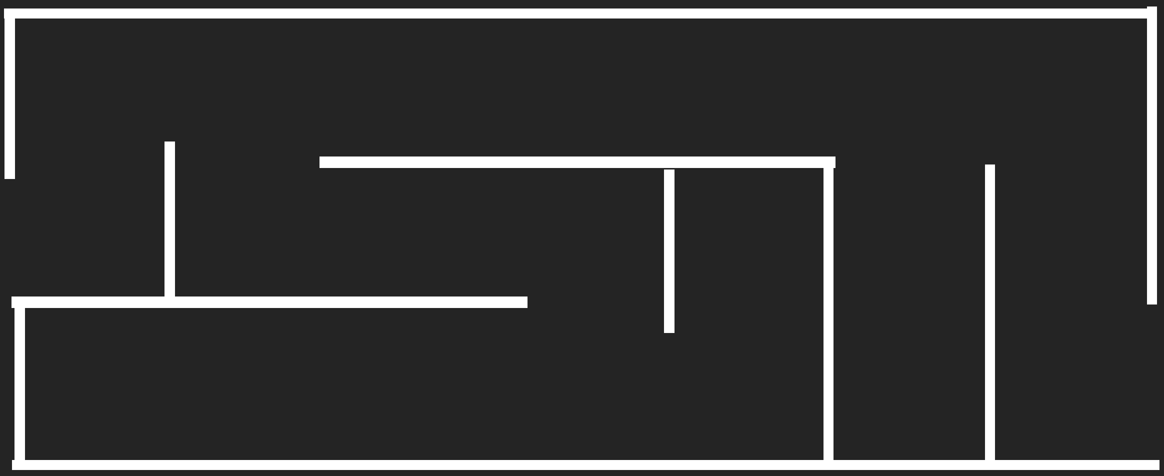


MAZE

ANCIENTS GODS LABYRINTH

WHAT

Final Report

WHERE

3D Game Programming

NCTU 2016

WHO

Zhang Zhexian - 0545080
zhangzhexian@outlook.com

Fanuel Wahjudi - 0556162
fanuelwahjudi.edu@gmail.com

Setiawan Wibowo P. - 0556165
setiawantong.cs05g@nctu.edu.tw

Content

Introduction

Jefa is the first game released by our team that designed and produced with unity3D. This game redefines traditional maze game in a 3D cube world which makes it engaging yet challenging.

Main features of Jefa include 3D cube maze minimap, multiple game levels and scenes, intuitive character movement control, as well as self-developed 3D models, sound effects and particle effects.

Game Story

In the beginning, when the universe was created, the gods brought life to the world by creating different creatures. Each god was responsible for creating one type of creature. After all animals and plants are created, Jefa, the god of wisdom, created human.

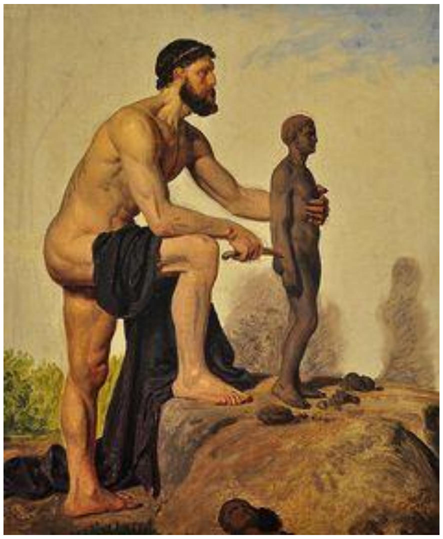


Figure 1. Jefa created human

Jefa loves the human beings he created, that he decided to go down to earth to live with them. He taught human beings the skills of making fire, healing, reading and writing.

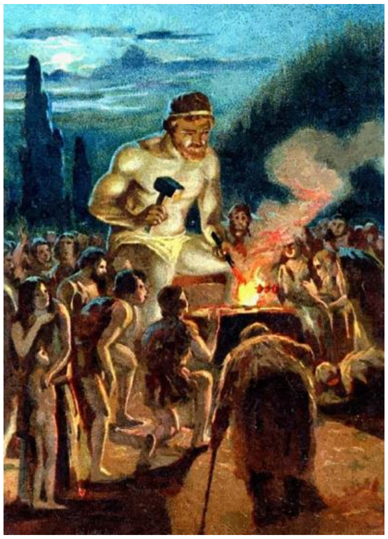


Figure 2. Jefa lives with human

Most of the other gods admired Jefa's creation, and were amazed by human beings' ability to think and learn. However, several gods were worried about human becoming too powerful to be controlled. They created a complicated maze on earth, and imprisoned Jefa, preventing him from returning to heaven to continue help human beings.

The maze is guarded by titans. The only way to get out of the maze is by collecting gems to unlock the exit gate. The gems are scattered in all three worlds: the land world, the sea world, and the sky world.

Game Type & Game Target

Jefa is a maze-type game. A maze itself means a path or collection of paths, typically from a starting point to a finish point. There is no specific gender or age to play this game, making this game playable for everyone else. Besides categorized as a maze game, Jefa can be categorized as an arcade game, regarding of the timer which forces the player to finish the game before it runs out.

The gameplay requires the player to collect gems which are scattered randomly in the maze. After a number of gems collected, the player must find the finish point which is also placed randomly in the maze. There are currently three levels in this game, and player needs to complete the existing level before unlocking the next one

Game Platform

This game is built on Unity game engine.

Requirements:

- Intel core i5 4th Generation or Higher
- 4GB of RAM
- 1 GB free space
- Nvidia GT 750M AMD Radeon HD 8790M, must support texture clamping.
- Windows 8.1 x64 or above
- Generic Mouse and Keyboard
- Speaker (Optional)
- Resolution: wait for build

Game Design

Motivation

The maze game genre is selected as we wish to create a game that not only engages players, but also educates them, in this case helping them practise the skills of navigating in complex paths with a reference map. We recognises the importance of spatial navigation skill especially for traveling and driving, and are proud of our game's potential benefits.

A 3D maze is created on top of the maze theme to add the challenging element and make our game stand out. The 3D maze with a finishing point on a different side from the player's start position encourages the player to refer to the minimap frequently. The point system that requires player to collect items from all sides of the maze forces the player to explore the entire complexity of the 3D cube map.

The hero's journey theme is adopted in our game as a trip of a central character that resolves a problem. Due to its simplicity and popularity, it allows us to outline the initial story and converge to a final story structure efficiently.

The view from player's point of view as the hero engages and encourages players to undertake the game mission. The story context of maze in three worlds: the land world, the sea world, and the sky world provides an opportunity for interesting and diverse graphics too.

Game Flow

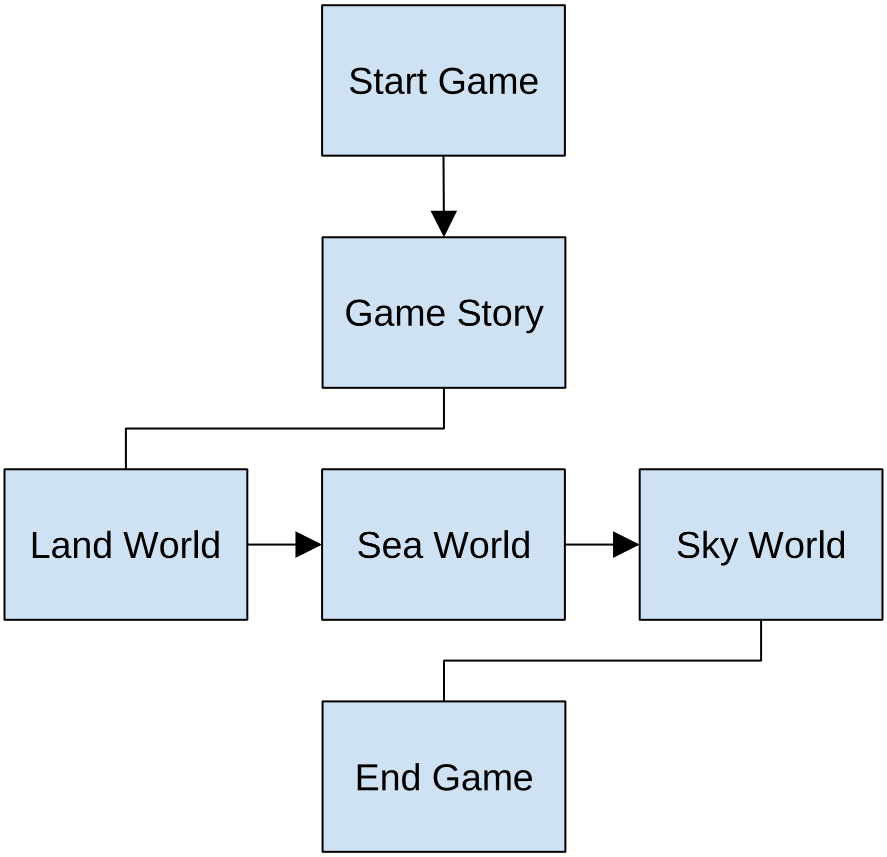


Figure 3. Main game flow

Game Level

This game contains of three levels, the land (level 1), the sea (level 2), and the sky (level 3). The ground level has a maze with its size 10x10. The complexity of the maze is highly reduced to create multiways and less dead-end.

User Interface

Jefa is the game that have the background that related with ancient god. To support this we design the menus interface as simple and have ancient look. We use the same background as the panel for button in the main menu, play menu, option menu, and credits menu.

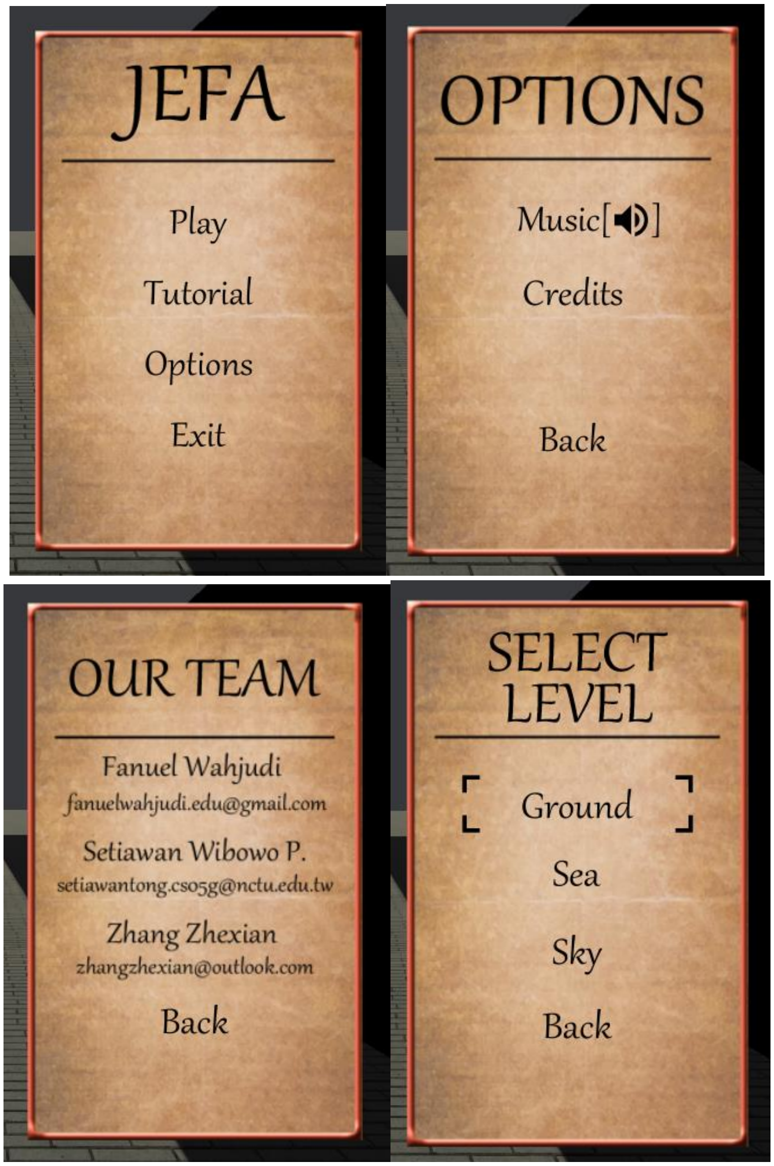


Figure 4. Menus Design
(Top Left-Main Menu, Top Right-Options Menu,
Bottom Left-Credits Menu, Bottom Right-Play Menu)

To strive for consistency and visual effects, we choose ‘gabriola’ font for all game buttons and overlay titles. When player’s mouse hovers over a button the corner decoration is shown for button hovering feedback.

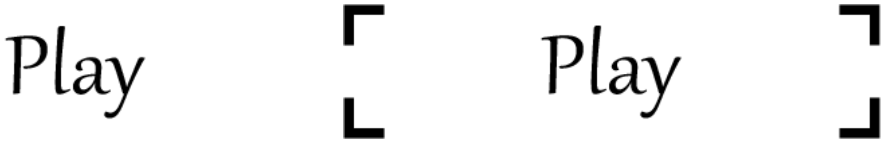


Figure 5. Buttons Design
(Left - Normal Play Button, Right - Play Button on hover)

]3D Game Objects

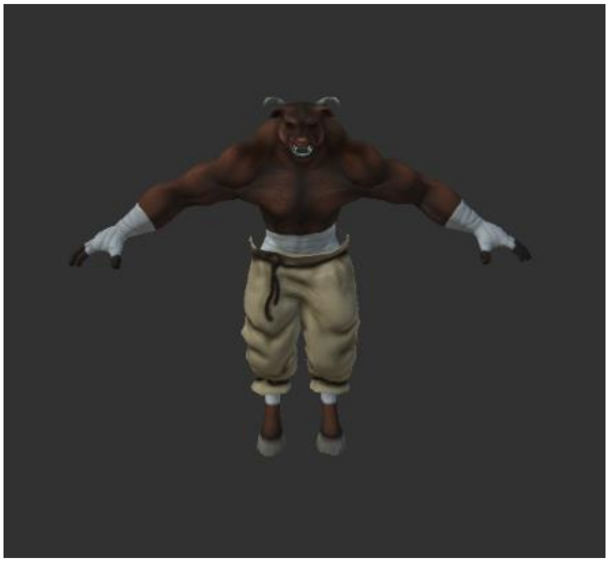


Figure 6. Character Main Player

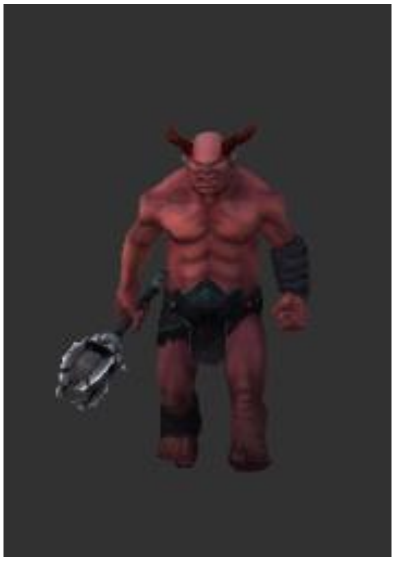


Figure 7. Character AI

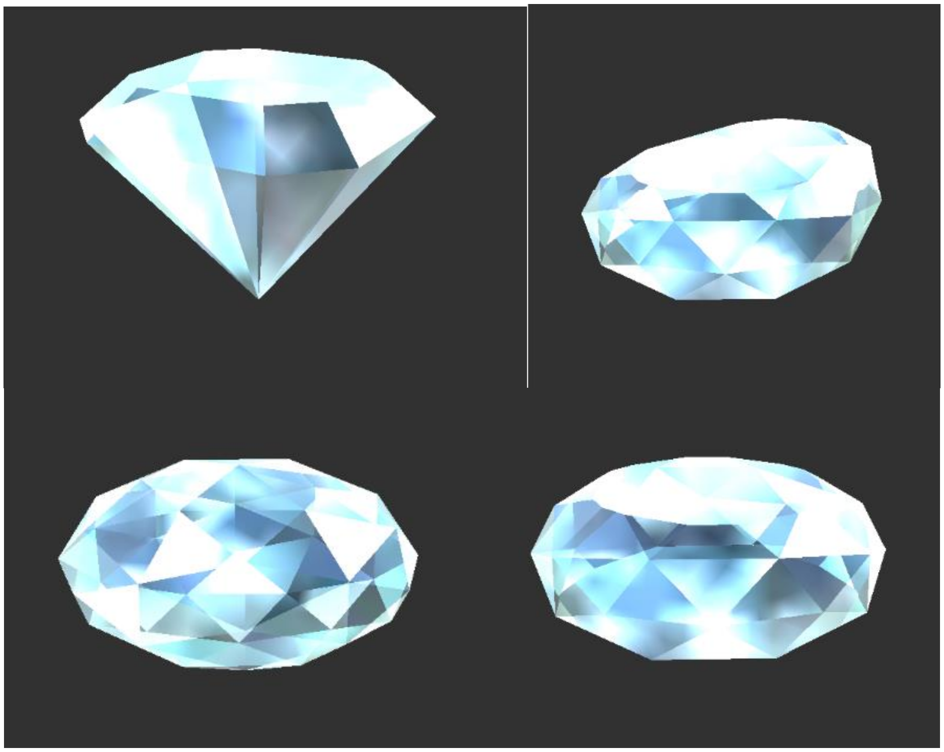


Figure 8. Gems

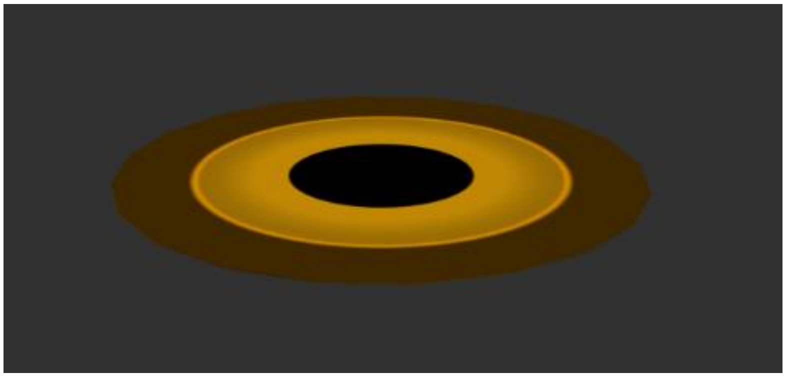


Figure 9. Teleport Point

Game Scene

Each level of the game scene shows the characteristic of that world.

