Story-based AI-Powered Puzzle game created with GMS2

Developer´s report

Sofia Musick

Examensarbete

Informationsteknik

2022

EXAMENSARBETE	
Yrkeshögskolan Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	20329
Författare:	Sofia Musick
Arbetets namn:	Berättelsebaserat AI-drivet pusselspel skapat med GMS2: utvecklarrapport
Handledare (Arcada):	Andrey Shcherbakov
Uppdragsgivare:	

Sammandrag:
När det kommer till spelutveckling ser många människor det som en avlägsen, skrämmande uppgift som endast är avsett för programmerare. Syftet med detta examensarbete är att beskriva och analysera utvecklingen av ett 2D-spel i GameMaker Studio 2. Examensarbetet går in på djupet av hur ett spel kan utvecklas i GameMaker Studio 2 och vilka problem utvecklaren kan stöta på. Examensarbetet beskriver det som behövs för att ett spel ska kunna utvecklas i GameMaker Studio 2 och utforskar de olika delarna av spelmotorn. Det går igenom specifika steg och processer för att skapa ett spel och använda verktygen från GameMaker Studio samt speltillverkarens språk, GML, för att på bästa sätt bygga upp spelet. Examensarbetet förklarar planen som gjordes för att utveckla det här spelet, hur själva spelet är uppbyggt, vilka tillgångar som behövdes implementeras för att får spelet att fungera och uppbyggnaden av själva spelet. I slutet av arbetet beskrivs slutsatser, hur det var att utveckla ett spel, vilka mål som nåtts och inte uppnåtts samt slutliga tankar och analys. Slutresultatet är ett fungerade spel med specifika mekaniker som var planerat för spelet.

Nyckelord:	Spelutveckling, 2D, GameMaker Studio 2
Sidantal:	
Språk:	Engelska
Datum för godkännande:	

DEGREE THESIS	
Arcada University of Applied Sciences	
Degree Programme:	Information Technology
Identification number:	20329
Author:	Sofia Musick
Title:	Story-based AI-powered Puzzle game created with GMS2: Developers report
Supervisor (Arcada):	Andrey Shcherbakov
Commissioned by:	

Abstract:

When it comes to game development, many people see it as a distant, daunting task, that is only made for programmers. The purpose for this thesis is to describe and analyse the development of a 2D game in Game Maker Studio 2. The thesis goes in depth as to how a game can be developed in GameMaker Studio 2 and what problems developers can run into. The thesis starts with an explanation of the details of what is needed for a game to be made in GameMaker Studio 2 and explores the different parts of the engine. It goes through specific steps and processes for making a game and using the tools of GameMaker Studio and the GameMaker Language to best create the game. The thesis explains the plan that was made to make this game, how the game itself is constructed, what assets needed to be implemented to make the game work and the building of the game itself. Finally, we end it with the conclusion, how it was to develop a game, the goals reached and not reached as well as final thoughts and analysis. The end result is a functioning game with specific mechanics that was planned for the game.

Keywords:	Game development, 2D, GameMaker Studio 2
Number of pages:	
Language:	English
Date of acceptance:	

TABLE OF CONTENTS

FIGURES

LISTINGS

1 INTRODUCTION

Game making is an interesting concept in today's world. When you hear about game development you think that it is a huge process that takes years to develop and is unreachable for the common person. Game development is set on such a high pedestal that it is impossible to reach and most stay away from it for that reason. The process can be quite simple if one is willing to set a little time aside and get into it. It can be quite daunting at first if you do not know where to begin. However, because of the time we live in, there are a lot of different tools and software that can be used to create the game you desire.

1.1 Background

GameMaker Studio 2 is a simple tool for game development. The difference between GameMaker Studio 2 and other game engines is that it is free to use, however with a few limitations. It is also quite easy to use, compared to a lot of other different types of software and is quickly adaptable and useable.

Compared to several other software, GameMaker Studio 2 has more variety of options to use and to build the game, if compared to, for example, RPG Maker. "RPG Maker does not really incentivize learning the ropes of game development as many of its official and community-made features can replace the need to know how to program. Plus, RPG Maker is fairly limited with what users can make within it." (Crewell, 2021, para. 12).

If you compare GameMaker Studio 2 to a bigger software like Unity, there is a big difference. Unity is a good program for creating games but can be quite hard for those who do not know how to code. It also focuses a lot more on 3D games instead of 2D which is where GameMaker Studio excels. A game developer who tested Unity and other 2D games came to the conclusion about Unity that:

The editor, on the other hand, is a bear to work with. To harness Unity's full potential, you need to spend a good long while wrestling with the UI to understand where everything is and how to use it /.../ if you're looking for a 2D game engine, your quality of life will be a lot higher elsewhere (Farzan, 2019, para. 15).

This thesis will focus on the development of a game in GameMaker Studio 2 and how the process of that would be. It goes through the different stages of development and the

process of game development in the engine and how the engine works. This way the reader gets a good insight into GameMaker Studio 2 and the process a game developer goes through to make a game with the engine.

1.2 Purpose and Goals

The purpose of this thesis is to go through the steps of the process of game development in GameMaker Studio 2 and give the reader an insight into the planning, development, production process and result based on a game made in GameMaker Studio 2. The purpose is also to see what problems you often run into when building a game and how to solve them.

The goal is to build a functioning game using the tools and features of GameMaker Studio 2 and to implement story, puzzles, and AI (Artificial Intelligence). The goal is also to go through the process of game development and make a report about the steps, problems and solutions, limitations, and different types of obstacles developers run into while making a game.

1.3 Limitations

Since this thesis focuses on a prototype of a full-fledged game, the focus is put on all the important functions that it needs to have. That is that the game has a decent story, puzzle mechanics, and three different types of AI that are implemented into the environment. The thesis will mostly focus on the development of the functions of the game and not so much on the design. It also is not a fully functioning game, in the sense that this game does not run without issues, and it does not contain all the steps it needs to have. To remove all bugs would take a lot longer, so the focus is just on the implementation of the most important features.

The limitations include focusing on one game engine and not comparing too much to other engines as well as focusing on the GameMaker Language itself and not implementing too many other assets and game features that you can add. It is possible to add different types of scripts to the game, however I decided to focus on the built-in functions and code.

1.4 Tools and Software

Here follows the presentation of different concepts and tools that are used in this thesis.

GameMaker Studio 2	GameMaker Studio is a multiplatform game engine. It offers the ability to create games using drag and drop or their own programming language GML. GMS is intended to be easy to use and allows game developers to create professional games in a short amount of time. The focus of GMS is on 2D games.
C#	C# is a programming language that is intended to be light, platform-independent, modern, and object-oriented. C# can be used both as a compiled language on a local computer and in ASP.NET
C++	C++ is a powerful, flexible, general programming language. It can be used to develop operating systems, browsers, games and so on. C++ supports different ways of programming such as procedural, object-oriented, functional, et cetera.
Data analysis	Data analysis is the process of systematically applying statistical or logical techniques to describe, illustrate, summarize, and evaluate data and information. It is a process where you inspect, and clean data with the aim to find useful information.
2D	2D videogames refer to action that takes place on a 2D plane and is usually either side-scrolling or vertical-scrolling. In addition, the characters and surroundings are usually made in 2D.
Game Assets	Game assets are tools and resources used for creating games. They can be things such as graphics, characters, logos, environments, music, sound effects and so on.
PixelOver	PixelOver is a software to make arts into pixel form and give the art animations.
Blender	Blender is a free graphics software for creating visual effects and animations.

1.4.1 GameMaker Studio 2

GameMaker Studio 2 is a game engine that uses simple coding or drag and drop to make 2D games. It is a cross-platform engine created by Mark Overmars in 1999 that started out as Animo but was changed to GameMaker in 2011. The program was made in a software called Delphi. Delphi is a programming language that is used to create different kinds of software, mostly for the development of Windows applications. “The greatest

strength of Delphi programming language it´s incredibly fast compiling speed, which enables programmers to write large code-based programs that can be run on a limited resources machine." (Imran 2020, para. 5). GameMaker Studio was then released in 2012 and GameMaker Studio 2 came out 2017.

GameMaker Studio 2 offers two different types of game making, GML visual and GML code. GML visual is a drag and drop function, that allows the creator to choose what they need from a library and sculpt the game they want. The drag and drop functions are very easy to use for beginners in game making and users who do not know much about coding but still want to make a simple game.

The second is GML code which allows the user to use the coding language of GML as well as JavaScript, C++, and C# and implement scripts and other functions to sculpt the game the user wants. GameMaker Studio is a free tool that can easily be downloaded from their website or bought from the digital distribution service Steam. The free version is limited so to access the full product it needs to be purchased either from their website or from the digital distribution service called Steam.

1.4.2 Blender

Blender is a free software used to make animations and arts. This tool was used to test how to make art for the game. The bow used in the game was made with Blender. Tutorials and other information that Blender provides were used to figure out how to make the bow. Blender is an open-source 3D creation software that implements animation, modeling, simulation, rendering, and so on to create the work. Blender is a cross-platform tool that can be used in any way the user sees fit when it comes to art and animations.

1.4.3 PixelOver

PixelOver is a software that turns art into pixel art. For this project, PixelOver was used to animate the bow, which was used in the game, into a spinning motion for an effect. The way to do this in PixelOver is to import the art over into Pixel, and then transform it into pixel art which then becomes an animation. Then the x and y-axis are set to move the motion from point a to point b and then that gives the animation movement. The movement for the bow is just a spinning move. After the bow was animated, it was imported

over into GameMaker Studio as an animated image. Animation in GameMaker Studio is simple because the creator can import any kind of animated or non-animated image.

2 METHODS

There are a lot of different methods when it comes to software engineering. Here is the presentation of a few of the most used ones. Waterfall methodology, Agil methodology, scrum methodology and spiral methodology. We go into what these methods are and what their benefits are versus their weak points. The chapter does not cover the whole scope of existing software development methods. The interested reader can find a more detailed overview in the book Software Methodologies: A Quantitative Guide by Capers Jones (Jones, 2017).

