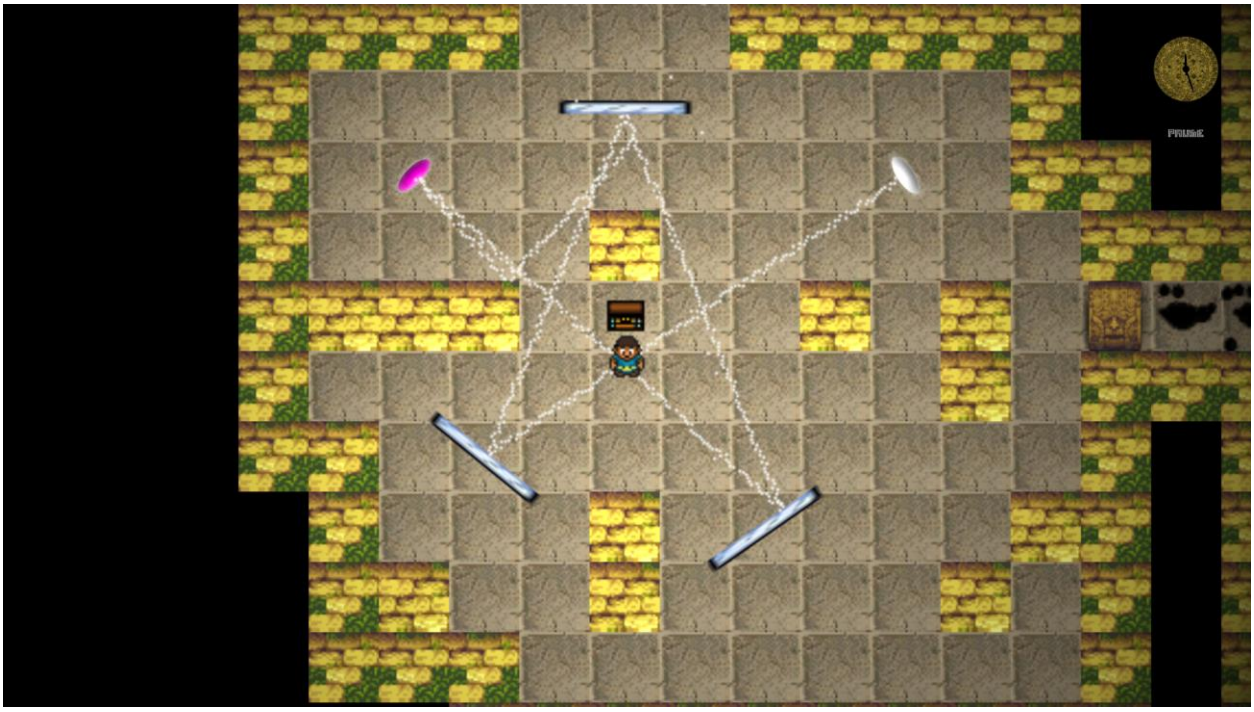
CE903/CE913 Group Project (Incorporating a Game Jam)
Requirements Specification & Project Plan

Rebirth



Raluca Gaina	1501104
Matthew Harrop	1203496
Paul Leonard	0920713
Jonathan Nichols	1504740
Olivier Thill	1500973
Ovidio Villarreal	1500946
Harvey Wigton	1500646

Table of Contents

1. Introduction

The Global Game Jam (GGJ) [1] is the largest event of its kind in the world, being organized at the same time in numerous physical locations around the globe. It aims to bring together artists, story-tellers, programmers and designers in order to create a new game in just 48 hours. Even though all the products obtained at the end of the event are meant to be unique expressions of their creators, they all have one thing in common: a secret theme, announced at the start of the event. The theme can vary from something quite specific ("As long as we have each other, we will never run out of problems", 2009 [1]) to a more abstract concept ("The Sound of a Heartbeat", 2013 [1]). This year's theme, which this project follows, is "Ritual".

While offering an intellectual challenge to participants and opening new opportunities in the community, the GGJ is not a competition, but a way for people with common interests to come together and collaborate in order to have a finished product by the end of the condensed development cycle.

This document offers an account of the requirements specification and project plan of the main project, which can be summed up as designing and developing a game for the Global Game Jam 2016, as well as continuing development in order to produce a fully fleshed out game. The methodology adopted is agile, chosen due to its high flexibility. In the context of the GGJ marking the proper start of the project, with the theme unknown before the event – a restriction that has significant impact on the project itself –, adaptability is a necessary benefit.

As the actual event, where the game is specified and created, takes place just 4 days before the deadline, the first few sprints cover team building and discussions about broad objectives regarding the project. The last sprint outlined in this document is concerned with the actual specification, design and development during the Game Jam. At the end of this sprint, a first version of the game is produced and tested accordingly.

The rest of this document is structured as follows: section 2 defines functional and non-functional requirements and the third section describes testing techniques and procedures. The document concludes with a detailed record of project planning tools, methodologies, sprints and a project backlog.

