

Milestone 2 report

IN3026 Advanced Games Technology

By Kevin La

Table of Contents

Overview	3
Gameplay	3
Main menu screen	3
Primitive-based Objects.....	4
Audio	4
HUD	5
Camera technique.....	5
Mesh-based objects	5
Lighting	6
Special Effects	6
Physics	7
NPC and AI.....	8
Gameplay elements.....	9
Discussion	9
Appendices	11
Asset list	13
Textures.....	13
Meshes.....	13
Audio	14
Libraries.....	14

Overview

As part of the Advanced games technology coursework requirements, I have made a game project using OpenGL C++ based on my own knowledge and content taught during Labs and Lectures. The title of the game is called Toxic Rain and this name was decided based on the theme that I had in mind. The theme of my game is post zombie apocalyptic theme similarly to already made titles such as Resident Evil franchise and Fallout. The theme was achieved by adding a skybox texture and terrain and adding minimal lighting. (*See Appendix 1 for theme*) The genre of my game is a 3rd Person FPS taking inspiration from The division Tom Clancy's and Resident Evil 4/5/6 allowing the player to move around and look around with the camera positioned behind the character model's shoulder.

Gameplay

In Toxic Rain, the player is tasked to kill the various beasts that have different characteristics e.g. Speed, Health, Damage. The player must survive and eradicate all present beasts and zombies before extraction (Before time runs out) in an open area to beat the game. The map is made up of walls surrounding the playing field, spikes located on the ground that damage you and containers that can act as cover.

Some key gameplay mechanics' feature:

- Shooting
- Grenade throwing
- Enemy AI
- Score system
- Timer
- Power ups - Health increase, Ammo increase etc.
- Difficulty Selection

Main menu screen

For the main menu, I have kept it pretty much the same only improving some areas so that it is easier to navigate and adding more options as I implement new features. For example, I have changed the colour of the instructions section to red like the rest of the menu as it was quite hard to see with a dark setting in the background. Before starting the game, I have given an option to select difficulty. This allows players to try the game at a higher difficulty if they feel like it is too easy in normal mode.

I have also added more control instructions as I have implemented more features. For example, Added instructions on how to jump, throw grenade and toggle flashlight. As promised in my previous report, I wanted to give the player the option to toggle between Inverted or non-inverted camera controls which I have done so. This can be done by pressing the key 'i' in the instruction's menu. There will be a clear indication that the option has been enabled or disabled in the instruction's menu. (*Instructions screen shown in appendix 2*)

Primitive-based Objects

The primitives in my game form a lot of the level or is used as pickup or projectile. 1 of the primitive objects I have added is a wall which is a variation of the cuboid primitive. It is much thinner to simulate a wall. I have added a gritty wall texture on it to fit the theme. To make it surround the map I used for loops to generate multiple walls but change the location of each wall. This was also done with the bounding box. I made 4 for loops, 1 for each side and in each for loop I have rotated it. (*See Appendix 3*) An alternate and possibly more efficient way of making walls surround the map instead of using a for loop is to change the scale of the wall on the x axis so it is wider. I would then loop the texture on the wall as have 1 texture stretched may not look so good.

Another primitive is a square pyramid. This was used for spikes that cause damage if the player is not wary of its surroundings. With the spikes I wanted to make each spike different, so I gave the top vertex a random X and Y and Z value. (*Examples of the spike can be seen in Appendix 4*). I then used a for loop to create multiple spikes and I used more random values to set the position so that each playthrough, spike locations will not be the same.

Triangular prism is another primitive object I created, and I used. Just like the square pyramid, I calculated the vertices and texture co-ordinates by drawing out the faces of the 3d shape. (*Seen in Appendix 5*), normal (using cross product) This acts as a power up which increases the battery level of the flashlight. I have added a plasma red texture to it represent energy which powers up the torch.

For the projectiles I used 2 primitive objects. 1 which is a sphere that was already made, I just scaled it down and added a texture and this was then used for the bullet. The other is a variation of the sphere. By reducing the stacks and slices, I was able to create a diamond shape and this was used for the grenade. I then added a futuristic plasma texture as in my mind, I wanted it to be a plasma grenade.