In the land world, the ground is covered with grass texture, the skybox shows white cloud on blue sky, and the maze walls are made of rock models.

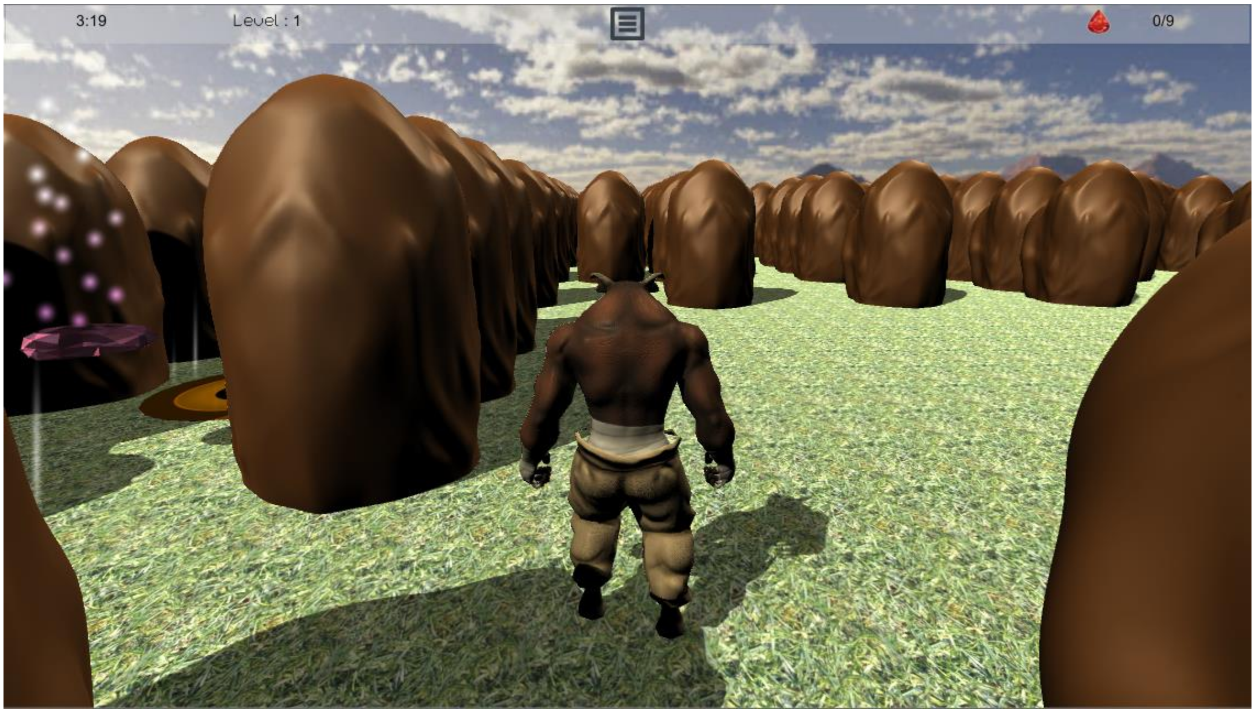


Figure 10. The land world

In the sea world, the ground is covered with water texture, the skybox shows blue crystal like atmosphere, and the maze walls are made of coral models.

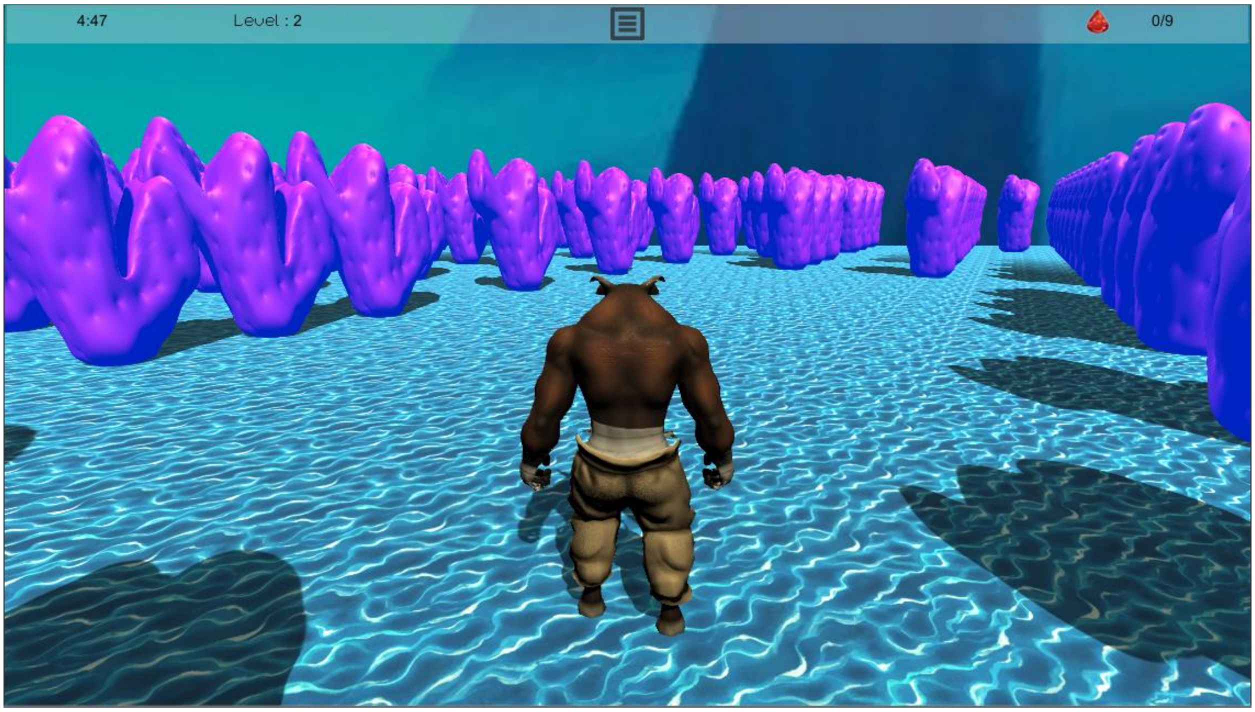


Figure 11. The sea world

In the sky world, the ground is covered with white texture that shows light blue shadow, and the maze walls are made of cloud models.

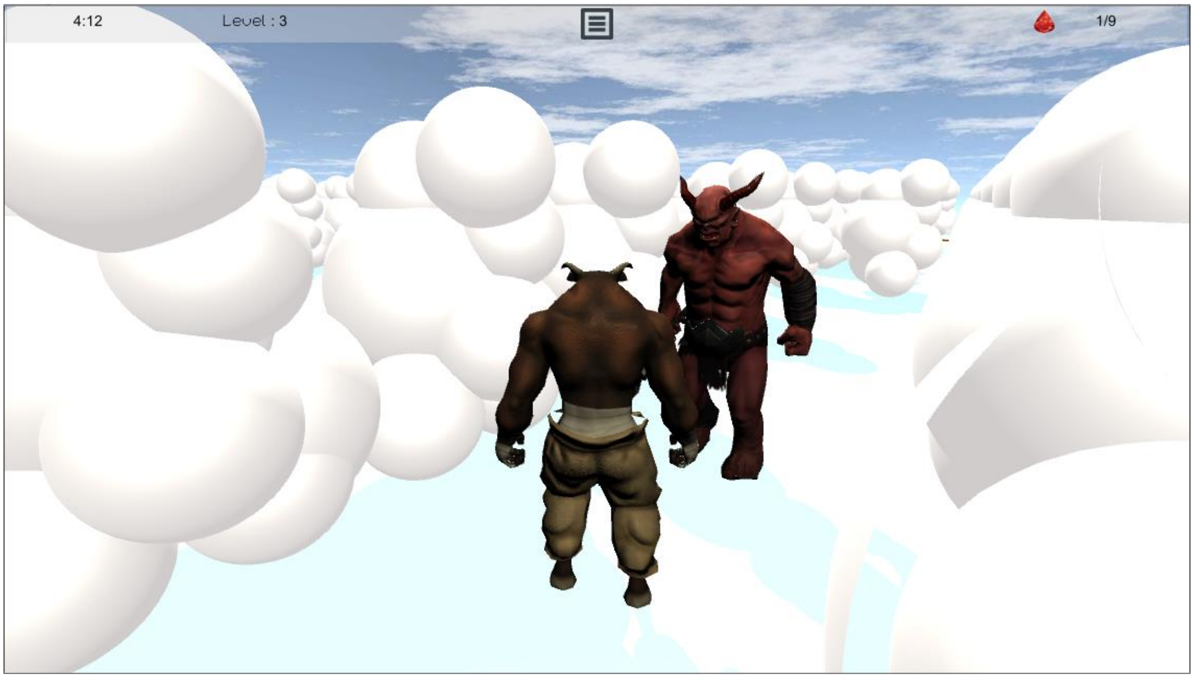


Figure 12. The sky world

Game Analysis

Scene System

There are four scenes in our game: the menu scene, the introduction scene, the game scene (the tutorial scene is based on game scene with some modifications), and the map scene.

We adopt the scene system for modularization of programs, so the scenes become independent and may be reused easily. It is also easier for game state transition and the state of a scene may be preserved even when another scene is loaded.

There is one more added value of having multiple scenes: it is easier for collaboration, as when there are editions on different scenes instead of all on the same scene, there are less conflicts or concurrency problems.

Mission System

We integrated mission system to motivate player to complete the given tasks. By the hero's story, the main character is trapped in the land world, and need to find special gems to unlock the exit.

If the main character is seen and caught by the maze guardian, he will be sent back to the starting location and part of the gems will be taken away.

The mission has multiple layers too. When one world is conquered, another world would appear with even more challenges to engage the player.

Character System

Our characters are simple and straightforward. There is one player and one non-playable character (NPC).

The player, i.e. the main character, is Jefa the god, also the hero in the hero's journey storyline.

The NPC is the guardian of the maze who aims to keep the main character inside the maze. It could not be be killed, but to be avoided. Once it catches the main character, the main character will be chased back to its starting point, and some of the gems will be lost.

Navigation System

To navigate among different sides of the maze, special teleportation points are added to the game as shown below:

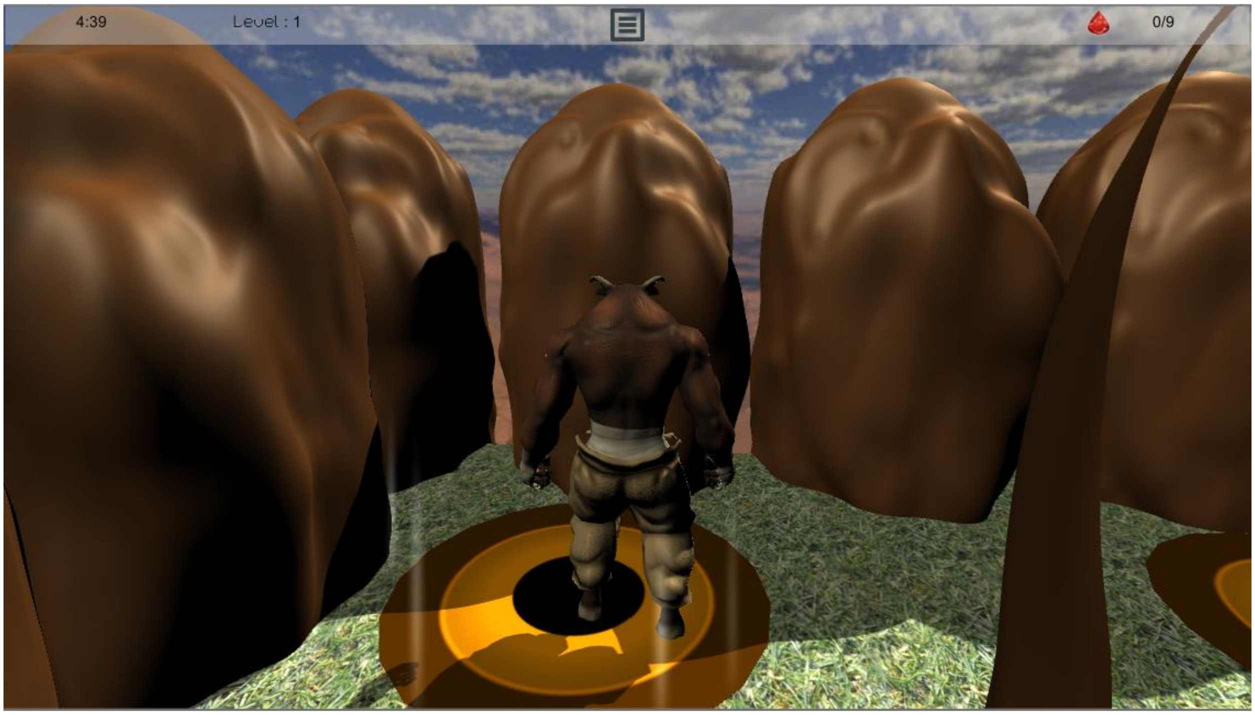


Figure 13. Teleportation Point

When the player stand on a teleportation point, it will be teleported to another side of the maze. The side to be teleported may be inferred from the following cube map:

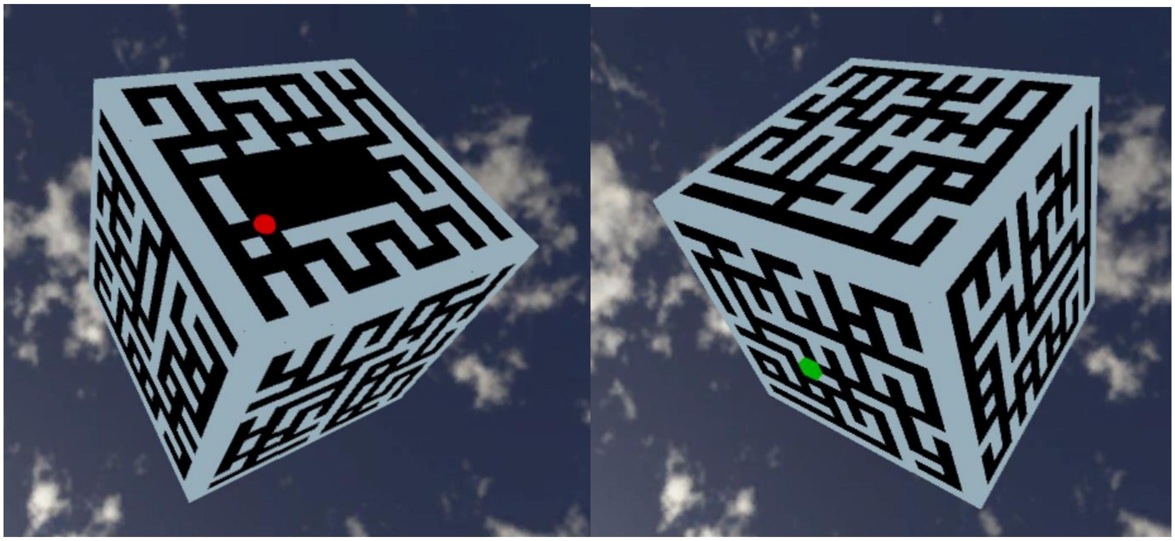


Figure 14. Cube maze map

Note that the red dot shows the player’s current location. This allows player to navigate in the labyrinth more easily. The green dot is the exit of the maze.

Control System

The game may be controlled by both mouse and keyboard.

In the game scene, the mouse movement will decides the view angle of the player, and the keyboard allows player to move the character in all four directions.

In the map scene, we implement the same set of keys to rotate the cube map, so as to minimize the learning curve of controlling the game in our multi-scene system.

Item System

Our main item in the game are the gems. Besides the gems, there are non-collectable items such as the teleportation point and the finishing point.

The collectable gems are generated at random locations in the scene. Locations of the teleportation points and finishing point, however, are predefined and will not change for a given round of the game.

