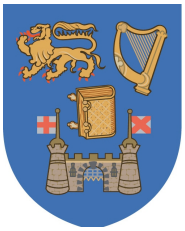
School of Computer Science and Statistics

Trinity College Dublin



Final Report

CS7038: Group Project

Yafie Qu, Hao Guan, Jeremiah Dunn, Saloni Sharma

April 8, 2015

1 Overview of the Project

1.1 Story

Our game takes place on a spaceship which is on a long journey. The ship's computer begins malfunctioning, different subsystems start failing and the security system begins to malfunction. The crew of the ship is in stasis and unaware of the situation. The player character, the maintenance droid is the only unaffected robot or computer on the ship. His task is to restore power to ship's subsystem while avoiding the security system. The ship's fate is in the hands of this humble maintenance droid.

1.2 Theme

This is the International Year of Light and Light-based Technologies. This is a United Nations observance that aims to raise awareness of the achievements of light science and its applications and importance.

Our game incorporates light as one of the main mechanics for exploring the ship. When the space ship is wrecked and malfunctions, it looses all its power sources and light. The player character returns power to the ship, gradually lighting it up in the process. Restoring light makes the ship easier to navigate and directs the player towards the exit of each level.

1.3 Initial Design

The game is an exploration based 2D platformer similar to games like *Spelunker*. It is designed to be played using the Unity web-player. The initial design was focused on randomised level generation. Using this a large variety of different looking levels can be created from a handful of prefabs. The gameplay is influenced by the dynamic lighting system which changes the appearance of the ship. These dynamic lighting effects are generated by the placement of point and spot lights and use if normal maps for different game objects.

2 Technical Overview

The game is made using *Unity*. Game physics are implemented using Unity 2D physics engine. Normal maps and height maps are created with the assistance of *SpriteLamp* and *Crazy Bump*.

2.1 Level Generation

Each level begins as 2D grid of an arbitrary size. A *solution path* is then generated through the level as shown in figure 1. First a point on the top of the grid is randomly picked as the start of the level. We then can step through the grid towards the bottom. Now a left/right is chosen randomly. There is then a random chance of stepping in this direction or stepping down. After stepping down, we chose our random direction again and repeat the process. When we reach the bottom floor the probability of going down is replaced

by a probability of choosing the exit.

We then assign different room types to each point on the *solution path*. These room types correspond to the exits/entrances that each room needs. After this, we draw a boundary around the level. Doing this minimises the amount of different exit types we need to design to the four shown in figure 2.

