The University of Melbourne

School of Computing and Information Systems

COMP30019 Graphics and Interaction

Assignment 2 Report

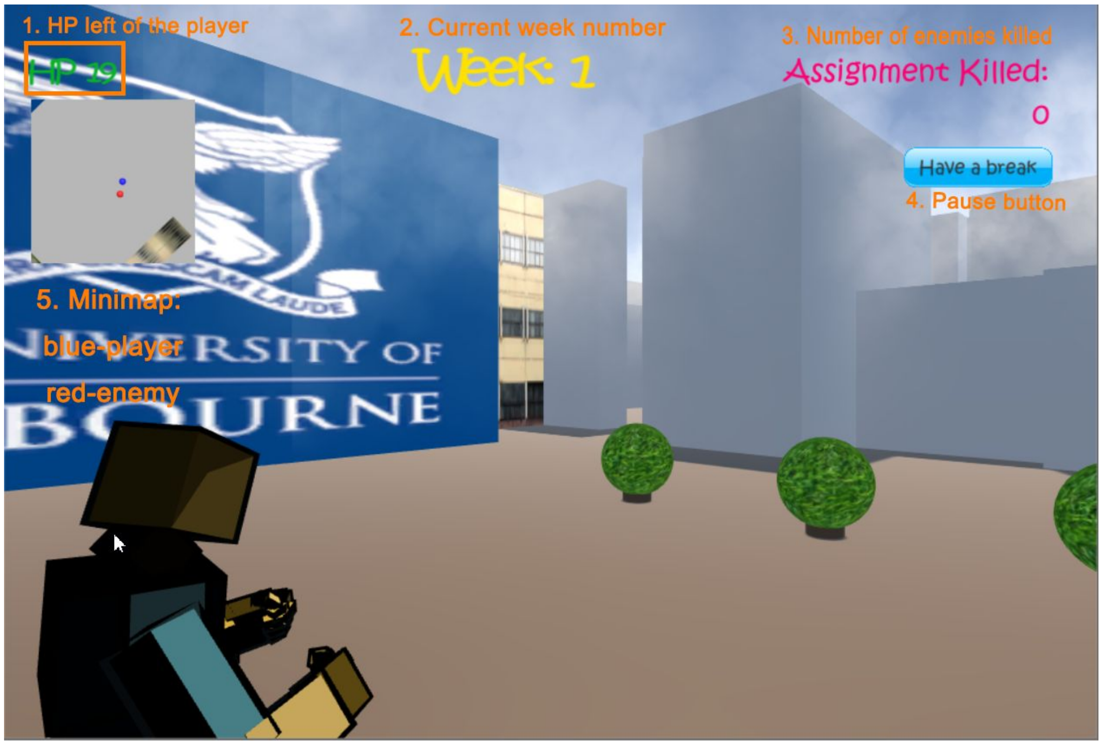
By:
Mayank Sharma (mayanks1@student.unimelb.edu.au)
Chao Li (chaol6@student.unimelb.edu.au)
Judd Guerrero (jguerrero@student.unimelb.edu.au)

A brief explanation of the game

The game is a third person shooter game deglamourising the unrealistic perception of a computer science university student. In this game, the player gets to play as a student at The University of Melbourne that woke up from a nightmare. The player combats computer software that are trying to haunt him in his sleep through weeks 1 to 12 of a semester. Every 6th and 12th week the game goes to a nightmare mode similar to a real uni student trying to accomplish all his assignments. The goal of the game is to get through weeks 1 to 12 without dying by attacking enemies with laptops.

How to use it (especially the user interface aspects)?

The controls are as follows:
W – move forward
A – move sideways to the left
S – move backwards
D – move sideways to the right
Left mouse button click – kick laptops towards where the player is facing
Mouse movement left and right – rotates player's vision to the left or right respectively.