I did attempt to change the bullet to be a hexagonal prism however I was not successfully in creating it. The main downfall was the indices and normal. The shape is there however the textures did not render on all triangles. (*As seen in Appendix 6*). In the future I could finish this off and learn more about normals. Main reason this was not done was due to time constraints and I felt like polishing the core gameplay elements was more important.

Audio

For the audio of the game, I have added various sound effects to enhance the users experience when playing. In addition to that, I have also added a new background music to fit the theme. The audio effects are managed by the `m_audio_manager` and each sound effect are activated on the following events:

- 2 different gunshot sounds that are randomly picked when the player shoots.
- When the player toggles the flashlight, a switch turning on sound is played.
- When player toggles it again a switch turning off sound is played to simulate the flashlight being turned off.
- When player gets damaged by a spike, a slash sound effect plays.

To improve on this section even more, I tried to add spatialised sound however it did not work as intended. The sound was not dropping off the further away I am from it and every zombie played the sound at the same time making it loud as there's at least 10 of the same sound effect being played at the same time. So, I removed it entirely. However, in the future I will improve on this by tweaking the min-max distance and using a randomizer so that I can randomly select which zombie the sound will play from.

HUD

For the HUD (Shown in Appendix 7), I have added text to indicate various information such as Health, Ammo, Grenades, Flashlight battery, Score and Timer. This is all coloured in red to fit the dark zombie theme and to make it visible in-game as the setting is quite dark. I have put the information at the edge of the screen, so it is out of the way. In the future I could improve on this by possibly adding images to indicate information for example, 5 grenade images to indicate that the player only has 5 grenades to throw. How I would go about doing this is, render multiple small 2d rectangles with a texture of the grenade mapped to the rectangles like how the crossfade was implemented which is talked about later. As a player throws a grenade, remove the rectangle and texture. (Would be the opposite for picking up grenades and increasing it)

Camera technique

For the camera technique, I have implemented a 3rd person camera which is also rotatable around the player depending on the mouse's input. The implementation is pretty much the same only some adjustments to the camera to fix an issue that would cause the camera far view to clip and remove a chunk of the map which would make looking in the far distance impossible. (See appendix 8). To fix this issue, I increased the far_z variable in camera.h from 100.f to 250.f.

In the future, I hope to add another camera which the player can switch from. This 2nd camera will act as the ADS (Aim down sight) camera and will allow the player to shoot wherever the camera is looking (Crosshair provided in this camera) instead of just 1 direction like it is now. This camera technique is used in games like Resident Evil and other 3rd person shooters. To do this, I will implement a first-person camera view and then switch to that camera when a key is held then switch back when key is released. I will then render the crosshair using a quad and add the crosshair texture on that quad. When the camera is on the ADS camera then render the crosshair then have an if statement stating that if player is in the ADS camera, shoot the ball from the camera instead of the player.

Mesh-based objects

In my game I have added various mesh-based objects based on the theme of the game. For the enemies I have 4 different mesh-based objects, 1 to represent each type of enemy. I then change the size of them by changing the scale value. I changed the value depending on the enemy type for example, the tank zombie which has the most health would be the largest, the fastest beast would be small and nimble. With these meshes, I have randomly placed them in the map so it's the player's objective to find and kill them.

To add to the feel of a zombie apocalypse I have added containers which can act as cover when versing the boss that shoots projectiles. These containers have also been randomly

placed in the map using random values within a range. This gives more replay ability as having containers in the same spot every playthrough may get boring.

I added a streetlamp in the level to represent a light source. This is positioned in the middle of the stage. I've also added a Torch and a Rifle just for looks as the player can use a flashlight and shoot bullets so it made sense to include them. What I could do in the future to improve the implementation of the Rifle and Torch mesh is attach it to a body part like the hand. Right now when it rotates, it moves position. How I would go about doing this is making a collision box for each hand and collision boxes for the Torch and Rifle. I'll then check if the Torch is touching the left hand box, if it is then keep setting the position of the torch to the position of that collision box. This will be called every frame in the `on_update`. I would then do the same to the Rifle.

All the meshes mentioned have a bounding box based on the size of the mesh.