2.1 Software Development Methodologies

2.1.1 Waterfall Methodology

The waterfall method is one of the most used but also one of the oldest methods out there. It follows a linear path where each step is completed before the other. It flows like a waterfall through all phases of a project. The common stages in the waterfall process are requirement analysis, design, implementation, testing and maintenance. According to Moira Alexander about the waterfall method:

> Waterfall allows for increased control throughout each phase. It offers a more formal planning stage that may increase the chances of capturing all project requirements upfront. It reduces the loss of any key information and requirements in the initial stages. One downside is that waterfall can be highly inflexible if a project's scope changes after it is already under way. (Alexander, 2021, para. 6).

The benefits for this method are that it follows a specific plan which makes it easier for the project to follow a plan without jumping all over the place. It helps keep the project in certain borders and get out the project when each step is done. The downsides are that there is very little flexibility when it comes to the project. The Lucid team added that:

> Saving the testing phase until the last half of a project is risky, but Waterfall insists that teams wait until step four out of six to test their products. Outside of the software industry, the testing phase could mean showing a new website design to a client, A/B testing content, or taking any number of steps to gain empirical data on the viability of the project. At this point, the project

has likely taken considerable time to complete, so large revisions could cause significant delays (LucidChart, 2017, para. 18).

2.1.2 Agile Methodology

The Agile Method was made as a follow-up of the waterfall method. Because the waterfall method is so strict, agile was made to make the process more flexible. When explaining it more, Wrike adds:

> The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. (Wrike, 2019, para. 1).

The process of the agile method is Define, Design, Build, Test, Release. The benefits for the Agile method are that it is very flexible and relatively easy to use. "Agile encourages teamwork, collaboration, self-organization and accountability. This helps with overall motivation and commitment to a project's outcomes and goals." (AIPM, 2021, para. 5). The cons are that it can be hard to predict the outcome and it can become a bit messy when developing and working on multiple things at the same time which can make documentation left behind or forgotten.

2.1.3 Scrum Methodology

Scrum is a method based of the Agile methodology. It was made to be more efficient than Agile Methodology and work more smoothly and more flexible. Digite has this to say about Scrum:

> Scrum is an agile development methodology used in the development of Software based on an iterative and incremental processes. Scrum is adaptable, fast, flexible and effective agile framework that is designed to deliver value to the customer throughout the development of the project. (Digite, 2019, para. 1).

The process of scrum is product backlog creation, sprint planning and creating backlog, working on sprint, testing and product demonstration, retrospective and the next sprint planning.

Benefits of scrum are flexibility, involvement of costumers for best result, fast results and simple testing and easy visibility of all stages. The downsides are, "If a member of the team leaves the process during development, it has a negative impact on the project

development.” (Ajmal, 2017, para. 3), no time limit so the workflow can be lacking and get behind. It requires a lot of strong commitment from all team members, it is mostly suited for small teams.

2.1.4 Spiral Methodology

Spiral method is another method used in software engineering. GeeksforGeeks adds:

> Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. (GeeksforGeeks, 2022, para. 1).

It is mostly used for risk management with parts of the waterfall method implemented. The process of the spiral method is, identify objectives, develop and test, review and evaluate, perform risk analysis. It is often used in large, expensive and complicated projects. The benefits, it is very flexible and can easily adapt items added later on. The main focus is risk handling so the process of this handled well. Customer satisfaction is a big benefit since the spiral method can handle customers feedback and easily implement them. The downsides are that it is very expensive and not suitable for small projects, it is very dependent on risk analysis. It is very complex compared to other methods. According to TechTarget the downside of Spiral is:

> Hard to manage time - Going into the project, the number of required phases is often unknown, making time management almost impossible. Therefore, there is always a risk for falling behind schedule or going over budget. (TechTarget, 2019, para. 16).

2.2 Software Development Method Applied

The method used for this thesis is the waterfall method. The waterfall method is a simple method used for work that needs to be quickly pushed forward. However, this might not be the best method overall and for the future of this game, the method would have to be adjusted to the Agile one, since that one is more flexible and better to use. But for this thesis, the waterfall method was the best choice. To reference a site speaking about the waterfall method:

> Waterfall is the oldest methodology on the block. It substantially simplifies the software engineering process into a linear process diagram, where the completion of the previous task is necessary for the engineer to be able to move onto the next one. (Guthrie, 2020, para. 2).

The waterfall method was used because it is a linear, up-front method with requirements fully defined before a project starts. Each step is taken in order and completed before proceeding with the next part. For this project that was the best approach because each step was completed before the other and it was the fastest to use when it comes to methods. The waterfall method is straight forward, and the steps are easy to follow because everything is already planned out. According to the Australian Institute of Project Management:

> Unlike Agile, this more linear approach often means that team members only need to be available for their specific project phases and can thus continue to focus in other areas. Equally a project's customers may only need to be involved heavily in the early initial scoping phase and then at delivery. (AIPM,2021, para. 6).

Compared to Agile, even though Agile might seem more flexible and better to use, "For teams new to working in an Agile way, there is a risk that people can feel unsure of what they should focus on when the scope is developing." (AIPM, 2021, para. 5).

2.2.1 Application

The steps this method includes are:

- Requirement Analysis
- System design
- Implementation
- Testing
- Deployment
- Maintenance

Each step of the method was planned out in advance and made in order. The requirements analysis phase was to prepare for the project and to make a specified list of all requirements and rank them to achieve more efficient time planning. The system design includes the building of the game´s architecture, product design modules, and interfaces. The implementation was building the game itself and implementing all the features. Some of the tools used for the implementation were GameMaker Studio 2 with all the tools and functions that the engine has, as well as, PixelOver and Blender which are two programs that help implement art into pixel art. The testing phase was to make sure everything works and that the game is playable. This will be done by playing the game and running through

all the different pathways to make sure it all works. There is also a beta tester who will test out the game. Deployment was the result and the finalizing of the game itself.

3 PROCESS

The process of this project follows the waterfall method. It goes through different stages to complete the project and get a final result. The process of this thesis goes through four significant steps before we come to the conclusion of it all. The process of the development of the game and all the steps are presented in this chapter.

3.1 Requirement Analysis

To start this project, and to follow the waterfall method the project needed to go through a requirement analysis. According to an analysis on requirement engineering, the definition is:

> Requirements engineering is a collection of different phases, processes and methodologies, which seek to take in information from a variety of sources and transform it into concrete requirements. (Lehtonen et al., 2020)

The requirement analysis begins with planning and defining the specifications that the project needs to meet. This was made by taking a few steps, shown in appendix G, to fully plan out the project and prepare the next step of the method.

The first step was to see who this project is for, who is the key audience and who will play the game. The key audience to this project is novice game developers who want to try game development with a low threshold, people interested in the game building environment and interested in how GameMaker Studio 2 works. The audience is also game development companies and other parties in the same field.

The next step was to gather the requirements. This was made by making a list of different requirements for the game. The main three requirements are story, puzzle, and AI. The requirements for the story were that there needed to be a story that the player could follow and that was interesting. The requirements for the puzzle were that there needed to be at least two puzzles in the game and that they should not be the same. The requirements for the AI were that there needed to be three different types of AI and that they should function properly. Other requirements were the assets of the world, such as background, forest

environment, desert environment and villages, areas to move to, textboxes for the story, collision boxes, and characters as well as coding the functions and animations. The requirements for the animation were that the player character should be able to move around using the W, A, S, and D keyboard keys and that the character should be able to shoot with the mouse button. The requirements for the enemies were that they should be able to move around randomly.

The next step was to analyse the requirements, determine which requirements are achievable as well as prioritizing the requirements, and make a list of which ones are the most critical to have and which ones are just a nice implementation. Most of the requirements were within the scope of things achievable but some were higher prioritized than others. The AI aspect was given the highest priority, over the story and puzzles. There needed to be at least three different types of AI so that was the highest on the list. The next one would be the puzzles, there needed to be at least two puzzles implemented into the game. The third one, on the most critical list, was the story. Most of everything else was just nice-to-have implementations. The fourth item on the list would be the environment, because of the way the game is built there needed to be an environment for the player to play in. Other quite important things were animation of movement, player interaction objects, like walls or boxes and different areas to move to.

It would not be feasible to implement specific details in the different environments and areas since this is a prototype, it would only be a nice-to-have implementation. Other items that would not be needed would be more than three AIs, more than two puzzles, extra movement for the characters, pleasing to the eye assets, and extra non-player characters.

The last step is to make sure that there are no major problems with the requirements. In this case, the only problem would be time management because there are quite a lot of requirements to go through for this project. To be able to get all the requirements and for them to work requires a lot of management.

After analysing and making sure that all the important requirements are feasible it was time to move into the next step of the waterfall method, the design.

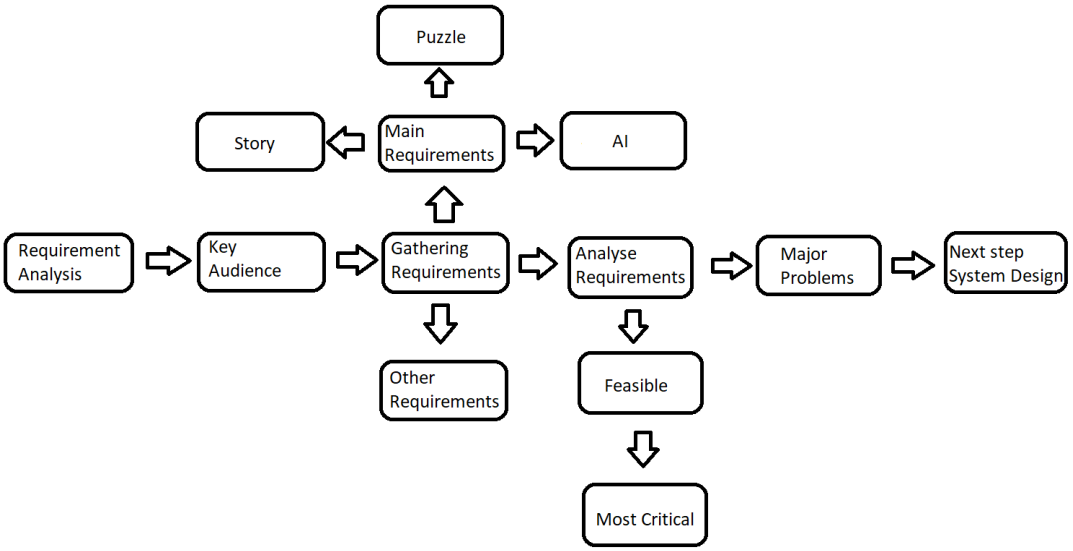

Figure 1. Requirement Analysis Flowchart

3.2 System Design

The plan was to make a story-based game in GameMaker Studio 2 with the implementation of AI and puzzles. The plan was to focus more on the AI and implementation of self-moving AI in three different ways, than the other functions. The focus was also to have a particularly good story to go with the AI as well as a few puzzles to make the game world feel complete. The plan for the game itself was to have a total of ten different instances or rooms that the player would go to and that each room had a different focus. The plans for the rooms were:

1. Title screen
2. Intro text, rolling up like Star Wars
3. Woods
4. Pathway – like the game Undertale
5. Crossroad
6. Desert
7. Forest
8. Dungeon
9. Castle Garden // meadows
10. Castle entrance
11. Castle

Each room in the list was supposed to have a specific function and was supposed to be different from the next so that there was a vast variety of content displayed in each room. There still had to be something connecting them together but overall, they were supposed to be different from each other.

