# IN3026 – Advanced Games Technology Milestone 2 (Full Build)

## Laurentiu Maties

# Overview

Since the Interim build, Cursefall has matured into a more polished game than the raw materials that were captured previously. I have now developed it into a two-level dungeon crawler experience where the player must solve puzzles and correctly interact with NPCs to progress. The dungeon idea was moulded into a modern yet abandoned house as the adventurer attempts to remove the creature (horror) that is haunting the mansion. Despite having a sword, the player may only cast a magic lightning ability learned through a pickup and a grenade that is used as part of the game’s main puzzle. Furthermore, some enemies are only vulnerable to either the grenade or lightning, while some are not harmed by either.

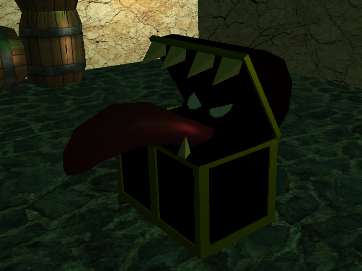
A picture containing text, green

Description automatically generatedAfter navigating through the main/options menu, the player spawns in front of the residence. The player’s main goal is to defeat the horror however to do so, they must find a way to get to the monster.

Furter away from the mansion, a lemur roams the area. Once caught, the player will be given a speed bonus and the ability to quickly turn by 180 degrees through pressing the G key. This buff is not necessary to complete the game however it will make the boss fight easier. This is an example of the variable difficulty present in the game.

Outside the residence, next to a barrel lies a spinning electricity power up. The player must pick this up as the horror is vulnerable to electricity, and invulnerable to grenade explosions.

Upon entering the residence, the player is welcomed with a health potion that they may pick up. This will shield the player from a single boss hit and is not necessary to beat the game, however it will give the player a slightly easier time.

Into the middle of the ruined room, a mimic patrols back and forth. The player must defeat it by throwing the single grenade at it, without startling it. If the player gets too close to the mimic, it will change its states to aware and if defeated in this state, it will not drop the key to the next level (set to a grenade for the time being however a key model would be more appropriate). If, however, the grenade makes an impact while the mimic is not aware of your presence, it will drop an item that once picked up, will teleport the player to the second level…the boss room. The grenade follows the football functionality from lab 6 with some adjustments

A picture containing green

Description automatically generatedOnce inside, you will be able to see the horror patrolling in triangles. Upon closing onto the monster, it will become aware of you and face the player if they remain within this radius. Any closer and the monster will begin chasing and attacking the player, dealing damage, and reducing their life count. The player can retaliate through using their electricity spell and healing with the potion they might have picked up earlier. The best strategy for defeating the horror is to kite it as the range of the lightning spell is slightly greater than the range of the horror’s attack. This is made easier with the lemur buff.

To move, the player walks forwards and backwards with the **W and S keys**, while using **A and D keys** to turn left and right. To throw a grenade, the player must press the **1 key**. You must use it wisely as you only get a single grenade, with the second one being awarded for correctly defeating the mimic. In addition to this, the lightning bolt attack is set to the **2 key**. This means that after gaining the pickup by the barrel (outside the mansion), the player will be able to cast lightning bolt, a high damage attack that has a 3 second cooldown. While the mimic is invulnerable to this. It is the only way to defeat the horror. Finally, upon catching the lemur, the player will be able to turn by 180 degrees by pressing the **G key**

# Part 1: Basic game modelling

### Intro screen

The intro screen is composed of a free-to-use image, to which I added features such as the game’s name and keyboard controls. The intro screen is in fact a physical 3D object in the scene, positioned underneath the map. Despite the cost associated with rendering the object, this implementation allows the game to transition to the main menu through a simple camera change. Alongside the main menu, I have also added an options menu that describes the game controls. The textured of these menus were created by me, using the Clip Studio Paint software (paid version) where I added the text/symbol layers on top of the original image. Furthermore, these objects recycle the pre-existing terrain model so there was no need to create another object.

### Objects using OpenGL primitives

The first primitive I have created was the healing potion. This is a hexahedron with two slanted faces and a cuboid stacked on top. I have chosen to texture this so that it is more visible than a regular coloured mesh.

Furthermore, the wall torch consists of two components: the base and the holder.The base is a fairly straight forward, angled hexahedron while the holder is more complex in that it theoretically connects a prism leading from the angle of the base, onto a cuboid (equilateral faces) on which the light sphere is meant to go. Instead of a prism below a cuboid however, the torch is rendered at once, with a slanted square backside.