Lighting

I have implemented 2 different light sources. 1 Spotlight and 1 Point Light.

The spotlight is used as a torch/flashlight as the setting is dark and there's not much light other than the point light so it is needed in order to see where the zombie is or if there's a spike in front of you. I made the spotlight follow the player and shine light in front of the player in a wide cone angle by setting the spotlight position and direction to the player's position and forward (See *appendix 9*). This is done in the `on_update` function.. The player can toggle the flashlight with the 'F' Key however this light is not unlimited. The "battery" decreases as you use it.

As for the Point Light which is to give the streetlamp a purpose of being there and to give some light to the player in case the torch runs out of battery. I positioned this at the top of the street lamp mesh so it lights up the surroundings of the lamp but not too much that it deems the torch useless and changed values of the `m_pointLight` such as ambient, diffuse intensity. Constant, Linear and Exp to restrict light from reaching the whole of the map. (See *Appendix 10*)

For the lighting, I made sure that these light sources impact each mesh and primitive objects so that it is realistic and allows the player to see objects instead of looking at a completely blacked out silhouette.

Special Effects

In the game there are 2 special effects that I have implemented to enhance the user's playing experience.

1 is a crossfade which activates when getting damaged by either an enemy or a spike. It also activates when the player dies. I wanted it to be like the game *Dark Souls* where when you die it fades to black and you cannot really see what is going on in the background. The crossfade is done by rendering a quad on the screen with a texture placed on top. The texture is a dark red texture giving the sense of losing blood.

2nd special effect is the alpha-sphere and was extracted from the special effect lab code. This is shown when the player throws a grenade and contacts an enemy. The alpha-sphere is then

activated at the last position of the grenade. I have tweaked the alpha-sphere so it is smaller and reaches the end of the animation much faster.

In the future, I wish to implement a fog special effect which was released in the Special Effect lab code. I feel like that would fit well with the game theme but due to time constraints I wasn't able to add it and I felt like the fog may have been a bit too much for the player to see anything alongside it being really dark. Hopefully in the future I can implement more point lights and then that would allow me to add a fog so it would not hinder the players vision as badly.

Physics

The game physics that I have used are Bounce, Acceleration, Gravity and Collision. I have included these into my game as they felt suitable for the gameplay.

For Bounce physics, this can be seen with the grenade and Shooting Beast's projectile that it fires. This is to add bounce just like a grenade would bounce and give ambiguity to Shooting beast projectile as having it bounce, there is a chance it can hit you or not. For example, It could straight up hit you or bounce and hit you or bounce and go over your head.

As for Acceleration & force, this can be seen implemented in the bullet for the player. When the player presses the fire button, it calls a function that passes through a value which is the force at which to shoot the bullet. I then multiply that by the value and the player forward vector to shoot the bullet at a constant speed in the direction the player is facing. In the `on_update` function for the bullet class, I have made sure to keep the y position constant as I do not want it to drop and bounce like the grenade.

For the Jump physics, I had to do a bit of research. I watched this video on how to implement jump and gravity in Unity ([Jump And Gravity - Game Mechanics - Unity 3D - YouTube](#), Accessed on 02/12/2020) and translated what I learned from there into OpenGL C++. The main key point I learned was the formula:

- $Y \text{ velocity} -= \text{gravity} * dt;$

After figuring that out I was able to implement the rest of the gravity. And just like the shoot function, I passed through a force to push me upwards and let the formula above deal with the player falling. The formula was coded in the `on_update` function so it got called every frame meaning every frame the players y position kept decreasing. I then added an if statement to ensure that the player does not fall through the terrain. (see Appendix 11 for the gravity code)

Collision detection was a big part of the whole game. This is where players get hit, player can kill and where players are restricted from going inside objects or out of the map. To implement collision detection, I used vectors of bounding boxes for game objects that have been rendered multiple times such as: Wall, Enemies, containers, Spikes. To check for collision, I had to use a for loop to iterate through each bounding box in the vector. (See appendix 12 for an example). I then checked for collision between objects that I know would collide such as Bullet and Enemies, Enemy bullet to player, Pickups, and player etc...With each collision, I called functions such as decrease health, push player back, activate sound or spawn a health pack at the position of the zombie that died.