There are particle systems above all the said items to guide the players to find them, even when the view is partially blocked by the maze walls.

Camera System

In the tutorial and the game scene the camera shows a 3rd person view. It is a perspective in which the player can visibly see the body of the controlled character. Below summarizes the advantages of using the 3rd person camera view:
- improved character visibility
- wider field of view
- easier to locate the items or the monster near the player

Level System

The game has three levels. There is a level saving functionality, that the levels unlocked will be stored in a file and the level may be resumed even if the game is accidentally closed.

Life System

The player wins a level when she collects enough gems and successfully find the finishing point.

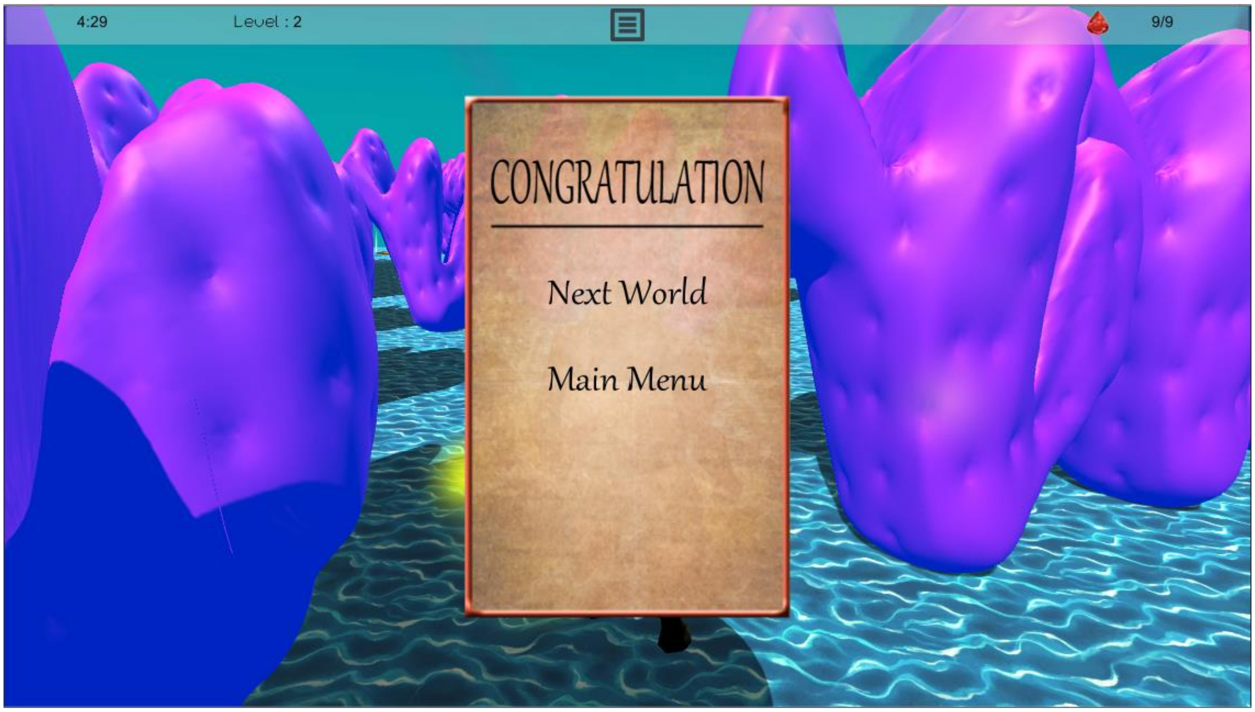


Figure 15. Game win scene

The player will not be killed by monster. However, it is considered game over when the count down timer reached zero.

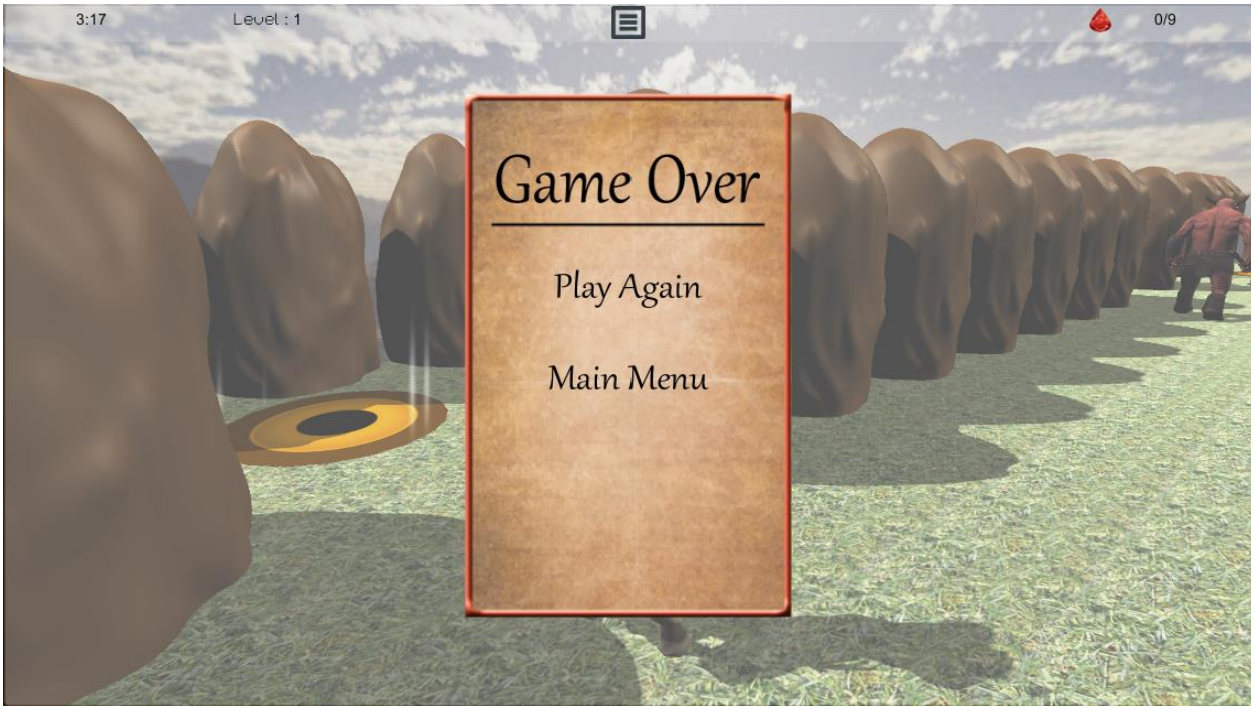
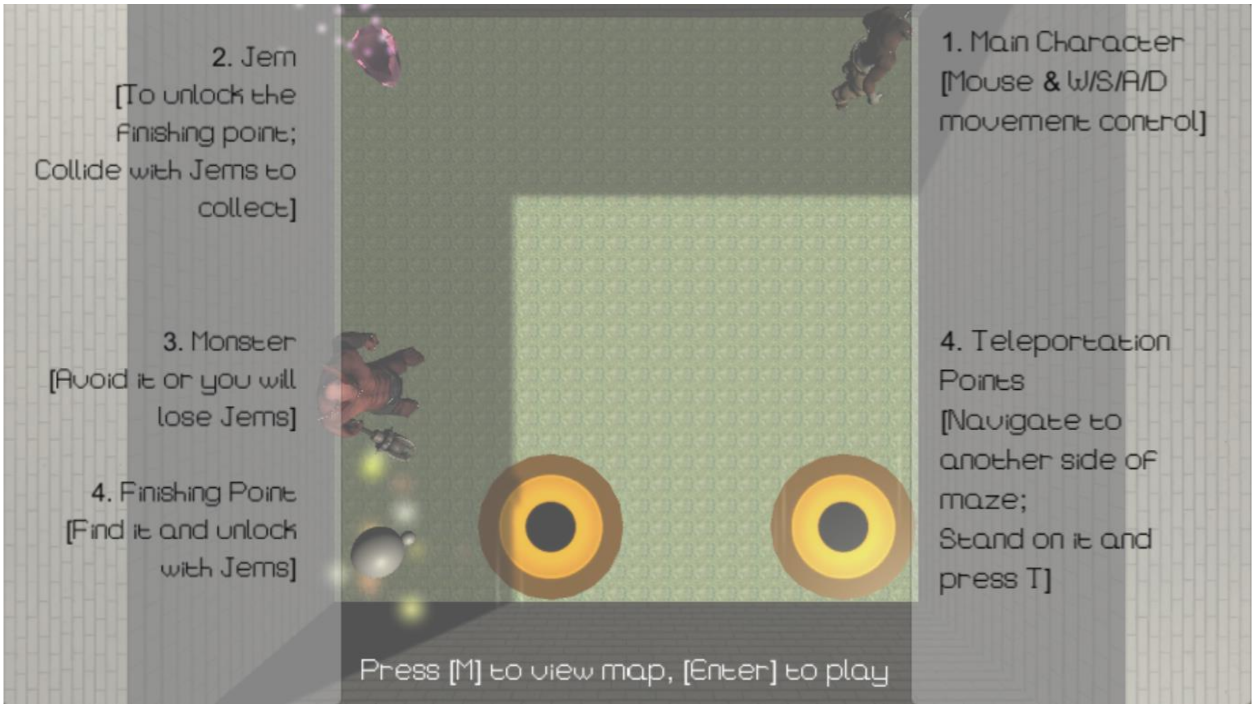


Figure 16. Game over scene

Help System

To help new user get familiar with the game, a tutorial is created to teach player the basics of the game.

As you may see, there are numbered legends describing the roles and control of each item in the game.



1. Main Character [Mouse & W/S/A/D movement control]

2. Jem [To unlock the Finishing point; Collide with Jems to collect]

3. Monster [Avoid it or you will lose Jems]

4. Finishing Point [Find it and unlock with Jems]

4. Teleportation Points [Navigate to another side of maze; Stand on it and press T]

Press [M] to view map, [Enter] to play

Figure 17. Tutorial scene

In addition to tutorial, there are legends in the game too to make the game user-friendly.

For instance, the following is legend in game scene, showing countdown timer, game level, menu button, as well as gem count.

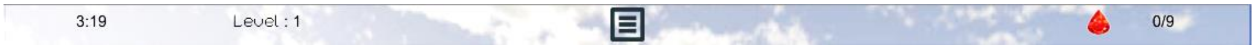


Figure 18. Game scene legend

The next legend is in map scene. There are location indicator and control keys explanations.

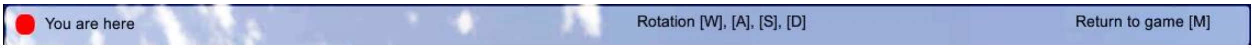


Figure 19. Map scene legend

System Architecture

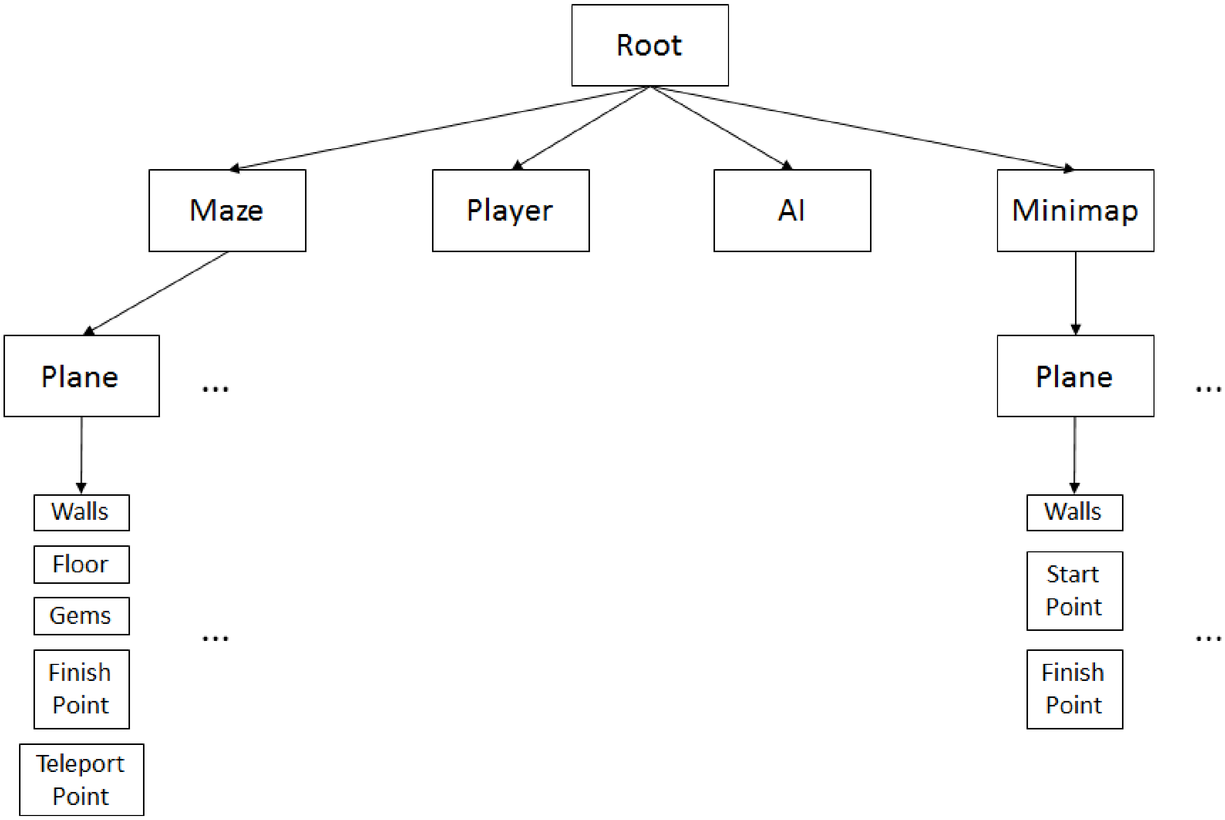


Figure 20. Objects Hierarchy

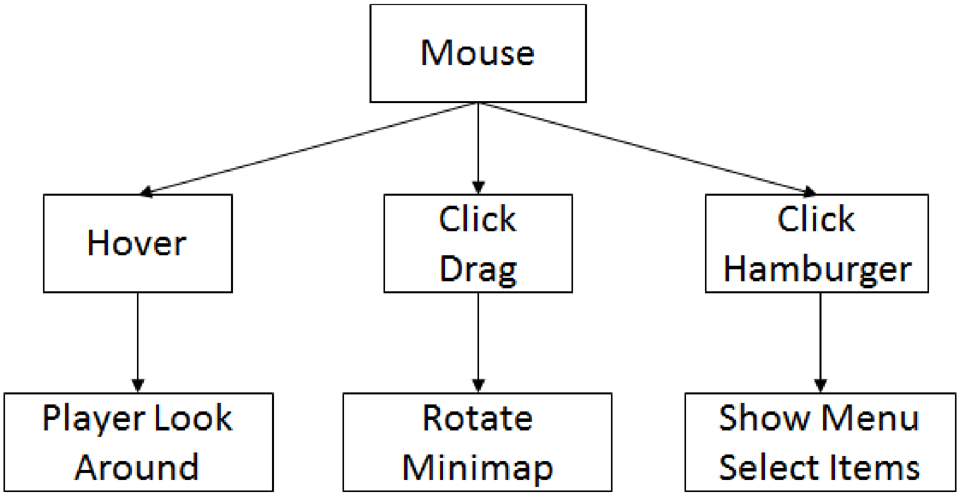


Figure 21. Controller Hierarchy (Mouse)

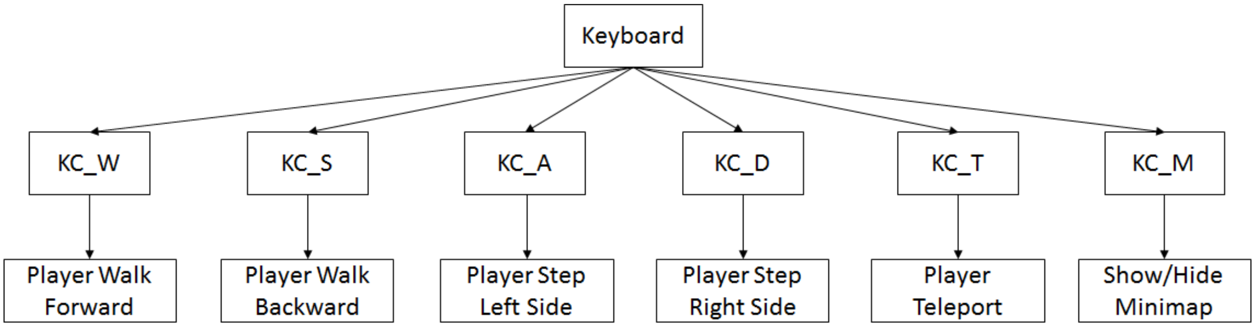


Figure 22. Controller Hierarchy (Keyboard)

Technical Document

Maze and 3D Map Generation

To create the maze, recursive division algorithm is used. It works by dividing an empty space into two division. One part of separator line that separate those two divisions are selected randomly and removed to create a way that connects those two division. For each division that is created, are divided again with the same method to create smaller division. This algorithm are repeated until there is no dividable space, which means that there are only the smallest space available. The visualisation of this algorithm can be seen on Figure 23.