2. Requirements Specification

2.1 Requirements Elicitation, Analysis and Specification

The requirements for this document were derived following three steps: idea elicitation (gathering), idea analysis and converting ideas into concrete specifications.

Gathering ideas can be split into two sections: those dictated by the rules of the GGJ and those decided upon by the team. The GGJ has only one definite requirement: the game must incorporate the given theme. However, there are a few requirements that, whilst not necessary, are useful for ensuring a successful game (e.g. a time limit in which the game should be completed). Other ideas were also supplied by the team, based on what each member thought would make a successful game.

After gathering the ideas, the value of each was discussed, together with how well the idea would work in the context of the game. This discussion resulted in narrowing down the specific required features of the game.

Finally, each feature was broken down to its base requirements.

2.2 System Architecture

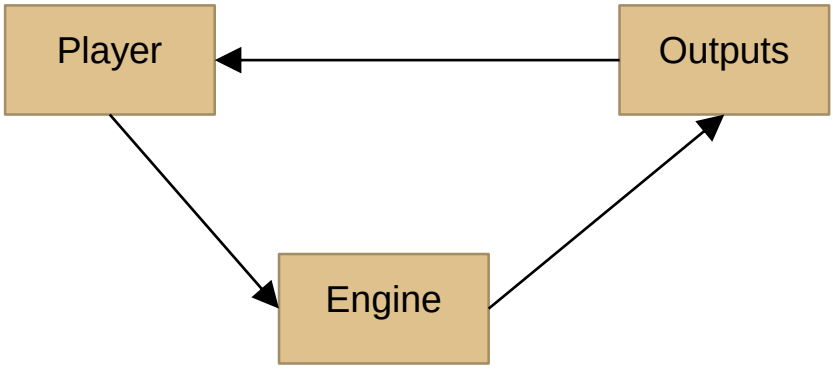


Figure 1. System Architecture

2.2.1 Player

The player (Figure 1) reacts to the outputs from the engine, either visual images on the game screen or various audio information. The inputs from the player are then passed on to the engine.

2.2.2 Engine

The player's inputs are interpreted and cause changes to the game world. As well as player actions, the engine (Figure 1) also controls non-player actions, such as timers, NPC actions etc.

2.2.4 Output

The output (Figure 1) from the engine is given to the player both visually, as updates to the display and via audio, as sound effects or music.

2.3 Functional Requirements (Design Behaviours)

2.3.1 Non-Domain Requirements

1. The game must contain multiple options/paths to complete any level.
2. The game must dynamically generate the paths between each room in the game (Each play through has a degree of randomness, increasing the replay value of the game).
3. The game must contain multiple puzzle rooms per level.
4. The game must allow the player to leave a puzzle room (and possibly attempt a different puzzle).
5. The game must contain a real-time combat system (combat can take place in any room or path at any time).
6. The game must allow the player to flee from combat.
7. The game must end each level with a single boss room (a boss is a combat puzzle where the player finds and exploits a weakness).
8. The game must reward the player's progress through each level by giving the player a new item or ability.
9. The game must allow the player to trade in currency in order to purchase a new item or ability.
10. The game must have different themes for each level.
11. The game must track the player's score.
12. The game must track the amount of time that the player has obtained (game currency).
13. The game must have a text based storyline.

2.3.2 Domain Requirements

1. The game must have a starting splash screen (before game starts, showing the name of the game).
2. The game must have a loading splash screen (between levels).
3. The game must have an options menu (allowing the user to adjust sound, brightness and difficulty, or to toggle full screen/windowed modes).
4. The game must have a start menu (single player, split screen, options).
5. The game must have sound effects.
6. The game must have background music.

2.4 Non-functional Requirements (Architectural Constraints)

1. The game must be accessible and playable from a web browser.
2. The game must be created using Unity and C#.
3. The game must have keyboard support.
4. The game must have mouse support.
5. The game must have Xbox controller support.
6. The game must have PlayStation controller support.

3. Testing Schedule

3.1 Overview

Three types of testing will be used in order to test each version of the game and its components: unit testing, integration testing and system testing. For unit testing, the functionality of each component that is developed for the game will be tested against its requirement, by its developer. Integration testing will be used to assure that developed components work together correctly and do not interfere with each other's functions. Finally, system testing will be used to test the version of the game in a realistic game environment.

3.2 Test Types

3.2.1 Black Box

One approach to testing is to take the specification of the program or that of a component and test if it gives the expected output for a specific set of inputs. This is known as functional or black box testing. If an object oriented programming paradigm is used, there is one caveat to this method: if testing methods of an object, the current state of the object has to also be taken into account when determining what the expected output in a test case should be [2].

3.2.2 White Box

The other common test approach is to analyse the code directly and, based on those findings, select a number of test cases such that as many execution paths as possible, preferably all, are covered. This is known as white box or structural testing [3].