NPC and AI

For NPC and AI, I have added 4 enemy NPCs that have different characteristics. These are:

- Normal Zombie
 - Average speed, Average damage, Average sized.
- Tank Zombie
 - A lot of health, Slow movement speed, Huge in size, Above average damage.
- Fast Beast
 - Nimble and quick, Small damage, Low maximum health.
- Shooting Beast (Boss)
 - Highest health, Able to shoot projectile, Large, Slow movement speed, Above average damage.

All these NPCs are based on an FSM that I have implemented, and this handles the movement of the NPCs. When initialising each enemy, I can set their health and speed (See appendix 13). Additionally, I added a new float variable called `difficultyMultiplier`. This changes depending on which difficulty has been selected. Normal would set it to 1 and Hard would set it to 1.5. Some of the NPCs attributes like Speed and Damage dealt are much higher to make it more challenging.

Each frame the NPCs position and rotation are updated based on the state they are in for example, when in idle state the NPC patrols moving up and down by adding the position with the forward, speed and frames (`Time_step`), when in on guard state the NPC faces the player, when in chasing state the NPC is moving forward based on its forward values. Initially the FSM had 3 states however I have added 2 extra states. An attack state which lunges at you once you are in attack range and a resting state which lets the NPC heal when below a certain threshold.

The attack state is like the `chase_player` state with some slight tweaks. I made a new variable called `attack radius` so when the distance between the NPC and player is less than the attack radius then lunges forward to attack. I have implemented the lunge by multiplying the objects forward values by 10 instead of `m_speed`. Due to it lunging forward, the player and NPC bounding boxes collide causing damage to the player and knocking the player back as the `pushback` function is called.

The resting state triggers when the NPCs health is below 50. When it is in a resting state it heals its health overtime. Once it is above 50 it will go back to patrolling (See Appendix 14) or another state depending on the distance between itself and the player. I added this as it encourages players to finish off their enemies and to not leave them alive as it may be troublesome down the road.

I can improve on this by having the NPC flee to a point in the map to rest. I can do this by making another state called `Flee` and setting a random location within the map for the NPC to move towards when health is below 50. Once NPC has reached the point, then enter the resting state in which it heals and then starts patrolling.

Gameplay elements

For the gameplay elements, I have chosen a Timer and power ups as I feel like these were perfect for what I have already. To get the timer up and running I had to make a new variable called frames. This variable increases every frame in the `on_update` function and with that in mind, I was able to convert that into seconds by using an if statement. I knew 60 frames = 1 second. I added another variable called seconds which increases by 1 every time the frames variable hits 60. (Shown in Appendix 15) After seconds was increased, I set frames to 0 to increase the seconds variable again. With this down, I was able to implement timers for things such as the flashlight duration which decreases only when the flashlight is on. I also implemented a small timer, so the Shooting Beast NPC fires a projectile every 3 seconds.

As for the power ups, I implemented these by rendering a box and putting a texture on the box. The power ups in the game are Health increase, Ammo increase, Grenade Increase and Flashlight battery increase. After creating the shape of the power up, I started creating a bounding box around it for the collision. This process was done to the triangular prism shape that I made specifically for the flashlight battery powerup. I then set the position of the power ups and their bounding boxes to the last position of the enemy that was just killed (See appendix 16). I then make a random number variable and which ever number comes out determines the power up dropped. I also made it so that power ups do not spawn if you have maxed out attributes like Health, Ammo, and grenades.

What I could do to improve this is combine the 2 elements for example have the powerup disappear after a certain amount of time. This makes getting loot from new enemies better as If the loot from previously killed enemy has not been picked up, the same type of loot won't spawn again until it has been collected and in some cases, loot may be hard to find if you don't remember the location. I could also make the pickup rotate to make it more noticeable and to point out that it is an item that is obtainable. This can be done by updating the rotational axis every frame.

Discussion

This project has been a great learning experience for me. As an aspiring game developer wanting to create games in Unity I felt like programming in Open GL C++ allowed me to realise that I can do things that I never thought I would be able to do. There were many times where I thought that I am really getting comfortable coding and debugging problems within my code much easier than before. I also felt like I did a decent job polishing some areas that wasn't required such as the movement animations.