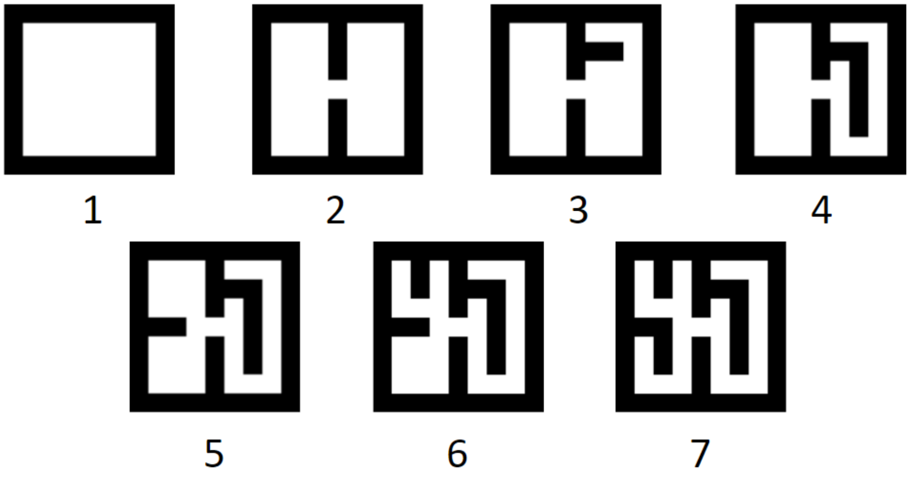


Figure 23. Recursive Division Technique

Mini Map

The cube map is created by assembling the six faces of game maps into a cube, by carefully arranging the faces' facing direction, orientation, as well as position. Testing is done to confirm that the relation between sides are accurate.

Also in the cube map, the goal position will be obtained from the map data and be represented with a green square. The player position is trickier, as it needs to be updated whenever the player moves. We thus store the player position in a global variable, and access the variable to update player position, represented with a red dot.

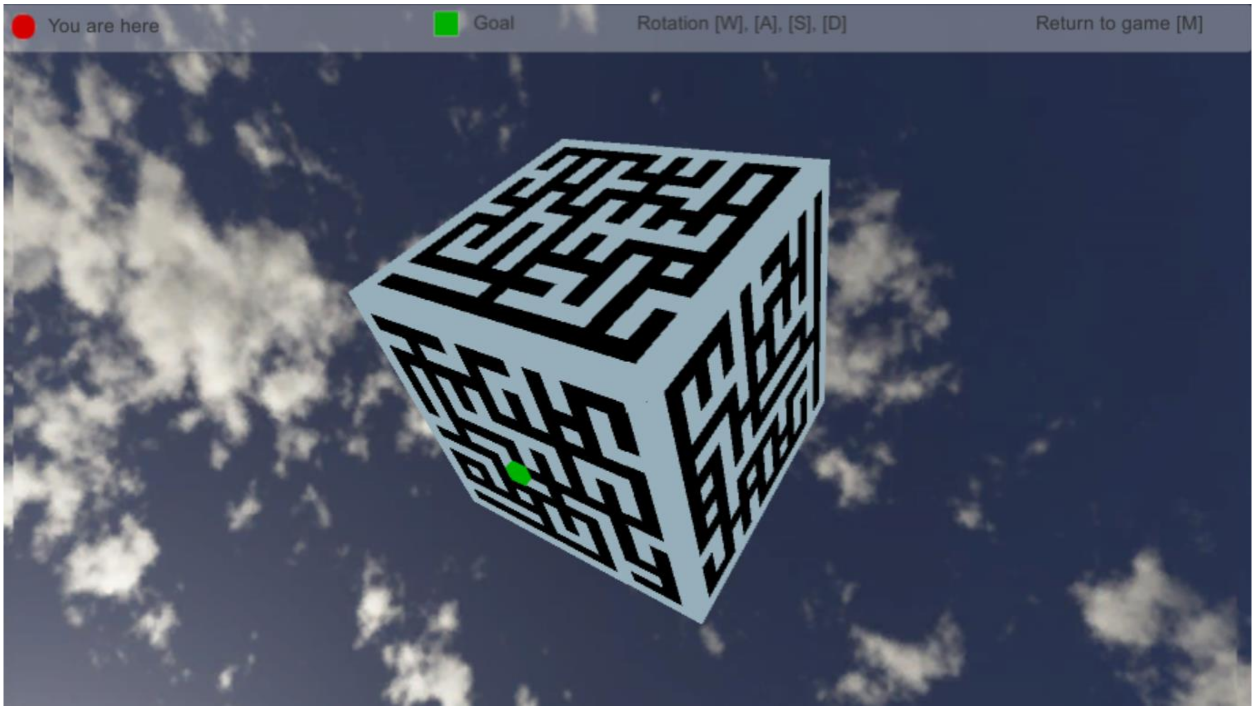


Figure 24. Mini map (3D)

Character Movement

[Keyboard] A - Moving Forward
[Keyboard] S - Moving Backward while player still facing forward
[Keyboard] W - Side walk to the left while player still facing forward
[Keyboard] D - Sidewalk to the right while player still facing forward
[Mouse] Swipe Left - change the facing orientation to the left direction
[Mouse] Swipe Right - change the facing orientation to the right direction

Character Behaviour

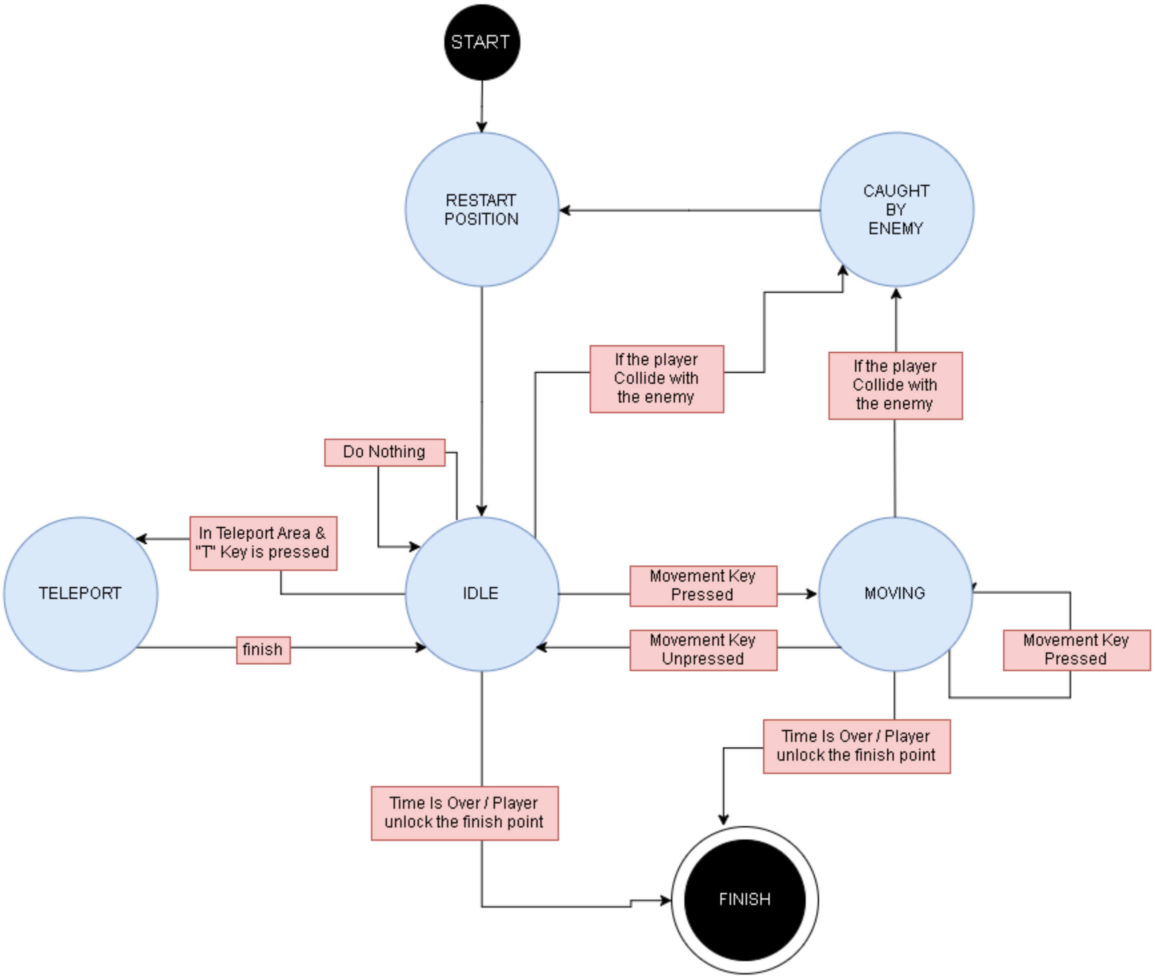

START
RESTART POSITION
CAUGHT BY ENEMY
If the player Collide with the enemy
If the player Collide with the enemy
Do Nothing
In Teleport Area & "T" Key is pressed
TELEPORT
IDLE
Movement Key Pressed
MOVING
finish
Movement Key Unpressed
Movement Key Pressed
Time Is Over / Player unlock the finish point
Time Is Over / Player unlock the finish point
FINISH

Figure 25.Character's State Machine

When the game starts, the player character will be located in the initial position in a maze and initiate the “idle” state. When the player pressed the movement key (A , W , S , D) in keyboard, the state changed to “moving”. If the player stop press the movement key, it will back to “idle” state. When the player is in the teleport point area, player is able to teleport to another side of cube maze. The “teleport” state happens under this situation, when the player press “T” key which is the teleport button. After the player is teleported to another side of maze it will come back to “idle” state. While in the “idle” state or “moving” state, the player is possible to collide with the enemy, and the player will be caught by enemy, and the position of the player will be relocated to the initial position as the same as the beginning of the game.

Artificial Intelligence (AI)

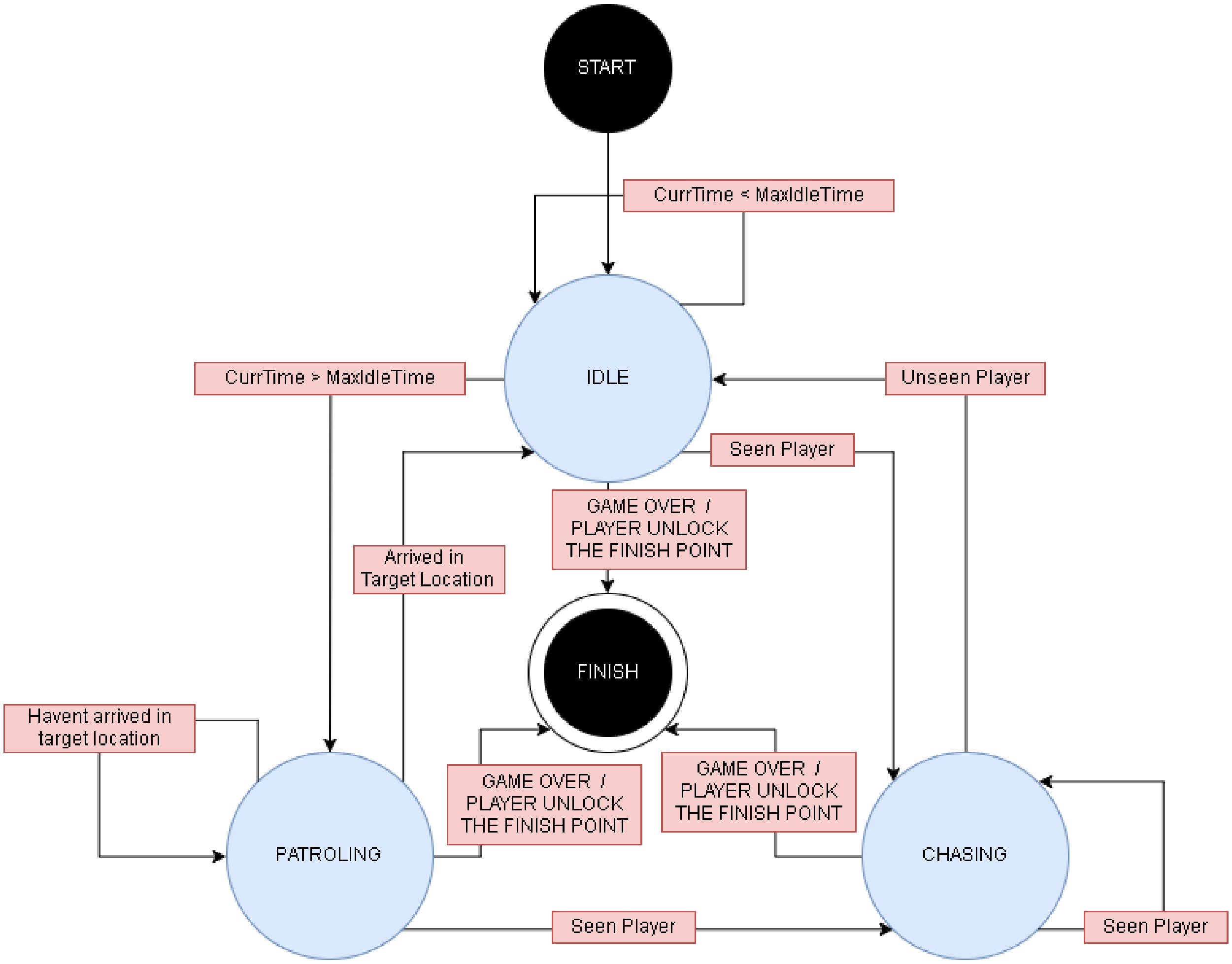


Figure 26. AI Machine State

In this game, we have a monster that implement AI. AI behaviour quite intuitive. When the game starts it is initiated with “idle” state. In idle state It does the idle for up to 3 second, and then changes the state to “patrolling”.

In patrolling state, first the AI will decide the target location, then it will calculate the path using A* algorithm, and then it moves to the target location. When the AI reach the target position, it change the state to idle state, and do the loop as long it doesn't see the player.

Seeing the player mean when the range between player and the AI is below the AI’s seen range and the player is not in the start position. When the AI sees the player it will immediately change the state to “chasing” state.

In chasing state first it determines the player’s location, then it calculates the path using A* algorithm. Then, it will move to the calculated position. For every 3 seconds it will update the player position and recalculate the path. And when it hasn't seen the player again, it change the state to idle again.

Collision Detection

There are two collision detection methods that are used in this game. The first method is the basic collision detection. If the object is closer than its bounding mesh, then it is colliding. This method can be done easily with Unity. The game object only needs a collider which can be seen on Figure 27. If those objects touch each other, the object are colliding.

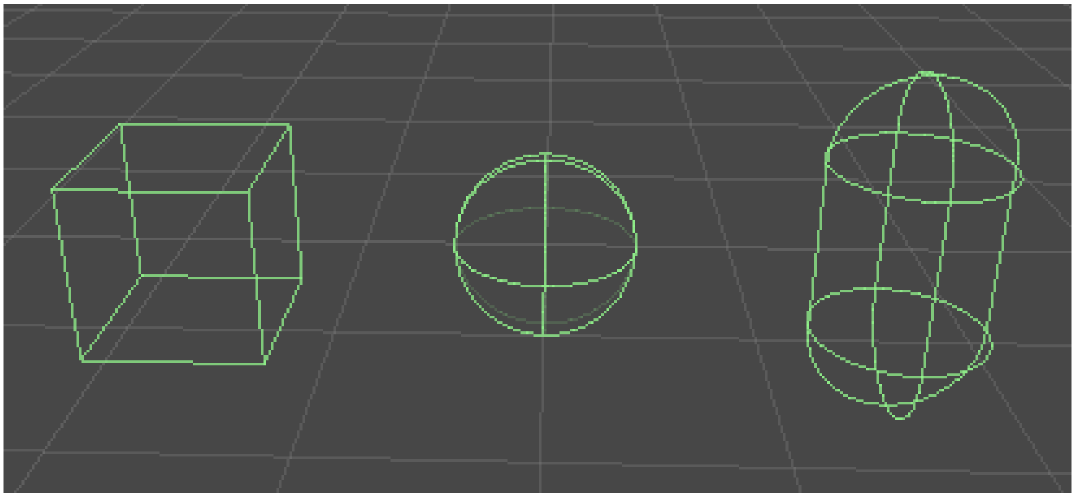


Figure 27. Bounding Box Technique

The second one is same-coordinate technique. We create this method to emphasize the effect of classic maze. Instead of giving an object a mesh collider, it checks the coordinate on the map of each objects. If two objects are on the same coordinate which can be seen on the right picture on Figure 28, it will collide. But if the coordinate position is different which can be seen on the left picture on Figure 28, it will not collide no matter how big the object is and clearly visible that it is colliding with naked eye. This method must be used carefully to prevent the effect that the object is colliding on vision, but is not colliding in the system. If this method is used, the first thing is to make sure that the size of the object is not bigger than the size of 1 area in the maze.

