thinking, as the algorithm is constantly searching for other solutions to the same problem.

In Lucas's work it seeks to provide a level layout as distinct as possible from the designer's layout. His algorithm only includes one constraint of having at least one walkable path from the start position to the end position.

Since Delgado's work focuses on generating puzzles for the current level design, his innovation algorithm aims for the number of map elements in a section to be more distinct than the one created by the designer. It is worth mentioning that the algorithm does not take into account the user's map gate-pressure plate connections.

- **Operators:** Lucas's Crossover parameters include 80% crossover probability using his modified **3x3** 2D square shape custom crossover. For the parent selection Lucas chose to use Fitness Proportionate Selection, which is also known as roulette wheel selection. This selection directly associates an individual's fitness to the probability of its selection. In terms of the replacement of Individuals he used generational replacement [16], where the best offspring replaces its least fit parent. Lucas's Mutation has a 15% chance of occurring and uses a custom mutation operator using swap mutation, that simply swaps relevant gene information between two genes. As for Elitism parameters, it has a 5% elitism percentage and prioritizes the fittest individuals, which means that only the top 5% of the fittest individuals of the population are carried over without suffering any modifications.

Delgado's crossover operator includes a 80% crossover probability using double point crossover available in the GAF. As for replacement strategy he chose generational replacement and Fitness Proportionate Selection for the parent selection. For the Mutation operator, Delgado used the GAF default operator that goes through each gene and with a 10% chance, inverts the current bit value. As for Elitism parameters he used the same as Lucas with a 5% elitism percentage that prioritizes the fittest individuals.

- **Termination:** Lucas's evolutionary run terminates after 80 generations with the final population being kept in memory for the next iterations. In contrast, Delgado's evolutionary run finishes after 50 generations. This discrepancy is probably due to the difference between the search spaces of each work. Delgado's search space is more heavily affected by constraints, as there are more variables that need to be considered in order to have an adequate solution.

## Objective Algorithms

The Objective algorithm evolves towards the generation of suggestions that contain evident design patterns, coherent with the objectives implemented and included in the objectives combo-box.

- **Population and chromosomes:** Lucas's initialization phase is the same as in the Innovation algorithm, it generates a 30 individual population with an average of 500 walkable tiles. Delgado's

initialization phase once again repeats the same pattern as in the Innovation algorithm with a 50 individual population that is pseudo-randomly generated, the first individual starts with all genes at zero and each subsequent individual has two more genes, potentially with the value one than the previous one, one for room sections, to generate pressure plates, and the other to corridors, for the generation of doors.

- **Objectives:** For this algorithm, Lucas and Delgado objectives are slightly different.

Lucas's values the creation of at least one valid path from the start point to the end point of the level, as well as following his implement sub-goals: the emergence of narrow halls and the emergence of ample space or rooms. If the fitness function gives preference to individuals whose genetic material consists in only having one or two walkable tiles, than the algorithm is evolving towards the first sub-goal. Whereas if it gives preference to individuals whose genetic material has more than two walkable tiles than it is evolving towards the second sub-goal.

Delgado's implementation also values if it is possible to reach the end point of the level. In order to check this, he uses his version of the Breadth-First Graph Search and it evaluates the level in three domains: Feasibility, which determines if the level can be successfully completed, Length, which is characterized by the number of map sections that a player must go through in order to finish the map and Backtrack, which is defined by the maximum number a certain section was visited on the most efficient path to reach the level end point. The algorithm attributes a fitness score considering all three domains.

- **Operators:** Lucas's operators for the Objective algorithm are the same as the Innovation algorithm with only one exception. The Mutation operator now has 10% chance of occurring.

Delgado's operators for the Objective algorithm are exactly the same as the Innovation algorithm.

- **Termination:** Lucas's evolutionary run of the objective algorithm once again terminates after 80 generation and Delgado's algorithm stops after 50 generations. He explains that this increase in generations is due to his Objective algorithm not being as heavy as his Innovation algorithm.

## User Map Algorithms

The User Map algorithm is independent from the previously mentioned algorithms and it evolves towards generating similar content to the designer's map. A note worth mentioning is that in Delgado's implementation, the algorithm does not take into account the designer's gate-pressure plate connections, only the number of puzzle elements per section of the map, just as his Innovation algorithm.

- **Population and chromosomes:** Lucas and Delgado have different approaches to this algorithm.



Lucas algorithm takes in the designer's level layout and calculates its respective chromosomes, he then proceeds in transferring them into the Objective population for the next iteration.

Delgado on the other hand, follows the same steps as he did in the Objective and Innovation algorithm, with 50 individuals and 46 chromosomes using his pseudo-random population generation.

- **Objectives:** The objective of this algorithm is quite clear, to generate content similar to the current designer's level. Delgado's approach is similar to his Innovation algorithm where he converts the level created by the designer into a chromosome. Then compares each pair of bits and converts them into a number. However, his algorithm does not guarantee that the map is feasible, it only evaluates if the number of puzzle elements is similar.
- **Operators:** As for operators, Delgado stuck with the same approach as his Innovation Algorithm.
- **Termination:** Just as his Innovation algorithm, Delgado's User Map algorithm terminates after 50 generations.

## Combined Algorithm

This algorithm is exclusive to Delgado's Editor Buddy Puzzle Mode, where the algorithm takes into account all three previous algorithms. Each generated chromosome is evaluated with two algorithms, Objective and Innovation, since the User Map is the inverse of the Innovation. Lucas addresses this problem by ensuring that the displayed suggestion to the designer is always the one which better adheres to the current global task objectives. This is done by selecting the suggestion that is more inline with the designer's switch/sliders present in the Editor Buddy's interface.

- **Population and chromosomes:** Delgado's initial population consists of the top  $\chi$  % of each of the previous algorithm populations, this percentage is calculated according to the sliders in the Editor Buddy's Puzzle Mode UI.
- **Objectives:** Delgado's objective for making this algorithm is to gather the best individuals of each population and re-evaluate them with his previously mentioned fitness function, where at the end the best individual will be the displayed as a suggestion.
- **Operators:** As for the crossover operator, Delgado used the same as he did in his previous algorithms.
- **Termination:** Since his algorithm uses the objective fitness function, that is computationally more intensive, he terminates the algorithm after only 25 generations.

## 4.7 Discussion

In this section we took a deep look into the previous works of Pedro Lucas and Gonçalo Delgado, which we will use as inspiration for our work. As previously mentioned, we will use Pedro Lucas's creation and develop our own algorithms to produce lighting suggestions. We will try to improve the user interface in regards to the results they both obtained in their user testing.

After analyzing how both works were created and how they function, we have gained a better grasp on how to proceed with the development of our tool. However it is also worth discussing the feedback each work received on their evaluation phase.

Lucas mentions that a common misconception amongst participants of his user case studies, expected a more contextualized feedback from the Editor Buddy, that it should be able to read patterns in the user's design. He also refers that personality traits and life experience are big factors that affect the creative process, which is one of the causes that, to some users, the potential use of such tools is greatly diminished.

With the feedback Delgado received from his evaluation phase, he points out certain aspects of his tool that could be improved. The first is to make his algorithms work for any type of level layout, as his only works for the map Trickster's Lair from the Legend of Grimrock 2 game. One change he considers to be one of the hardest to tackle is, for his algorithms to take into account the user's pressure plate and gates connections. As his algorithms, when reading the user's level design, discard these connections. He also refers that users have shown interest in the addition of other puzzle elements, rather than only having pressure plates and gates. Lastly in terms of the tool's UI, participants mentioned some improvements such as a dedicated box for puzzle metrics and different colours for the sections the algorithm has passed through.

# 5

## Solution Model

### Contents

---

5.1 Computer colleague paradigm . . . . .	39
5.2 Solution Model . . . . .	40

---



In this chapter, we explain our solution model and how we modeled the computer as colleague paradigm. We also detail the thought process behind our user interface as well as our algorithms and application behavior. Afterwards, we address how our tool integrates with the Legend of Grimrock 2 Dungeon Editor and how a user is expected to use our solution. Lastly, we have a brief discussion on the topics mentioned.

## 5.1 Computer colleague paradigm

Our solution resides in tackling the paradigm of computer as colleague, so it is crucial that the user's perspective of our tool is that it is working alongside him/her in the process of level design creation. We have decided to continue with the logic behind Pedro Lucas's three algorithms: Innovation, Objective and User Map, as we consider they represent three of the most important creativity thinking process when creating a new level.

- **Innovation:** when a level designer's focus is to diversify the constructed level and is hoping to come up with new fresh ideas, with more value or novelty than what is created.
- **Objective:** is one of the most important thought processes behind level design, as the level must be possible to be completed and if it is a simple guided experience or a more challenging adventure.
- **User Map:** as every level designer has to keep his/hers imagination in check, so that the already built level does not go to waste in the pursuit of new ideas.

Obviously, there are other thinking processes that level designers go through, however these three were already tested by both Pedro Lucas and Gonçalo Delgado and delivered positive results.

As the three algorithms will be running simultaneously and generating suggestions accordingly to their purpose, it can be considered that they are working against each other, which may sometimes produce results that do not correspond to the user's expectations. These suggestions may be considered as unwanted or as inadequate results, however we must keep in mind that in the process of stimulating creativity, these non-ideal results are a natural occurrence. Just like the human mind in a creative state, also produces results that the subject does not find acceptable and is quick to dismiss.

In order to diminish these unwanted results, we provide all the means to properly guide our tool in the generation of content. As it is described further in this document, the user can control a set of variables to maneuver our algorithms into generating suggestions more aligned with his/hers expectations. This does raise the question if it can be considered creativity if we are controlling the program and not letting it run freely, however we see this behavior has a communication method between two partners. Similarly to two human colleagues, one can ask the other for their opinions or different approaches to a problem.

We also believe that by giving the user this control over the tools parameters, it mitigates frustrating experiences, as the generated suggestions are more valuable to the user's level. In the end, it all comes down to the designer's judgement, whether the generated suggestion has value and he/she wants to export it to their design or if it is worthless and generating another suggestion might be more of their interest.

## 5.2 Solution Model

Throughout the next section we give an overview of the implemented solution, going into some detail on the user interface, the tool's behavior and how it relates with the Legend of Grimrock 2 Dungeon Editor, the selected level editor.