3.3 Schedule

The testing schedule is designed according to the agile methodology. Each finished component must pass all of the test cases associated with it before it is considered "done". These tests must be rerun each time a feature has been added to the component [4].

3.4 Unit Tests

Unit tests are needed for the components below:

3.4.1 Characters

Player: movement, ability use, interaction with the environment and graphical display. These can be implemented as a black box test using a test scene.
Concrete tests:

- Does react correctly to movement keys.
- Does move at the correct speed.
- Does push blocks in the right direction.
- Inventory shows the correct items and abilities.
- Abilities have the correct effect.
- Only abilities the player has acquired can be used.
- Sprite is drawn and animated correctly on the screen.

NPC: artificial intelligence, interaction with the environment and graphical display. These can be implemented as a black box test using a test scene.

3.4.2 World

Puzzles: solvability, proper reset, dead-end free. Each puzzle room needs to be play tested. If inter puzzle dependencies exist, all paths must be tested.
Concrete tests:

- Solvable.
- No dead ends, or the player can recover from a dead end.
- Resets correctly.
- No unintended solutions.
- All puzzles which depend on one puzzle (Puzzle A) cannot be solved without solving Puzzle A.

Levels: the same rules for puzzles apply.

3.4.3 Menus

Dialogue: display, correct ordering.
Concrete tests:

- Dialogue is not hidden by other elements.
- All dialogue options are reachable.
- The correct dialogue appears at all times.

Menus: the same rules for dialogue apply.

3.5 Testing of Non-functional requirements

3.5.1 Quality of Assets

Audio: general quality.
Background music loops throughout the game and it is not noticeable to the player when this happens. Sound effects play according to each interaction, as intended.

Graphics: tessellation where applicable, general quality.
Each sprite must be checked for quality and each one of them is designed to portray the theme of the game. Each sprite has a transparent background and the dimensions of each one of them is a multiple of 32px.

Text: grammar, orthography.
Each text displayed in menus, dialogue or graphics needs to be checked.

3.5.2 Consistency of Assets

Audio: the quality is consistent and all audio assets fit the theme of the game.

Graphics: the quality is consistent and all graphical assets fit the theme of the game.

Text: the quality is consistent throughout the game and all text assets fit the theme of the game.

Story: the story is consistent and there are no plot holes.

3.5.3 Balance

Difficulty curve: it has to match the shape as specified.

Skill balance: all of the choices a player can make during the game are viable, irrespective of skill.

Test Case ID: Move_0001			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): High			**Test Designed date:** <Date>	
Module Name: Player/Character			**Test Executed by:** <Name>	
Test Title: Character Movement			**Test Execution date:** <Date>	
Description: Check the movement of the character according to each one of the movement keys bindings.				
Pre-conditions: Game must have already started.				
Dependencies: None				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Press the LeftKey button on the keyboard.	LeftKey	Character moves to the left and executes the animation accordingly.	If there is a wall, only the animation will be executed.
2	Press the RightKey button on the keyboard.	RightKey	Character moves to the right and executes the animation accordingly.	If there is a wall, only the animation will be executed.
3	Press the UpKey button on the keyboard.	UpKey	Character moves upwards and executes the animation accordingly.	If there is a wall, only the animation will be executed.
4	Press the DownKey button on the keyboard.	DownKey	Character moves downwards and executes the animation accordingly.	If there is a wall, only the animation will be executed.
Post-conditions: Character moves successfully on the map.				

Table 1. Test Case #1

Test Case ID: Move_0002			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): High			**Test Designed date:** <Date>	
Module Name: Player/Character			**Test Executed by:** <Name>	
Test Title: Interacting with Movable Blocks			**Test Execution date:** <Date>	
Description: Character pushes the corresponding block one space in the appropriate direction.				
Pre-conditions: Character must be next to a movable block.				
Dependencies: None				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Face the movable block.		Character animates and faces the movable block.	
2	Press the E key on the keyboard and a direction key to move the movable block.	E Key	Movable block and character move.	The block moves the size of a tile (32x32px)
Post-conditions: Character moves the movable block one tile of distance (32x32px).				

Table 2. Test Case #2

Test Case ID: Tile_0001			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): High			**Test Designed date:** <Date>	
Module Name: Tile			**Test Executed by:** <Name>	
Test Title: Character Press Switch			**Test Execution date:** <Date>	
Description: Character presses switch on the ground.				
Pre-conditions: Game must have started.				
Dependencies: None.				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Character is next to a switch tile.			
2	Character moves onto the switch.		Tile animates, changes sprite and makes a triggered sound.	
Post-conditions: Switch changed sprite to represent that it is triggered.				