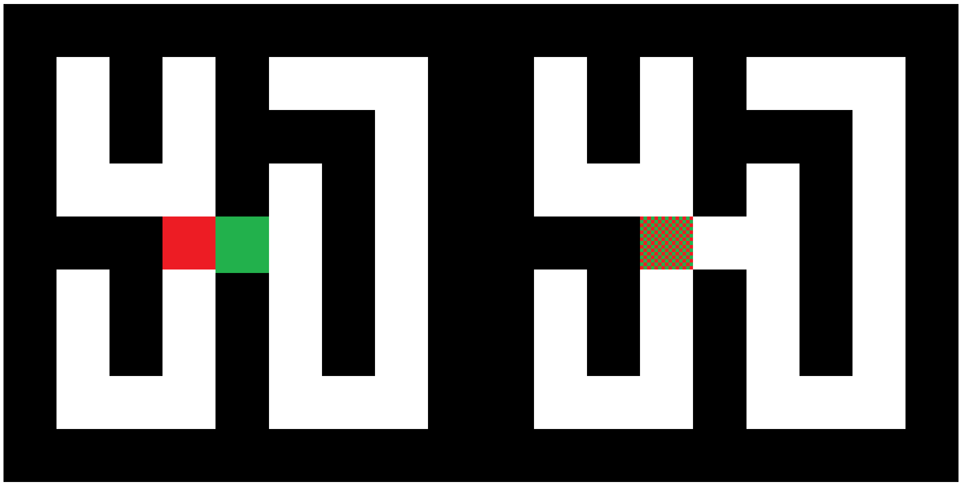


Figure 28. Same-Coordinate Technique

Camera Movement

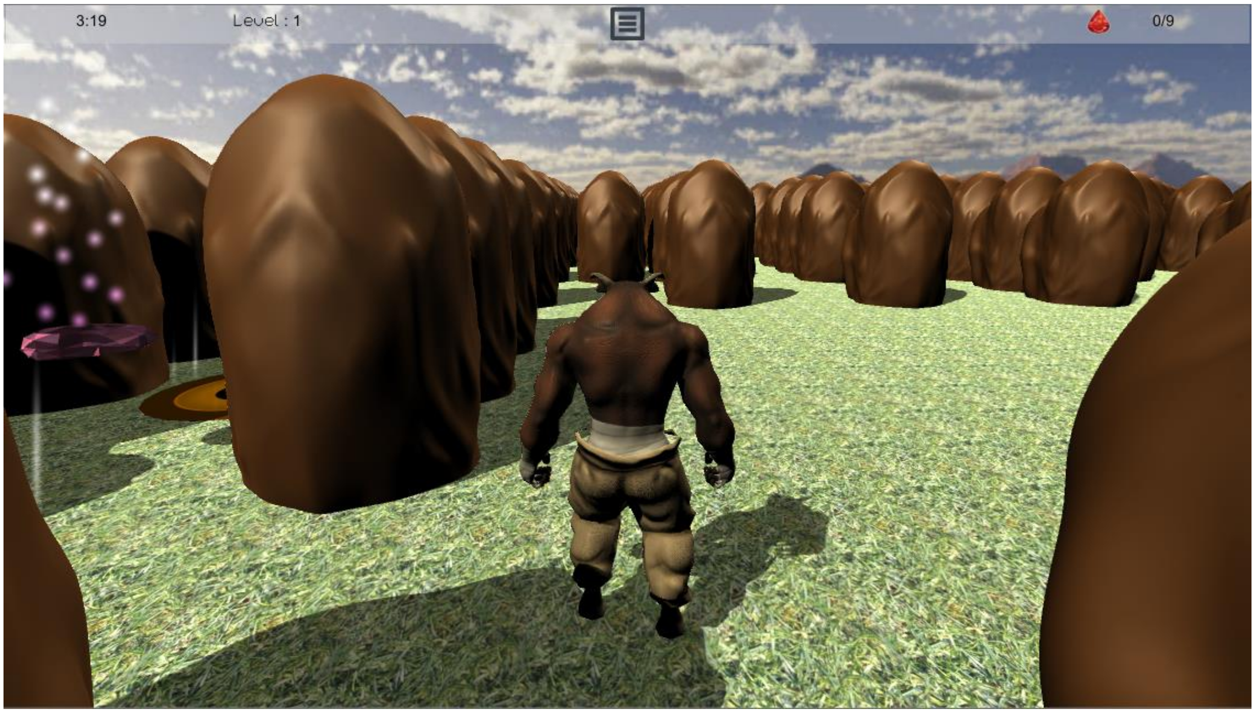
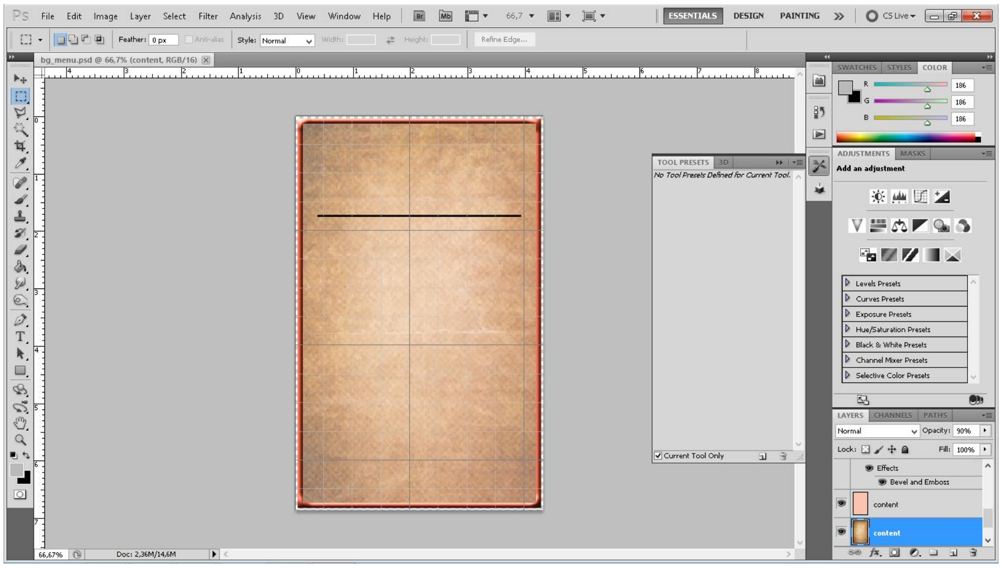


Figure 29. 3rd person Camera

In the game the camera located behind the player. The camera acts as 3rd person camera. It will move follow the direction of the player facing. Whenever player is moved by keyboard control movement or mouse control movement, the camera will move as well.

User Interface

User Interface of this game was created with the aid of photoshop. For the buttons the ‘gabriola’ font is used. It is simple and matches with the background of the menu, and shows the ancient feel in this game. The background of the menu is created by old paper textures for similar ancient look. Similarly, the introduction story background uses another type of old paper texture.


bg_menu.psd @ 66,7% (content, RGB/16)

Feather: 0 px

Style: Normal

R 186 G 186 B 186

No Tool Presets Defined for Current Tool.

Normal Opacity: 90%

Fill: 100%

Effects

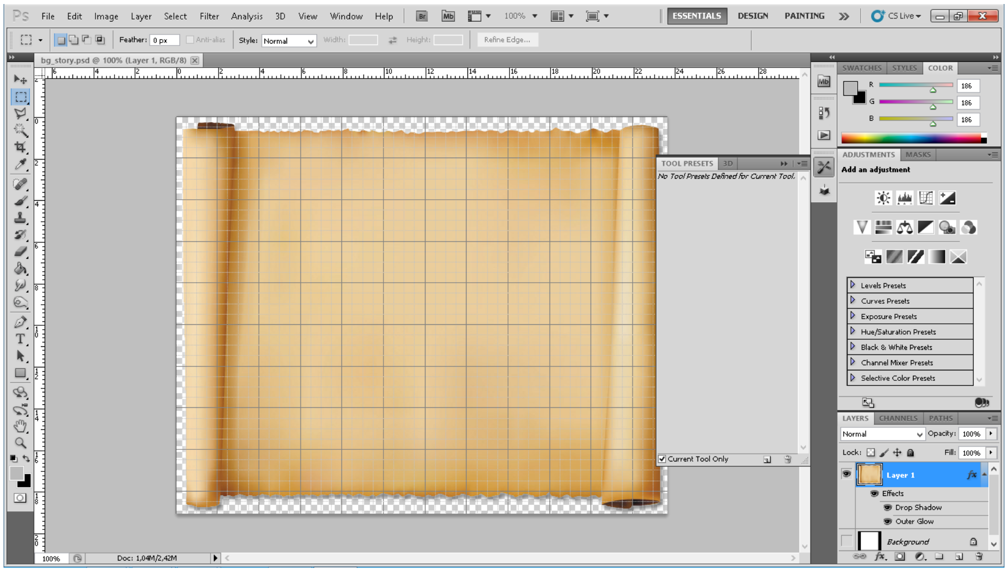
Bevel and Emboss

content

content

66,67% Doc: 2,36M/14,6M

Figure 30. Menus Background


bg_story.psd @ 100% (Layer 1, RGB/8)

Feather: 0 px

Style: Normal

R 186 G 186 B 186

No Tool Presets Defined for Current Tool.

Normal Opacity: 100%

Fill: 100%

Layer 1

Effects

Drop Shadow

Outer Glow

Background

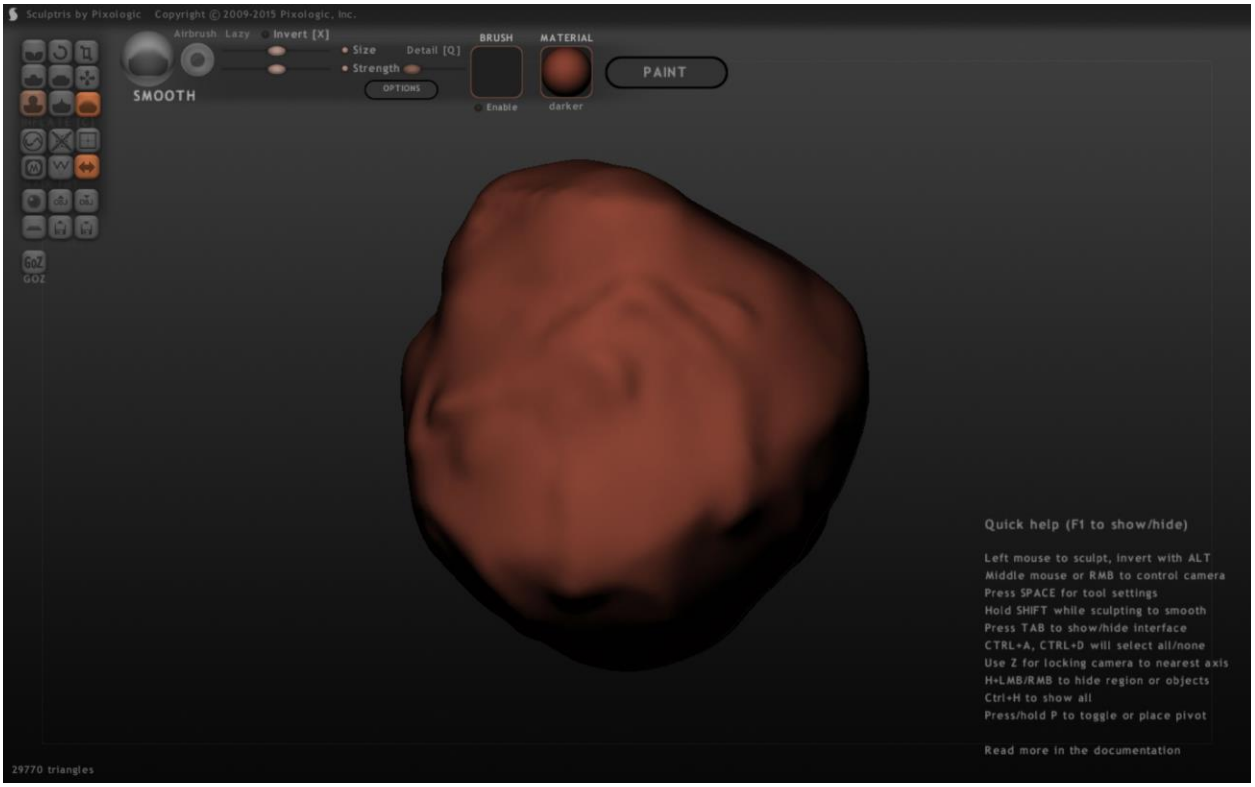
100% Doc: 1,04M/2,42M

Figure 31. Introduction Story Background

Custom Meshes

The meshes are made with free software Pixologic Sculptris (link in reference section). It allows user to shape a sphere (the base geometry) into the form they like, by means of pinching, flattening, smoothing, etc.

Using the software, three meshes: rock, coral and cloud are created as shown below:

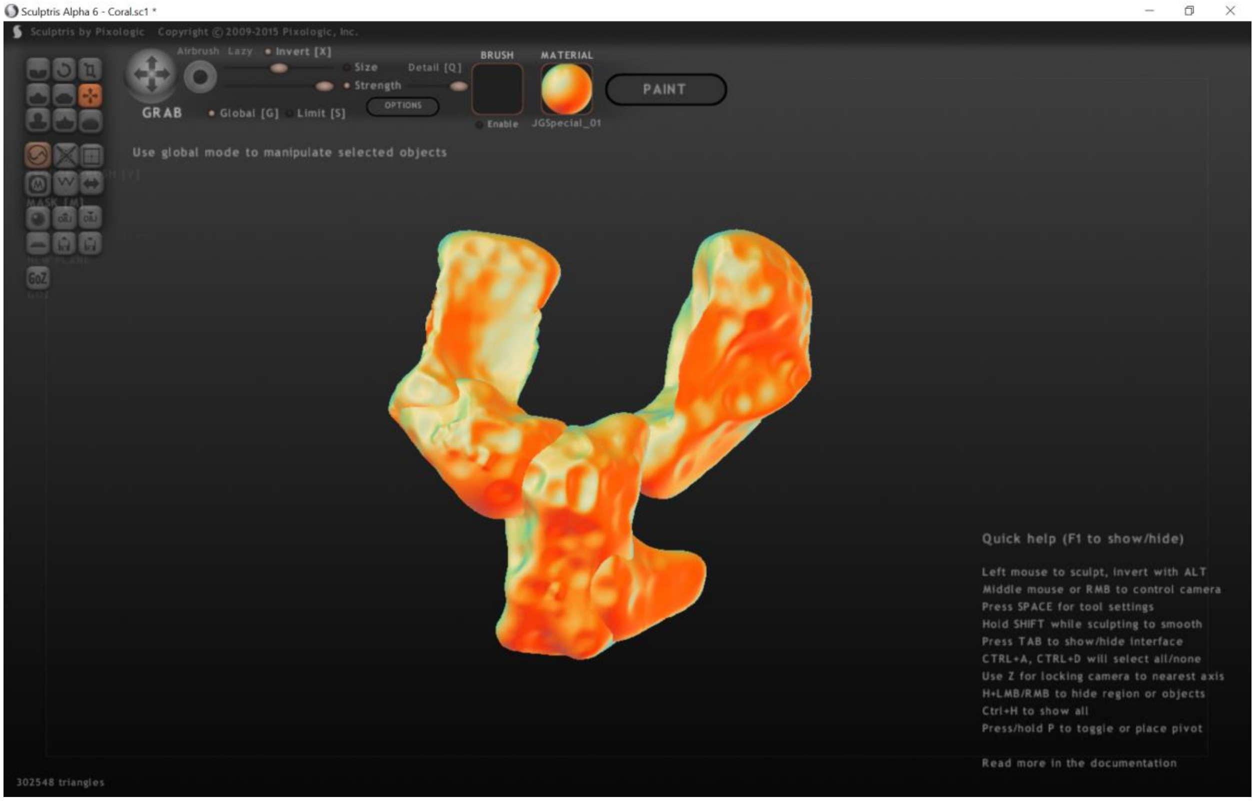


Quick help (F1 to show/hide)

Left mouse to sculpt, invert with ALT
Middle mouse or RMB to control camera
Press SPACE for tool settings
Hold SHIFT while sculpting to smooth
Press TAB to show/hide interface
CTRL+A, CTRL+D will select all/none
Use Z for locking camera to nearest axis
H+LMB/RMB to hide region or objects
Ctrl+H to show all
Press/hold P to toggle or place pivot

Read more in the documentation

Figure 32. Land world mesh (rock)