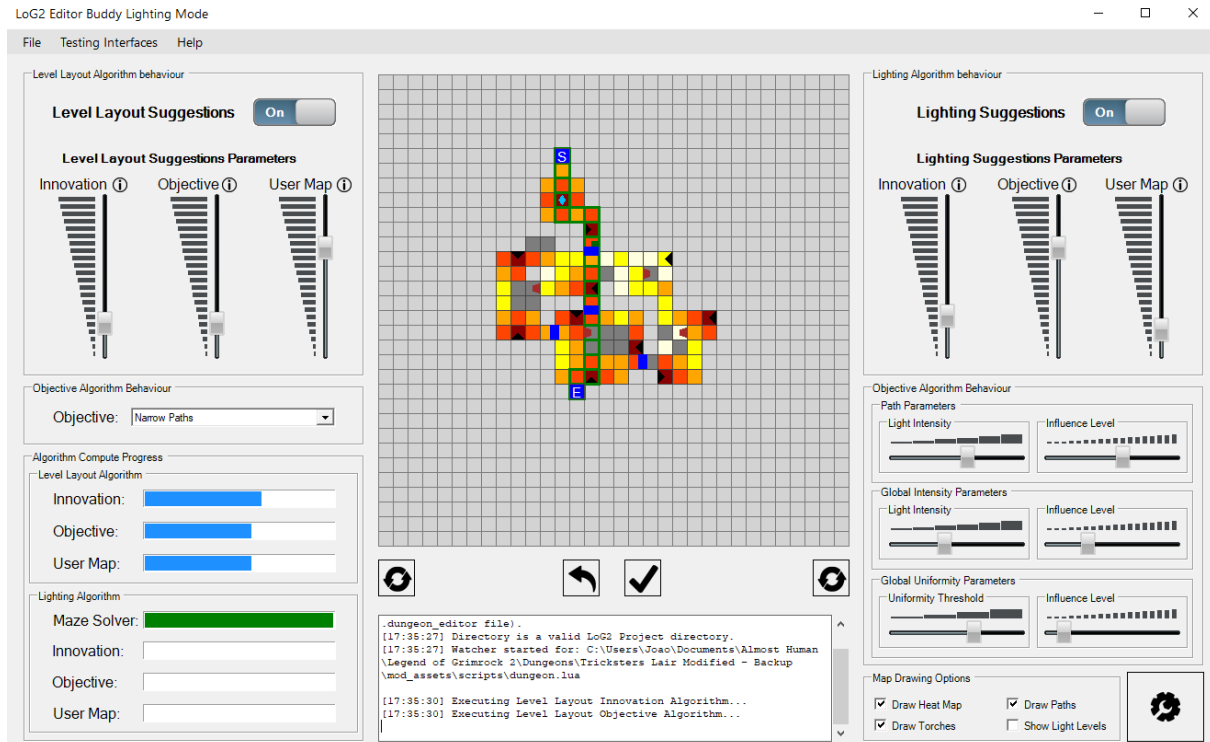
As our solution's foundation is the Editor Buddy developed by Pedro Lucas, we decided to name our solution Editor Buddy Lighting Mode (EBLM) as its main focus is the illumination of levels. It consists in a Graphical User Interface (GUI) based application that works along side the LoG2 Dungeon Editor to stimulate creativity. The tool, in conjunction with the level designer, participate in a co-creative process when creating a level. The tool's creativity is represented by the generation of suggestions, which essentially are possible modifications to the user's current level design. The Figure 5.1 represents the EB Lighting Mode final version. The user interface contains several sliders as well as a map canvas, which is used to show the generated suggestions. The mentioned sliders are responsible for defining the EB Lighting Mode behavior. Its behavior is constituted by a total of four algorithms, two algorithms created by Lucas that beside user created map, generate level layout suggestions and two algorithms developed by us, that also take the user created map as an input and generate lighting suggestions.

### 5.2.1 Interface

The Editor Buddy Lighting Mode interface consists of these main components:

- Controls in the form of sliders that can be used to adjust the application's behavior.
- A 2D canvas where the program generated suggestions are displayed.
- The canvas can also be used to draw out paths in the map. These paths can be used to influence the application's generated suggestions.
- Two buttons to interact with the suggestion. Revert and Export.
- Two buttons to generate a new suggestion. One for level layout and the other for illumination.
- Two buttons to deactivate each one of the algorithms.

- A set of buttons to show or hide certain elements of the displayed suggestion.
- A set of progress bars that display each algorithm's computational progress.



**Figure 5.1:** Editor Buddy Lighting Mode interface.

As can be observed from the figure above, the Editor Buddy Lighting Mode interface is divided into three distinct sectors:

**The left sector** is dedicated to level layout content generation, it includes the slider controls to adjust the parameters of each of the algorithms present in Lucas's work - Innovation, Objective and User Map. As well as a combo-box element, which can be used to set the goal of the Objective algorithm. The left zone also includes an area dedicated to show the computational progress of all of the algorithms.

**The center sector** has two responsibilities, displaying information and the interaction with the generated suggestions. The large 32x32 grid is where the suggestions are mapped out, it includes visual indicators level elements such as: enemies, light intensity, treasures, puzzles and others. The canvas can also be used to select paths. The level designer can accomplish this by selecting, with two left mouse clicks, in two of tiles of the canvas, then our search algorithm will determine and display the shortest path between those two tiles. If the level designer wants to create a more specific path, other than the shortest path between two tiles, he/she can simply create multiple linked sub-paths that form a longer course. If the level designer wants to delete any of the created paths, a simple right mouse click on either point will eliminate it.

This zone includes a total of four buttons, Two buttons for the generation of new suggestions (one for level layout and another for level illumination). One button to revert the current level back to its original state and one button to export or apply the generated suggestion to the level. To complete this section, there is a log box, that keeps the user informed of the application's current state.

**The right sector** is committed to lighting content generation, it includes a total of nine sliders and a group box with some options to change the visualization of suggestions. The three sliders at top are used to adjust the parameters of each of the lighting algorithms - Innovation, Objectives and User Map. The rest of the sliders are in groups of two and are directly linked with the Objective algorithm. Each group contains an intensity slider and an influence slider, where the intensity determines how strong should the light be for a certain objective and the influence regulates the weight of the objective in the suggestion. The three objectives consist of:

- **Path Objective:** is responsible for generating the illumination levels in the selected paths.
- **Global Intensity Objective:** determines the light intensity for the entire level.
- **Uniformity Objective:** its goal is to ensure that the light intensity is constant throughout the level or that it does not fall beneath a certain value.

## 5.2.2 Behavior

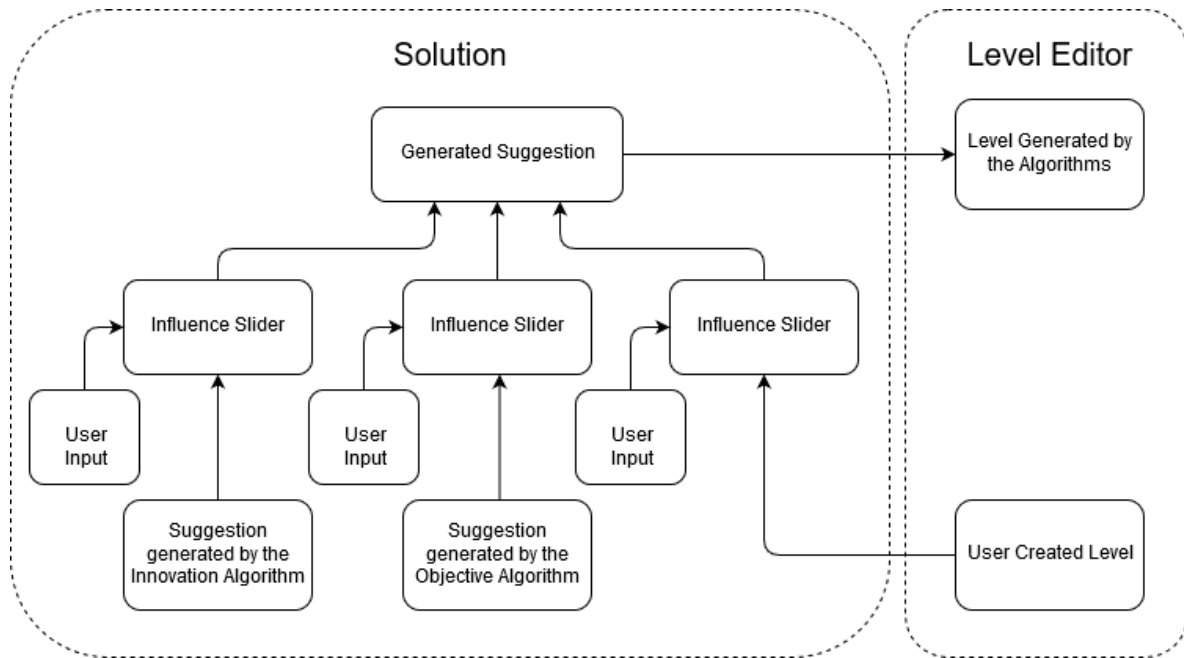
The Editor Buddy Lighting Mode concept is to act as a colleague in co-creative level design process. To accomplish this, we followed the same concept as Pedro Lucas, the creation of two Genetic Algorithms. These two algorithms serve different purposes, one is for calculating the Innovation component of the suggestions and the other is responsible for generating suggestions according to the required Objectives. The User Map algorithm functions differently from these two. A certain percentage of individuals, measured by the User Map slider, is inserted into the initial population of each of the algorithms.

The algorithms uniqueness, is ensured by the fundamental concepts of Genetic Algorithms, where individuals are selected to breed amongst them in order to spawn a stronger offspring. However the algorithms need to ensure that the population is evolving towards a solution fitting of the user's expectations. This assurance is kept by the fitness functions, which associate a score to each individual, the higher score the better the user's expectations will be matched.

The suggestions generated by the algorithms are a direct result to designer's interaction with the controls in the EB Lighting Mode and the current level he/she has created in the LoG2 Dungeon Editor. Every time the designer makes an alteration in their level in the LoG2 Dungeon Editor, the EB Lighting Mode will automatically start generating new suggestions. However, the process does not reset, both the new level and the previous population from the generated suggestion, are used in the next algorithmic iteration. With this approach the algorithms' initial population will consist of an already evolved



population, meaning that instead of restarting the entire growth of the population, the algorithms continue developing the previous one into an even better population, thus making the suggestions better with every algorithm cycle. This operation also results in a cooperative creative activity with the level designer, where the generated suggestions might influence the user's decisions and where the designer's decisions influence the applications outcome. Figure 5.2 illustrates the behavior of the Editor Buddy Lighting Mode.



**Figure 5.2:** Diagram of the interaction of the solution with a level editor.

## Summary

In this chapter, our objective was to provide a simple overview of our proposed solution model, including its distinct elements and their functions to the overall design. Although, we have developed our solution model to a particular level editor, we believe that our approach of modeling co-creativity in level design can be adapted to be used with other similar level editors.