Table 3. Test Case #3

Test Case ID: Tile_0002			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): High			**Test Designed date:** <Date>	
Module Name: Tile			**Test Executed by:** <Name>	
Test Title: Movable Block Press Switch			**Test Execution date:** <Date>	
Description: Movable block presses switch on the ground.				
Pre-conditions: Game must have started.				
Dependencies: Movable block.				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Movable block is next to a switch tile.			
2	Character pushes block onto the switch.		Tile animates, changes sprite and makes a triggered sound.	
Post-conditions: Switch changed sprite to represent that it is triggered.				

Table 4. Test Case #4

Test Case ID: Tile_0003			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): High			**Test Designed date:** <Date>	
Module Name: Tile			**Test Executed by:** <Name>	
Test Title: Trap Switch			**Test Execution date:** <Date>	
Description: Character presses switch on the ground and activates a trap.				
Pre-conditions: Game must have started and trap should not have been activated previously.				
Dependencies: None.				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Character is next to a switch tile.			
2	Character moves onto the switch.		Tile animates, changes sprite and makes a triggered sound.	
3	Trap activates.		A floor tile is changed to a pit tile through which the character can fall down.	
Post-conditions: Switch changed sprite to represent that it is triggered. One or several floor tiles are now pits through which the character can fall.				

Table 5. Test Case #5

Test Case ID: Clock_0001			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): Medium			**Test Designed date:** <Date>	
Module Name: Clock			**Test Executed by:** <Name>	
Test Title: Room Reset			**Test Execution date:** <Date>	
Description: Player clicks the clock and restarts the puzzle they are in.				
Pre-conditions: Character must be in a puzzle room already modified.				
Dependencies: None.				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Player clicks on the clock.		Animation and sound of the clock activates. Room resets to its original state.	
2	Player stays at the same position.			If block respawns at the character's position, the character is moved to closest available position.
Post-conditions: Rooms returns to its original state.				

Table 6. Test Case #6

Test Case ID: Pit_0001			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): Medium			**Test Designed date:** <Date>	
Module Name: Pit			**Test Executed by:** <Name>	
Test Title: Pit Fall			**Test Execution date:** <Date>	
Description: Character falls through pit and respawns at the start of the puzzle room.				
Pre-conditions: Game must have started.				
Dependencies: None.				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Character is next to a pit tile.			
2	Character moves onto the pit.		Falling sound is triggered.	
3	Character respawns at the start of the puzzle room.			Puzzle does not restart.
Post-conditions: Character at the start of the puzzle room.				

Table 7. Test Case #7

Test Case ID: Chest_0001			**Test Designed by:** Ovidio	
Test Priority (Low/Medium/High): High			**Test Designed date:** <Date>	
Module Name: Reward			**Test Executed by:** <Name>	
Test Title: Reward Obtained from Chest			**Test Execution date:** <Date>	
Description: Character opens the chest and gets the reward.				
Pre-conditions: Puzzle must be solved.				
Dependencies: None.				
Step	**Test Steps**	**Test Data**	**Expected Results**	**Notes**
1	Character is next to the chest.			
2	Character opens the chest.		Chest animates, changes sprite and makes a reward sound.	
3	Reward is saved in the player's inventory.			
Post-conditions: Player now has a new reward in the inventory.				

Table 8. Test Case #8

4. Project Planning

In this section, we discuss the project's management methodology, direction of the project, work done so far and associated tools used to reach this stage.

4.1 Project Management Methodologies

Picking an ideal methodology to be used as a framework is important for maximising our work potential. There are many methodologies that have been devised for handling different teams and projects, and so the team's initial discussions were centered on which one we should use.

We require a flexible framework which would allow us to work within the limited time frame we have and to cope with the changeability of the project. Due to the nature of a Game Jam, in which we are not told the theme of the project until the day of the event, we cannot plan specific details about the project and instead have to be cautious. Prioritising development over documentation will prevent the team from planning to excess.

4.1.1 Agile Methodology

An Agile management methodology [5] was ultimately chosen due to its flexibility and focus on productivity instead of documentation. To elaborate, agile methodologies are designed to be able to adapt to changing circumstances and objectives. In contrast to Waterfall, another planning methodology, the framework is far less strict, focusing instead on development and implementation; it directly encourages change and has systems in place to handle and adjust to any unforeseen circumstances.

4.2 Tools

The project team selected a varied range of software tools to support the creation of both the software product and report documents. Tool selection can have a big impact on the productivity of the team, as well as how easily that productivity can be achieved. As a team, we have very specific requirements for our tools. First, they must be cheap, free or obtainable without the use of money. They must also be game-suitable and able to export to formats that are compatible with our chosen development platform and engine, Unity. Some of the tools used are covered below.

4.2.1 GitHub

Github [6] is used for code management and versioning to make team development easier and more streamlined. It will provide a central online location in which the entirety of the project assets are stored, so that the team can collaborate in real time. Without such a tool, working together would entail using another method (e.g. email, portable hdds, non-development-centric file sync services) to transfer files and synchronise all of our work. It also provides tools for handling some of the problems that can arise from teamwork, such as conflicting code and concurrent modification of a single file from multiple people. As the service is also free, it is very suitable to our needs.