1 of the more enjoyable parts of the project and probably my strong point was the implementation of NPCs. I was able to use what was given to me and expand on it by adding various types of enemies and states. I am also proud that I was able to figure out how to render and update multiple enemies while checking collision for every 1 of them using vectors and for loops.

Another proud moment is getting the camera to be significantly different to the template code and nearly replicating camera techniques used in popular franchises like Resident evil. I put a lot of time and effort trying to learn the code and what some things meant like `front_vector` and so on.


I would also say that Physics was a good area for me too as I was able to implement 3 different types. Gravity is the physics that I am most proud of as I was able to translate a video tutorial of implementing jump and gravity in Unity into OpenGL C++. Despite the Jump not working as smoothly, it still did the job and the gravity worked perfectly.

On the other hand, I feel like my weaknesses are shown in the primitive objects. I was not able to meet the requirement of creating 3 primitive objects from scratch. I did not fully understand how to use what I learned with some of the primitive shapes that I aimed to create such as the hexagonal prism however in the future, I will focus more on this area.

I feel like if I were to expand on the game even more I would polish some areas like animations for the player and NPCs. I would like to add various gun types for example, I could have a shotgun which will create multiple bullet objects from the player position with slight value change for each bullet in the x and y value. Then these bullets will shoot in the direction of the player forward value but have a spread just like shotguns do in other FPS or 3rd person shooters.

Overall, this project was a great learning experience and has taught me a lot. I definitely feel more comfortable using C++ and I hope to use this skill in the future. May it be creating more projects in OpenGL or using an engine like Unreal.

Appendices

Appendix 1		
Appendix 2		
Appendix 3	<pre>// North Wall from initial position for (int i = 0; i < 18; i++) { glm::mat4 wall_transform(1.0f); glm::vec3 wall_position = glm::vec3(-29.f + i * 4.f, 4.5f, -38.f); glm::vec3 wall_boundingBox = m_wall->bounding_shape(); wall_transform = glm::translate(wall_transform, wall_position); wall_transform = glm::rotate(wall_transform, 1.571f, glm::vec3(0.f, 1.f, 0.f)); wall_transform = glm::scale(wall_transform, m_wall->scale()); engine::renderer::submit(textured_lighting_shader, wall_transform, m_wall); }</pre>	
Appendix 4		
Appendix 5		
Appendix 6		
Appendix 7		

Appendix 8	
Appendix 9	<pre>m_spotLight.Position = m_player.getPosition(); m_spotLight.Direction = m_player.getForward();</pre>
Appendix 10	<pre>m_pointLight.Color = glm::vec3(1.0f, 1.0f, 1.0f); m_pointLight.AmbientIntensity = 3.f; m_pointLight.DiffuseIntensity = 3.f; m_pointLight.Attenuation.Constant = 1.3f; m_pointLight.Attenuation.Linear = 0.3f; m_pointLight.Attenuation.Exp = 0.3f; m_pointLight.Position = glm::vec3(1.f, 4.0f, 1.0f);</pre>
Appendix 11	<pre>// Player Jump upwardsSpeed += gravity * (float)time_step; if (m_object->position().y <= 0.52f) { upwardsSpeed = 0; isGrounded = true; } else { isGrounded = false; m_object->set_position(glm::vec3(m_object->position().x, m_object->position().y + upwardsSpeed * (float)time_step, m_object->position().z)); }</pre>
Appendix 12	<pre>// Check container collision for (int i = 0; i < numOfContainers; i++) { if (m_player_box.collide(m_container_box_vector.at(i))) { m_player.moveBackward(1.f * time_step); } }</pre>
Appendix 13	<pre>tankZombie.initialise(m_tankZombie, tankZombie_props.position, m_tankZombie->forward(), 0.5f* difficultyMultiplier, 100);</pre>
Appendix 14	<pre>if (m_state == state::resting) { m_health += time_step * 5.5; if (m_health >= 50) { m_state = state::idle; } }</pre>
Appendix 15	<pre>////** Timer **** timerFrames++; if (timerFrames == 60) { seconds--; if (seconds <= 0) { std::cout << "Time is up"; m_player.setisTimerOut(true); m_player.setisDead(true); m_crossfadeDeath->activate(); } timerFrames = 0; }</pre>
Appendix 16	<pre>if (m_fastBeast_vector.at(i).getHealth() <= 0) { glm::vec3 lastposition = m_fastBeast_vector.at(i).last_position(); if (randNum == 0 && healthSpawned == false && m_player.getHealth() < 75) { m_healthPack->set_position(glm::vec3(lastposition.x, lastposition.y - 0.1f, lastposition.z)); healthSpawned = true; } }</pre>