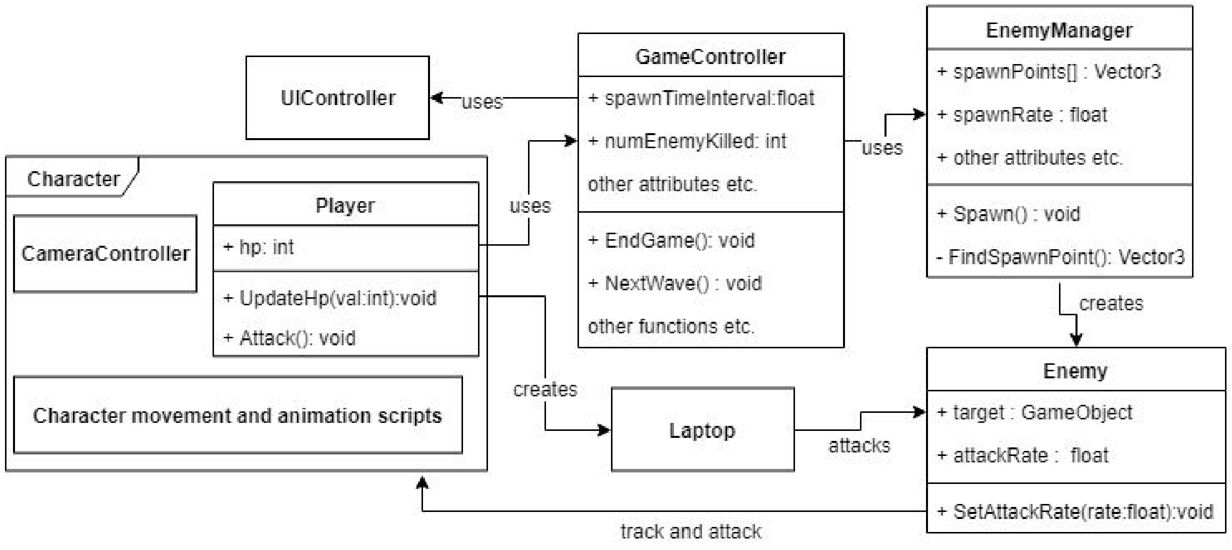


(Figure 1. UI explanation)

It should be noticed that the shader sometimes behaves weird when playing the game in Unity editor. While everything goes will in builds.

How you modelled objects and entities?

There are mainly six components in this game: map, character, camera, weapon, enemy and UI. The map was generally modelled after the general University of Melbourne Map, which is built by static cubes and cylinders. Walking camera was implemented as its the most suitable camera type for our game. Navigation system provided by Unity is used for enemy pathfinding.



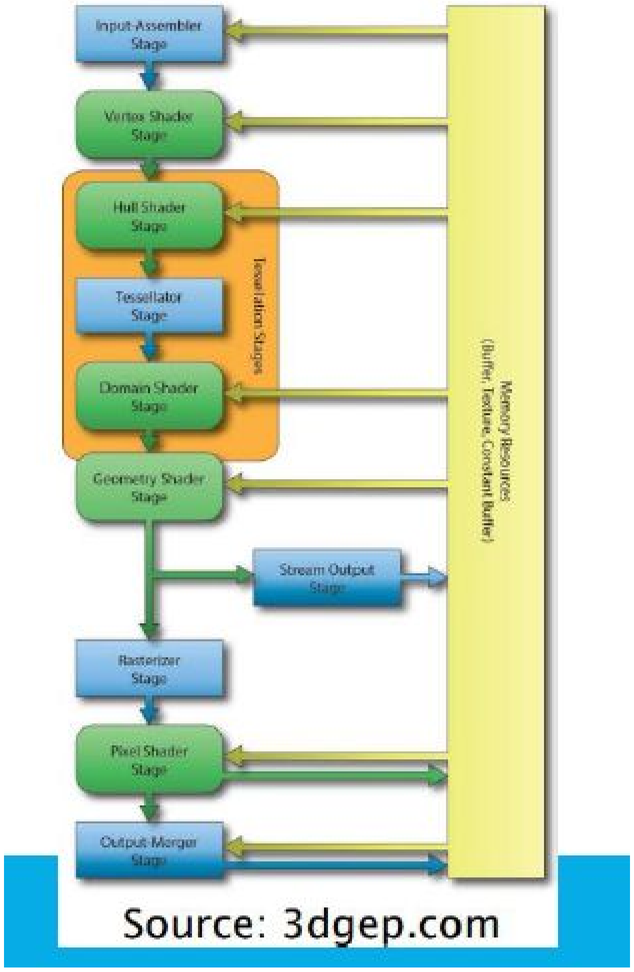
(Figure 2. The brief architecture of the game)

Figure 2 shows a brief design of the game. GameController handles the main game mechanism, such as adjusting parameters for each wave and uses UIController to update UI displays. UIController handles the relationship between function calls and UI elements, such as the display of game status and button functions. EnemyManager is used to spawn enemies. It will select spawn points from pre-set points set, which satisfy given conditions and spawn enemies repeatedly. The texture of the enemy is randomly chosen from 4 different materials. The enemy class just represents the enemy and record the target player it's heading to. It will automatically attack player based on attack rate if the player is in attack range. Player class represents the player, containing information like the current hp of the player. And when left mouse button clicked, a laptop will be thrown out at the direction player is facing. The laptop will check objects collide with it, if it is an enemy, and the laptop has not collided with other objects before, both objects are destroyed immediately. If it collides with objects other than an enemy, it will be destroyed after 5 secs and is not able to destroy enemy during that time.