4.2.2 Unity

The development environment the team decided to use is Unity [7], as it is specifically designed for game software development. Unity is a platform which specialises in 2D and 3D games or interactive content. It features a wealth of domain specific tools and features, which make it perfect for our project. Most importantly, it is completely free for us to use within the project's context. Many members of the team have used the software before, potentially removing some of the initial learning problems that may arise. Unity also has plugins to integrate with SVN tools; having native support for this will be key to seamless and painless collaborative development.

4.2.3 Microsoft SharePoint

Due to the discussions of the team members needing to be evidenced for the marker of the course module, SharePoint [8] is used for out of meeting communication, where the required information is accessible for all relevant parties. SharePoint is a web-based, password protected hub for all sorts of project types. It doesn't explicitly have any tools for developers, but the document sharing and forum style communication system should reduce the likelihood of any correspondence accidentally being missed or lost.

4.2.4 Google Docs

For creation of the first report, Google Docs [9] was used. Google Docs enables teams to simultaneously work on a document together, accounting automatically for layout, sync, and concurrent editing issues. Not only is this service free, but it is highly robust, allowing all members to access the document with a fast browser based editor, as well as removing the hassle of having to merge separate documents at a later stage of the document creation process. Using this service did have the disadvantage of not having the full Microsoft Word complement of features, none of which were required however for this document.

4.3 Gantt Chart

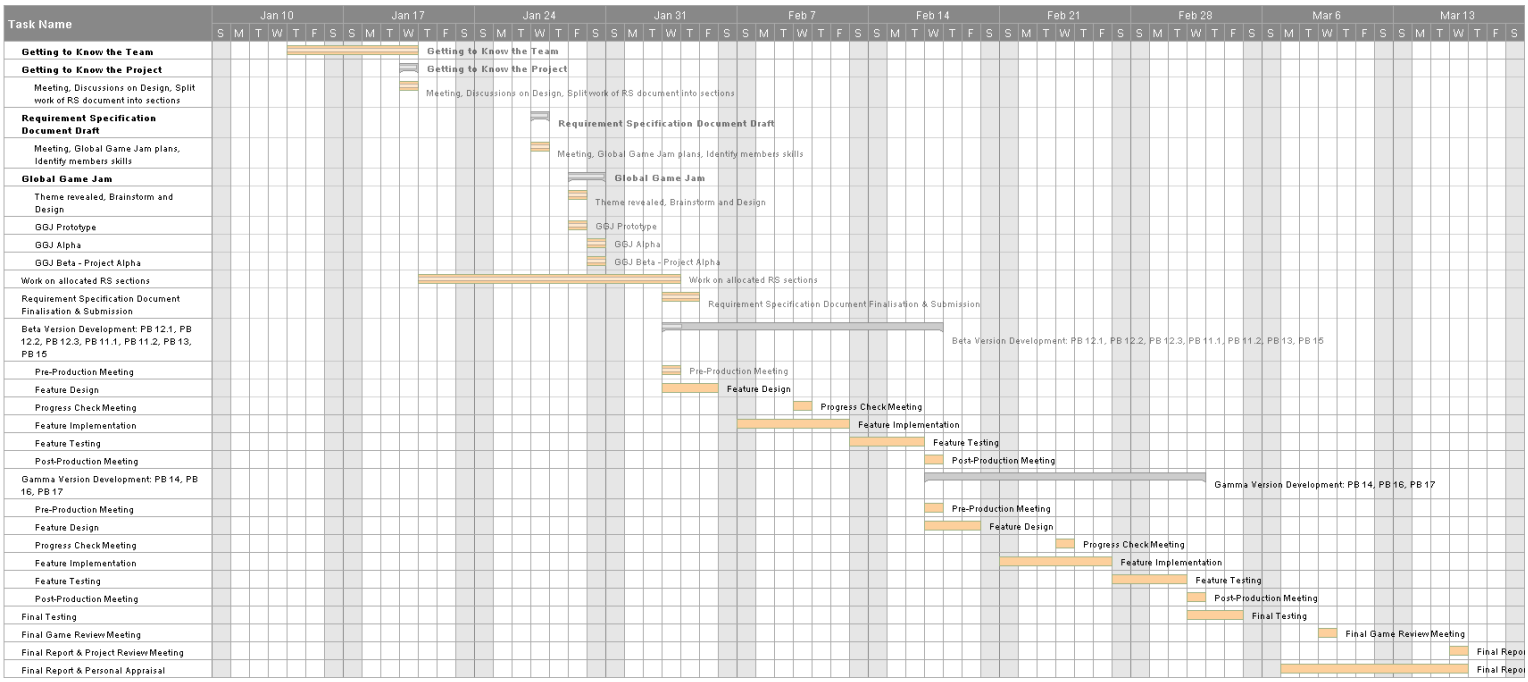


Figure 2. Gantt Chart

4.4 Project Backlog

4.4.1 Future Tasks

Reference	Title	Breakdown
11	Refactoring	Tile Engine
		Mirror Sink
12	Dynamic Map Creation	Puzzle Rules
		Puzzle Insertion
		Connecting Rooms
		Puzzle Creation
13	Items/Abilities	Design and Implementation
		Item/Ability Shop GUI
14	More Puzzles	
15	More Assets	
16	More Temples	
17	Extend Storyline	