3.2.1 Story Aspect

The story was implemented to support the puzzle and AI aspect of the game and to make it more fun to play as well as see what a player would do with the certain choices, they had to make. The story was divided into two and depending on the player´s choice the outcome was different. The story is an adventure-puzzle story where both come into play to complement each other. The story has both mystery and adventure in it as well as random fun but also parts where the player has to imagine things for themselves to make it more exciting. The story can be found in Appendix F.

3.2.2 Puzzle Aspect

Puzzles were implemented to make the game more fun and challenging. There are only two big puzzles added to the game. One is a box pushing puzzle to see if the player can figure out that when they push a box onto a button the button then spawns another button that the player must push. To get out of the room the player must get the door to spawn by pushing all the boxes in a certain order. The second puzzle is a small maze with spikes that the player must navigate to get out of while small slime enemies run after the player. If the maze does not kill the player, the slime might.

3.2.3 AI Aspect

The AI was the biggest part of the project. Learning how to implement simple AI into a GameMaker program and how to make it more advanced, making it so that the non-player characters move on their own randomly, and even implementing learning AI, was the most challenging. Learning AI, according to IBM cloud education “is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.” (IBM, 2020, para. 2).

The plan was to implement three different types of AI into the game. There is a horde AI that follows the player wherever they go, an Undertale-inspired AI that puts the player in another room to fight a bigger boss and the third one is an area AI enemy, where if the player gets into a certain area they come and attack the player. There is also random AI moving around the game environment.

3.2.4 Game Breakdown

The game is a 2D top-down game where the player starts out as a cat character lost in the woods. The first room is just a simple forest room with a couple of AI slimes just moving around randomly without any feature. The goal is to try to figure out why we are in the forest and what happened to us to end up here. The game starts with a black screen that has a scrolling text that makes the story begin.

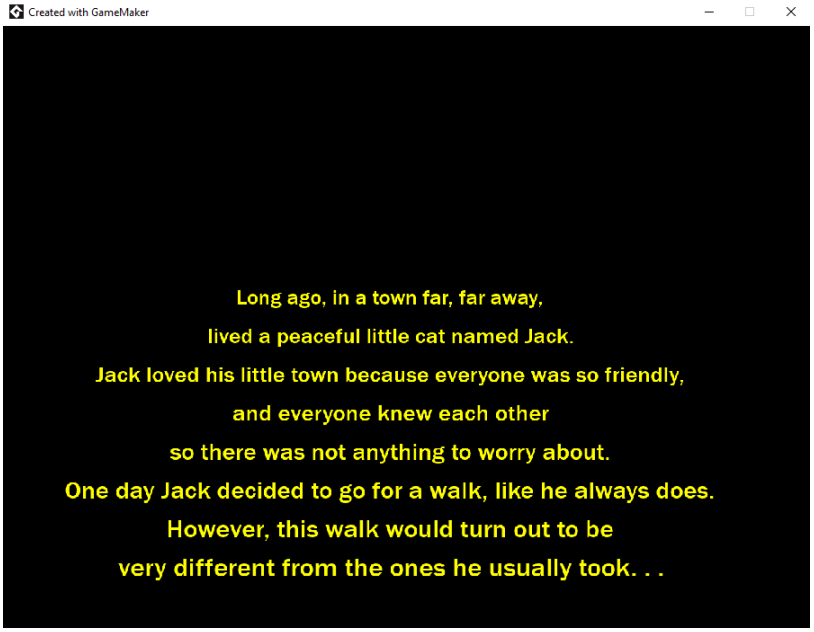

Figure 2. Intro Scene

Then we move on to the next room which is the forest. Pressing the keyboard keys W, A, S, and D makes the cat character move, and we can run around the whole map apart from where there are walls. There is also teleportation to the next room which is invisible but if we just follow the path we end up in the next room.

Figure 3. First Room

There is also a health-bar shown in the figure above, which will come into play later in the game. The game then continues to the next room where there is a sign. Here is where the story aspect really comes into play. The story is that we are lost in the forest and now need to find our way out. The sign tells us that we have two pathways to choose from and that we can either go north or south but we will not be able to return. However, because this game is a prototype, the south room has not been made so instead there is a blockade. The north room has a chest that now tells us that we can use the left mouse button to shoot. If we walk forward, we come into the first part of the AI-implementation. This is a moving AI boss that shoots at the player, and it is an Undertale inspired fight.

Figure 4. Pathway and Pathway blocked

Figure 5. Chest

After the player defeats the boss, a portal spawns and the player can leave. The player can then continue forward to the next room which is a puzzle room. To get out of the puzzle room the player must push four boxes over four buttons. After the player completes this puzzle, the player comes to a village. The village is quite underdeveloped because the game is a prototype, but when the player comes to the last house, the player falls down a portal which takes the player to a desert dungeon. This dungeon is filled with enemies and puzzles to get through. First, the player comes across a horde of slime that will follow the player wherever the player goes, and if the player cannot defeat them, the player dies. After the hoard, there is a small spike puzzle, where the player must navigate through a

small maze to get out of the spikes. When the player has gotten past that, there is a big slime boss that spawns little slime, the player must defeat this boss AI to be able to continue the game. The boss drops a key that unlocks a chest. After the player defeats the boss, the player walks into an area with a chest that is guarded by four guards and that is triggered by how close the player is to them. The player must defeat them to get to the chest and that is where the big prize is. The player gets to return home, or so the player thinks. It spawns the player back at the beginning of the game.

Figure 6. Village

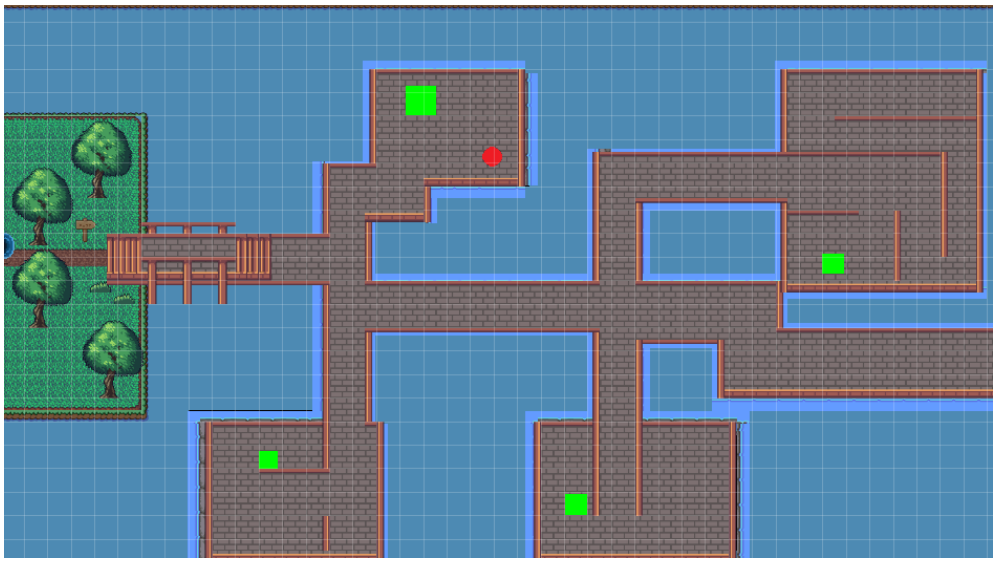

Figure 7. Puzzle Room

Figure 8. AI slime and AI slime boss

Figure 9. End chest and AI guards

3.3 Implementation

This game is built in the GameMaker Studio 2 environment. Players move around a small character by pressing the W, A, S, D, on the keyboard. and the goal is to get out of the forest and get back home. The construction of the game is as follows,

- Rooms to place the character on
- Background

- Character
- AI – Enemies and Characters
- Story
- Visual effects
- Movement
- Assets for the environment
- Assets for items
- Puzzles
- Collision boxes
- Room transition

Other implementations made to this environment include adding it all to the GameMaker environment using the assets provided by GameMaker Studio or downloaded through free for commercial use products.

3.3.1 Assets and Coding

The assets used in this game are freely downloaded content from different websites that allow the user to use the images and backgrounds for commercial use. The assets were downloaded and implemented into the game by adding a sprite, an image that represents game assets, to GameMaker Studio and then making the sprite into a tileset or an object. The tilesets are simple to add as background or flooring while the objects can be programmed and that is how you make the different enemies and allies interact with each other.

To start coding, a base is needed. The base of this game is heavily focused on assets obtained from different sources. To implement them there needed to be some research and analysis of data to figure out how to implement the assets in the right way. After adding the assets to the engine there needed to be a character for the player to use. That was a hassle because adding the picture itself was not easy and then getting it to animate was another problem. The issue was that there was a picture of multiple cats, which could not be implemented as one. The solution was to import the image, not as an image but as a strip image to then edit the boxes to make them into separate images. See the example below.