# 6

## Solution Implementation

### Contents

---

6.1 Solution Implementation . . . . .	45
---------------------------------------	----

---

In the following chapter, we present the implementation of our solution, where we explain our approach, its constraints and how we used the genetic algorithm framework built by John Newcombe. We also go into detail on how each algorithm works and explain some more technical aspects of them. Next, we describe our solution's architecture and explain why some constraints had to be made in order to avoid computational problems. Lastly, we describe our algorithm to build a heatmap of the level and its importance to the solution.

## 6.1 Solution Implementation

This section addresses the more technical aspects of the proposed solution, design decisions, constraints and respective workarounds regarding the Editor Buddy Lighting Mode behavior, interface and its development cycle. It is also worth clarifying that in this section, we will be solely focused on the lighting concept of our solution, as the level layout generation concept was already developed and tested in Pedro Lucas's Editor Buddy.

### 6.1.1 Lighting Concept

After creating the parser to read the current level and displaying it in our interface, we faced our first obstacle as we needed to develop a way for the level designer to see the lighting in our interface's 2D canvas. Our solution was to develop an algorithm that calculated a lighting heatmap of the level.

The first step was to analyze the Legend of Grimrock 2 game lighting behavior. This was accomplished by performing a series of tests with the Dungeon Editor and the placement of several lighting objects. These are the most relevant findings:

- There are multiple objects for illuminating the level, however they all operate by the same rules.
- Torches, the most common form of light objects in dungeons, illuminate a radius of 5 tiles, but can be seen from 12 tiles away.
- There is no reflection when light bounces off walls, meaning that light does not travel through corners.
- Having multiple light sources does not increase the light intensity of a tile. Meaning that having numerous torches close to each other illuminating a certain tile, does not increase how well illuminated the tile is.

After gathering this information, there were some challenges we had to figure out before developing our heatmap algorithm. The first design decision we made was we would only consider torches as light sources, as it is the most common form of illumination in dungeons. With this decision taken into

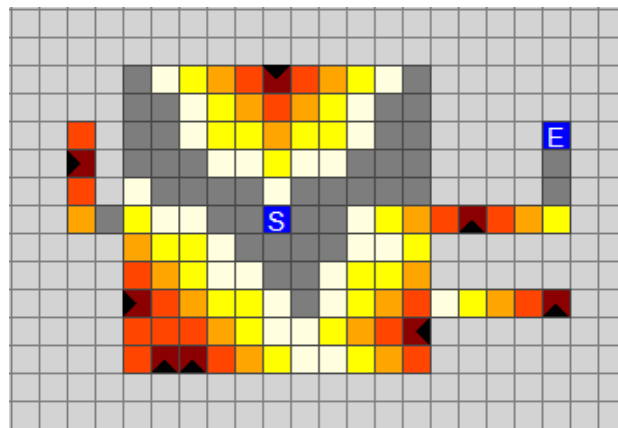
account, our light levels for a certain tile could only be within 0 and 5, the first being that the tile is completely dark and the latter that the tile has at least one torch, meaning it is fully illuminated. This scale is based on the distance a tile is from its closest light source, for example if the tile is three tiles away from the closest torch and its path is unobstructed, the light level of the tile is 3.

When developing our algorithm three main constraints had to be taken into account:

- The algorithm must know where the closest torch for each given tile is and provide the correct light level considering the distance between them.
- It must always check if there is any wall obstruction between a torch and the tile in question.
- The algorithm must always verify all torches within reach (5 tiles) of a tile before committing to a light level.

After building the algorithm with these constraints in mind, we had to optimize it as much as possible since, as it will be explained in a further in the document, this heatmap will also be used by the fitness functions of our genetic algorithms.

Figure 6.1 represents how the heatmap algorithm's outcome is displayed in our interface. With the black triangles representing the torches and the colors red to white each tile's light level, red being the highest and white the lowest. We performed a brief test with a small group of users and its results demonstrated that this color scheme was the best approach.



**Figure 6.1:** Example of the Editor Buddy Lighting Mode 2D canvas when displaying the generated heatmap.

### 6.1.2 Editor integration

The Editor Buddy Lighting Mode is not a standalone program, it is meant to work in parallel with level editors. The chosen level editor is the LoG2 Dungeon Editor that was introduced in chapter 3.

Each new project firstly needs to be created in the level editor. Once this process is finished, the designer can select the newly created project directory in the EB Lighting Mode, in order to start receiving suggestions while developing his/hers content in the level editor.

There are only four instances where level designers should directly interact with the EB Lighting Mode:

- when adjusting the sliders or selecting paths in order to receive suggestions more adequate to their needs.
- when changing suggestion display options, for example enabling or disabling the visualization of the light heatmap.
- when exporting a suggestion, where the EB Lighting Mode will apply the generated suggestion to the level designer's design.
- when asking the EB Lighting Mode to generate a new suggestion.

Meaning that the EB Lighting Mode serves its purpose of only provide suggestions, leaving all the level creation and editing to level editor.

### **6.1.3 Genetic Algorithm Implementation**

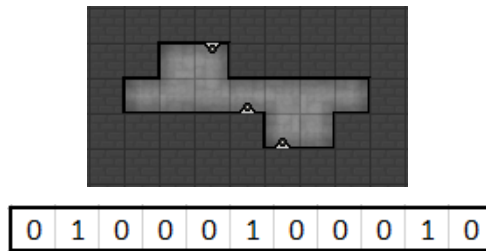
As previously mentioned, genetic algorithms use the theory of natural evolution, where each possible solution for a problem is considered an individual and each individual reproduces with another to generate a better offspring, thus evolving the population to stronger group of solutions. In this section, we will provide an overview of how we took this concept and applied to the development of lighting suggestions in a level design environment.

As Lucas and Delgado have already used and demonstrated that John Newcombe's genetic algorithm framework is reliable and compatible with the type of concept we are developing, we also decided to use it and we will explain how we adjusted it to work with our solution.

#### **Lighting to chromosome representation**

When transforming our level to a chromosome, we decided to go with the most logical approach possible. Our representation consists of transforming, in a sequential manner, all walkable tiles in the level into bits, where each bit has the value 0 if the tile does not have a torch and 1 if it does, as illustrated by figure 6.2.

This method of chromosome representation proved to be the most efficient and reliable in our internal testing. In addition, doing the opposite transformation of receiving a chromosome and converting it to



**Figure 6.2:** Example of level and its chromosome representation, where the light grey tiles represent the walkable tiles of the level and the triangle shaped objects the torches.

a level, is also straight forward. This transformation is performed by copying the previous level without the lighting objects, then checking the chromosome for the bits with the value 1 and attributing a light source to the corresponding tiles. However, there is one unfavorable limitation of this process, due to the shift in the placement of the lighting objects, the entire level's heatmap has to be recalculated and since the heatmap algorithm is computationally heavy, it results in a loss of efficiency. Although this is a vital step, since our fitness functions do not attribute a fitness score to each individual by comparing chromosomes. They use the level's heatmap to compare individuals, the better the individual's heatmap fulfils the designer's goals the higher its fitness score will be. Without the constant update to the level's heatmap, the fitness functions would not be able to properly attribute a fitness value to each individual.

On a more positive note, since our chromosome representation was simple and effective, there is no need for us to build custom genetic operators, such as custom crossover operator or a custom mutation operator. We used the already built in operators of the John Newcombe's genetic algorithm framework.

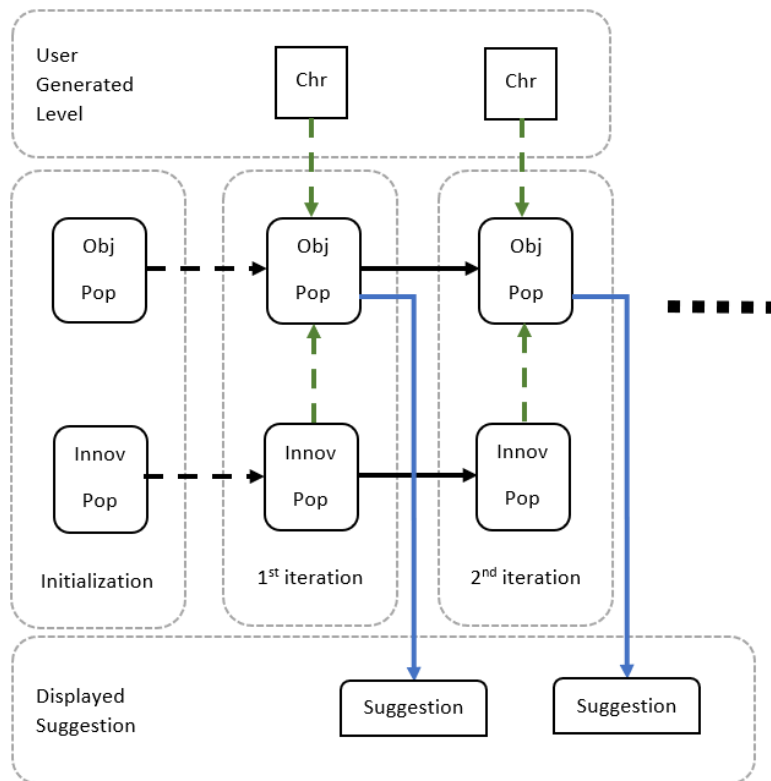
#### 6.1.4 Editor Buddy Lighting Mode execution flow









Over the next subsection, we detail the Editor Buddy Lighting Mode's life cycle, which is illustrated by figure 6.3. We address when and how it starts, as well as providing relevant information regarding algorithm interaction and how it chooses which suggestion to display.

The Editor Buddy Lighting Mode algorithm execution is comprised of two separate stages: An Initialization phase, where the algorithm first generates the initial populations and a post-initialization phase responsible for generating the suggestion, which takes the previous algorithm generated populations and evolves them to stronger populations until a suggestion is formulated. Each time the level designer makes modifications, creative or destructive, to the level, in terms of both level layout and placement of light objects, it triggers a new evolutionary run in the Editor Buddy Lighting Mode.

**Initialization Phase:** It consists in the generation of the initial population for both the Innovation and Objective algorithms. This phase only occurs when one of two scenarios happen:

- When the Editor Buddy Lighting Mode is first launched.



Element								
Description	Chromosome	Objective Algorithm Population	Innovation Algorithm Population	Generated Suggestion	Initialization Run	Algorithm Run	Chromosome Transfer	Suggestion Display

**Figure 6.3:** Editor Buddy Lighting Mode algorithm execution flow diagram and its element captions.

- When the level designer changes any of the sliders in the interface.

**User Input:** Identified as User Generated Level in the execution flow diagram, it is independent from the Innovation and Objective algorithms, due to its development being solely performed by the level designer. The EB Lighting Mode job is to read it and use it in the generation of suggestions. The only two times the EB Lighting Mode interacts with the user generated map, is when the level designers wish to export the generated suggestion to their design and when revert the button is pressed in the interface.