Diagram, schematic

Description automatically generatedMoreover, the final primitive consists of a table and can be found scattered throughout the two levels. Seeing as this was a bigger piece, I wanted to take a more automated approach by manually defining the vertex coordinates for the initial table leg and reflecting these coordinates to create the other table legs. The issue with this is that the method of reflection resulted in incorrect winding and normal coordinates. As highlighted in this figure, a reflection will end up giving you the opposite coordinates on the plane that you are operating on. This mean that in order to fix this, I had to reverse the indices so that the vertex coordinates are reflected, and fed into the mesh creator as intended.

Text

Description automatically generatedText

Description automatically generatedIn addition to creating these primitives, I have also changed the skybox and ground textures. The issue with the ground texture however, was that it looked extremely stretched for the terrain size I was using. I have resolved this issue by modifying the engine::cuboid constructor to require an additional float parameter named repeat. This float would then replace all of the points with a Cartesian value of 1 (0s were left untouched) in the texture coordinates for every declared vertex in cuboid. Following this, the Terrain constructor was also changed so that it accepted the additional A picture containing stone, ocean floor

Description automatically generatedparameter and upon usage of this object, I have called terrain with a repeat value of 1. In the case of the ground however, I called terrain with a repeat value of 60 which allowed me to have an extremely smooth looking terrain texture. The downside of this is that the texture used must be seamless such that one will not be able to easily spot the repetiton. Furthermore, the repeating texture combined with the high quality of it makes it seem like the rocks in the scene are 3D at a first glance. This change was inspired by something I have noticed during the Computer Graphics module, where all of the simple primitives were instantiated with a repeat value for the texture.

The night-time skybox was also chosen such that it provides a relevant – ominous – feel to the scene. This, alongside the cold-looking stone floor sets the first impression of the game right away.

### Audio

For audio, I have added 9 different sounds. When launching the game, the player will hear the ominous track as they scroll through the menu. Once the game has started, this track is paused and three different tracks are layered and played at the same time. A more muted and equally sinister track plays alongside a recording of birds and the sound of rain. These three tracks give the game a relevant amosphere. The audio implementation is similar to the one from the Lab exercises, with the addition of more sounds and event-based audio triggers such as the pickup, healing and throwing sounds effects that trigger when the respective action is taking place.

### Chart, funnel chart Description automatically generatedHead’s up display

The Head’s up display is constructed by a combination of, text and images. Upon game start, the HUD is enabled. In the top left corner, I have used the text manager and renderer to display the HP of the enemy such that the player is able to judge this when fighting it. This was done using a new getHealth method added to the enemy class. In addition to this, the player’s own HP is shown in the top right corner. As the players’ HP is reduced, the image rendered changes to a less-filled heart. I have used Clip Paint Studio on the original heart texture to create multiple other images that are loaded into a vector at initialisation of example\_layer.cpp. Following this, whenever the player’s HP hits specific thresholds (80,60,40,20,0), the corresponding texture will be loaded in the corner of the screen. This is done by rendering a textured quad, similar to the cross-fade effect, with smaller dimensions so that it does not take up the entire screen. The downside of this implementation is that whenever the cross-fade effect takes place, the health quad is not affected. Have I had more time, this would have been something that I would have liked to change.

# Part 2: Camera, meshes, lighting

### Camera motion technique

For the camera motion technique, I have opted for a first-person view where the camera is displaced as the player is walking, giving off a more immersive feel. When the game starts, the camera begins lower than usual, in attempt to give off the feel of the adventurer standing up. Furthermore, the player may also turn using the A and D keys. I went for this motion technique since I believe it gives the game a more retro feel such as DOOM. The camera is set such that it follows the player’s forward vector and stays positioned next to the player. Furthermore, the player’s sword is positioned based on the camera’s front vector where the blade rotates about the y axis based on the x and z coordinates of the front vector where:

glm::rotate(sword\_transform, glm::atan(m\_3d\_camera.front\_vector().x, m\_3d\_camera.front\_vector().z), glm::vec3(0.f, 1.f, 0.f))

### Mesh based objects

For the mesh-based objects, I have added the grenade, barrel, mimic, lemur, and animal models. These are all static objects, with the only animated mesh-based object being the mannequin used by the player. With more time, I would have implemented an additional 3rd person camera angle that would show off the mannequin animations. For this reason, I have decided to keep the animations. Furthermore, I would have also liked to implement some of my own animations as I was able to take the mimic model and split the pre-existing animation (from the blender model) into 200 individual frames. Using a similar technique to the player HP texture, I would substitute the model with the next frame in line, during the on\_update() method. This way, the mimic appears as it is moving its mouth while patrolling.