How you handled the graphics pipeline and camera motion?

The camera motion takes into consideration the time that the game has passed since the last update. The movement of the camera depends on this thus going in increments since the last update. The camera in this game is a sub-child of the player offset to the back of the player.

The general graphics pipeline contains stages as follows:



Our approach was Pixel shader stage heavy as we used shaders like the Nightmare Mode Shader and Cel Shader which were completely Pixel Shaders. All the calculations of the two shaders happened in the Pixel Shader Stage.

Besides the Cel Shader, we did not deal much with the lighting of the game and chose to work with the general directional lighting provided in the unity from the beginning.

Descriptions of how the shaders and the particle system work.

There were quite a number of shaders used in our game:

Nightmare Shader

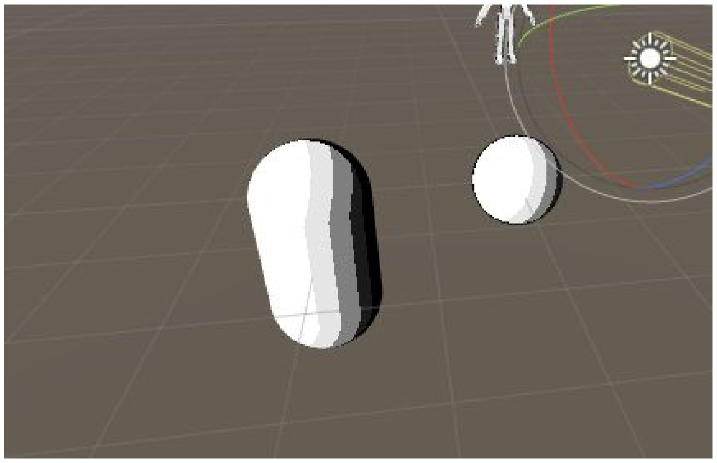
The Nightmare Shader was originally planned as the fog shader but it did not work as plan. Luckily, we applied the same fog shader with modifications to the camera and the effect generated was the nightmare mode. The mechanism works based on the exponential square formula:

$$f = e^{-(d*b)^2} = \frac{1}{e^{(d*b)^2}} \quad \textit{where } d = \textit{distance}, \quad b = \textit{attenuation factor or fog density}$$

According to the formula, the further the object is, the lighter it appears and the closer the object is, the darker it appears. However, we want the shader to appear lighter nearer and darker further away. Hence, the fog factor was inverted by subtracting the factor from 1 and adding it to the Return Colour variable. This shader was applied to the camera only when it is 6th week or the 12th week.

Cel Shader

Similar to the Phong Shader, this shader does not have specular component there for the specular component is not added to the final Return Colour variable. In cell Shader, the colours do not up as a blend from one shader to another. Rather, the colours appear separately as different visible shades on top of each other.



There are a few factors that help to control the shader. Object Outline helps to fix the object border width. The larger the value, the thicker the value. Diffusion Threshold enables the number of layers of colour that can appear on the object. The colours appear from light shade to dark shade (Top to bottom). The greater the number, the more the colour layers. Finally, the ambient component here is set as 0.3 to control the amount of dark shadow on the object. The lower the value, the greater the shadow on the object. A 1000 value is multiplied to the outline width to prevent to the outline from blending into the player texture. This shader was applied to the player. Since all the objects including the player of low poly. The cel shader did not should a wonderful effect as postulated.

Other Shaders
Other Shaders such as Building Shader, Git Shader, Eclipse Shader, Unity Shader, ColourShader, Chrome Shader and Grass Shader were all default standard shaders of unity to provide texture and colour to objects.

Particle System (Fog)
The particle system was used from the standard assets shader. The values of the shader were tweaked to suit our game environment. The colour of the fog was set as white in colour and the duration of the loop was set to 15s. Start speed was set to 0.5 and start lifetime was set to 10s. Also, the max particles were set to 500 particles. The Particle System was learnt from the following link: https://www.youtube.com/watch?v=POFKoQzI5pU

Description of the querying and observational methods used.

1) Participants for the evaluation techniques were 5 students of Software Engineering and CIS at the University of Melbourne. The participants were pretty knowledgeable with Unity, software development and game development. From this, it could be said that they have a keen eye when it comes to mistakes and things to improve on. The participants were the same for both the observational and query evaluation techniques.