Each time level designers make a modification to their level, the EB Lighting Mode automatically detects it and proceeds to use current level as an input in the generation of a new suggestion, thus provoking a new evolutionary run of the algorithms.

**Configuration and Population / Chromosome transfer:** Each action from the level designer trig-

gers a new algorithm run, in which the first step the EB Lighting Mode performs is a verification of the current state of the interface's sliders. Depending on how they are configured, the pool of potential suggestions will be composed of different types and/or amounts of individuals. The Objective algorithm is the algorithm responsible for the selection of which suggestion to display. This is due to the fact that, according to the value of the sliders, a portion of both the level designer's level and the Innovation algorithm's population are inserted into the initial population of the Objective population. For example, if the innovation slider is set to 25%, then a quarter of the top population generated by innovation population is selected to be a part of the Objective algorithm's initial population for the next iteration.

**Algorithm execution:** After the Objective algorithm receives a percentage of individuals from the Innovation algorithm's population and the designer's level, a new algorithm run starts, where each algorithm runs independently. The Innovation algorithm generates a solution as distinct as possible from the current level. Whereas, the Objective algorithm generates a suggestion that best fits the parameters set by the user in the interface.

**Displayed suggestion:** When both algorithms' execution is completed, the EB Lighting Mode must decide which individual is the best suggestion to be displayed to the level designer. This selection process always adheres to the current state of the sliders of the interface. For instance, if all the sliders are set at a third but the Global Intensity Objective slider is set at 100% with its light intensity at 0, then the suggestion chosen is the one who has the least amount of global illumination possible.

**Algorithm re-initialization:** The re-initialization of the Innovation and Objective algorithms has to occur every time the level designer switches the application's behavior, which means every time the sliders in the interface are altered. Consequentially, the process of initializing populations also has to take place. The next scenario is a perfect example of why this process must happen: imagine a designer has been using the EB Lighting Mode for while without changing the sliders, only asking the application to generate new suggestions. If now, he/she were to change both the Innovation and User Map sliders to 0 and if there was no re-initialization, the next suggestion might still display elements of both, due to the fact that there were still individuals in the Objective algorithm's population.

### 6.1.5 Algorithms

The following subsection provides a more detailed look of the algorithms' implementation and their corresponding configuration parameters, as well as any constraints we had in the development and their respective workarounds.



## Innovation Algorithm

The Innovation algorithm's purpose is to constantly evolve the individuals into content as distinct as possible from the user's level. Its objective is to spark the level designer's creativity, in order to pursue alternatives to their design. At a lighting perspective, innovation means that the torch placement is as different as possible from the user's created level. The algorithm accomplishes this by evaluating the heatmap from the designer's level and comparing it to the individual's heatmap, the more distinct it is from the user's level, the higher its fitness value is.

**Population and chromosomes:** The initialization phase of the algorithm generates a population of 20 individuals. Where each gene corresponds to a walkable tile of the current designer's level, meaning that if the level has 45 walkable tiles then the chromosome will consist of 45 genes. However in this phase, the algorithm starts with all genes set at 0 and randomly selects genes to set their value as 1. The number of genes selected is the same as the number of torches in the user's level. As we seek the most innovative level possible, the most logical initialization is to transform the user's level into a chromosome and invert all bits. However, upon testing this approach it quickly became apparent that the results were inadequate, as usually the level became flooded with torches and the results were beyond what we consider to be creative content.

**Objectives:** As mentioned the objective of the Innovation algorithm is to generate lighting suggestions as distinct as possible from the designer's level. However, this can not be measured by torch position, so we must calculate the difference between the designer's level heatmap and each of the individuals heatmap the innovation algorithm produces. This discrepancy is calculated, by iterating through each tile of the level and calculating the difference between its light level in the designer's level and its light level on the individual. The total sum of the differences will determine the fitness of the individual.

**Operators:** The crossover parameters include an 80% crossover probability using the default crossover operator from the GAF and with generational replacement as our individual replacement strategy, where the offspring are generated from the existing individuals to create a new population. As for parent selection we chose to use Fitness Proportionate Selection, also known as roulette wheel selection, which associates an individual's fitness to the probability of its selection. As for our Mutation operator and its parameters, we used the GAF's default operator with a 1% chance of flipping the value of a bit. This mutation percentage might appear to be low, however from our internal testing, this value presented the best results in terms of suggestions. This mutation rate will later be discussed in greater detail in the design decisions section. As for the Elitism parameters, they are made of a 5% elitism percentage that prioritize the fittest individuals, meaning that the top 5% individuals get carried over to the next generation without suffering any modifications. This values were chosen after some internal testing and suggested to be the most adequate.

**Termination:** The algorithm terminates its evolutionary run after 35 generations, which is lower than what we wished for, this is due to the necessity of our heatmap algorithm having to be called for every individual, resulting in a heavy performance hit. We had to compromise for a low value to maintain the concept of the EB Lighting Mode being a colleague in a co-creative process. If the suggestions takes too long to be generated, it sheds this concept.

## Objective Algorithm

The Objective algorithm evolves towards generating suggestions according to three sub-objectives, Path illumination, Global illumination and illumination Uniformity. The path illumination sub-objective generates suggestions with only the selected paths in mind. These paths can be set by the level designer in the interface, as well as the light intensity the algorithm will target in those selected paths. The global illumination sub-objective generates suggestions where the global light level is the same as the value set on the interface. Meaning that it calculates the average light level of the level and checks if it meets the required value. The sub-objective illumination Uniformity generates suggestions that can be either more or less uniform in terms of lighting, meaning that it checks if the illumination throughout the level is consistent or not. This is achieved by checking sectors of the level and verifying if the discrepancy is according to the threshold set on the interface. All sub-objective's influence on the Objective algorithm can also be set in the interface.

**Population and chromosomes:** The initialization phase of the algorithm generates a population of 20 individuals, where its creation method is the same as the Innovation algorithm. After the initialization stage, each iteration of the algorithm evolves the previous population towards a solution that follows the criteria and parameters of the three sub-objectives.

**Objectives:** As mentioned, the Objective algorithm has three sub-objectives and in order to choose the most appropriate individual of full filling the designer's goals, it has to determine the fitness value for each of the sub-goals. To do so we developed four fitness functions:

- **Path fitness function:** It evaluates all the tiles in the selected paths, and calculates the difference between each tile's light level to the light level set by the designer in the interface. The smaller the global difference, the higher the individual's fitness value is.
- **Global Illumination fitness function:** It calculates the average light level for the entire level and compares it to the light intensity level set in the interface. The smaller the global difference, the higher the individuals fitness value is.
- **Global Uniformity fitness function:** It divides the level in small sectors and calculates the average light level for each sector, comparing it to the sectors around it, the difference between each

sector must fall within the threshold set in the user interface. The fitness value is calculated by taking into account all the differences and the threshold.

- **Objective algorithm fitness function:** It is the main fitness function and is responsible for taking all the previously mentioned fitness values, (path, global illumination and global uniformity) and calculating an overall fitness value for the individual. This is accomplished by a simple mathematical operation, where each of the three fitness values are multiplied by their influence value (also set in the interface) and then summed.

**Operators:** All of the crossover parameters are the same as in the Innovation algorithm, however as a recap:

- GAF's default Crossover operator with 80% probability and generational replacement as the individual replacement strategy.
- Fitness Proportionate Selection as the parent selection method.
- GAF's default Mutation operator with 1% probability for same reasons as the Innovation Algorithm.
- An Elitism chance of 5%, that prioritizes the fittest individuals.

**Termination:** The algorithm terminates its evolutionary run after 60 generations, which is almost double of the Innovation algorithm, however not as high as we wanted. Once again, this is due to the heavy hit in computational performance the heatmap algorithm delivers every time it is executed.

### 6.1.6 Design decisions

In this subsection, we will enter in detail on some of the design decisions we made in the final implementation and why these decisions were necessary.

#### **Global algorithm objectives:**

When deciding how to give creativity to the EB Lighting Mode, we decided to go with the same model that Pedro Lucas developed, our objective was to keep it simple and since it was already tested by him, we decided it was a safe approach that would produce the results we intended. Pedro Lucas said in his work that although the Editor Buddy was only adapted for level layout, the concept that he used could be transfer to other areas. So we took his concept and applied it to the concept of illumination in co-creative environment in level design.

The Innovation algorithm objective was obvious, since at its core, its objective is to be opposite of the designer's created level. So we took this concept and applied it by comparing the generated level to the designer's level, which in the end proved to be a good approach.

The Objective algorithm on the other hand, was reasonably more challenging as we needed to define what could be considered as an objective in a lighting environment. Upon studying the concepts of level illumination, we decided to go with the three mentioned sub-goals. We believe they represent three of the cores in level illumination.

The illumination of paths, where level designers often use this tactic to guide players through levels, either being to the level's conclusion or to important moments in it. It worth mentioning that when we speak of illumination it can also mean the absence of it. Level designers often use reduced illumination in their tactics, for example in horror-games or to hide secret paths.

The global light intensity goal was perhaps the most obvious goal. When a level designer starts building a level, usually already has an idea of how dark or bright the level is going to be, all this before actually starting the development. Certainly, this is not true for all cases, however for experience level designers it is usually the case.

As for the global uniformity goal, it came more as a necessity. As in the generated suggestions, we would sometimes observe that the torch placement was centered around a particular area of the level, leaving the rest of the level completely dark. Although, this did not happen frequently, we thought it would be a good addition to the Objective algorithm, to have an option to tune the level's lighting uniformity.

### **Control over the generated content:**

There are advantages and disadvantages of the model we chose to generate suggestions, the one find the most relevant to discuss is the concept of the user interaction with the interface. In the context of a co-creative process, a user should not need to tune the colleague's opinion to get a suggestion. This is a disadvantage since it hinders the concept, however applying it to an application the overall experience becomes much more refined, as the user can guide the suggestions into more favorable content, which in the end is an advantage. Adding to this point, just like in a co-creative experience, one person can ask the colleague for an opinion on a certain manner, for example in our context: a level designer can ask his/hers colleague for an opinion on the illumination of a path, which we believe is similar to asking the application to generate a suggestion focused on the paths, by adjusting the interface's sliders.

### **Suggestion quality vs execution time:**

Possibly the biggest compromise we have decided to make in our solution. GAs generate better solutions with every iteration, however we had to make small sacrifices in the quality of the generated solutions, in order to keep the concept of the computer working as a colleague. If the application took too long to generate a suggestion, the level designer would loose interest, thus breaking the experience. Our ultimate goal was to keep the time it took to generate a suggestion below 3 secs, with ideally being

around the 1 to 2 seconds, with the highest amount possible of population generations. These values were based on the time it took for Lucas's EB to generate a suggestion which is approximately 2 secs, as well as the results from our internal testing.