Asset list

Textures

All Assets have been accessed recently on 6th December 2:40AM. (I went to double check each asset to see what website they are from)

- Terrain
 - <https://freestocktextures.com/texture/grunge-gray-blue-wall,1232.html>
 - License: <https://freestocktextures.com/license/>
- Skybox
 - <http://www.vwall.it/wp-content/plugins/canvasio3dpro/inc/resource/cubeMaps/>
- Wall
 - <https://www.pinterest.co.uk/pin/493707177874111147/>
 - License: <http://downloads.photoshoproadmap.com/paperbackgrounds>
- Steel Texture (Bullet)
 - <https://www.dreamstime.com/stock-photo-brushed-steel-metal-texture-background-image46381395>
 - License: <https://www.dreamstime.com/terms>
- Acid Texture (Enemy projectile)
 - <https://www.dreamstime.com/green-grunge-acid-texture-grim-oil-liquid-hazardous-design-element-green-grunge-acid-texture-grim-oil-liquid-image178950860>
 - License: <https://www.dreamstime.com/terms>
- Ammo box
 - Crate texture: <https://opengameart.org/content/3-crate-textures-w-bump-normal>
 - License: <https://opengameart.org/content/faq>
 - Ammo bullet pixel art: <https://www.dreamstime.com/vector-pixel-art-bullet-isolated-cartoon-vector-pixel-art-bullet-image150887311>
 - License: <https://www.dreamstime.com/terms>
- Grenade Box
 - Bomb image: <https://opengameart.org/content/bomb-sprite>
 - License: <https://opengameart.org/content/faq>
 - Crate texture: <https://opengameart.org/content/3-crate-textures-w-bump-normal>
- Grenade Plasma texture
 - <https://www.deviantart.com/ndugger/art/Tileable-Futuristic-Grid-312360821>
- Battery energy texture
 - <https://unsplash.com/photos/F8NIC0FAEgc>
 - License: <https://unsplash.com/terms>
- Spike steel blood texture
 - <https://www.dreamstime.com/royalty-free-stock-photo-steel-background-drop-blood-image19752795>
 - License: <https://www.dreamstime.com/terms>

Meshes

- Enemies (All types)

- Normal Zombie: <https://clara.io/view/c584ee5d-e6d5-479f-af01-df569e20390e>
- Tank Zombie: <https://clara.io/view/9ad817e6-09ef-499c-9dd1-9879e666f1f5>
- Shooting Beast: <https://clara.io/view/1d3c6c00-476e-464d-b918-2b2d5b8e10ff>
- Fast Beast: <https://clara.io/view/53ad3f71-eb8d-44ec-a0f7-40a2c74ab241>
- Rifle: <https://sharecg.com/v/18828/browse/5/3D-Model/colt-xm177-Machine-gun>
- Torch: <https://sharecg.com/v/81969/browse/5/3D-Model/torches>
- Street Lamp: <https://www.sharecg.com/v/72379/related/5/3D-Model/Antique-Street-Lamp>
- Container: <https://free3d.com/3d-model/container-169022.html>

Audio

- Torch sound effect (Switch on and off): <https://www.zapsplat.com/sound-effect-category/torch/>
 - License: <https://www.zapsplat.com/license-type/standard-license/>
- Gunfire1 and 2 and Slash sound: <https://www.fesliyanstudios.com/royalty-free-sound-effects-download/gun-shooting-300>
- Music from: <https://www.fesliyanstudios.com/royalty-free-music/downloads-c/dark-music/12?page=2>
- License: <https://www.fesliyanstudios.com/policy>

Libraries

No additional libraries used.