Table 9. PB – Future Tasks

4.4.2 Completed Tasks

Reference	Title	Breakdown
1	Tile Engine	
2	Interactive Objects	Movable Blocks
		Reward Chest
		Mirrors/Sink/Source
		Switches
		Collectable Items
		Traps
		Pits
		Door
3	Inventory	
4	Win Conditions	
5	Puzzles	Dark Room
		Movable Block Room
		Mirror Room
		Trap Room
6	Puzzle Reset	
7	Graphical User Interface	Introduction/Backstory
		Room/Puzzle Reset clock
		Pause Screen
		Credits
		Game Over
		Victory Screen
8	Story	
9	Player/Character	Movement
		Camera Tracking
10	Art Assets	Sprites
		Animation Object/Player
		Sound/Music

Table 10. PB – Completed Tasks

4.5 Sprints to Date

Sprints for this project encompass a week time period due to team members' other responsibilities and the module requirements. The sprints to date are overviewed below:

4.5.1 Week 1 - Getting to Know the Team (5 hours)

In the first week the main objective was to meet the team members and the supervisor. This included finding a room to meet in, as well as identifying a time when we were all without a prior engagement. We also vaguely talked through what to expect from the module and in particular the Game Jam.

4.5.2 Week 2 - Getting to Know the Project (8 hours)

Week 2 had a long meeting in which we tried to narrow down the scope of the project and make some decisions on what the game will actually be like. This had to be done, as best we could, before the Game Jam and the associated theme were released because of the small amount of time between the end of the Game Jam and the report submission deadline.

4.5.3 Week 3 - First Report Drafting (10 hours)

Using Google Docs, the team have been working collaboratively on this report. The meeting was thus highly focused on the report and its structure. This was also our final official meeting before the Global Game Jam, therefore we voiced any final questions about the Jam and how we will proceed as a team.

4.5.4 Week 4 - Game Jam and Finalising/Submitting the Report (55 hours)

Week 4's meeting involved the review of the Requirements and Specification report, as well as assessing the current game state after the progress made during the Game Jam.

All team members attended the Game Jam and produced a working prototype, according to the rules of the event (including respecting the theme, game completion within two minutes and all of the features being introduced in the first thirty seconds). The team worked well together, dividing work between group members as follows: two members worked on art assets, two on the level design, one on the graphical user interface, one on implementing core functionality and one on testing and debugging. With a sound foundation set from the Game Jam, continued development will be carried out over the following project period.

During the GameJam, an idea of the game was developed and implemented. The idea of the game is a puzzle game where the main character needs to gather relics in order to revive himself with the help of the gods. Four puzzle rooms and a main chamber where designed and went through its first versioning. These rooms can be seen in Figures 3-7 below.

Figure 3. Game representation of the first puzzle.

Figure 4. Game representation of second puzzle.

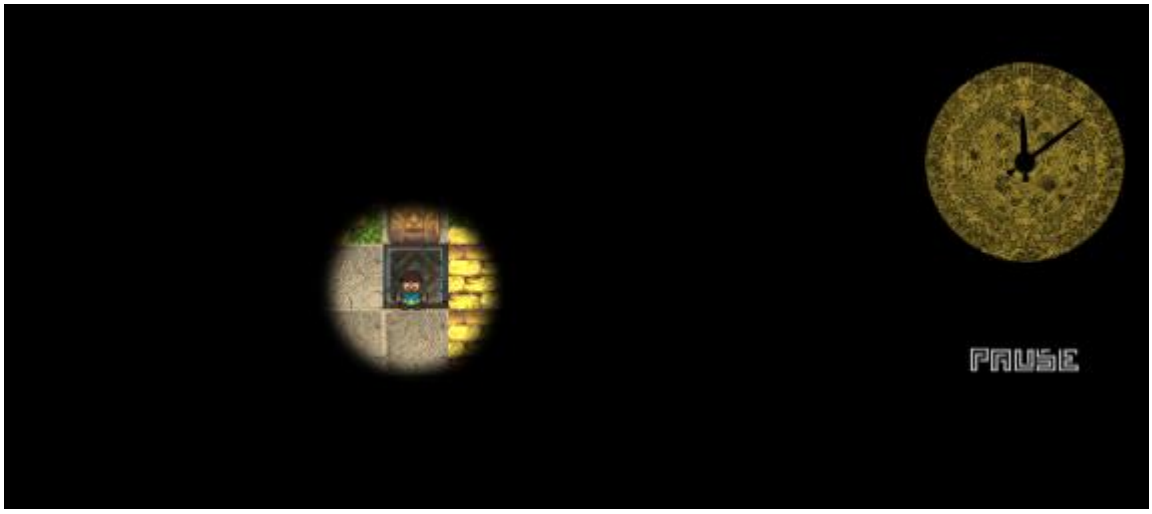


Figure 5. Game representation of the third puzzle.