Use global mode to manipulate selected objects

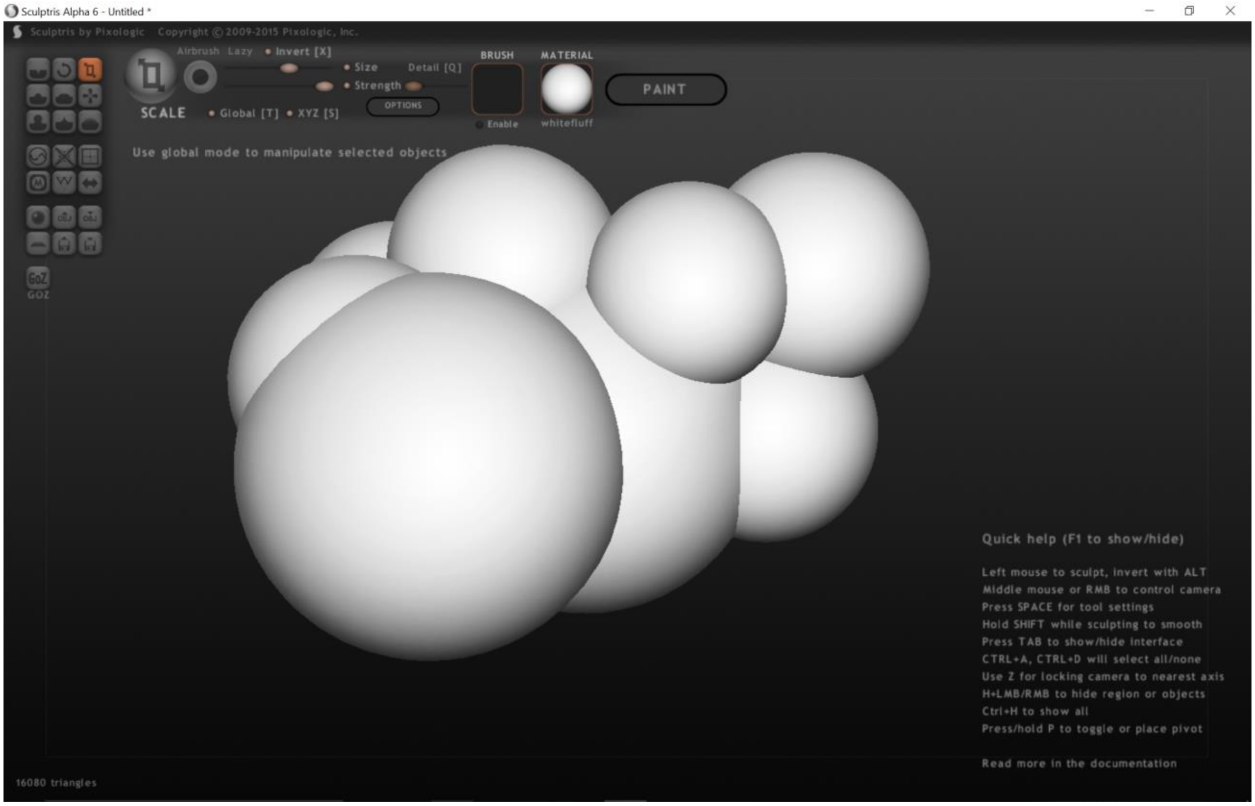
Quick help (F1 to show/hide)

Left mouse to sculpt, invert with ALT
Middle mouse or RMB to control camera
Press SPACE for tool settings
Hold SHIFT while sculpting to smooth
Press TAB to show/hide interface
CTRL+A, CTRL+D will select all/none
Use Z for locking camera to nearest axis
H+LMB/RMB to hide region or objects
Ctrl+H to show all
Press/hold P to toggle or place pivot

Read more in the documentation

302548 triangles

Figure 33. Sea world mesh (coral)



Use global mode to manipulate selected objects

Quick help (F1 to show/hide)

Left mouse to sculpt, invert with ALT
Middle mouse or RMB to control camera
Press SPACE for tool settings
Hold SHIFT while sculpting to smooth
Press TAB to show/hide interface
CTRL+A, CTRL+D will select all/none
Use Z for locking camera to nearest axis
H+LMB/RMB to hide region or objects
Ctrl+H to show all
Press/hold P to toggle or place pivot

Read more in the documentation

16080 triangles

Figure 34. Sky world mesh (cloud)

Particle Systems

In Unity, creating particle system is easy as adding particle system on the desired object. However, customizing the particle takes a long time.

In this game, there are 6 particle systems which can be seen in Figure 35. On top left, the color of the particle changes gradually based on the lifetime of the particle. On top middle, the shape of the particle is small vertical line, and stay on the edge of teleport object. On the top right, each particle has its own unique color which is randomized from rainbow spectrum.On the bottom left, there is a particle system which is implemented on gem-water mesh, creating an effect of bubbles. On the bottom middle, the particle is similar to the top right picture, which emits with random color from a preset given. On the bottom right, the particle is softer than the other to make the effect of shiny particle.

These colors, shape, gravity, direction, and position of emitting can be easily modified on inspector tab. As an example, the start color, color over lifetime, color by speed, are parameters to change the color of the particle.

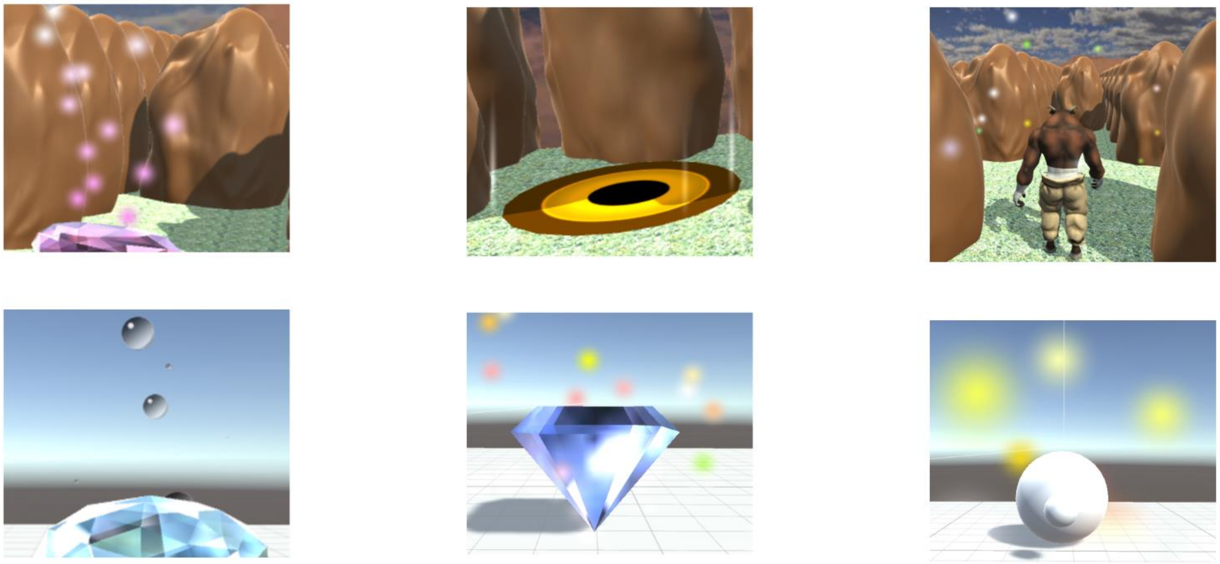


Figure 35. Particle systems used in the game

Sound Effects

There are 2 types of sounds used in this game, background music and sound effects. Due to limited human resource and time in the development of this game, background music are taken from artists. While sound effects are created by ourself with unique methods. Here are the list of background music and sound effects used in this game:

Background Music:

Main Menu

Land Level

Water Level

Sky Level

Sound Effects:

Button Click

Finish

Hit by monster

Foot Step Land

Foot Step Water

Foot Step Sky

Teleport

To create sound effects, digital audio workstation application, Acoustica Mixcraft 7 (Figure 36), and audio-editing application, Adobe Audition 1.5 (Figure 37), are used serially. Acoustica Mixcraft 7 is used to create the sound effect. It provides lots of sound effects, from percussion, choir, base, synthesizer, and many more. The application can be connected to MIDI controller, which usually comes in electric-piano shape to give a convenient way to type the music.

An example is finish sound effect, choir sounds are used to give the feeling of being on the ancient era with lots of princess singing together. This sound effect comes in major chord and transposed upwards every time to create the sensation of achieving something.

The other example is foot step which the sounds are recorded. For “foot step land” sound effects, it sounds like stepping on the ground. But the making of that sound is not by recording someone stepping on the grass, instead, a tissue paper is pressed to recreate the effect of stepping on a grass.

For “foot step water”, the sound of kicking in the water is created by fictioning a hand on the body of the laptop while the laptop’s microphone is recording. To make the effect of being underwater, the high frequency sound is eliminated using Adobe Audition. On the equalizer, the frequency from 8kHz above is cut 36dB.

The sound of “foot step sky” is similar to “foot step water”. To create the sound of the wind, which feels like an inaudible sound, the wave of the sound is modified using pitch bender, an effect editor on Adobe Audition. The wave is stretched to make the frequency lower while preserving its quality. With low frequency, the sounds seems to come from nearby *whooshing* to our ear.

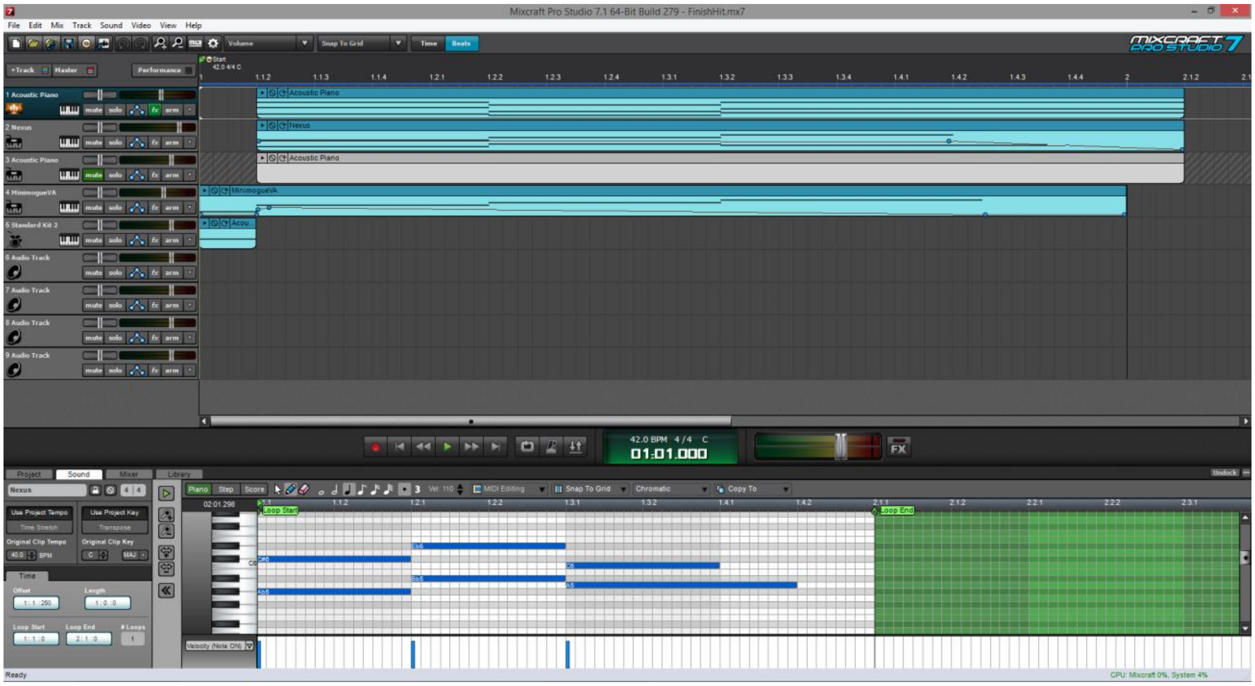
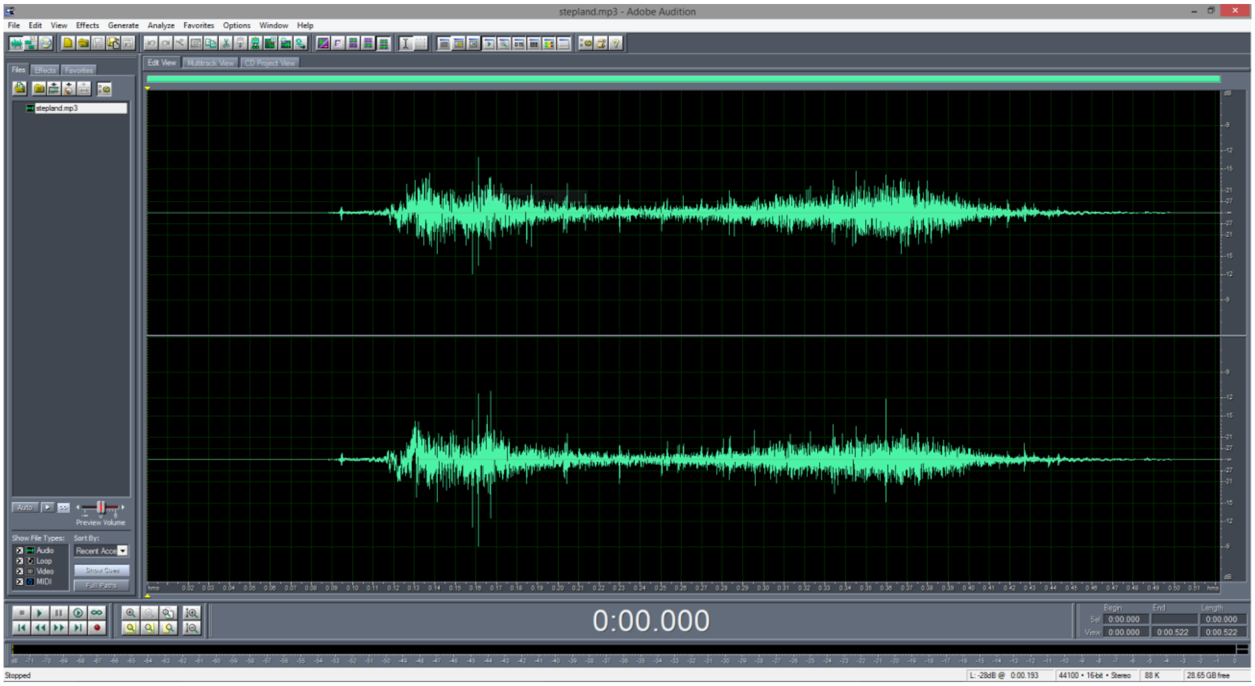


Figure 36. Acoustica Mixcraft


stepland.mp3 - Adobe Audition
File Edit View Effects Generate Analyze Favorites Options Window Help
Files Effects Favorites
stepland.mp3
Edit View Multitrack View CD Project View
Preview Volume
Show File Types:
Audio
Loop
Video
MIDI
Sort By:
Recent Acce
Show Cues
Full Paths
0:00.000
Stopped

	Begin	End	Length
Sel	0:00.000		0:00.000
View	0:00.000	0:00.522	0:00.522

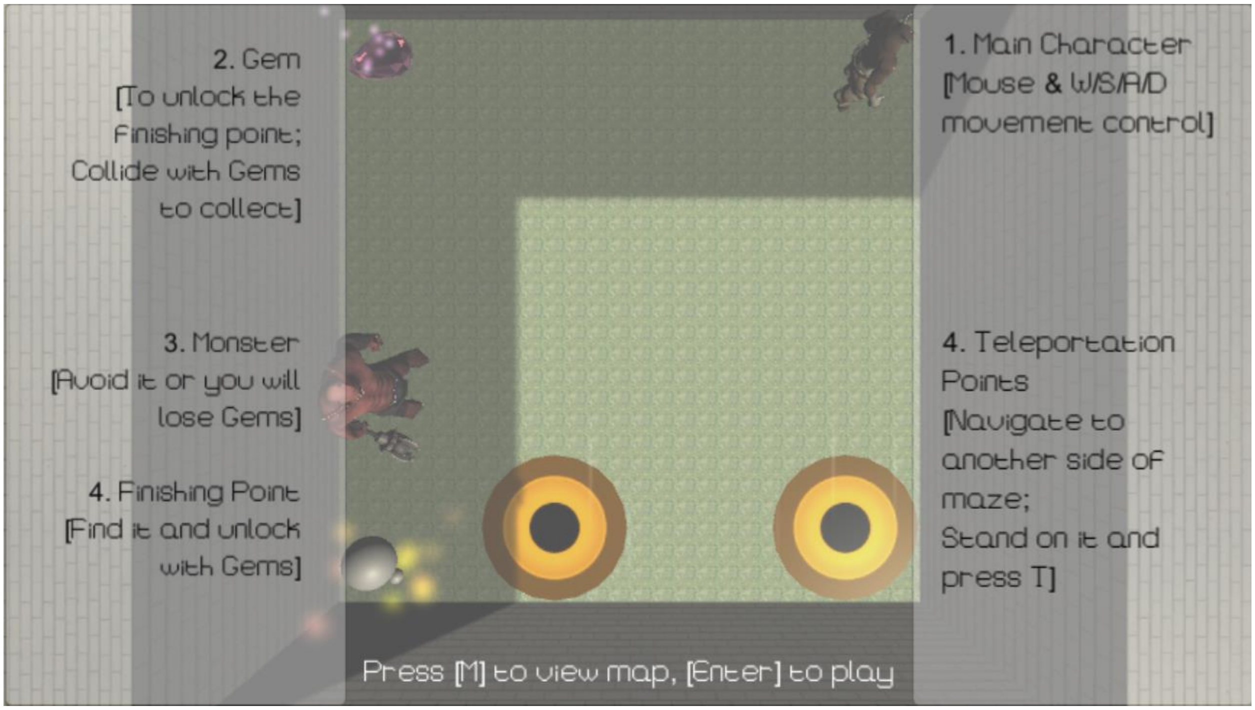
L: -28dB @ 0:00.193 | 44100 • 16-bit • Stereo | 88 K | 28.65 GB free

Figure 37. Adobe Audition

Tutorial

Tutorial is built based on the game scene, but with the game set to level 0. In this mode, the maze only consist of a simple space surrounded by four walls. There are two teleportation points, one gem, one finishing point, as well as one monster for demonstration.