The table 6.1 gives an example for a test map and how the generation limit affects performance:

Innovation Algorithm:			Objective Algorithm:		
	Generations:	Time:		Generations:	Time:
	35	1.18s		60	2.39s
	70	2.19s		90	3.29s
Difference:	+35 (+100%)	+0.9s (+85.59%)	Difference:	+30 (+50%)	+0.9s (+37.66%)

**Table 6.1:** Simple test with two Generational Limits and how they affect the algorithm's execution time.

However, we believe to have strike a good balance between performance and quality. The parameters and the generated solutions were thoroughly tested in order to achieve suggestions with quality in the shortest time as possible.

### **Mutation rate:**

In the Editor Buddy Lighting Mode algorithms, the mutation rate is set at 1% by default. Although in the concept of genetic algorithms the mutation rate is supposed to be a low value, in our testing we witnessed an interesting interaction when tempering with it. We could not set it to 0% as the mutation operator would loose its purpose, however when setting to a value higher then 1%, the chromosomes would be flooded with bits set at 1, thus creating a level filled with torches. Although, this is the normal behavior when increasing the mutation rate, what we found odd is the impact of setting the mutation rate to 2%, the results were levels with an irrational amount of torches.

### **Heatmap algorithm:**

The generation of suggestions in the Editor Buddy Lighting Mode relies on the developed heatmap algorithm, which is computationally heavy. The decision to create this algorithm was made early in the development (before the development of the genetic algorithms) as a way to display the illumination of a level to the designer, which is a feature that the LoG2 Dungeon Editor lacks. A level designer can experience the lighting of the level through the preview window in the Dungeon Editor, however it can be a tedious process when illuminating a level from scratch to be constantly walking around the level in the preview window.

The heatmap algorithm became even more relevant when we started exploring with different concepts on how to implement our genetic algorithms. We soon realised, a procedure to compare the lighting between levels was necessary. By comparing only the chromosome, we would only retrieve the

information about the position of each torch, which is insufficient data to perform a detailed comparison. By constructing a heatmap for each of the resulting levels obtained by the chromosomes, we get a real read of the level's lighting, allowing for a much more exact comparison. However, this approach comes with a computation penalty, since for each chromosome produced by the genetic algorithms, we must calculate the corresponding heatmap in order to obtain a fitness value. Nonetheless, we believe that this trade off in performance is worth it, as it allows us to achieve much more valuable results, due to the entire lighting of the level being considered towards the calculation of the fitness value, instead of only the positioning of the torches.

## **Summary**

With this chapter we hope to have provided a more profound understanding of all the components developed for the lighting element of the Editor Buddy Lighting Mode, as well as how it is integrated with the Legend of Grimrock 2 Dungeon Editor. In addition we also hope to have given a deeper explanation to why certain design decisions were made, as well as a extensive description of how the Objective and Innovation algorithms function and the reason behind their parameters.

# 7

## Evaluation

### Contents

---

7.1 Solution evaluation . . . . .	59
7.2 Methods . . . . .	60
7.3 Results . . . . .	63
7.4 Study Conclusions . . . . .	71
7.5 Recommendations . . . . .	72

---





In this chapter we discuss our evaluation strategy for the Editor Buddy Lighting Mode. We start by introducing the study's goals and a brief study description. Afterwards, we describe the methodology, present the collected data and our interpretation of the results. To finish, there is a conclusion of the overall evaluation process with our key-findings.

## 7.1 Solution evaluation

### Study Goals

The objective of this evaluation is to test the utility and the efficiency of the Editor Buddy Lighting Mode. Meaning that we want to test the usefulness of our solution in the generation of lighting suggestions in a co-creative process, by observing its contribution to a designer's when creating a level. Another goal of this evaluation is to test the designer's interaction with the EB Lighting Mode's user interface. In other words, evaluate if the designers interaction with the UI is fluid and if they adjust the algorithm's parameters to achieve suggestions that they find valuable.

The evaluation's purpose is to only test the lighting component of the application, meaning that we will not be testing the level layout component. This is due to the fact that Lucas has already performed an evaluation of his work and retrieved results regarding the usefulness of the level layout component. In order to not distract the participants, we have created a different user interface that omits the level layout component of the EB Lighting Mode, which can be observed in figure 7.1.

### Study Description

The evaluation process was conducted with the final version of the Editor Buddy Lighting Mode and since the application's target audience is novice level designer's, the tests were performed by people that either have acquired knowledge in level game design and might have previously built levels, or by people that game on a daily basis and have limited knowledge in level design. We decided not to focus our attention on more experienced level designers that work in the field, as they already have a specific mindset when building a level, thus diminishing the usefulness of the EB Lighting Mode. Additionally, when researching Lucas's evaluation results, one of his key-findings was that professional level designers would not interact with his application until they were finished creating the level. As he stated: "...professional participants demonstrated less inclination towards receiving and interacting with suggestions, because they felt like they were being evaluated themselves in how well they could accomplish the task, instead of us being evaluating our solution. " [13] in the Interpretation section of his results.

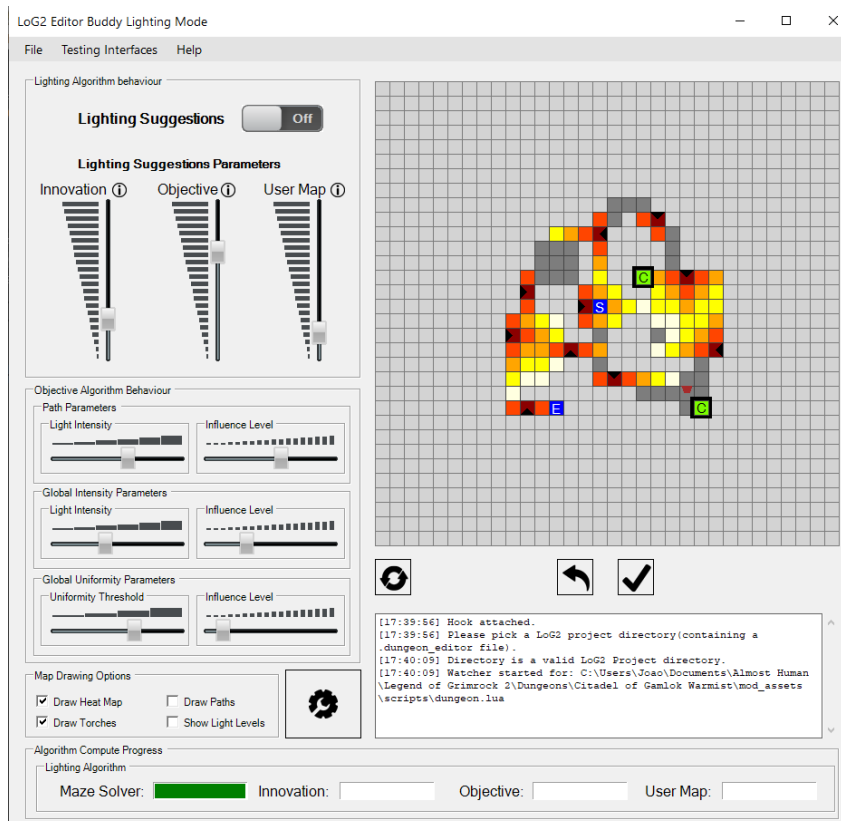


Figure 7.1: Editor Buddy Lighting Mode user testing UI.

## 7.2 Methods

### Study Design

The study was performed by a small group of participants in the facilities of IST-Taguspark campus. All the participants play videogames regularly and have a background in level design, except for one, who is an avid gamer but lacks knowledge in level design. All the participants were given a scenario, where they were working for videogame design company and they were responsible for all illumination on the current project. For each of the levels they were only given one guideline about the level's timeline in the overall videogame plot. The reason behind it, was to evaluate how participants change their lighting process considering the difficulty of the level. It is also worth mentioned, that we strongly advised the participants to use the Dungeon Editor's preview window as a way to validate their development.

First the participants were given a document with information about the test, as well as brief guide on how to operate the LoG2 Dungeon Editor. The participants were also given about two minutes so that they could get familiar with the level editor. Afterwards, we presented the first task, which consisted in asking the participants light up an already built level, illustrated by figure 7.2. This task did not have a time-limit but ideally it should not take more than 10 minutes to complete, although the expected time

was around 7 minutes.

The second task was similar to the first one, however this time we introduced the Editor Buddy Lighting Mode to the participants. Before the beginning of the test, we gave an extensive introduction of the application and its components, explaining the how to operate the EB Lighting Mode and the function of each slider, we also demonstrated how to select paths within the 2D canvas and how to generate and export suggestions. Once we clarified all the possible doubts the participants had, we began task number two, where the goal was the same as task one. Participants had to illuminate the same level, although this time they could do it with the help of the EB Lighting Mode. The test had the same duration of the first and when the participants were satisfied with the level's illumination the test was concluded.

The third and final task, was similar to the second one, however the level was different. The level the participants were asked to illuminate, is the last level of the Legend of Grimrock 2, entitled "Tricksters Lair", illustrated by figure 7.3. To the participants, this task could be seen as identical to the previous but it had a different purpose in the process of evaluating of our tool. Similarly to the other tasks, this task did not have a time limit, yet we predicted that it should take around 10 to 15 minutes to complete.

Level guidelines:

- **Task 1:** This level is one of the first level's of the game, use lighting accordingly.
- **Task 2:** This level is one of the first level's of the game, use lighting accordingly.
- **Task 3:** It is the last level of the game and it includes hard puzzles and a difficult boss fight.



**Figure 7.2:** The level participants were asked to illuminate in task 1 and 2.



**Figure 7.3:** Tricksters Lair level from LoG2 that participants were asked to illuminate in task 3.

## Data Collection Methods

We used three types of data collection, **participant observation**, **questionnaire** and **screen recording**.

Participant observation was conducted throughout the entire evaluation phase, including when the tasks were being performed. The data obtained was collected by observing the participants and taking notes. Some of the notes include the participants questions or behavior when performing the tasks and when interacting with the Dungeon Editor or the EB Lighting Mode. Figure 7.4 represents the setup used for the evaluation, using a single computer with the Dungeon Editor on the right and the EB Lighting Mode in user testing interface on the left.

The collection of data by screen recording the participants when performing the tasks, was conducted so that we could retrieve relevant information after the tests. After analyzing the data we hoped to be able to observe patterns in the participants interaction with the EB Lighting Mode or to draw out conclusions by analyzing their behavior.

Upon concluding the tasks, participants were asked to fill out a questionnaire, which can be found in the appendix A regarding their experience of building a level with and without the Editor Buddy Lighting Mode. This questionnaire purpose was for us to get a personal opinion of the participants on the usability of developed solution.