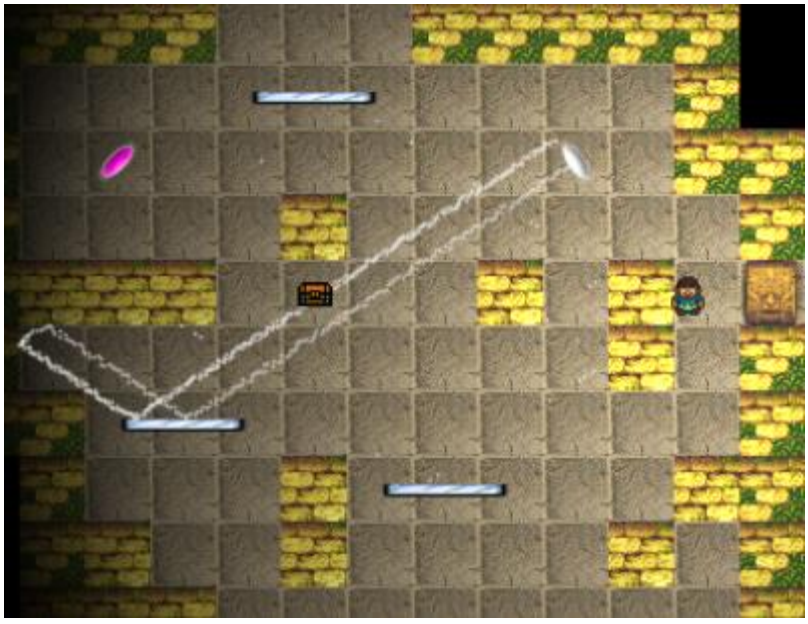


Figure 6. Game representation of the fourth puzzle.

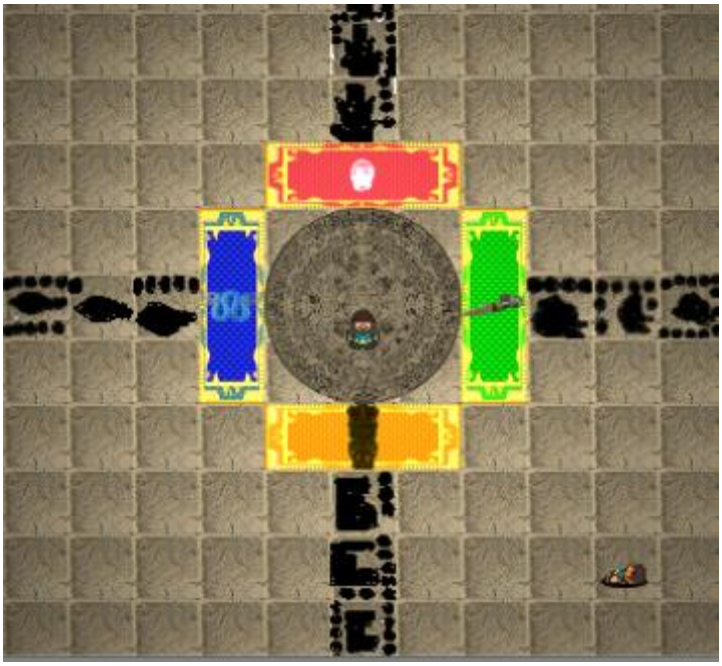


Figure 7. Game representation of the main chamber.

The game design of the map and the puzzles is given by reading a text file that gives the characteristics, details and positions of the objects in the environment seen in Figure 8-9 below.