When tutorial is loaded, the player first sees the top-down view of the map with annotations for different objects.



2. Gem
[To unlock the Finishing point;
Collide with Gems to collect]

3. Monster
[Avoid it or you will lose Gems]

4. Finishing Point
[Find it and unlock with Gems]

1. Main Character
[Mouse & W/S/A/D movement control]

4. Teleportation Points
[Navigate to another side of maze;
Stand on it and press T]

Press [M] to view map, [Enter] to play

Figure 38. Tutorial instruction mode

Once [Enter] key is pressed, the player will enter the actual first person view (exploration mode) in the tutorial scene to try out the game.

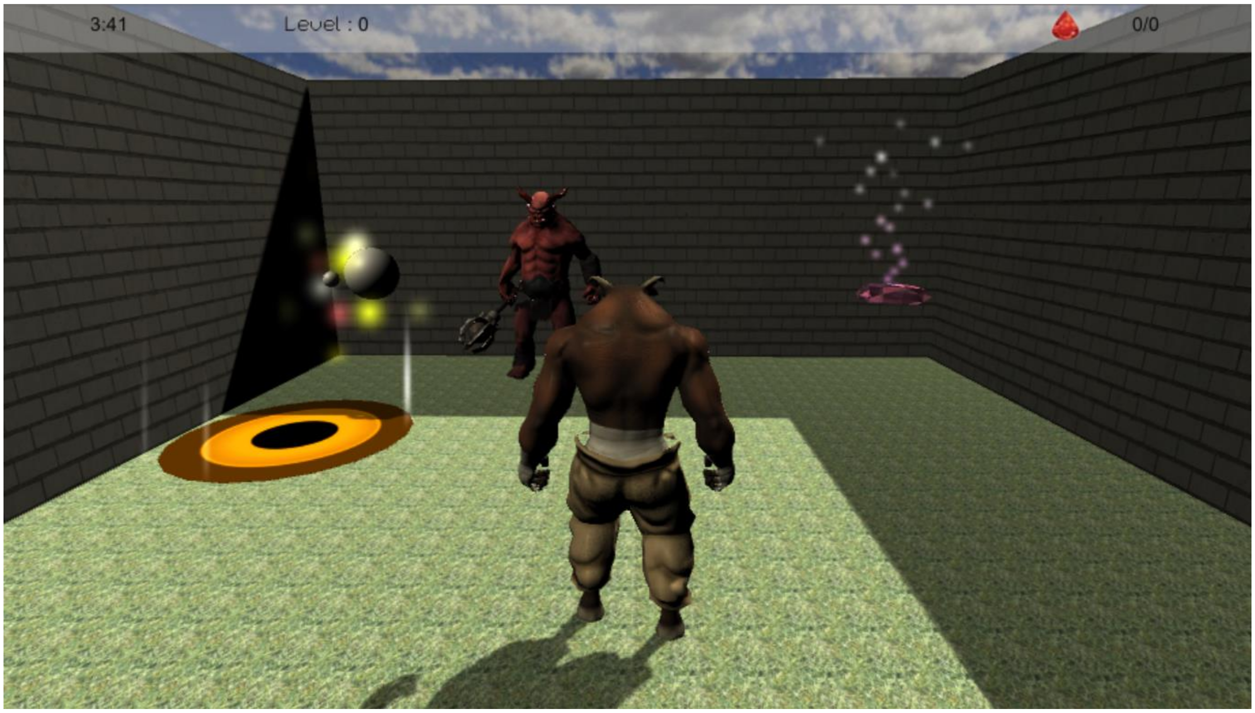


Figure 39. Tutorial exploration mode

Game Data

In the C# script we wrote some code that allows us to serialize our game data and convert it to a file, which can be saved and later restored by deserialize the file into object of the class.

Playing this game for the first time only receives access the land world. It will not able to access the sea world and the sky world. After we finish the land world it will unlock the sea world and save the game automatically. It works the same when we finish the sea world and unlock the sky world.

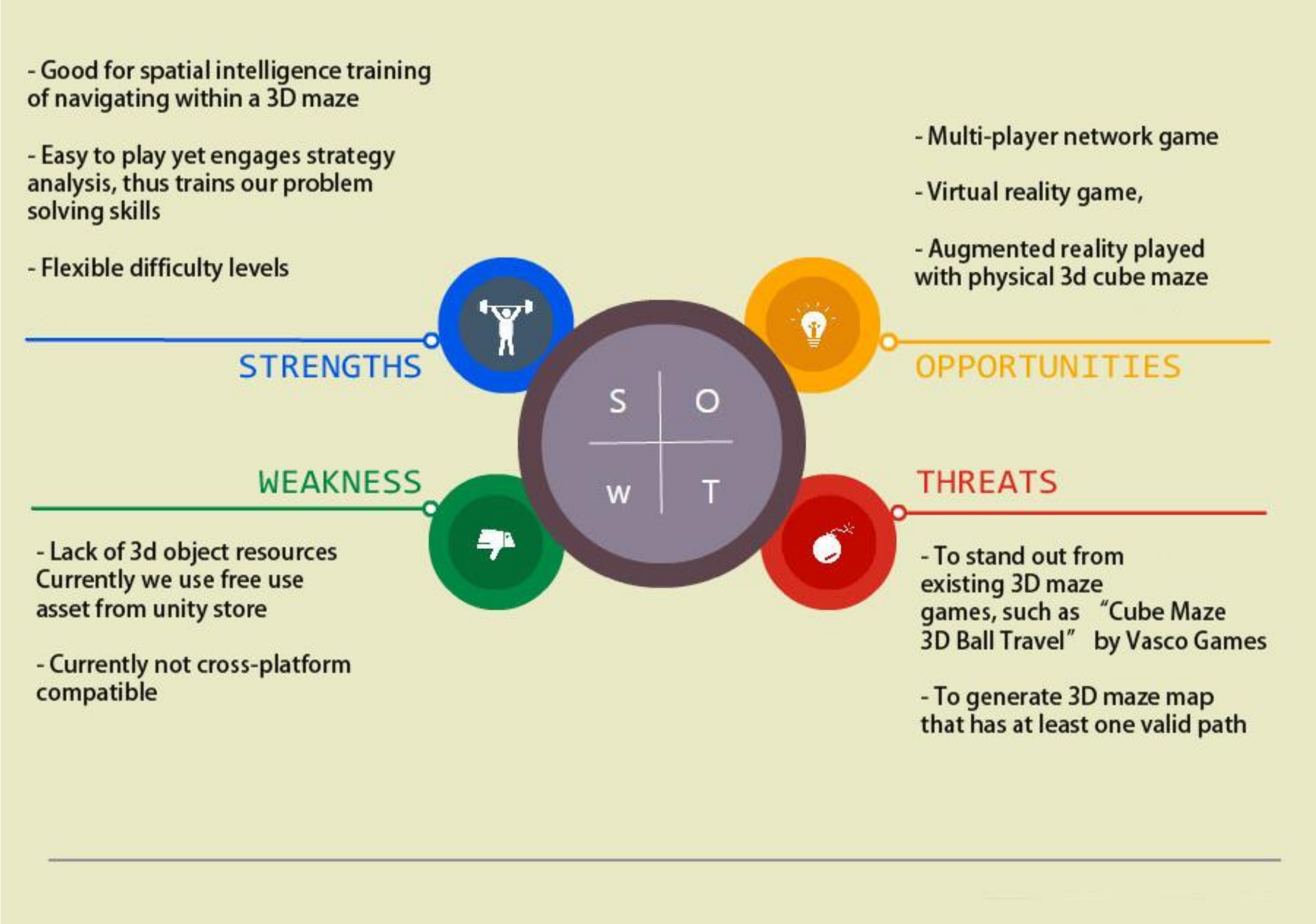
Once the game is saved automatically, we can close the game without hesitation, as the game will be restored when we resume.

Development Shortcuts

We implemented cheat keys for development and testing:

C - Finish the task completely,Win the game, and “Congratulaion” layout will show up
O - Player lose the game, and “Game Over” layout will show up.
B - Go Back directly to Main Menu

SWOT Analysis



STRENGTHS

- Good for spatial intelligence training of navigating within a 3D maze
- Easy to play yet engages strategy analysis, thus trains our problem solving skills
- Flexible difficulty levels

OPPORTUNITIES

- Multi-player network game
- Virtual reality game,
- Augmented reality played with physical 3d cube maze

WEAKNESS

- Lack of 3d object resources Currently we use free use asset from unity store
- Currently not cross-platform compatible

THREATS

- To stand out from existing 3D maze games, such as "Cube Maze 3D Ball Travel" by Vasco Games
- To generate 3D maze map that has at least one valid path

Figure xx. SWOT graph

Manpower

The following three tables summaries the task distribution among three team members:

First member Fanuel mainly works on the game menu, character control, as well as the AI algorithms:

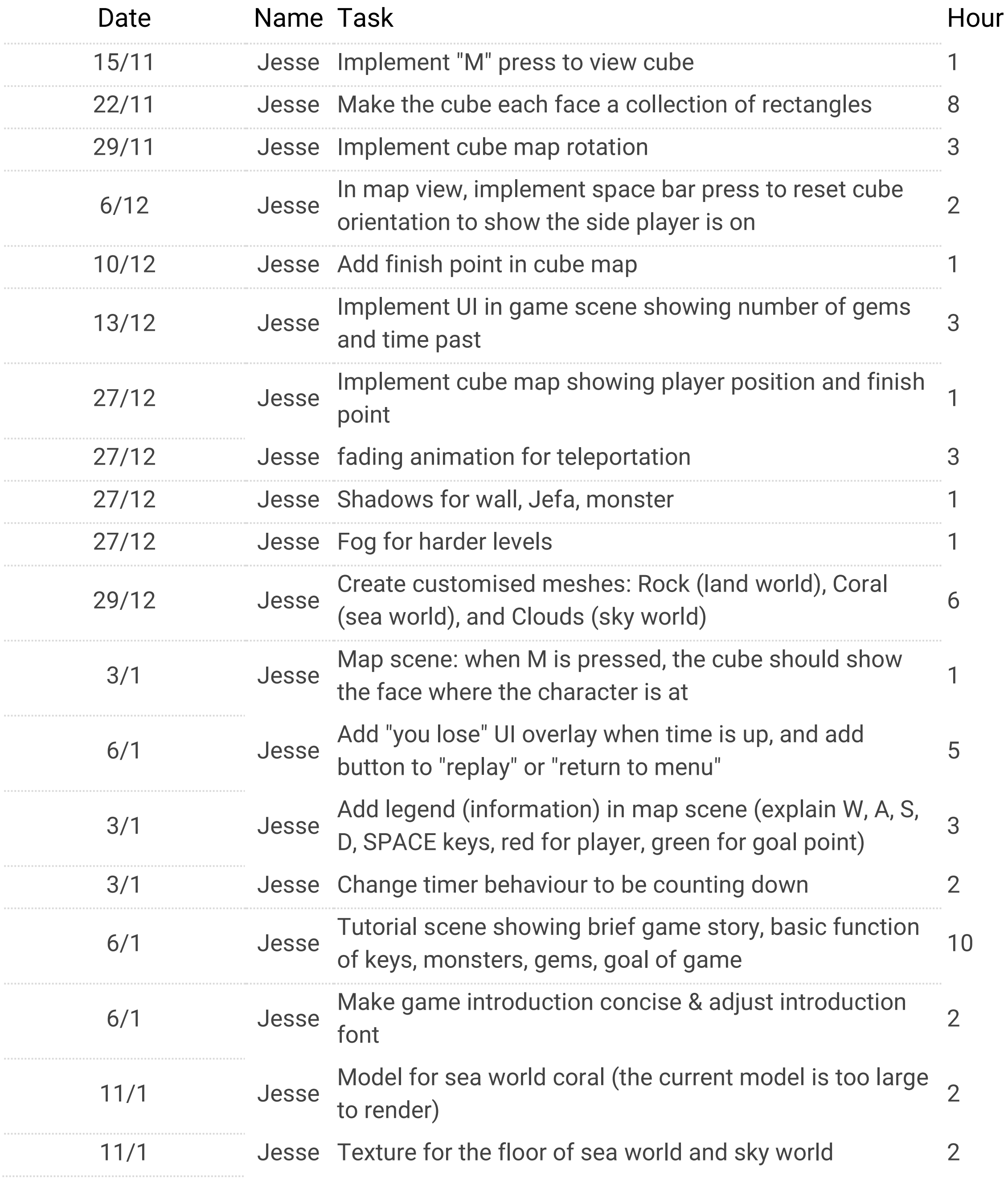
Table 1. Fenuel's tasks

Date	Name	Task	Hour

15/11	Fanuel	Animate the character	5
22/11	Fanuel	Choose the avatar for monster and add it to the maze	2
29/11	Fanuel	Design game start menu screen	5
6/12	Fanuel	Refine start menu UI and implement it	5
13/12	Fanuel	Modify menu UI to make it more visible	1
27/12	Fanuel	change movement using mouse	3
27/12	Fanuel	Put the camera.y lower, so the player cant see the maze	1
27/12	Fanuel	Position character to the starting point of the maze when the game starts	1
27/12	Fanuel	Choose models for gem fragments	1
27/12	Fanuel	AI for monster engage algorithm	20
27/12	Fanuel	Find a solid image for maze floor	1
27/12	Fanuel	Implement pause scene function on game-scene when onpressed M	2
27/12	Fanuel	Health bar add game level indication	2
27/12	Fanuel	At least 3 finite state machines	1
3/1	Fanuel	Level selection scene show unlocked levels	2
3/1	Fanuel	Create fnish scene similar as gameover (nex world - main menu -)	1
3/1	Fanuel	Add "Select level" option after "Play" is pressed in menu	1
3/1	Fanuel	Option scene showing turn music on or off, information of the developers (us)	1
3/1	Fanuel	"Tutorial" option in menu	1
3/1	Fanuel	Implement AI function -- player lose half of the gems (should be integer number, and if gem = 1, lose 1 gem) + reset location	2
3/1	Fanuel	Add game story before main scene is loaded (enable pressing ENTER to skip story) Game story: Jefa need to collect gems from three world: the land world, the sea world, and the sky world	1
3/1	Fanuel	Teleport AI when player is teleported to another side	2
3/1	Fanuel	Add fading effect while player restart	1
3/1	Fanuel	create winning scene	1
3/1	Fanuel	generated scatered gem in the the place where it was taken	3
3/1	Fanuel	disable running on all character	1
3/1	Fanuel	Cheat keys to move to the next level & to trigger game win	1

The second member Jesse mainly worked on the mini cube map, the game tutorial, as well as the customized meshes.

Table 2. Jesse's tasks



Date	Name	Task	Hour
15/11	Jesse	Implement "M" press to view cube	1
22/11	Jesse	Make the cube each face a collection of rectangles	8
29/11	Jesse	Implement cube map rotation	3
6/12	Jesse	In map view, implement space bar press to reset cube orientation to show the side player is on	2
10/12	Jesse	Add finish point in cube map	1
13/12	Jesse	Implement UI in game scene showing number of gems and time past	3
27/12	Jesse	Implement cube map showing player position and finish point	1
27/12	Jesse	fading animation for teleportation	3
27/12	Jesse	Shadows for wall, Jefa, monster	1
27/12	Jesse	Fog for harder levels	1
29/12	Jesse	Create customised meshes: Rock (land world), Coral (sea world), and Clouds (sky world)	6
3/1	Jesse	Map scene: when M is pressed, the cube should show the face where the character is at	1
6/1	Jesse	Add "you lose" UI overlay when time is up, and add button to "replay" or "return to menu"	5
3/1	Jesse	Add legend (information) in map scene (explain W, A, S, D, SPACE keys, red for player, green for goal point)	3
3/1	Jesse	Change timer behaviour to be counting down	2
6/1	Jesse	Tutorial scene showing brief game story, basic function of keys, monsters, gems, goal of game	10
6/1	Jesse	Make game introduction concise & adjust introduction font	2
11/1	Jesse	Model for sea world coral (the current model is too large to render)	2
11/1	Jesse	Texture for the floor of sea world and sky world	2

The third member Setiawan, worked mainly on maze generation and implementation, particle system, sound effect creation, and testing and bug fixing.