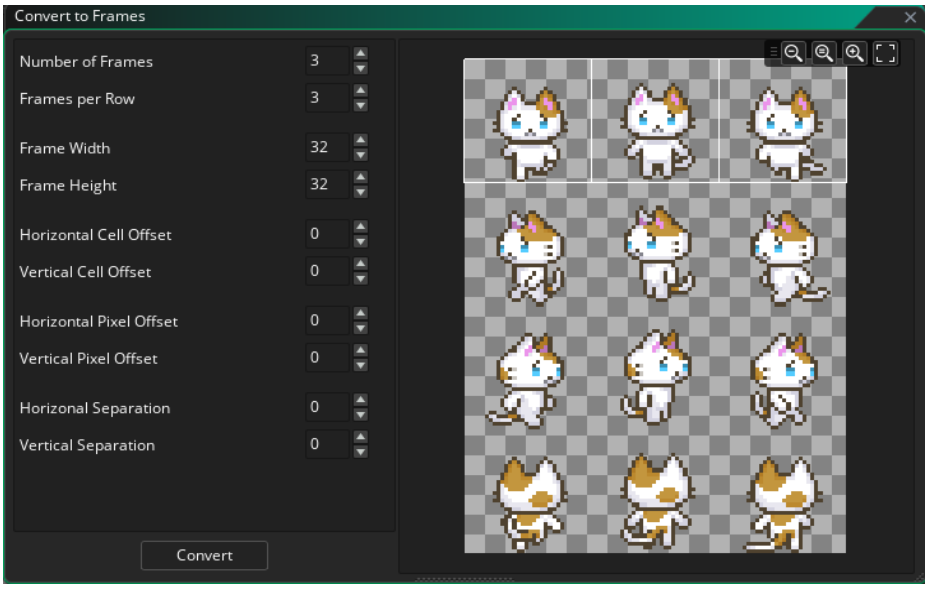

Figure 10. Image getting converted

3.3.2 Movement and Collisions

Collision boxes are the walls and all the objects that the player runs into and cannot get through. To make the collision box, a simple box asset or square was added features added to it. There is a feature that GameMaker uses called collision, so all that needed to be done was to add and see if the box collides with the player. The box is solid, which means that the player cannot pass through it.

3.3.3 Cameras and Display

The next step was to set up the camera to follow the player. To set this up a simple line of code was added to the player and to the room. The viewport needs to be active, and the width and the height of the room need to be set. When set up correctly, the camera will follow the player wherever the player goes.

The player code looks like:

```
halfViewWidth = camera_get_view_width(view_camera[0]) / 2;
halfViewHeight = camera_get_view_height(view_camera[0]) /2;
camera_set_view_pos(view_camera[0], x - halfViewWidth, y - halfViewHeight);
```

Listing 1. Player Code

After setting up the viewport the effect of when the player walks behind trees, they go invisible or fades out was added. To do this, the image alpha was lowered, making the trees transparent. GameMaker has an `image_alpa` feature that was used. Before the player goes behind a tree the `image_alpha` is equal to 1, when the player goes behind the tree it is set to 0.25. After the player leaves the tree, the developer has to set it back to 1 or the tree will remain transparent.

A room was added, named first invisible, this was a simple room with no images, just code so that when the game launches, everything would run more smoothly. The `R` button as a restart point was added so that if anything happens, such as the player getting stuck or dying, the player can just go back to the beginning of the game. An escape button was added, so if the player clicks `escape` it exits the whole game. This was simply added by if keyboard check is `R` game restart and if keyboard check is `escape`, game end.

3.3.4 Textbox

The textbox was one of the biggest and hardest parts that were added to this project. Figuring out how to implement the textbox correctly was challenging and time-consuming. It uses multiple image assets and objects to make it look a certain way. Four image sprites were made, a frame of the picture, the picture, a namebox, and a textbox to build the textbox itself. An invisible textbox object was made where the text itself would appear.

A draw event was created to draw the box on the screen, and to make it appear the feature `if place_meeting`, was added. `place_meeting` is another function to use for the developer to check if the object is colliding with another object, in this case, the player. To add the box on the screen, `if place_meeting` was implemented to see if the collision is with the player. If true, spawn the textbox or `instance_create` the textbox and then if the player moves away from the textbox, destroy the textbox.

The textbox also has the feature of moving to the next page, it has not been implemented into the game on the textbox itself. The button to click next page, was made for the simple fact that a lot of games use this button to click to the next page, and that is the keyboard button E. More about the code and the function of the textbox is shown in Appendix A-C.

```
text[0] = "Choose your path";
text[1] = "You can either take the road North to Greenville ";
```

```
text[2] = "Or you can take the road South to Sandville.";
text[3] = "However, whichever path you take, you cannot go back,";
text[4] = "for a curse lies upon this road,";
text[5] = "and whatever you choose, you must stick with. ";
```

Listing 2. Textbox - Choose your path

3.3.5 Health System and Attacking

There was a need for a health system for the player and the non-player characters. Because there is fighting in the game there needed to be some indication of health. The first health bar that was made looked very poorly made and I found that GameMaker Studio has a built-in function for drawing a health meter. The health meter is a visual indicator of the value of a character's attribute in the game that keeps track of how many more hits or collisions a character can take before being removed from the game. The way it is built is that when a player collides with an enemy or boss, the health of the player goes down. A variable was added called `health_bar` that goes down one step for as many times as an enemy touches the player. The code for the health system is shown in appendix H.

A bullet was made that has the same mechanism, but every time that touches the player, the health goes down by 0.1. It only goes down very little for testing purposes. To be able to make the health bar work it was converted into a percentage and the way to do that was to take the player's health divided by max health times 100. If the health reaches 0, the instance is destroyed, meaning the player object is destroyed from the world. There was a significant issue where all the non-player characters' health w initially tied together and not separated upon collision. The cause of the issue was that the child object was shared across all parent instances. For example, if the spawn is three slime instances, the slime object has instance IDs for all three slime instances, but only one instance ID for the child object `(npc_health)`. The issue was fixed by removing the `npc_health` object and putting the code in the main slime object instead. The drawing of the health was created using GameMaker's built-in function `draw`. The function `draw_healthbar` is used here and the parameters it takes are just placing it on the *x* and *y*-axis and how wide it is supposed to be as well as coloring it. The developer can mix and match how the developer wants it to be because all the parameters it takes are whatever is set.

The main attack for the player is the left mouse click which gives out a meow shout attack. The way this was made was to use a sprite index function that calls the bullet, the meow,

and then use `instance create` to display it on the screen. A point direction was used so that the bullet spawns and shoots in the direction the mouse was pointing. The bullet had to be a bit different depending on the rooms because the Undertale room was different than the rest. If the player shoots in the normal, non-boss, world, the range is much smaller than that of the boss room, because the boss is further away from the player. To make the bullet work the transparency function `image_alpha` of the engine, was used. `Image_alpha` has a range of 0 to 1, 0 is completely transparent or invisible and 1 is completely opaque, solid. To make the bullet shoot but also slowly disappear the `image_alpha` was used and set to `image_alpha - = 1 / (room_speed * 1)` for the normal rooms and `room_speed * 10` for the Undertale battle. `Room_speed * 10` makes the bullet have a bit longer range, meaning, when the bullet collides with an enemy their hp goes down. The code for enemy taking damage was made by doing `hp -= 0.1`, which would make them take 0.1 damage from each bullet. The bullet and the attacking functions are shown in appendix I.

3.3.6 Weapons and Art

The Meow attack was made using plain text from a text editor. In addition to the meow attack, a bow was created and a sword which was supposed to be used in the story, however because of the time limit they were not implemented how they should have been implemented. There is however a trick to getting them to spawn. In the desert room, there is a hidden secret button that if the player steps on it, the chest with the weapons spawns and the player can choose either the sword or the bow to spawn from the chest to then pick up. This is a secret of the game that was implemented last second as a fun easter egg.

The sword is a simple sprite that was implemented into the game. The bow was made in a program called Blender. It was animated with the help of a program called PixelOver and imported over into GameMaker Studio.

Figure 11. Bow and Sword

3.3.7 Animation

The animation of the characters was made by importing image strips to the engine and then using the built-in feature to make the images play in a loop to look like they are moving. After creating the image loop, the sprite index function was used to show the image on the screen. For example, when the player is moving up and down it calls the function `sprite index` cat walking left if the left button is clicked, in this case, the A key on the keyboard. The same function is used for the rest of the W, A, S, and D keys. If any of the keys are pressed the `sprite index` is changed to the direction, they are pressing, and the image speed is slowed down. The image speed is originally quite fast, so to slow it down, the code `walkspeed` was used and set to 2.5, divided by 3. Slowing down the image made it run smoothly. Then if the player presses no key, the character stands still. The code for the player's movements shows in Appendix D.

3.3.8 Intro Screen

The intro was at first a bit tricky to make, it was quite easy to get the text to pop up on the screen but to make it scroll was a different story. Variables were made for the movement of the text, variable `i` for how fast the text should go up, `s` for the scaling of how the text will go from big to smaller when it moves upwards. Finally, there was added a key check, so that if the player does not want to watch the intro again, they can just press spacebar to make it stop. A draw event was made so that everything would render on-

screen and a for loop was added to go through each line of the text. Finally, a special font was made in the built-in fonts and set to a certain size that would display on the screen.

3.3.9 Bosses and AI

There are two large bosses in the game. The first boss the player runs into is the bee boss set in an Undertale setting. This boss is made from a new sprite and made quite simple. It has health and spawns in a new room set as Undertale boss room. It has a few triggers that happens when the player enters the room. Bullets shoot out from the boss when they come in contact with the player it also has two triggers that if the player is near and the boss health points is 65 or lower, it starts shooting double bullets. Another trigger happens when it has 35 or lower health, then it starts shooting triple bullets. More about the boss's movements can be found in Appendix E.

A health bar was added to the boss, to keep track of the health, and a pathway. There is a feature in GameMaker Studio that is called animation curves, where the developer can build a pathway for the player or non-player character to walk. That was then added to the bee boss so that it moves in an upside-down U shape. After the player defeats the boss the instance in which the boss was created gets destroyed and a portal spawns to let the player go.