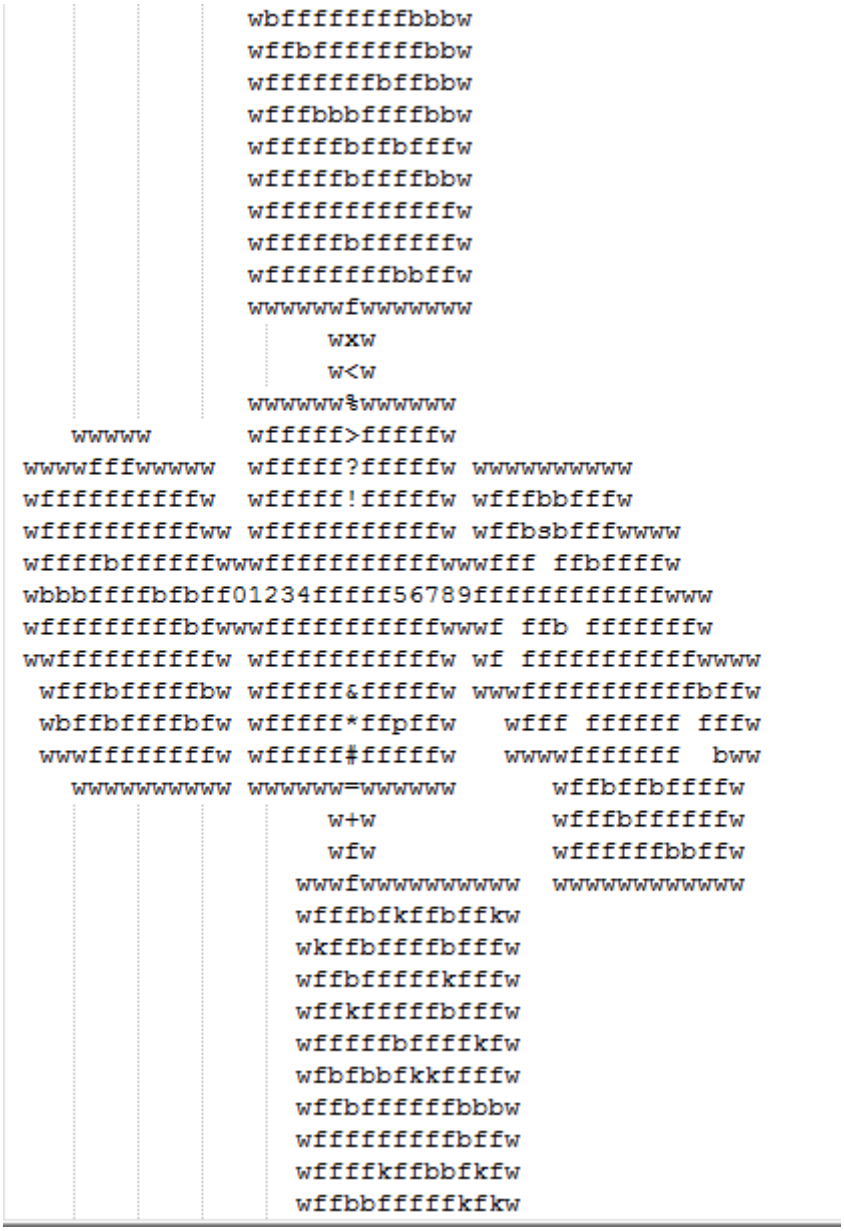


Figure 8. Design of the map.

```
- Objects0
15 13 27 25 Offset
p 9 10 Player
z 10 10 Corpse
h 6 6 center
q 6 4 northCarpet
e 8 6 eastCarpet
u 6 8 southCarpet
y 4 6 westCarpet
- Rules0
northCarpet : Reward1
southCarpet : Reward4
eastCarpet : Reward2
westCarpet : Reward3
win : northCarpet & southCarpet & eastCarpet & westCarpet & center
```

Figure 9. Text representation of objects on the main chamber.

5. References

[1] Dustin Clingman, Susan Gold, Lindsay Grace, Foaad Khosmood, Gorm Lai, Global Game Jam, Global Game Jam, Inc., [Online], Available: http://globalgamejam.org/about [Accessed: 29 01 2016]

[2] Turner, C.D.; Robson, D.J., "The state-based testing of object-oriented programs," in Software Maintenance ,1993. CSM-93, Proceedings., Conference on , vol., no., pp.302-310, 27-30 Sep 1993

[3] Barus, A.C.; Hutasoit, D.I.P.; Siringoringo, J.H.; Siahaan, Y.A., "White box testing tool prototype development," in Electrical Engineering and Informatics (ICEEI), 2015 International Conference on , vol., no., pp.417-422, 10-11 Aug. 2015

[4] Hellmann, T.D.; Chokshi, A.; Abad, Z.S.H.; Pratte, S.; Maurer, F., "Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences," in *Agile Conference (AGILE), 2013* , vol., no., pp.32-41, 5-9 Aug. 2013

[5] t. f. e. Wikipedia, “Agile software development,” [Online]. Available: https://en.wikipedia.org/wiki/Agile_software_development. [Accessed 29 01 2016].

[6] Github, “GitHub · Where software is built,” [Online]. Available: https://github.com/. [Accessed 29 01 2016].

[7] Unity, “Unity - Game Engine,” [Online]. Available: https://unity3d.com/. [Accessed 29 01 2016].

[8] Microsoft, “What is SharePoint 2013 – Overview of Features,” [Online]. Available: https://products.office.com/en-us/sharepoint/sharepoint-2013-overview-collaboration-software-features. [Accessed 29 01 2016].

[9] Google, “Google Docs - create and edit documents online, for free.,” [Online]. Available: https://www.google.com/docs/about/. [Accessed 29 01 2016].