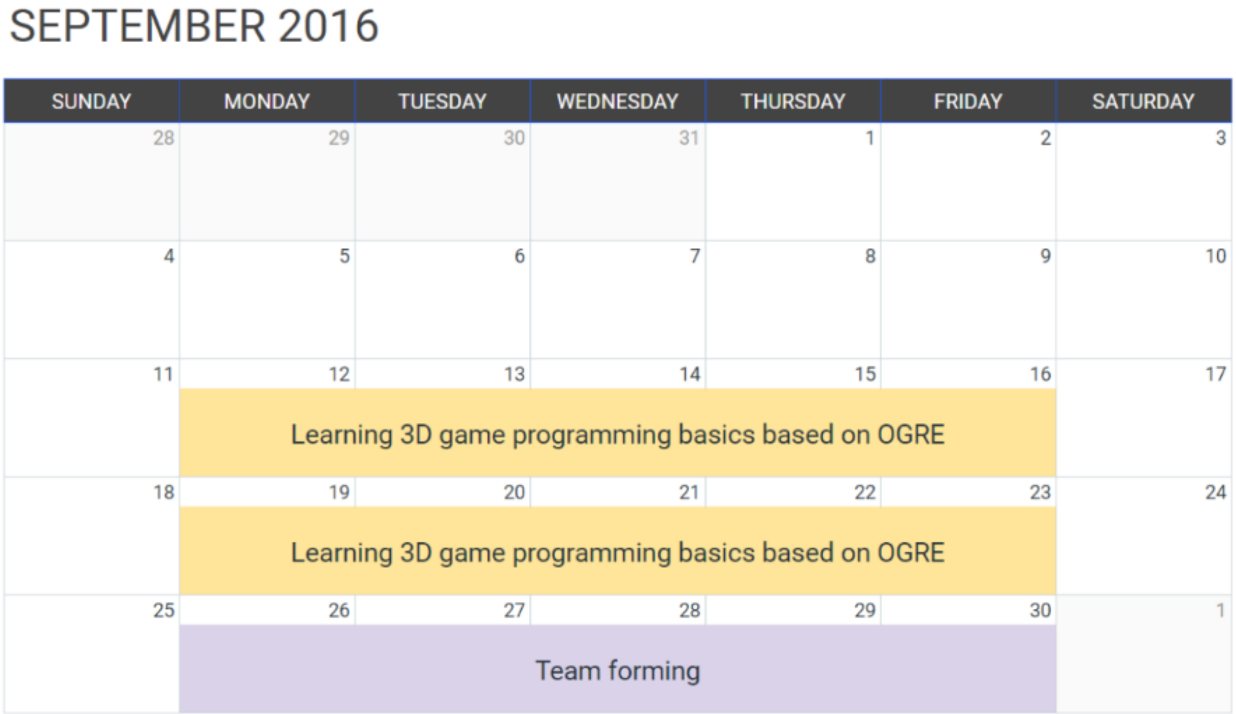
Table 3. Setiawan’s tasks

Date	Name	Task	Hour
15/22	Setiawan	Create Maze Generator	10
22/11	Setiawan	Replace ground to maze, implement map to game	1
15/11	Setiawan	Implement collision detection between Jefa and maze wall	1
1/12	Setiawan	Check if maze cube face orientation is correct	3
13/12	Setiawan	(Copy code to current version) revised cube map showing correct orientation, more zoomed in view, and "Space" key reset to starting orientation	1
13/12	Setiawan	Create a ball on finish point (Add some particle system too)	1
13/12	Setiawan	Scatter 3 gems in each side of cube	1
15/12	Setiawan	Create global variable	1
16/12	*Setiawan*	*Create a particle system ground to indicaate that the ground is a teleport point*	*1*
27/12	Setiawan	Implement teleportation point to move from one side of maze to another	2
3/1	Setiawan	In map scene, when SPACE is pressed, shift maze to the side where player is, instead of the starting side	0.5
3/1	Setiawan	Fix main player move by itself when colliding with walls and AIs	2
3/1	Setiawan	Add particle system for the gem	1
3/1	Setiawan	Put level data on global variable	0.5
3/1	Setiawan	Redefine player movement algorithm	1
3/1	Setiawan	Create three levels: different texture, harder maze, monster, time, more gem fragment, and time, (optional: fog)	3
3/1	Setiawan	Create SFX, record sound effect, and music, and implement it on the game, added music enabled or enabled option	8
3/1	Setiawan	Each level = each gem type, each teleport point type, and each finish point type, and each particle system type	2

5/1	Setiawan	Add particle system when we collect the gem	1
5/1	Setiawan	Redo uncommited changes (character controller, gem collision)	1
3/1	Setiawan	Make the side wall lots higher, and the inside wall higher, to make the camera cant see the player	1
1/4	Setiawan	Tutorial map	1
11/1	Setiawan	Fix teleportation point mapping	2
11/1	Setiawan	Skybox for scenes	3

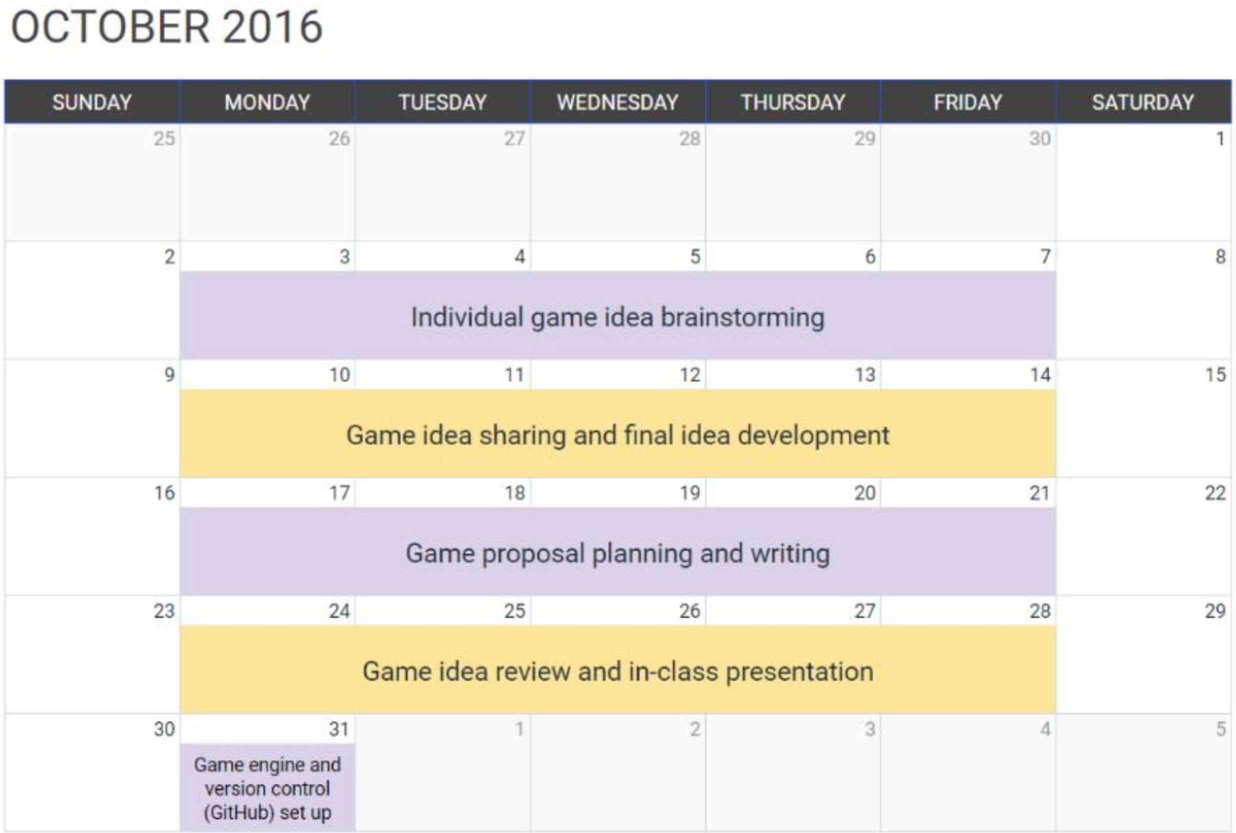
Milestones

Below is our original milestone timeline:



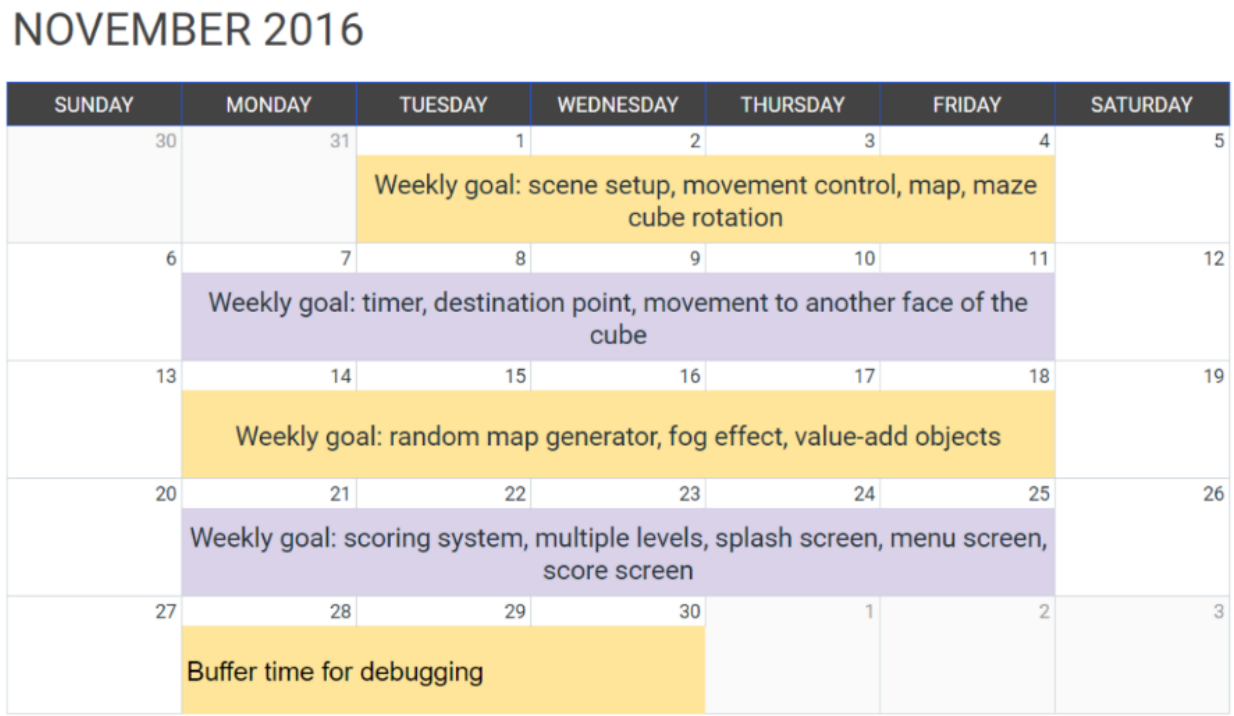
SEPTEMBER 2016

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
	Learning 3D game programming basics based on OGRE					
18	19	20	21	22	23	24
	Learning 3D game programming basics based on OGRE					
25	26	27	28	29	30	1
	Team forming					



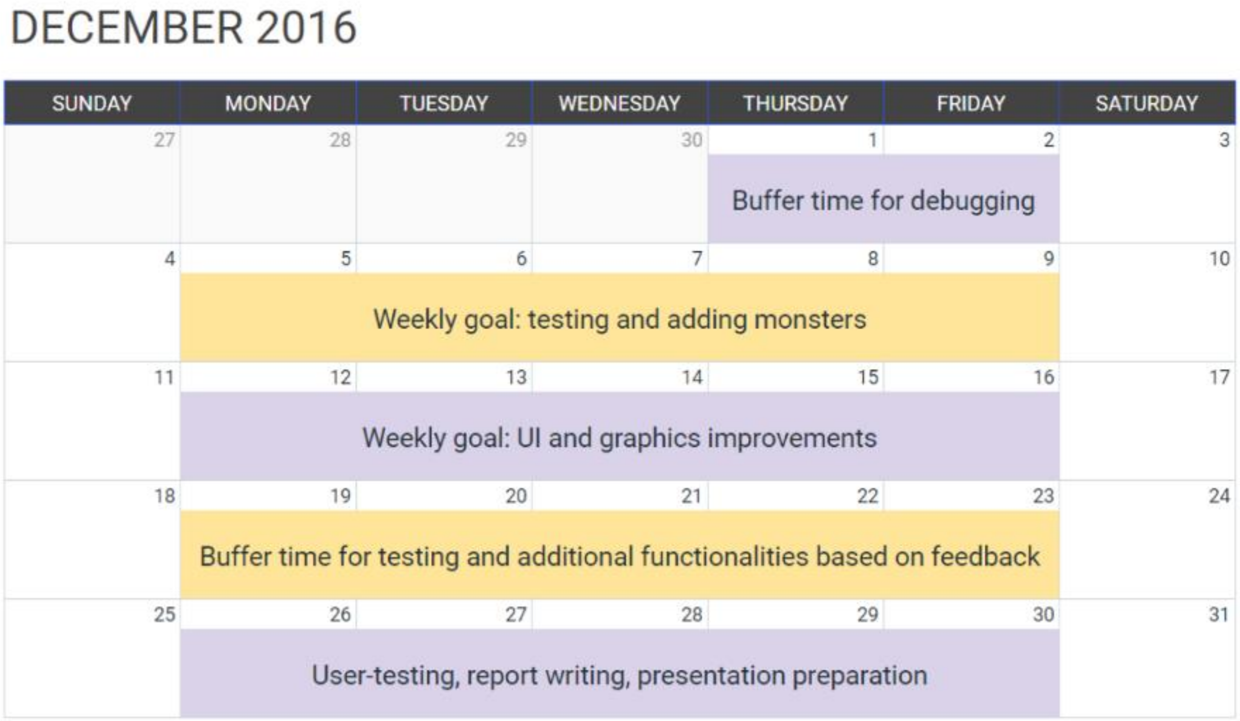
OCTOBER 2016

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
25	26	27	28	29	30	1
2	3	4	5	6	7	8
	Individual game idea brainstorming					
9	10	11	12	13	14	15
	Game idea sharing and final idea development					
16	17	18	19	20	21	22
	Game proposal planning and writing					
23	24	25	26	27	28	29
	Game idea review and in-class presentation					
30	31	1	2	3	4	5
	Game engine and version control (GitHub) set up					



NOVEMBER 2016

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
30	31	1	2	3	4	5
		Weekly goal: scene setup, movement control, map, maze cube rotation				
6	7	8	9	10	11	12
	Weekly goal: timer, destination point, movement to another face of the cube					
13	14	15	16	17	18	19
	Weekly goal: random map generator, fog effect, value-add objects					
20	21	22	23	24	25	26
	Weekly goal: scoring system, multiple levels, splash screen, menu screen, score screen					
27	28	29	30	1	2	3
	Buffer time for debugging					



DECEMBER 2016

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
27	28	29	30	1	2	3
				Buffer time for debugging		
4	5	6	7	8	9	10
	Weekly goal: testing and adding monsters					
11	12	13	14	15	16	17
	Weekly goal: UI and graphics improvements					
18	19	20	21	22	23	24
	Buffer time for testing and additional functionalities based on feedback					
25	26	27	28	29	30	31
	User-testing, report writing, presentation preparation					

Looking back to our original timeline, we did follow the schedule closely for the first two months. However, we were slow in developing new features, so the time scheduled for testing and integration were not saved, but being used to develop the delayed features. This results in a stressful workload nearing the submission date.

Discussion

Game wise, given that our game takes some time to load, we would need to add a UI element to show loading process to reduce possible anxiety from the players.

Planning wise, we should have scheduled more time for fine tuning. In our case, we finished the game right before submission with limited time to play without using cheat keys. As a result, our game is too challenging to be completed.

Time and work management wise, some part require a lot of more time than we expected before, and other parts are dependent on other teammates' progress.We also realised that GitHub is not quite suitable for Unity code version control due to binary file merging issues.

Conclusions

By doing this term-long project, we learnt the processes in developing a game.

We practiced the programming and design skills such as creating game elements, apply mathematics and physics knowledge in game world, C# scripting in Unity, as well as creating multi-media elements such as sprite, meshes, and sound effects.

Furthermore, we gained soft skills in the software development process, including working in a team of diverse background, work flow management under tight timeline, as well as presentation skills in both verbal and written forms.

The game programming is very fun. You create your own world!

Acknowledgement

We would like to express our special thanks to our professor Sai-Keung WONG who gave us the opportunity to do this project.

We are grateful for our TAs' technical assistance, as well as the encouragement and inspirations from other course mates.

References

1. https://en.wikipedia.org/wiki/Maze
2. http://pixologic.com/sculptris/

3. http://www.giantbomb.com/third-person-perspective/3015-464/
4. https://www.cs.umd.edu/users/ben/goldenrules.html
5. https://www.assetstore.unity3d.com/en/#!/content/27594
6. https://www.assetstore.unity3d.com
7. http://www.freeiconspng.com/free-images/menu-icon-19348
8. https://gamedevelopment.tutsplus.com/tutorials/how-to-save-and-load-your-players-progress-in-unity--cms-20934
9. https://s-media-cache-ak0.pinimg.com/236x/f3/7f/41/f37f41ce5c0a5ff52fa17f0f7e42555e.jpg
10. https://s-media-cache-ak0.pinimg.com/736x/a6/d1/18/a6d1189afc169c03b0b1f089f6373b90.jpg