Furthermore, it is important to mention that all participants gave their consent for us to record the computer's display, as well as to take personal notes during the duration of the tests.

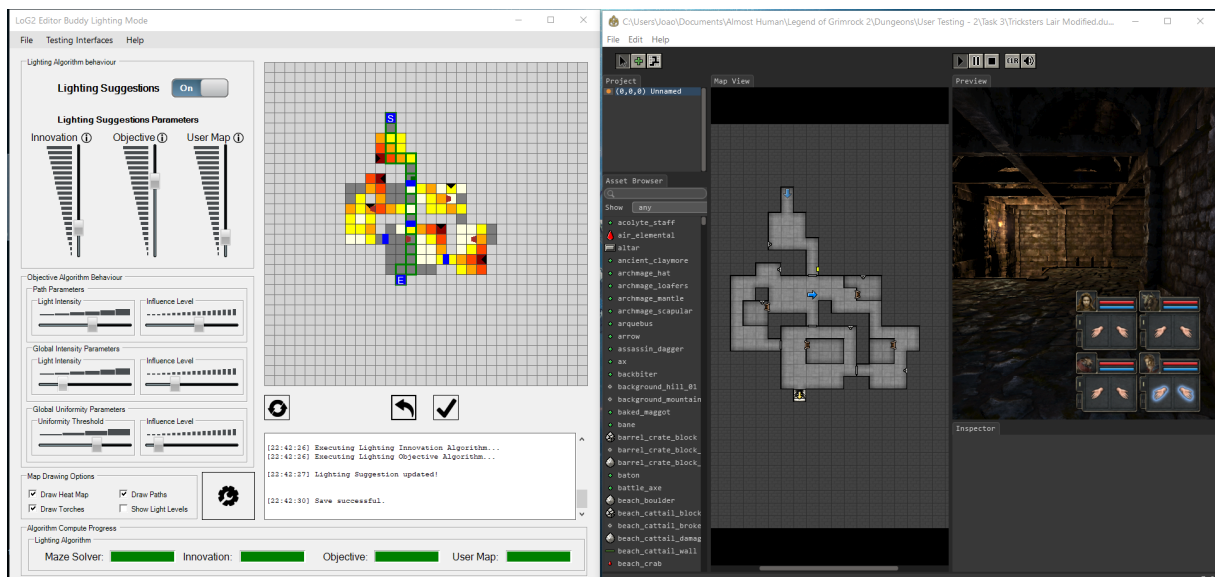


Figure 7.4: Setup used in the user testing.

## Data Analysis Methods

To determine the usefulness of the EB Lighting Mode, we used three data analysis methods, questionnaires, recordings of the computer's display and participant observation. Each had a different purpose, allowing us to gather more specific information of the participants' experience.

The **questionnaires**' purpose was for us to retrieve the participants' opinion of the usefulness of the Editor Buddy Lighting Mode. The questionnaires were anonymous and in terms of the data collected, it was better analyzed after being displayed in charts, since it allowed us to observe patterns and discrepancies in the results.

The **recordings of the computer's display** were crucial for analyzing the participants' interaction with the EB Lighting Mode, the list below represents some of the most important aspects we wanted to observe.

- The level of comfort the participants showed when using level editors like the LoG2 Dungeon Editor, in order to evaluate how well versed they were at level construction.
- Number of distinct interactions the participants had with the EB Lighting Mode.
- Number of suggestions the participants asked the application to generate.
- How many times did the participants adjusted the parameters in the UI.
- How much time did the participants spent on adjusting the parameters in the UI.
- Key-issues that might have occurred during the tasks.
- Chronology of the participants actions over time.

**Participant observation** had a different purpose than the other methods of data collection. Its purpose was for us to gather extra information that could not be observed through screen recording or through the questionnaires. We were specially looking out for things such as questions about the EB Lighting Mode, the evolution of their interaction with the application or simple details in the participants behavior.

## 7.3 Results

Throughout this section we present our collected data regarding the questionnaires, the screen recordings and the player observations. Furthermore, we present an interpretation of the collected data as well as our key-findings. At the end, we give a conclusion of the obtained results.

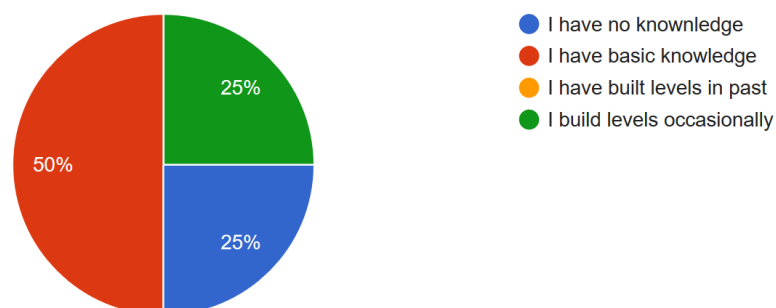
## Evaluation Environment

The evaluation tests were performed by 4 individuals, that at the time of testing were students of the Information Systems and Computer Engineering course at Instituto Superior Técnico. By no particular reason, all participants were male and between the ages of 23 and 28. Three of the participants had knowledge in level design, noticeably taking courses of game design, while one of the participants lacked a background in game design, however he was an avid gamer thus having some experience on the logic of level design. The test were performed in different locations of IST-Taguspark, however they were all conducted in meeting-rooms, where the participants were encouraged to ask questions about any doubts they had on both LoG2 Dungeon Editor or the EB Lighting Mode. As previously mentioned, the test were performed on a single computer with the EB Lighting Mode and the LoG2 Dungeon Editor were placed side-by-side, as seen in figure 7.4.

## Questionnaires

The beginning of the questionnaire there were three questions with the purpose of defining the demographic of the participants. Question 3 of the questionnaire, was the most important of the three as it allowed us to classify the participants' knowledge of level design, to which the results can be found in figure 7.5.

How familiar are you with videogames level design?



**Figure 7.5:** Chart regarding the questionnaire's question 3.

As can be observed, our small group of individuals contains at least one participant for every level of knowledge in level design, individuals that have no knowledge of level design, individuals that have basic knowledge and individuals with practical experience in building levels.

Regarding their limited involvement with Editor Buddy Lighting Mode, the common opinion the partici-

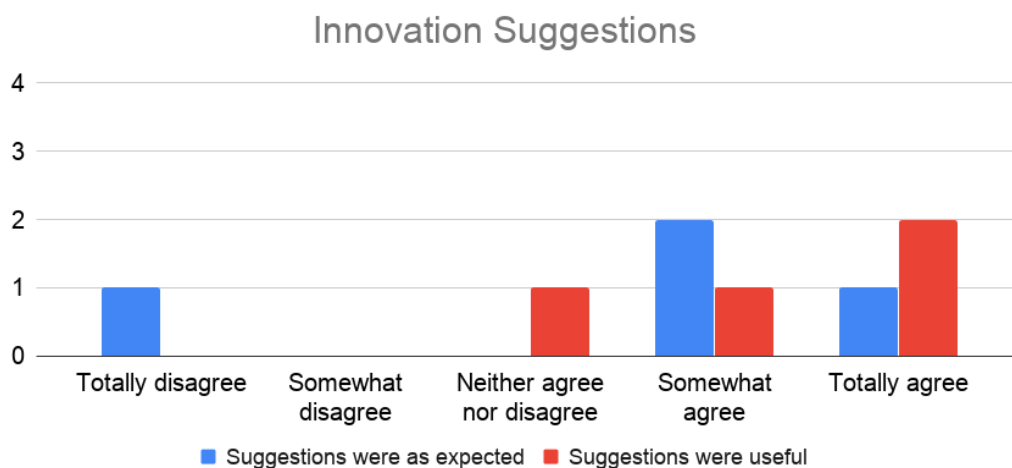
participants expressed that it was a useful tool considering their level design experience. The full questionnaire can be found in Appendix A and the list below represents questions 4 through 7 from the questionnaire's usability section and the table 7.1 the respective responses from the participants:

- **Question 4:** It was easy to edit the level's lighting using the Legend of Grimrock 2 Dungeon Editor.
- **Question 5:** It was easy to configure the Editor Buddy Lighting Mode behavior using the available interface controls.
- **Question 6:** It was easy to create and delete paths on the Editor Buddy Lighting Mode interface.
- **Question 7:** There were no communication issues between the LoG2 Dungeon Editor and the Editor Buddy Lighting Mode.

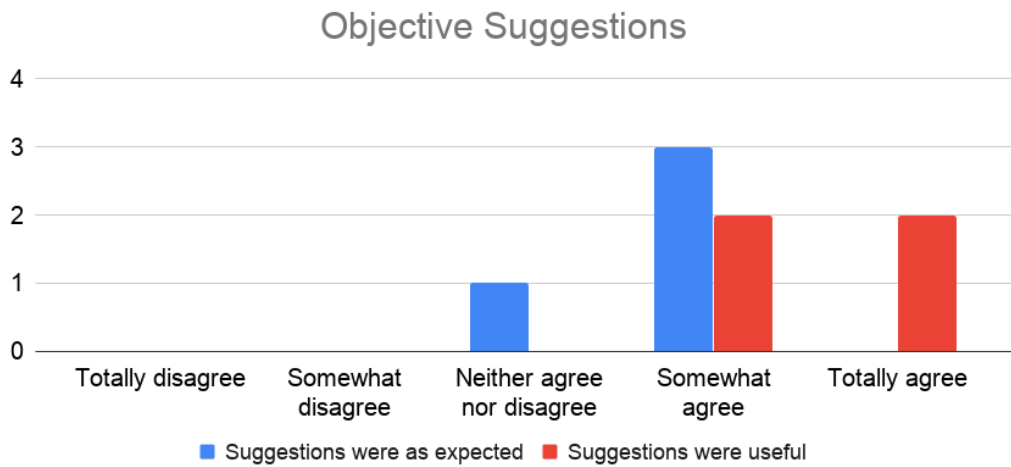
LoG2 Dungeon Editor and Editor Buddy Lighting Mode usability				
Scale	Question 4	Question 5	Question 6	Question 7
Totally disagree	0	0	0	0
Somewhat disagree	0	1	0	0
Neither agree nor disagree	0	1	1	0
Somewhat agree	1	0	0	1
Totally agree	3	2	3	3

**Table 7.1:** Questionnaire usability results

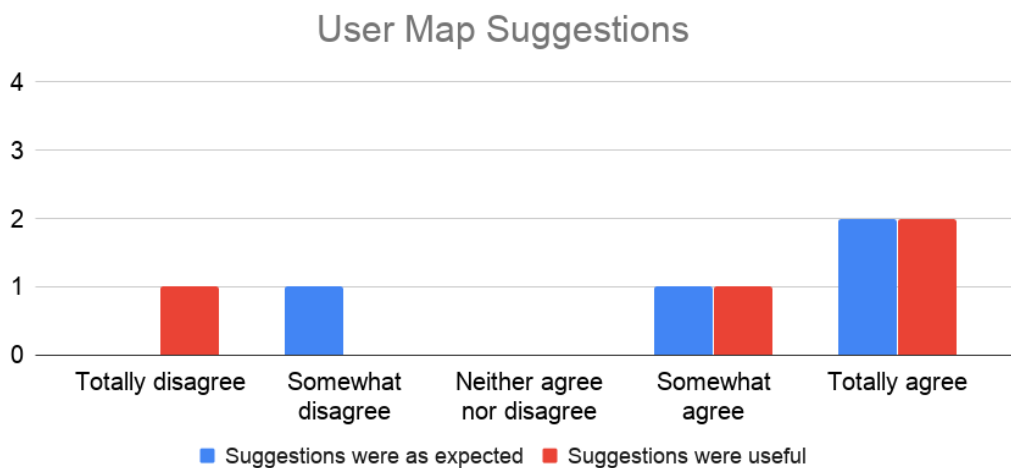
Regarding the EB Lighting Mode's behavior, the following figures, figure 7.6, figure 7.7 and figure 7.8, correspond to the responses of the participants on the questions 8 through 13 of the questionnaire. They represent the participants' opinion on the results generate by the algorithms when adjusting the UI's sliders.