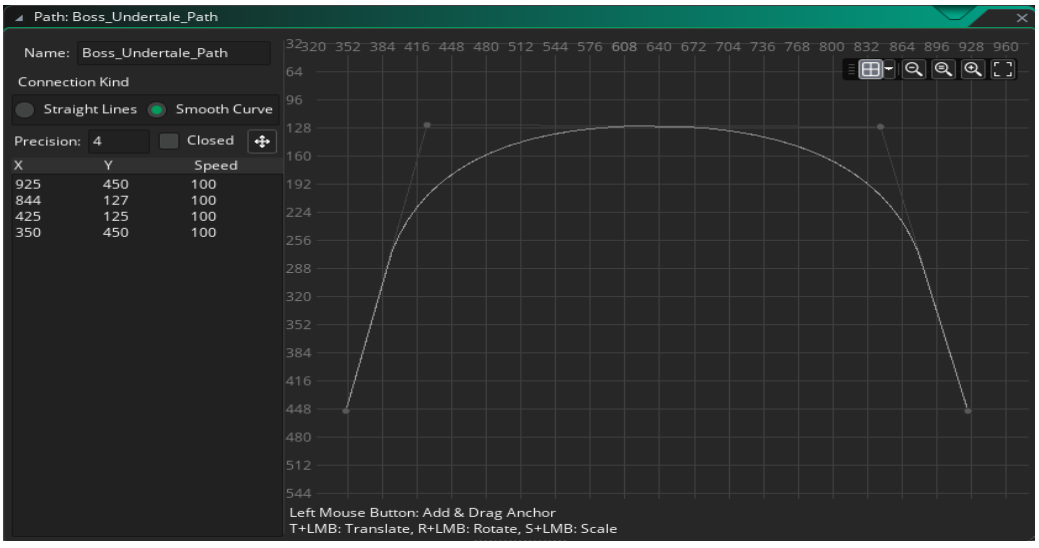

Figure 12. Boss bee movement curve

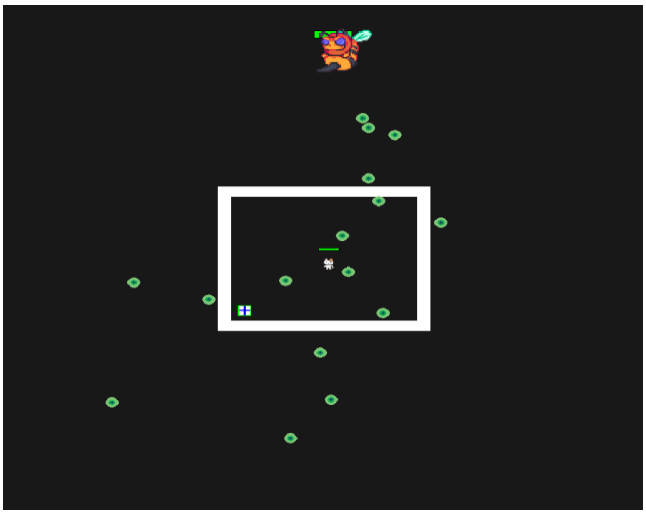

Figure 13.Undertale boss

The second boss is a slime boss that the player finds in the desert. This boss is made from the sprite of a slime and made bigger with some added features. If the boss is in radius of the player, it starts attacking the player. GameMaker Studio has a feature called point direction which allows the non-player character to look at the player or go in that direction. This boss has a lot of health in comparison to the players who caused the damage. The boss does not have an attack but spawns little baby slimes that attack for him. The spawn of the slimes is created by the `instance create layer` event so that every ten seconds a new slime spawns to attack the player if the player is within range. The other non-player characters return to their starting position after a while, but the boss slime keeps coming after the player until the player kills him. A feature was planned to be added where the slimes appear around the boss by using random range, random range returns a number within a specific range. By then adding it to the slime you could get them to move in whatever direction you wanted. For example, this code makes the slime move randomly around.

```
moveX = 0;
moveY = 0;
idle = choose(0,1);
if(idle == false){
  var dir = choose(1,2,3,4);
  switch(dir){
    case 1: moveX = -spd; break;
    case 2: moveX = spd; break;
    case 3: moveY = -spd; break;
    case 4: moveY = spd; break;
  }
}
alarm[1] = random_range(2.5, 4) * room_speed;
```

Listing 3. Random AI movement

However, the `random_range` function did not work properly, the only thing it did was spawn slimes under him instead.

The slimes were supposed to go after the player straight away, however, they do not have that function but act like the other normal slimes instead. When the boss dies, he drops a key that unlocks the final chest. This was easily made by using the code if the non-player character's health is zero then `Instance create` the key and `instance destroy boss.`

There are a couple of enemy AIS running around the game. The first one the player runs into is the boss bee, which was explained in a previous paragraph. The second one the player runs into are the slimes. The slimes have a hoard mentality so that when the player comes close to them, they attack the player. They take off one health for every frame for as long as they are near the player. They do not roam around but they remain inactive until the player comes in range.

A small circle was made to print out on screen for bug testing so to see how far the range goes. The slimes, in addition to previous mentioned features, have the feature that if the player is within range they come towards the player. There is, in addition to the range function, a function added to the slime, `irandom_range,` which makes the slimes not cluster together and not go straight for the player but sometimes drifts a little off the player.

The other enemy AI is the guards. They are like the slimes, but their range is smaller, and they are supposed to wander around guarding the chest. They are supposed to have a smaller radius than the slime and only attack when they are within the radius. Since they are guards, they should have a weapon they attack with which is different from the slimes who only attack with their bites and bodies. However, because of time limit some of these features were not implemented, such as the weapons for the soldiers, so the acting of the guards is like the slimes.

The addition of a random wandering AI was necessary for this project. The plan was to add more than just a few but the ones added were some simple slime AIs that move randomly around the world. The implementation of the slime AI was done with the help of a simple tutorial online. (FriendlyCosmonaut, 2017). To make the slime a speed and a move function was needed, as well as a collision function to see if they would switch directions if they collided with any walls. Then to create all these functions, an alarm

function had to be called that sets off the motion so that after a certain amount of time they start moving in a different direction. The functions made it so that if the slimes either collided with a wall or were just moving around, a time limit was set on their direction, so that they started moving in a different direction after a certain time.

3.3.10 Puzzle Room

The puzzle room was quite interesting to make. The first part was to set up the rooms so they would be easy to get to. The boxes were made and the buttons which had the code that if they collide with each other then it goes to the next step. The buttons were made so that if the box collides with a button, the next button will spawn or `instance_create`. However, the issue was that it started spawning thousands of instances instead of one. The way the issue was solved was to check if the button instance already existed, if it did not exist, then spawn the next button in the order. By adding the instance check all the buttons spawned in order and finally the portal to exit the puzzle room. The boxes were made bright green and the buttons red for bug testing. The buttons were made larger than the boxes to see if the box is on the button or not. The buttons all had a similar code to each other so that when box two hits button two the third button spawns. In this phase, the collision walls did not get implemented, even if the walls are showing, there is no collision for them in the puzzle room.

The hardest part of the puzzle room was to figure out how to push a box. Exploring the GameMaker forums (GameMaker Community, 2018). helped add a simple solution to the problem. To be able to push a box, there needed to be a check to see if the box collides with the player, and then as keyboard check is “direction” the box moves in that direction. Example:

```
if place_meeting(x-2, y, Cat_walking) and keyboard_check(ord("D")){
  if !place_meeting(x+4, y, puzzle_box1) x+=4;
}
```

Listing 4. Cat interacting with box

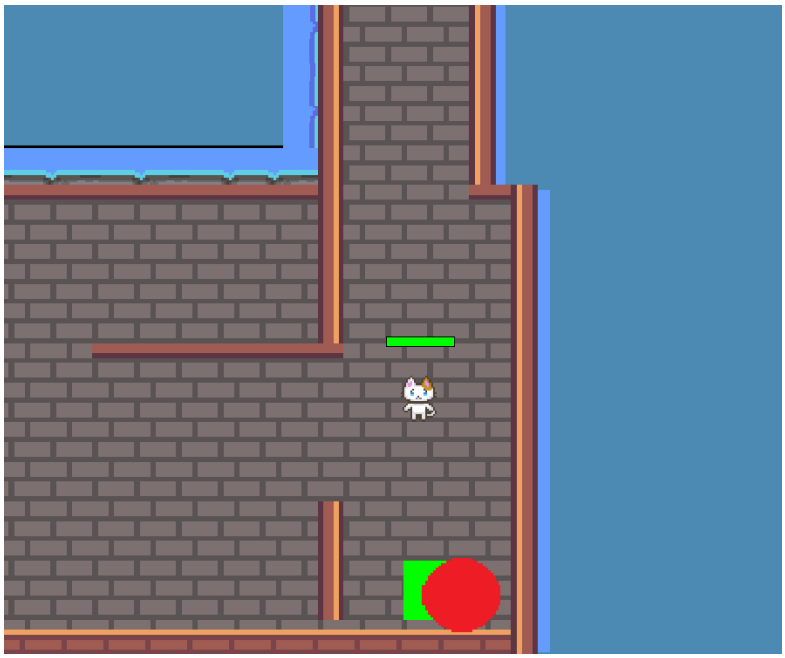

Figure 14. Puzzle Room

The box interacting with the button:

```
if place_meeting(x+2, y, puzzle_box1){
   if(mybox == noone){
      if(!instance_exists(puzzle_button2)){
         instance_create_layer(478,727,"Spawn",puzzle_button2);
      }
   }else{
      if(mybox != noone){
         instance_destroy(mybox);
         mybox = noone;
      }
   }
}
```

Listing 5. The box interacting with the button

3.4 Testing

The testing phase was to run it to see that everything works properly and that all the features were implemented. When starting the game, the title screen came up and text started scrolling up. After the text, coming into the world the character was on the screen and the player could move around using W, A, S, and D. However, there was no indication of what buttons to use to move around. Moving around the first room the player could not escape but only go through the door because of collision boxes or walls. To move to the next room there was a small transition to show that the move to the next room happened.

Moving up to the sign in the second room, made a small textbox appear. Going into the next room a chest opened up, textbox appeared again and then there was the first boss fight. The boss shot at the player and the player could shoot back. The chest before had a text where it told the player to use the mouse button to shoot. However, the boss's health was only at 1 so it died instantly. Moving on to the puzzle room, the puzzle room had no collision walls because making all the possible collision boxes was not important to this proof-of-concept project, it was not implemented, meaning, moving through it, was quite easy. Moving the boxes onto the button worked as intended and it opened the door to the village. When entering the village there was a number of houses and the last one made the player fall down a hole into the sandvillage. The sandvillage is filled with evil creatures and the player has to navigate through to get to the chest at the end. The walking slimes attack right away and then the player entered a maze while running from the slimes. After the maze was a large slime boss standing in the way. The slime boss takes a lot of hits before he dies because the weapon does not do a lot of damage. However, the slimes can kill the player quite easily, so the player has to watch out when attacking them. After killing the boss, which seemed like it took forever, there were four guards watching over a chest. The movements of the guards were quite stale, they only face one direction and do not rotate at all. The chest animation was lacking. When then finally killing everything and getting the chest the game ends.