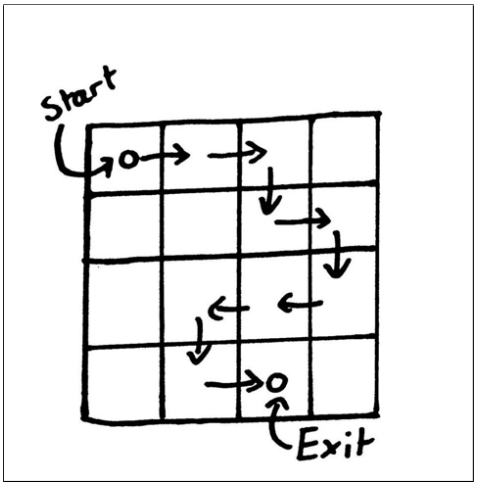


Figure 1: Solution Path Generation

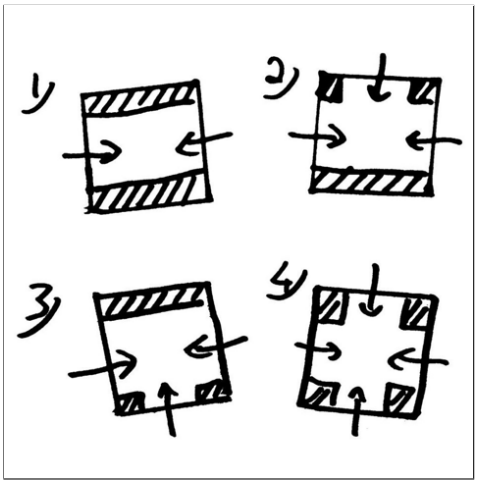


Figure 2: Room Types

2.2 Rooms

Rooms are 16x16 grids of tiles, there are different rooms for the different exit types. Rooms are represented by *CSV* text files an example of which can be seen in figure 3. In the text files, different characters are used as identifiers for different tile types. For example *G* represents a solid, walk-able surface, while *T* represents a trap tile. Modifiers to the tiles, such as the radius of a light, can be added after the identifier. These can be easily swapped out of the game to create and test new levels.

2.3 Lighting System

To follow the theme, the game uses light to change the appearance of the level as the game progresses. We use three different sources of light in our game.

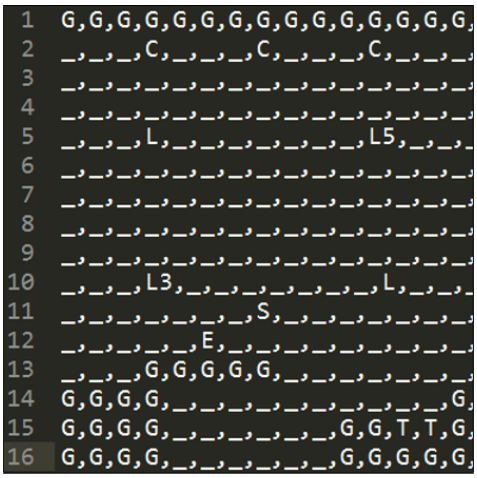


Figure 3: Input text file

Figure 4: Unity Editor Output

The first type of light is the spot light which is used for general illumination and ambience. The second type of light is the point light which is used to light specific areas. They are also used as spinning red alarm lights. These have three hard levels of dark, dim and bright which changes as components near them become reactivated.

The third method of lighting the scene is changing the ambient light level. As the player moves through the ship activating components the rooms towards the exit begin to light up. By varying the ambient light we can make dark areas harder to navigate and already explored areas brighter which helps highlighting where to go.

To emphasise the light effect on the level, normal mapped sprites are being used so as lights turn on and red alarm lights spin the level appears to react to it.

2.4 Security System

The security system can be tripped when the player re-activates components of the ship. To turn it off the player needs to find another component of the ship and interface with it.

When the ship enters an alarm state, the security droid begins chasing the player and electric traps are activated. The trap tiles kill the player with

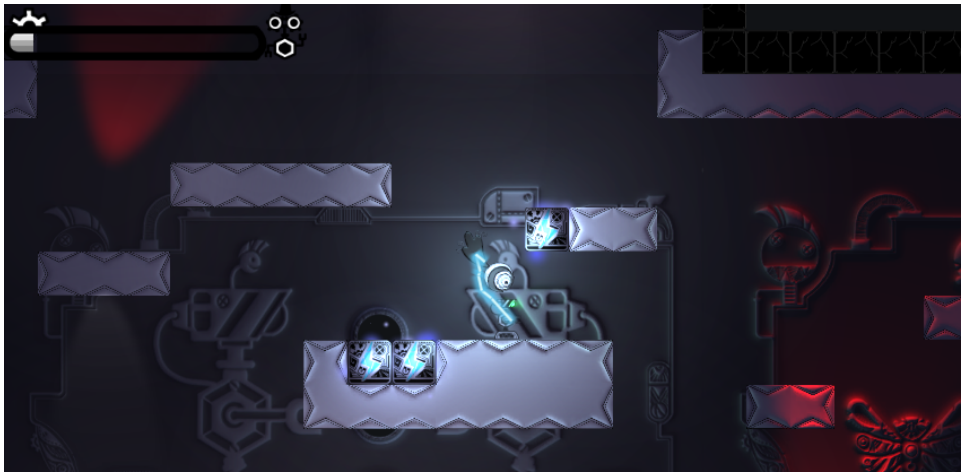


Figure 5: Security System

an electric bolt if they come in contact with them. The security droid can chase and attack the player by draining his energy. This forces the player to move fast to find the next ship component while dodging traps.