**Figure 7.6:** Innovation algorithm generated suggestions - expectation vs usefulness



**Figure 7.7:** Objective algorithm generated suggestions - expectation vs usefulness



**Figure 7.8:** User Map algorithm generated suggestions - expectation vs usefulness

## Screen Recordings

When evaluating the screen recordings we were searching for both significant patterns when the participants were constructing their level as well as some performance metrics.

The table 7.2 represents some of the metrics and their respective values obtained from the results of the tasks 2 and 3.

It worth mentioning that one of the participants took more time adjusting the parameters and generating suggestions in task 3 than the others. The results without taking into to account the values of this participant are represented by table 7.3

Upon analyzing the values of the metrics and through our observation of the recordings, there are



Editor Buddy Lighting Mode task 2 and 3 interaction results		
	Task 2	Task 3
Average number of interactions	1.7	1
Average number of generated suggestions	4.25	9.75
Average number of times sliders were adjusted	2.5	3.5
Average number of paths created	0.75	0.25
Average number of exported suggestions	1.75	1
Average total time spent on EBLM	1min28secs	1min58secs

**Table 7.2:** Screen recordings task 2 and 3 results.

Editor Buddy Lighting Mode task 3 interaction results		
	Task 3	Task 3 without a participant
Average number of interactions	1	1
Average number of generated suggestions	9.75	2.33
Average number of times sliders were adjusted	3.5	1
Average number of paths created	0.25	0.33
Average number of exported suggestions	1	1
Average total time spent on EBLM	1min58secs	57secs

**Table 7.3:** Comparison between the screen recordings task 3 results with and without one of the participants.

some interesting findings that can be extracted:

- The participants when familiarized with the EB Lighting Mode take significantly less time adjusting the parameters in the UI to generate a suggestion they are satisfied with, except for one of the participants.
- In task 2 and 3, all the participants started by using the EB Lighting Mode before interacting with the LoG2 Dungeon Editor.
- In task 3 none of the participants felt the need to adjust the sliders and generate new suggestions, after the first exported suggestion.
- The participants did not seem to have much interest in the Path Objective component of the EB Lighting.
- All the participants showed more interest in the Objective algorithm, more noticeably the Global Intensity sub-goal.
- Two of the participants in both tasks, removed or placed torches after exporting a suggestion.
- All of the participants' level in task 2 had considerably more torches than of the participants' level in task 1.
- In task 3, all participants created a similar lighting to the original LoG2 level "Tricksters Lair" with an average discrepancy of 2.5 torches.

## Participant Observation

The participant observation proved to be a valuable component of the evaluation, as we managed to identify some of the problems of the EB Lighting Mode. The most noticeable problem we encountered in our interaction with the participants, was a difficulty in understanding the purpose of the algorithms. Some participants required a more detailed explanation than the others in the function of some of the algorithms, specially the three sub-objectives of the Objective algorithm. Another problem we have observed was that some participants required a second explanation about the influence sliders of the Objective algorithm.

One key-finding we observed, was the evolution in comfort of the participants between task 2 and task 3. It is natural given that they already had gained experience when experimented with the application in task 2, however when observing the participants in the beginning of task 3, all but one quickly adjusted the parameters and generated a suggestion with the goal they had in mind.

Another curious aspect was how the participants clearly created different lighting for the task 3. As previously mentioned, the only difference, besides the level layout, between tasks 1 and 2 and the last task, was the guidelines. The guideline for the tasks 1 and 2 was that the level they were illuminating was one of the first levels in the game and all the participants interestingly created a well lit level, where none of the sections in the level had a low light intensity. Whilst on the last task, all of the participants illuminated the level with a low light intensity, all of the levels generated had at least one dark section and very few areas had more than one torch.

When observing the participant who took longer than the others to complete the test, we identified that he was having trouble adjusting the parameters of both the Global Intensity slider and the Uniformity slider. He also referred that the Uniformity suggestions although useful, were not generating the results that he was quite expecting, thus the increase in the number of the generated suggestions he asked of the EB Lighting Mode.

One detail we found particularly interesting was that a few of the participants did not verify the complete level with the LoG2 Dungeon Editor preview window. Upon completing the tests, we questioned the participants the reason behind it, to which they replied that they mostly relied on the EB Lighting Mode heatmap to visualize the lighting of certain parts of the level, while only using the preview window to check certain sections where they had doubts about the lighting.

## Interpretation

In this subsection, we present our interpretation of the results as well as address some of the problems and the reasons behind them. Firstly, we describe the two major problems we have extracted from the evaluation and then we analyze the participants difficulties regarding the EBLM's algorithms.

The first of the major problems of the EBLM we have identified through the evaluation, was that the participants rarely used the application in a co-creative process, they mostly operated it as a level lighting generator. This is proved by the low number of interactions the participants had with the EBLM, as can be observed in the table 7.2. In the best case scenario, which happened in the first task the EBLM was introduced, the participants averaged a number of 1.7 interactions with it. This is further supported by the fact that all of the participants always started the requested tasks, by exporting a suggestion and then proceeding to test it, with only two of the participants actually editing and refining the exported level. Furthermore, in task 3, none of the participants asked the EBLM to generate new suggestions after exporting one, thus making us believe that they saw the application as level lighting generator instead of colleague. Despite the concept of co-creation not being apparent to the participants, there is one positive note that we can extract from the low number of interactions with the EBLM, which is that the participants when exported a level, they believed that the generated lighting by the application was at least acceptable and appropriate to level they were illuminating. Thus making the EBLM a useful tool when creating a level's lighting. This is also supported by the fact that only two users felt the need to modify the exported level and that the changes made were minor.

The other major problem we observed was the difficulty the participants had in understanding all of the components of the EBLM user interface. Most of the participants required a very detailed explanation for each of the sliders and in some cases the tests were interrupted so that we could clarify any remaining doubts the participants had. To combat this problem, we believe that the user interface should be improved, in order to make the purpose of each the algorithm's slider more understandable. Before the beginning of task 2 of the test we did interact with the EB Lighting Mode and demonstrated a few examples of how each of the sliders influence the generated suggestions, however we think that using a video demo would have been a better approach, as it is a more controlled environment and the effects of each of the sliders on the generated suggestion could have been more clearly demonstrated.

Another reason that might be behind the participants confusion, is the abundance of sliders present in the user interface. The EBLM has a learning curve due to requiring the participants to learn the purpose of every single one of the nine sliders present in the user interface. Despite its learning curve, we believe that the core concepts of each of the sliders, were simple to comprehend after the participants had time to test them by themselves. This assumption is backed up by the retrieved numbers from the screen recordings. If we compare the metric values of task 2 and 3 without one of the participants, it is evident that the participants felt much more comfortable with the EBLM and its sliders after having time to experiment their effects. Thus resulting in a decrease in total time spent of EBLM, as well as a decrease in the number of generated suggestions and the number of times the participants adjusted the sliders. The participants in task 3, adjusted the sliders and generated suggestions one third faster than in task 2. Although we can not take the participant who took longer than the others out of the question.

Since the volume of participants in our user testing was quite small, this participant might represent a larger audience. If that were the case, then some major changes would be required to our application, as the understanding of all its parameters is not that clear. More specifically the purpose of each algorithm and how to control each of their influences' on the generated suggestion.

Another obvious point we retrieved from the user testing, was the lack of use of the Innovation algorithm, one of the participants even rated that its generated suggestions were not as he expected in our questionnaire. From our observation, all of the participants mainly focused on using the Objective algorithm. The User map algorithm was also used to generate suggestions, however not in nearly to the same extent of the Objective.

Furthermore, when analyzing the results of the questionnaires, it is obvious that the Objective algorithm was the participants favourite. It was the only one who did not receive a negative evaluation, where both of the Innovation and User Map algorithms received mixed evaluations. We suppose that the shortage in use of the Innovation slider was directly linked to the chosen levels for the evaluation. Given that both levels used in the tasks, in their original state did not have any light sources and the goal of the Innovation algorithm is to generate suggestions as distinct as possible from the current level, it meant that suggestions generated by the EBLM were mostly filled by torches. This translated into the participants being overwhelmed by the amount of torches in the suggestion. This may represent a possible problem with the EBLM, where some of the generated suggestions have too many light sources.

The User Map algorithm suffered from the same problem as the Innovation algorithm, however the effect was the opposite. Since the participants always started the tasks using the EBLM, when they attributed a high value to the User Map slider, the generated suggestions would represent levels with hardly any torches. Which is the expected behavior for the User Map algorithm, but its usefulness might have been dismissed due to the chosen levels for the tests. Another fact that might have influenced the participants dismiss the value of the User Map algorithm, was that they always started the tasks by generating suggestions and rarely went back to the EBLM after exporting a level, they never actually got to discover the true value of the User Map algorithm. The User Map algorithm becomes much more relevant once a level is already fully or partly illuminated and the level designer wants to generate similar lighting to the existing but with a few modifications.

One issue we did not foresee, was the absence of path creation. The results obtained from the question 6 of the questionnaire, reveal that the process of creating a path is simple, so its lack of use must be due to its usefulness. As can be observed in the table 7.3, only one of the participants decided to create a path in task 3. What is even more curious is the fact that all of participants throughout the task 3, did adjust the path objective sliders. Our first impression was that three of the participants were either satisfied with the automatically provided path (the shortest path between the starting point and the ending point of the level), or they misinterpreted its use. When questioned about it at the end of the

test, one participants responded he did not feel the need to create new paths as the level was too small to make a difference, while the other two responded that the provided path was as they wanted. This leads us to conclude that the participants did see value or usefulness in the path sub-objective, however they believed it was more useful to control the lighting intensity on the path that leads towards the level's end, instead of creating new different paths.