During the testing phase there were a lot of problems that came up. The movements of the AI were not properly programmed. The transition between rooms were not working properly in the sense that the character should be standing still when they move through rooms, however the character could still move while switching rooms. There were quite a lot of collision boxes missing throughout the world that had not been put into place. Other smaller details like the bosses' health, the movement of the slimes, the popping of the textbox above the character and the indication of what keys to press were not implemented, that was noticed in testing. However, the key factors of the game still worked. The AI moved around like it should, the story was a bit short but there was still a story and there were two puzzles implemented.

The testing phase was in addition to the developer, done by a beta tester who noticed quite a lot of small details missing from the Sandville area, especially the movements of the slime, and that the spike puzzle was very easy to complete.

4 RESULT

This chapter goes into what goals were reached and not reached during this project. It gives the reader a walkthrough of what the developer has done and what conclusions the developer has come to regarding the project. There is an analysis of the results as well as reflections.

4.1 Goals Reached

The goals that were set for this thesis was to build multiple rooms with different types of mechanics. The mechanics were supposed to be three different types of AI with puzzles and a good story to build on the game. The implementation of AI was a success in the sense that there are a few different types of AI in the game. There is the free-roaming AI slime that walks around with no one guiding them or doing anything specific. There is a horde of slimes that attack the player when they get too close, there is the soldier or guards that guard a chest and when the player gets too close, they attack. The final AI enemy is the Undertale one that moves in a certain pattern and shoots bullets for the player to avoid.

The story was implemented in the sense that there is more of a story in the beginning and then the player has to figure things out on their own apart from a few tips and tricks here and there. The story has almost been implemented in the way that it was meant to apart from a few steps.

The puzzles were implemented into the game, as a prototype. There are a total of two big puzzles the player has to get through to go to the final part of the game. The first one is a box pushing puzzle where the player has to think for themselves to figure out how to get out of it. The second puzzle is a small maze, quite easy to get out of, however the player still has to navigate the maze on their own.

Another goal reached was the implementation and importing of the player character and enemy AI. The goal to get the enemies to move around the game and attack the player was reached. The goal for the player character to attack the enemies and move around the game was reached as well.

The goal “give the reader an insight in planning, development, production and result” has been reached. The thesis has gone through the process of planning, the development of the game, the implementation and the results.

4.2 Goals Not Reached

To start off, not all rooms were made that were planned to be made. The plan was to make a total of ten rooms, at least nine. That did not happen because the time planning was not going according to schedule so had to be cut short. There were supposed to be two path-ways and the game ended up with only one. The total of rooms came to about seven, including the Undertale boss room. The rooms that are missing are the pathway to the desert room, a dungeon room, and a meadows room where the player can see the castle and the castle itself.

The story got cut short in the sense that there was supposed to be way more story and more puzzles added to the game. However, because of the time limit that had to be cut short.

The mechanics of the player got cut. The player was supposed to be able to pick up a weapon of choice, either a bow or sword depending on which direction they picked at the beginning. That would then have been used in the fighting scene. Later on, that would be discarded, and the meow attack implemented in the dungeon. Because adding the choice feature took so long, the decision to remove the weapons all together and just add the meow attack was the best approach.

The AI was supposed to be of more variety. They were supposed to walk around and guard and then if the player comes in range only one attack instead of many. That could be easily implemented by moving them a bit further away from each other, but the random walking was not implemented for the guard AI.

The plan was to make the story well rounded so that even if the game is not completed the story would still make sense and still have a point. This story is short and do not live up to those expectations. There were a lot of small details that were supposed to exist in the game that did not and that all came down to planning and time management as well as time limit.

4.3 Critical Analysis

The project all in all is good, however, there is always room for improvements. Because time management was very poor, the game was also lacking in many aspects. Much more of the planned content would have been able to be implemented if the preparations were a little bit better for the project. The structure of the game is still good, and the added assets are interesting and fun to look at. However, important information is still missing for anyone to be able to play the game. For example, the information about what keys to press to move the character around was missing. The project otherwise, reached the goals it had set out to reach.

5 CONCLUSION

When making a game, one must think about a lot of different things to make it happen. Time management and workflow are two major things that come into play. Assets and development in general have a lot of different points that one must think about when making a game.

Time limit was one of the biggest issues during the game-making process. One has to make a plan and set up milestones along the way when making a project this big to even get close to the end goal. A lot of milestones and steps for each day were added when making and planning the game. However, you never anticipate what problems you will run into and how long certain parts will take. Some can take a few minutes while others can take hours. Something that was noticed was that there were times where the plan for a certain aspect was set but, in the end, took much longer than planned.

The method used was not the best. It was very stiff, and you cannot change much about it. The biggest issue with it was that you have to complete every step to be able to go to the next one and one cannot for example jump back and forth into the testing phase. For future work and projects, the Agile or Scrum method would be a better fit for game development.

GameMaker Studio 2 is a very simple engine to use while making games. It can help to implement pretty much any 2D game idea. The only problem I had was to figure out how to actually implement it. Even though there is a lot of tutorials and documents out there on how to build a simple game with the engine, there was always the issue of that there

was not enough information or documentation to be able to follow. Instead, the best solution was to try and take time, and process the information and figure it out by oneself. There is a still a lot of information and documentation missing when building a game in GameMaker Studio 2 experimenting and information seeking is key.

6 SVENSK SAMMANFATTNING

6.1 Introduktion

Detta arbete handlar om utvecklingen av videospel i spelmotorn GameMaker Studio 2. Målet är att göra ett spel som innehåller olika funktioner. Detta spel skall innehålla berättelse, AI fiender samt olika sorters pussel. Syftet med detta arbete är att beskriva och analysera utvecklingen av ett 2D-spel och gå in på djupet av hur ett spel kan utvecklas i GameMaker Studio 2 miljön samt vilka problem utvecklaren kan stöta på. Syftet är också att följa med spelutvecklarens olika steg som utvecklaren måste ta för att göra ett spel.

6.2 Verktyg och mjukvara

De verktyg och program som använts för detta arbete är GameMaker Studio 2 men också andra program och verktyg som hjälper till att skapa ett spel. Andra verktyg som använts är, olika programmeringsspråk så som C#, C++, GML. Annat som har använts är olika speltillgångar och konstverktyg så som Blender och PixelOver.

6.3 Metoder

Detta arbete går in på olika metoder som finns inom programvaruutvecklingen och ser på deras positiva samt negativa egenskaper. Projektet beskriver vilken sorts metod som använts för detta projekt, det vill säga, vattenfallsmetoden. Vattenfallsmetoden är bra för små projekt där man önskar planera hela projektet i förväg och sedan gå igenom samtliga skeden en efter en. Det passar bra för en nybörjare eller för ett litet team. Andra metoder som nämns är, Agile, en väldigt omtyckt metod och en av de bättre metoder när det kommer till projekt, Scrum, en metod som kommer från Agile som är ännu mera flexibel än Agile och Spiral en av de svårare metoderna som passar bäst för ett litet team eftersom det kan vara relativt komplext.

6.4 Process

Processen i detta arbete går genom några olika skeden. Kravanalys, Design, Genomförande, Testande och Slutförande, Resultat. Dessa olika skeden måste klargöras innan den nästa kan påbörjas.

6.4.1 Kravanalys

Processen med kravanalys är att gå igenom de olika kraven för arbetet. Att analysera och gå igenom vad som behöver göras och vad som inte är så viktigt och komma fram till alla krav som detta arbete har. I detta fall så är det stort krav på berättelse, AI och pussel. Att kartlägga och planera är en stor del av förberedelserna för att göra ett spel så planen var att lägga upp ett spel med många olika rum som spelaren kan gå igenom och som har de olika kraven.

6.4.2 Design

Designen går igenom förberedelse och planering för hur spelet kommer att se ut, vad spelet skall innehålla och hur det kommer att vara uppbyggt. För detta spel så var planen att det skulle finnas ett flertal olika rum som spelaren skulle gå igenom. Det skulle finnas en titelskärm som berättar vad spelaren skall göra. Det skulle finnas ett Intro, som har en text som scrollar upp som berättar hur allting börjar. Ett rum var spelaren börjar, som är en skog. Sedan skall spelaren gå vidare och komma till ett annat rum med en skylt som berättar åt vilket håll spelaren kan gå. Det skulle finnas två olika vägar att ta, antingen en väg som gå mot skogen eller en väg som går mot öknen. Sedan skulle det finnas en fängelsehåla som spelaren skulle falla ner i. Till slut skulle spelaren komma till en äng som det skulle finnas vakter på och i bakgrunden ett slott. I detta slott skulle slutligen den stora onda bossen finnas. Med detta skulle alla olika aspekter av spelet bli fullgjort då det skulle finnas en bred berättelse att följa och varje rum skulle ha egna funktioner och saker att göra.

6.4.3 Genomförande

Spelet är skapat i GameMaker Studio 2. För att genomföra detta spel måste olika saker implementeras samt göras för att få ett resultat. Olika speltillbehör importerades in i spelet

för att få en bakgrund och en spelplan att börja med. Sedan att börja programmera in alla olika funktioner så att spelet löper smidigt. Första objektet att implementera var spelkaraktären. Spelkaraktären i detta spel är en liten kattfigur som flyttar sig upp, ner, vänster, höger. Att få animationen att fungera tog sin tid men problemet löste sig till slut. Nästa steg var att få världen att ha kollisions väggar. Alltså att när man går mot en vägg så stoppar väggen dig och du kan inte gå rakt igenom den. Detta gjordes genom att använda en bild och sedan göra det till ett objekt. Efter det så var det bara att lägga till att när spelaren går mot detta objekt, eller rör vid detta objekt så blir det solid, så att karaktären inte kan gå igenom objektet. Nästa steg var kamera, att få kameran att följa spelaren. Detta gjordes genom en enkel kod som bara programmerades på spelaren och då följer kameran spelaren.

Text boxen var en relativt stor implementation som lades till för att berättelse aspekten skulle löpa smidigt. Den är gjord med många små och olika text boxar för att tillsammans bilda en stor text box. För att sen kunna skapa puzzel och AI och också fiende AI, lades det till en stång för hälsa som indikerar när spelaren eller en fiende blir skadad. Detta ger oss också följande punkt, attackering. Attackering samt vapen behövdes läggas till för att stången för hälsa skulle komma till användning.