2.5 Player Character

The player controls the character directly as a traditional 2D platformer. The gameplay includes an energy system for the player. On re-activating a ship component, the player receives a portion of its energy back. The energy drains over time, encouraging the player to advance through the level.

The camera is dragged along by the player and moves in 3-dimensional space, the player can also hold the down button to move the camera down to check for traps lurking below drops. There are two lights attached to the player character. A spot light is used to light up the area in front of the player in dark rooms. A point light is used to draw attention to where the player is on screen and make him stand out.

3 Team Organisation

Scrum was used for the development of our game which is an iterative and incremental agile software development methodology for managing product development.

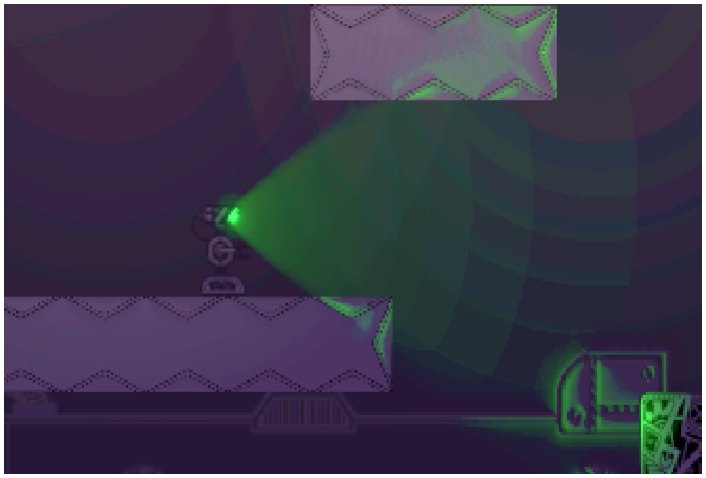


Figure 6: Player Character

The team has used Jira for project management and Git for source control, synchronising progress across team and for maintaining stable and development branches.

The team created high level overview of tasks called Epics. The game different main components were divided into the following Epics: *Level Generation*, *GUI Creation*, *Game Logic* and *Player Character*.

These Epics were further broke down into smaller, more manageable stories. The project was divided into 4 sprints. Each sprint lasted for 2 weeks. At start of each sprint we had grooming sessions to estimate or re-estimate times for different stories and give more detailed descriptions of their implementations. Stories which were considered to be overly-complicated were broken down into smaller sub-task which could be more accurately estimated. Time estimation for different tasks was decided by using a voting system and then averaging everyone's estimates.

Team had regular stand-ups, an average of three times a week. Stand-ups were aimed to last for 10 minutes with any additional discussions happened in separate meeting outside this time. The team had a different scrum master for each sprint from each one of the team members.

Team prioritized the project stories by keeping Level generation and core game play features as high priority features and features secondary to a functional prototype such as a GUI for starting the game as low priority features. This was implemented for an organized way of development so that major game-play could be implemented on time keeping an account of time spent on other assignments of coursework. During the course of project, team learned more about Unity and skills/efficiency of different team members.

4 Conclusion

The team has implemented the core game mechanics as planned. Starting from the level generation with file handler to the lighting system, the alarm system and the enemy AI droid. There is also a rought user interface for starting and ending games and transitioning between levels.

5 Future Work

The way in which the game is created would allow for user created room files which would have been nice feature to explore if we had more time. A dedicated, visual, room editing tool was also discussed but the time-constraints of the project meant we had to stick with manually editing text files.

6 You Tube Link

`https://www.youtube.com/watch?v=Sa106SV3YMg&feature=youtu.be`

7 Acknowledgements

We would like to express our very great appreciation to Dr. John Dingliana for his valuable and constructive suggestions during the planning and development of this game. We would also like to acknowledge our friend Ting Zhang for her ideas for the asset creation and to work in collaboration with the whole team.