The participant who took longer than his colleagues in task 3, expressed that he was having trouble in adjusting the Uniformity objective, as well as seeing any significant changes to the generated suggestions. This observation confirms one suspicion we had, that the uniformity fitness function might need a rework, since its results are not clear in the generated suggestions.

## 7.4 Study Conclusions

To begin, we you like to thank all the participants of the evaluation tests, thanks to them we found some key issues and areas where we can improve the Editor Buddy Lighting Mode.

The first aspect we would like point out is that although the EBLM was mostly used as lighting generator, it does not discredit its worth in working as colleague in the illumination of levels, this concept could have been further verified with more extensive testing. We also believe that we should have pursued different tests to prove the computer as colleague paradigm. For example, the addition of a task, where the participants first had to build their lighting in the Dungeon Editor and only after be allowed to use the EBLM. Also the addition of bigger levels to the tests, might also been useful, as the it would demand more light sources and modifications to the generated suggestions, thus making the cooperative experience more compelling.

The problem of the sliders' purpose, we accept it needs improvement. This might have been prevented with an earlier evaluation phase to test the user interface. However, we still consider the built UI to be a good approach, as the experience the participants gain in task 2 was directly translated to task 3. The participants in task 3, were much more proficient in adjusting the sliders and generating suggestions.

We would also like to point out the results produced by the algorithms were on par with what we expected, specially the Objective algorithm. Despite one of the users did not feel the same way as we do, we consider that the value of the Innovation and the User Map algorithms was crucial. After analyzing the screen recordings and the generated suggestions, we think that the participants failed to realize their influence on the overall suggestions. We assume that their dismiss towards these algorithms is directly related to the fact they focused their attention to the Objective algorithm, specially the Global Intensity sub-objective.

In terms of the EBLM performance, none of the participants had any sort of complaints. Therefore

we consider to have reached a good balance between suggestion quality and computational time.

To conclude, we believe to have proved that the Editor Buddy Lighting Mode is a useful tool in helping inexperienced level designers create lighting for level, however the concept of working as colleague could have been more clearly demonstrated with more appropriate tests.

## **7.5 Recommendations**

There are two recommendations we can give to anyone who is trying to conduct a similar study. One is more specific to people developing a similar work as ours and the other is towards everyone who is planning on performing an evaluation.

The first recommendation, is to focus the tests on the computer as colleague paradigm. It is not an easy concept to test and it requires a more extensive test than what was performed in this work. We suggest the addition of dedicated tests that encourage the participants to have multiple interactions with the solution when building a level. More specifically, the inclusion of tests that encourage the participants to be constantly switching between working on the level editor and interacting with application.

The second recommendation is, if the participants give their consent, to record the entirety of the evaluation phase. If we had recorded the whole interaction with the participants, we might have been able to extract even more detailed results. For example the participants' questions, although we wrote almost all of them on notepad, the interaction could be further analyzed and perhaps allowed us to have a more profound understanding of the participants' difficulties.

# 8

## Conclusion

### Contents

---

8.1 Solution performance . . . . .	75
8.2 Future Work . . . . .	75
8.3 Final Remarks . . . . .	76

---





In this last section of the document, we provide some conclusions about the topics involving this work, as well as give our final opinion about the developed solution and our assessment on the impact it had on the participants during the evaluation tests. We also discuss future improvements or modifications that can be worked into EB Lighting Mode. These improvements or modifications include assessments that have been suggested in the evaluation or general developments that we did not have time to accomplish.

## 8.1 Solution performance

We classify the Editor Buddy Lighting Mode impact on the participants as a success. Despite the issues pointed out in the Study Conclusions section, the overall impact it had on the participants experience of illuminating a level was positive. The participants interacted with the application's sliders, generated levels and exported several suggestions, even though we clearly specified that the use of the EB Lighting Mode was optional, all the participants decided to use it. Another reason for us to consider it a success, is the fact that all the participants shared the same opinion that the EB Lighting Mode is useful tool for inexperienced level designers to create a level's illumination.

As we mentioned, one area we wish we could have tested more thoroughly was the computer as colleague paradigm. The participants always interacted with the EB Lighting Mode to generate the lighting of a level, but rarely returned to it and asked for different suggestions. Although we believe this might be a problem of how the tests were performed, we still consider that the EBLM can be used and would do quite well in a computer as colleague environment, which could be proven with further testing.

In the end, we hope to have created a useful tool that helps novice designers in the process of illuminating a level.

## 8.2 Future Work

With the help of the evaluation we realized that several aspects of the EB Lighting Mode could be improved or that it would benefit from the addition of certain features. These are the most relevant aspects we consider to be worthy of a future work:

Making changes to the user interface, in order to make clear the purpose of each of the sliders. Also the sub-objectives of the Objective algorithm should be better defined as being part of it. We noticed that some participants were at first confused by this element of the UI.

Merge the Uniformity sub-objective with the Global Intensity sub-objective. This makes for a cleaner and simpler UI and would prevent some of the doubts the users might have. A user can always make parts of the level darker or brighter by selecting paths.

Adding an option to create zones in the level. This can be achieved in the current state of the EBLM

by creating multiple paths, however having the addition of simply selecting an entire zone would make more sense as user's would waste less time. This could be performed by holding a mouse click and dragging over the desired area.

In terms of overall algorithm improvements for a future work we recommend the implementation of some sort of restriction on the number of the light sources that can be added per suggestion. We realized this when observing participants adjusting the parameters of the Innovation algorithm and the generated suggestions would at times fill the level with torches. Another alternative is to check how much does one torch contribute to the heatmap. For example if two torches are placed side-by-side, the overall difference on the heatmap would be insignificant, therefore one of the torches could be removed with almost no impact to the level's lighting.

Regarding the evaluation tests, we have already described our recommendations, however to summarize we suggest a heavier focus on the computer as colleague paradigm, as the participants might not understand that it is the application's purpose to be used in this concept.

One aspect we did not have time to pursue but it would be an interesting addition to this work, is the incorporation the EBLM with Delgado's Editor Buddy Puzzle Mode. With this addition, level designers would have a tool that worked as colleague in the process of creating a level layout, illuminating the level and that generated puzzles, for a more complete experience.

To conclude, a more ambitious goal is to incorporate the EBLM with other videogames and their level editors, this is arduous task as the EBLM is deeply integrated into the LoG2 environment.

### **8.3 Final Remarks**

I would like to express that the entire process of research, development and evaluation was truly an enriching experience, both personally and professionally. From the beginning of the research process, where I gained a deeper understating of the current state of the computer as a co-creative partner, to the arduous task of developing an entire solution that attempts to accomplish this paradigm, it sincerely was an amazing rewarding learning experience. I can only wish for our contribution to bring some value to the co-creation in level design community.

# Bibliography

- [1] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-aided game level authoring." in *FDG*, 2013, pp. 213–220.
- [2] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.
- [3] R. E. Franken, *Human motivation*. Brooks/Cole Publishing Company Pacific Grove, California, 1998.
- [4] M. A. Boden, *The creative mind: Myths and mechanisms*. Routledge, 2004.
- [5] —, "Computer models of creativity," *AI Magazine*, vol. 30, no. 3, pp. 23–23, 2009.
- [6] E. De Bono and E. Zimbalist, *Lateral thinking*. Viking, 2010.
- [7] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.
- [8] K. Binsted, H. Pain, and G. D. Ritchie, "Children's evaluation of computer-generated punning riddles," *Pragmatics & Cognition*, vol. 5, no. 2, pp. 305–354, 1997.
- [9] H. Cohen, "The further exploits of aaron, painter," *Stanford Humanities Review*, vol. 4, no. 2, pp. 141–158, 1995.
- [10] T. Lubart, "How can computers be partners in the creative process: classification and commentary on the special issue," *International Journal of Human-Computer Studies*, vol. 63, no. 4-5, pp. 365–369, 2005.
- [11] C. Darwin, A. R. Wallace *et al.*, "Evolution by natural selection." *Evolution by natural selection.*, 1958.

- [12] J. Feil and M. Scattergood, *Beginning game level design*. Thomson Course Technology, 2005.
- [13] P. Lucas and C. Martinho, "Co-creativity in videogame level design."
- [14] G. Delgado and C. Martinho, "Co-creativity in videogame puzzle creation."
- [15] J. Newcombe, "Genetic algorithm framework."
- [16] C. W. Totten, *An architectural approach to level design*. AK Peters/CRC Press, 2018.
- [17] S. De Jong, *The Hows and Whys of Level Design*. Selbstverl. des Verf." Hourences", 2008.
- [18] M. Seif El-Nasr, "Intelligent lighting for game environments," 2005.



## **Questionnaire Appendix**

# Editor Buddy Lighting Mode Form

The main goal of the form is to evaluate the utility of the Editor Buddy Lighting Mode application and its efficiency at providing an enhanced experience in the creative process of level design, more specifically in its value as colleague in the illumination of levels. Furthermore, it will allow us to understand the current state of the user interface, as well as the behavior of the developed algorithms and their ability to generate suggestions in the computer as colleague paradigm.

\*Required

1. What is your age? \*

---

2. What is your sex? \*

Mark only one oval.

- Male  
 Female

3. How familiar are you with videogame level design? \*

Mark only one oval.

- I have no knowledge  
 I have basic knowledge  
 I have built levels in the past  
 I build levels occasionally

## Interface usability

---

On a scale of 1 to 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.

4. It was easy to edit the level's lighting using the Legend of Grimrock 2 Dungeon Editor \*

Mark only one oval.

	1	2	3	4	5	
Totally disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Totally agree

5. It was easy to configure the Editor Buddy Lighting Mode behavior using the available interface controls \*

Mark only one oval.

	1	2	3	4	5	
Totally disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Totally agree

6. It was easy to create and delete paths on the Editor Buddy Lighting Mode interface \*

Mark only one oval.

	1	2	3	4	5	
Totally disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Totally agree

7. There were no communication issues between the LoG2 Dungeon Editor and the Editor Buddy Lighting Mode \*

Mark only one oval.

	1	2	3	4	5	
Totally disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Totally agree

## Algorithm behavior

---

On a scale of 1 to 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.

8. Suggestions generated using the Innovation controls were in line with my expectations \*

Mark only one oval.

	1	2	3	4	5	
Totally disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Totally agree

9. Suggestions generated using the Innovation controls were useful \*

Mark only one oval.

	1	2	3	4	5	
Totally disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Totally agree