AI gjordes i olika former, en AI som bara slumpmässigt går runt och inte gör något speciellt, en fiende AI som attackerar när spelaren kommer inom en viss gräns och en AI flock, alltså en mäng AI som börjar komma emot spelaren genast då de ser spelaren. Detta implementerades till en del för att försöka se hur AI fungerar i GameMaker Studio 2.

Nästa steg var att implementera puzzel. Puzzel var den tredje och sista stora delen som behövdes implementeras för att spelet skulle vara nära sitt mål. Ett rum gjordes till ett puzzelrum. Detta rum innehåller en låda som spelaren skall skuffa runt och skuffa den på en knapp som sedan skapar en ny knapp som spelaren måste skuffa en låda på. När alla knappar har en låda öppnas en dörr.

6.4.4 Testande

Testandet av spelet gick ut på att spela spelet från början till slut några gånger och se vad för olika problem eller ting man som spelare råkade ut för. Eftersom detta är en prototyp

är det omöjligt att ha ett felfritt spel, men uppnår detta spel de målsättningar som gjorts på förhand?

Enligt beta testaren som testat spelet samt när vi testade spelet själva, märkte vi att spelet har en hel del små detaljer som skulle kunna finslipas innan den visas för någon som en prototyp men att det ändå går att spela som ett enkelt litet spel. En av de stora detaljerna var att det inte fanns väggar överallt samt att en del fiender var för lätta att döda och sedan andra var för svåra att döda.

6.5 Resultat och slutförande

Slutförande var att framställa resultaten och se vad som skulle ha kunnat göras bättre samt vad som gått väldigt bra, samt se de olika resultaten, vad som fullgjorts samt vad som inte riktigt blivit klart.

Målet med detta projekt var att bygga olika rum med olika mekanismer. Det skulle finnas tre olika sorters AI, det skulle finnas puzzel och en bra berättelse. Implementationen av AI var ett mål som fullgjordes. Det finns tre olika sorters AI i spelet, även om två är väldigt lika varandra. De olika AI som var implementerade var, random AI, slime som går runt världen i en "random motion", AI som attackerar när spelaren kommer i närheten och AI som är som en "horde" när de ser spelaren så rusar de emot spelaren.

Berättelsen var implementerad, men inte helt och hållet. Det fattas många delar som var med i planen men som inte blev implementerat i slutprodukten. Pussel implementerades så som de skulle vara. Det var planerat att det skulle finnas åtminstone två pussel och det blev byggt.

Mål som inte blev fullgjorda var, rummen som skulle implementeras, det skulle finnas många fler rum men det blev inte gjort. Berättelsen blev också kapad itu för att passa in med resten av spelet. Vissa mekanismer som spelaren skulle ha blev inte implementerade, spelaren skulle ha möjligheten att plocka upp vapen från kistor men det blev inte gjort. AI skulle också göra mer än de gör nu, de skulle gå runt och patrullera och bete sig mer som riktiga människor.

Kritisk analys, projektet är bra som det är men det skulle alltid kunna förbättras. Tidsplaneringen var väldigt dålig och på grund av detta så föll projektet lite i en grop. Mycket

mer av planeringen skulle ha kunnat implementerats om förberedelserna var lite bättre för projektet. Det som ändå är bra är hur spelet är uppbyggt och det är lätt att förstå och kunna spela spelet. Men det fattas viktig information för spelarens skull, till exempel vilka tangenter man skall trycka på för att flytta på karaktären. Allt som allt så är projektet ändå lyckat.

6.5.1 Slutsats

När man gör ett spel så måste man tänka på så många olika saker för att få det gjort. Tidshantering, planering och arbetsflöde är några viktiga saker att komma ihåg då man börjar med spelutveckling. Dessutom alla olika saker som måste implementeras och byggas upp för att få utvecklingen att löpa glöms lätt bort. Många milstolpar och steg för varje dag lades till för utvecklande av detta spel. Du förutsätter dock aldrig vilka problem du kommer att stöta på och hur lång tid vissa delar kommer att ta när man slutligen börjar.

GameMaker Studio 2 är en mycket enkel motor att använda när du gör spel. Det kan hjälpa dig att implementera i stort sett vilken 2D-idé du kan ha. GameMaker Studio 2 hjälper dig långt på vägen med hjälpmedel och information online för att bygga upp ditt spel. Men för att få den bästa produkten måste man ta hjälp av andra program och programmeringsspråk.

REFERENCES

AIPM (2021). Agile vs Waterfall: What´s the difference? [Online] Blog of Australian Institute of Project Management. Available at: https://www.aipm.com.au/blog/agile-vs-waterfall-whats-the-difference . [Accessed 28 Apr. 2022].

Ajmal, S. (2017). Pros & Cons of SCRUM Methodology [Online]. QuickStart. Available at: https://www.quickstart.com/blog/pros-and-cons-of-scrum-methodology/ [Accessed 28 Apr. 2022].

Alexander,M. (2021). Top 20 project management methodologies [Online]. CIO. Available at: https://www.cio.com/article/244577/top-project-management-methodologies.html [Accessed 28 Apr. 2022].

Blender (2019). Home of the Blender project - Free and Open 3D Creation Software [Online]. blender.org. Available at: https://www.blender.org/. [Accessed 24 Apr. 2022].

Creswell,J. (2021). Which Game Making Tool Is Right for You? [Online]. CBR. Available at: https://www.cbr.com/game-making-software-unity-source-game-maker-rpg-maker/ [Accessed 26 Apr. 2022].

Digite (2019). What Is Scrum Methodology? & Scrum Project Management [Online]. Available at: https://www.digite.com/agile/scrum-methodology/ [Accessed 28 Apr. 2022].

Farzan, M. (2019). I attempted to make the same 2D game prototype in React, Unity, Godot, Construct, Game Make and Phaser. Here´s what I found [Online]. FreeCodeCamp. Available at: https://www.freecodecamp.org/news/how-i-made-a-2d-prototype-in-different-game-engines/ [Accessed 26 Apr. 2022].

FriendlyCosmonaut (2022). FriendlyCosmonaut [Online]. Available at: https://www.youtube.com/channel/UCKCKHxkH8zqV9ltWZw0JFig [Accessed 24 Apr. 2022].

GameMaker (2018). GameMaker:Studio 1.2 – Shaders, YYC and More [Online]. GameMaker Help Center. Available at: https://help.yoyogames.com/hc/en-us/articles/216753668-GameMaker-Studio-1-2-Shaders-YYC-and-more [Accessed 24 Apr. 2022].

GameMaker (2022). Easily Make Video Games with GameMaker [Online]. Available at: https://gamemaker.io/en/gamemaker [Accessed 24 Apr. 2022].

GameMaker Community (n.d.). GameMaker - need help with pushing an object. [Online]. Available at: https://forum.yoyogames.com/index.php?threads/need-help-with-pushing-an-object.49791/ [Accessed 4 May 2022].

Gardiner, B. (2015). GameMaker Cookbook, Packt Publishing Limited, Birmingham, United Kingdom. Available at: Perlego. [Accessed 25 Apr. 2022].

GeeksforGeeks (2019). Software Engineering | Classical Waterfall Model [Online]. Available at: https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/ [Accessed 24 Apr. 2022].

GeeksforGeeks (2022). Software Engineering | Spiral Model [Online]. Available at: https://www.geeksforgeeks.org/software-engineering-spiral-model/ [Accessed 28 Apr. 2022].

Guthrie,G., (2020). 13 software development methodologies explained [Online] Backlog. Available at: https://backlog.com/blog/13-software-development-methodologies-explained/ [Accessed 24 Apr. 2022].

IBM (n.d.). What is Machine Learning? [online]. Available at: https://www.ibm.com/cloud/learn/machine-learning [Accessed 4 May 2022].

Imran,M. (2020). Delphi Programming Language - IDEs Versions, Code, Videos, Tutorials [Online]. Folio3AI Blog. Available at: https://www.folio3.ai/blog/delphi-programming-language/ [Accessed 4 May. 2022].

Jones,C. (2017). Software Methodologies: A Quantitative Guide. CRC Press.

Lehtonen, M., Lu, C., Nummenmaa, T. and Peltonen, J. (2020). Adoption of Requirements Engineering Methods in Game Development: A Literature and Postmortem Analysis. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering [online]. Available at: https://trepo.tuni.fi/bitstream/handle/10024/122949/adoption_of_requirements_2020.pdf [Accessed 6 May 2022].

Let´s Learn This Together (2022). Let´s Learn This Together [Online]. Available at: https://www.youtube.com/channel/UCHfgp0ynCQhlTyMiTPvPBkg [Accessed 24 Apr. 2022].

LucidChart (2017). The Pros and Cons of Waterfall Methodology [Online]. LucidChart blog. Available at: https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology [Accessed 28 Apr. 2022].

PixelOver (n.d.). PixelOver - Turn art into pixel art [Online]. Available at: https://pixelover.io/. [Accessed 24 Apr. 2022].

Steampowered.com (2019). Welcome to Steam [Online]. Available at: https://store.steampowered.com/ [Accessed 24 Apr. 2022].

TechTarget (2019). Spiral model, [Online]. Agile, DevOps and software development methodologies. Available at: https://www.techtarget.com/searchsoftwarequality/definition/spiral-model [Accessed 28 Apr. 2022].

Wrike (2019). What is Agile Methodology in Project Management? [Online]. Project Management Guide FAQ. Available at: https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/ [Accessed 28 Apr. 2022].