2) For the observational method, the method that was used cooperative evaluation. In this situation, the participants were treated as product owners so can get a feedback on how they would create/modify the game if they were the developers. No questions were asked here but they discussed amongst themselves their suggestions regarding the game's flaws. They mentioned that the game has to have a clear goal and a tutorial at the beginning so players wouldn't figure out what to do. They would also change the aesthetics of the game that would be more catchy and interesting. And finally they decided if we were to have low polygon objects then the aesthetics have to be similar to the games Rodeo, Hole.IO' aesthetics but if we were to

have a high polygon game it needs to be able to make the player feel that they are really in a nightmare by adding a lot of gory textures and implementing particle systems that will add to the game's action adrenaline.

3) For the query evaluation technique, the methodology used was questionnaires. The data was recorded after they've played with the game without any questions in mind whilst playing the game. After the game, the participants were given questionnaires that they answered based on their gameplay experience. The questions in the questionnaire are as follow:

What was wrong with the game?

1. What do you think needs improvement?
2. Was the game enjoyable?
3. Was the concept unique and do you think this would be something that we should pursue and make a complete game out of?
4. Are there any other suggestions you think we can change in the visuals and entities in the game?

Question #	1	2	3	4	5
Participant #1	Players can move outside the map	The enemy spawn rate needs improvement	The game has a cool concept but it isn't something I would crave to play	The game is a unique reskin of the third-perso n genre games.	Add more details to the objects in the map
Participant #2	The camera can't move up-down	The camera movement	The game was interesting because of the theme	Worth pursuing if you can make it look more engaging	Use more complex shapes
Participant #3	I don't see anything wrong with the game	Being able to navigate the map. I think a minimap showing where in the map the player currently is	The game was funny and so making it kind of enjoyable to play.	Yes I think if you clean and finalise the concept this could work. Similar to Baldi's Basic that	Higher level of detail

		would help			
Participant #4	Mouse visibility. The game doesn't need to know where the pointer is as it's not collecting things anyway.	Freedom in movement. Incorporate jumping	The game was quite weird, it's a cool twist in the third-person shooters.	Could be if there's a clear goal.	Make it look more cartoony
Participant #5	The ratio of the size of things to the character. The character also has an unnatural animation.	The gameplay needs improvement. I feel like it needs to be able to do more things.	The game was interesting to an extent but starts to get boring after a while. I feel like there needs to be more surprises in the game.	If you were to create a whole game out of this I feel like there needs to be more levels and more goals to follow.	The aesthetics needs more gore.

Document the changes made to your game based on the information collected during the evaluation.

1. Increased spawn rate and movement speed of enemies
2. Add obstacles around edges of the map to prevent player from moving outside the map
3. Rather than only rendering player and enemies on minimap, buildings will also be rendered on minimap to give user a better idea of the environment around
4. Add some bushes on empty space of the map
5. Add textures to part of the buildings to make them looks more like a building
6. Adjusted size of buildings according to the size of character

A statement about any code/APIs you have sourced/used from the internet that is not your own.

We did import some of our characters, textures and some shaders associated with those imported characters and textures.

1. RPG Character Animation Pack FREE
   This package is not ours and was imported from the assets store to import our main character and scripts associated with it.
   Scripts attached to the player but not actually used: GUIControls, RPGCharacterControllerFREE, RPGCharacterInputController and RPGCharacterMovementController.
   We **strictly** created our own camera and player movements along with our UIController. These scripts were attached to the player ideal state and removing them gave errors hence they were left attached to the player.
   There were also other scripts like gravity, state machines and DebugDraw, Math3d in the package that were not used.
   Reference link:
   https://assetstore.unity.com/packages/3d/animations/rpg-character-mecanim-animation-pack-free-65284

2. Laptop Package
   This package was also not ours and was used to provide laptop as a weapon for our player.
   Reference link:
   https://assetstore.unity.com/packages/3d/props/electronics/free-laptop-90315

3. Standard Assets
   This package was imported for unity standard packages. The Particle System was generated as a dust storm and the particle colour was changed to white. Also, the number of
   Particles were reduced to 500 and the total duration was kept to 15s loop.

A description of the contributions made by each member of the group.

Chao -

Map Modelling – creating the map, creating the buildings in the map,
User Interface – Main menu screen, player and game status display, game over screen
Script – Game state control (pause, restart, wave control etc), shader switch, code refactoring
Enemy AI – Enemy spawning points, enemy creation, the enemy path
MiniMap – showing the position of player and enemies nearby
Judd -
Camera control – character control, visible camera visual
Scripts – Weapon projectile, Object collision detection
Mayank -
Shaders – cell shader, Filter for the nightmare mode
Textures - textures to buildings, and enemy
Particle System – Fog Particle System of the game