APPENDICES

Appendix A shows the code for the Textbox create function; it explains how the textbox is created.

```
box = text_box;
frame = txt_frame;
portrait = txt_img;
namebox = txt_namebox;

box_width = sprite_get_width(box);
box_height = sprite_get_height(box);
port_width =  sprite_get_width(portrait);
port_height = sprite_get_height(portrait);
namebox_width = sprite_get_width(namebox);
namebox_height = sprite_get_height(namebox);

port_x = (global.game_width - box_width - port_width)* 0.5;
port_y = (global.game_height*0.98) - port_height;
box_x = port_x + port_width;
box_y = port_y;
namebox_x = port_x;
namebox_y = box_y - namebox_height;

x_buffer = 18;
y_buffer = 8;
text_x = box_x + x_buffer;
text_y = box_y + y_buffer;
name_text_x = namebox_x + (namebox_width/2);
name_text_y = namebox_y + (namebox_height/2);
text_max_width = box_width - (2*x_buffer);

portrait_index = 0;
text[0] = "Choose your path";
text[1] = "You can either take the road North to Greenville ";
text[2] = "Or you can take the road South to Sandville.";
text[3] = "However, whichever path you take, you cannot go back,";
text[4] = "for a curse lies upon this road,";
text[5] = "and whatever you choose, you must stick with. ";

page = 0;
interact_key = ord("E");
name = "Sign";
text_col = c_black;
name_text_col = c_black;
font = Sign_text;
draw_set_font(font);
text_height = string_height("M");
```

Appendix B shows how the textbox was drawn onto the game with the Draw GUI function.

```
//Draw box
draw_sprite(box,0,box_x, box_y);
// draw portrait box
draw_sprite(frame,0,port_x,port_y);
// portrait
draw_sprite(portrait,portrait_index,port_x,port_y);
// portrait frame
draw_sprite(frame,1,port_x,port_y);
//namebox
draw_sprite(namebox,0,namebox_x,namebox_y);
```

```
draw_set_font(font);
//name
var c = name_text_col;
draw_set_halign(fa_center); draw_set_valign(fa_middle);
draw_text_color(name_text_x, name_text_y, name, c,c,c,c, 1);
draw_set_halign(fa_left); draw_set_valign(fa_top);
//text
c = text_col;
draw_text_ext_color(text_x, text_y, text[page], text_height,text_max_width,
c,c,c,c, 1);
```

Appendix C shows the textbox step function, what happens when the player interacts with the textbox object.

```
if(keyboard_check_pressed(interact_key)){
   if(page < array_length_1d(text) - 1){
      page++;
   }else{
      instance_destroy();
   }
}
```

Appendix D shows the player's step function, when the player moves around, what code is used.

```
image_alpha = 1;
if(keyboard_check(ord("D")) and place_free(x + collisionSpeed, y)){
   x += walkSpeed;
   image_speed = walkSpeed / 3;
   sprite_index = Cat_Walking_Right;}
if(keyboard_check(ord("A")) and place_free(x - collisionSpeed, y)){
   x -= walkSpeed;
   image_speed = walkSpeed / 3;
   sprite_index = Cat_Walking_Left;}
if(keyboard_check(ord("W"))and place_free(x, y - collisionSpeed)){
   y -= walkSpeed;
   image_speed = walkSpeed / 3;
   sprite_index = Cat_Walking_Up;}
if(keyboard_check(ord("S"))and place_free(x, y + collisionSpeed)){
   y += walkSpeed;
   image_speed = walkSpeed / 3;
   sprite_index = Cat_Walking_Down;}
if(keyboard_check(vk_nokey)){
   image_speed = walkSpeed / 4;
   sprite_index = Cat;
}
health_bar = (player_hp / max_hp) * 100;
if (player_hp <= 0){
   instance_destroy();
}
```

Appendix E shows the Undertale boss code, what code he has and how it moves and works.

```
// BOSS PHASE 1
if (instance_exists(Cat_walking)){
   with (instance_create_layer(x,y,"Instances", Boss_Udertale_Bullet)){
   direction = point_direction(self.x, self.y,
   Cat_walking.x, Cat_walking.y);
   speed = 2.5; }}
```

```
alarm[0] = 30;
if (instance_exists(Cat_walking)){

// BOSS PHASE 2
if (npc_hp <= 65){
 with (instance_create_layer(x,y,"Instances", Boss_Undertale_Bullet)){
 direction = point_direction(self.x, self.y, Cat_walking.x +
irandom_range(0,100), Cat_walking.y + irandom_range(0,100));
speed = 3;}}
// BOSS PHASE 3
if (npc_hp <= 35){
 with (instance_create_layer(x,y,"Instances", Boss_Undertale_Bullet)){
 direction = point_direction(self.x, self.y, Cat_walking.x +
irandom_range(0,200), Cat_walking.y + irandom_range(0,200));
speed = 3.5;}}}
```

Appendix F shows the story that was planned for the game.

Story for the game:

Long ago, in a town far, far away, lived a peaceful little cat named Jack. Jack loved his little town because everyone was so friendly, and everyone knew each other so there wasn't anything to worry about. One day Jack decided to go for a walk, like he always does. However, this walk would turn out to be very different from the ones he usually took.

You wake up in the middle of the woods, lost and confused. How did you get here? You stumble on to a road that leads to somewhere. But there are two paths to take. Where will you go? You notice a piece of paper stuck to a sign by the road, you read it:

You can either take the road west to Greenville or you can take the road east to Sandville. However, whichever path you take, you cannot go back, for a curse lies upon this road, and whatever you choose, you must stick with. Two scenarios will play out depending on the user's choice, first puzzle comes into play already here. First battle will happen upon the road they choose but the battle will also be different depending on what they choose and the weapon they get will also be different.

Scenario 1: You chose to go to Greenville, you walk west and start to notice how every-thing becomes green, it truly is a land of green. You find a chest in the middle of the road, open it? yes or no.

No – You leave the chest alone and continue walking.
Yes – You open the chest and find a bow and arrow, take it?
No – You leave the bow and arrows in the chest but keep wondering if you should have taken them anyway just in case.

Yes – You take the bow and arrows and put then on your back.
You continue to walk and all of a sudden there is a sound. You enter into battle!!

1. Without bow:

A small but curious frog appears. He wants to know what u taste like. If you had the bow, you could have fought this frog. Now all you can do is run.

- Run from battle

2. With bow:

A small but curious frog appears. He wants to know what u taste like.

- Fight the frog
- Run from battle

1. You run from the frog, and he just stares at you while you run. You hit a sign and it knocks you out.

2. You win the battle, and the frog lies defeated on the ground, you are so happy you fail to notice a sign right in front of you, you hit your head and it knocks you out.

You wake up moments later with a weird creature hovering over you. The creature gets frightened and runs away. You get confused but stand up, the sign you hit is still there. It says: WELCOME TO GREENVILLE!! You keep walking, you step on something that says click. A cage falls on top of your head, the only way to get out is to crack the code. (puzzle) You got out of the cage but there is another puzzle in your way. (Puzzle) You finally made it to Greenville.

Scenario 2: You chose to go to Sandville, you walk east and start to notice how everything becomes warmer and sandier all around its starting to feel like a desert. You almost fall over something in the road, dig it up? yes or no

No – You leave the item alone and continue walking.
Yes – You big up the item, it's a chest!!! Open it?
No – you leave the chest alone but wonder what's inside
Yes – You open the chest and inside is a sword! Take it?
No – You leave the sword in the chest, it's too dangerous to take it with you.
Yes – You take the Sword and strap it to your back.
You continue to walk and all of a sudden there is a sound. You enter into battle!!

1. Without sword:

A large lizard appears. It looks hungry. If you had the sword, you could have fought this lizard. Now all you can do is run.

- Run from battle

2. With sword:
A large lizard appears. It looks hungry.

- Fight the lizard
- Run from battle

1. You run from the lizard, and he just stares at you while you run. You trip over a rock and pass out.

2. You win the battle, and the lizard lies defeated on the ground, you are so happy you fail to notice a big rock on the ground you trip over it and pass out.

You wake up moments later with a weird creature hovering over you. The creature gets frightened and runs away. You get confused but stand up, you see a sign right in front of you that says: WELCOME TO SANDVILLE!! You keep walking, you step on something in the sand, it says click. You fall down a hole, only way to get out is to solve a puzzle. (puzzle) You climb out of the hole in the ground, but now there is a bunch of sand dunes in the way!! (Puzzle) You finally made it to Sandville.

You walk around the town and all of a sudden you fall down a hole. You lose your weapon and stumble around in the dark. You find some light and something shiny on the ground. You pick it up, it makes you glow. You cannot climb out of the hole, so you keep walking forward. All of a sudden you step into a room with a lot of enemies. You fight off all the enemies and finally stumble out of the cave. You get unto a beautiful meadow with lots of guards walking around and a big castle in the distance. We need to get into the castle to defeat the evil villain that lives there to be able to return home. We meet an old lady. The old lady tells us that we have ended up in a different world and we need to save them all to be able to return home. You get into the castle and there is a small maze.

You finally reach the end and destroy the villain. But now u have to make a choice, either you stay here forever and the world you are in now will be fine, or you can leave and this whole world will be destroyed, and all its people die.

Appendix G shows the roadmap to the implementation of the game:

1. Story done
2. Puzzles to implement
3. AI implementation
4. When does everything need to be done?
5. Writing the text
6. Building the game
    a. Building phase 1: Start
    b. Phase 2 implementation of puzzles
    c. Phase 3 implementation of AI
    d. Phase 4 end
7. Finishing up

Appendix H shows the code of the health bar system.

```
Create event:
event_inherited();
parent_obj = object_get_parent(object_index);
parent_name = object_get_name(parent_obj);
player_hp = 100;
min_health = 0;
max_health = player_hp;
for (var num = 0; num < instance_number(parent_obj); ++num;){
parent_inst[num] = instance_find(parent_obj, num);}
Step event:
if (player_hp <= 0){
player_hp = 0;}
Draw event:
for (var num = 0; num < instance_number(parent_obj); ++num;){
draw_set_color(c_red);
draw_rectangle(parent_inst[num].id.x-5, parent_inst[num].id.y+20,
parent_inst[num].id.x+20, parent_inst[num].id.y+15, false);
draw_set_color(c_green);
draw_rectangle(parent_inst[num].id.x-5, parent_inst[num].id.y+20,
parent_inst[num].id.x-5+(player_hp/4), parent_inst[num].id.y+15, false);
draw_set_color(c_white);
draw_rectangle(parent_inst[num].id.x-5, parent_inst[num].id.y+20,
parent_inst[num].id.x+20, parent_inst[num].id.y+15, true);
draw_set_color(c_yellow);
draw_text_transformed(parent_inst[num].id.x+8, parent_inst[num].id.y+16,
player_hp, .10, .10, 0);
}
```

Appendix I shows the shooting and attack mechanics of the player, when pressing the mouse button.

```
sprite_index = animation_meow_attack;
with (instance_create_layer(x,y,"Instances", cat_bullet)){
direction = point_direction(self.x, self.y, mouse_x, mouse_y);
speed = 2.5;}
```