### Use of lighting or colouring effects

For lighting, I have made use of all three types (spot, point, directional). The directional light is used to illuminate the world with a soft green while the spotlight in the boss room is intended to act as a torch, with the light source being placed inside of the red sphere. Furthermore, the spotlight is used to shine on the intro and option screens. Since the scene is not very well light (helps the atmosphere of the game) and the objects are underneath the map, the spotlight is essential. A directional light would not function as it would end up placing more light into the scene, and a point light would not be able to provide the same round, blurred edge effect as a spotlight. When the player switches between the main menu and options menu, the spotlight moves along with the current selection, which means that there is no need to instantiate multiple spotlights. Another addition I would like to have is give the player their own spotlight to act as a flashlight, and further dim the scene’s lights.

A screenshot of a video game

Description automatically generatedAn issue I have had with the point light, however, was that the light would shine on every surface, regardless of whether there was a wall in front of it. While this is slightly confusing, it does not completely ruin the mood of the game. With more time, I would have liked to fix this issue however it is not a game breaking bug as the point light itself works as intended.

### Special effects

The three special effects in Cursefall are the billboard, crossfade and lightning bolt. For the cross-fade, I have adapted the implementation to make the transition lighter such that the transparency begins at a lower value that it was initially set in the lab. Furthermore, this special effect is used when the player drinks a healing potion (green crossfade effect) and when the player is damaged by the horror (red cross-fade effect). For the billboard, it felt appropriate that it must be used in conjunction with the grenade such that when the grenade contacts the mimic, the billboard explosion is rendered. Finally, I have altered the implementation for the lightning effect such that it is much shorter, and it always points in the direction of the player’s forward vector.

# Part 3: Physics, AI, and gameplay

### Use of game physics

The physics used in Cursefall are collision boxes, grenade bouncing, gravity and character movement (velocity, momentum, resistance, acceleration, drag, friction) . The grenade bounce is inspired from the football lab exercises, with less gravity. The downside of the grenade implementation is that the grenade bounces through the walls of the house and is not easy to remove on command. This is a bug however and with more time, I would have been able to get to the bottom of it.

The single barrel at the entrance of the house, the mimic, horror, and lemur have their own collision boxes. The player catches the lemur when their collision boxes interact and when the horror’s and player’s collision boxes interact, I have used the lab code to ensure that the two units do not intersect.

I have made an attempted at jumping (commented in the code, player class) however if I had more time I would have liked to make this function as intended. This only works partially now however I will be working on this later.

### Non-player characters / AI

Cursefall has three individual non-playable characters that use AI to make the game more engaging. The main monster has 5 states:

* Patrolling: The horror moves in a triangular motion (based on changing its front vector and velocity.)
* on\_guard: The horror notices the player once they are within the detection radius an faces them.
* Chasing: the horror begins to move towards the player once they enter the trigger radius
* Resting: once the player has completely gotten out of the monster’s detection radius, the horror takes a breather (after chasing the player), standing in one position for 4 seconds, after which going back to patrolling.
* Attacking: once the player is within the attacking radius of the horror, it will start hitting the player on a cooldown. This works hand in had with other states as the horror will be chasing the player and trying to get as close as possible.

The Lemur also patrols, however in completely random directions. The forward vector of the lemur is randomly chosen every few seconds. Furthermore, instead of chasing, the lemur’s flagship state is “scared”. This means that when the player gets close to the lemur, it will try to escape and run away by adapting the forward vector of the player and continuously moving in that direction until the player is no longer in range. Once the player is out of range, it will slightly speed up before going back to its patrolling state.

Finally, the mimic patrols in a straight line, much like the cow from Lab 8, except that when the player enters its detection radius, it becomes aware of it’s presence and begins to spin. The inspiration for this was taken from the pickup code (Lab 2).

### Gameplay elements

The potion primitive is used as a pickup item that allows the player to heal 20 health by pressing the 3 key. Once done, the potion is consumed and may no longer be used. This slightly decreases the difficulty of the game as the player will be able to tank an extra hit from the horror.

The lightning symbol next to the barrel at the house’s entrance makes a sound when it is picked up and gives the player, a necessary ability that is the lightning bolt.

Catching the lemur gives the player the ability to move faster and carry out a 180 degree turn with the press of a button (G key). This is the most significant indicator of variable difficulty as the final boss fight can be done without it, however it is significantly more difficult to do so without the speed buff.

An addition I would make would be to further develop the lemur and mimic, while also adding an additional NPC that would perhaps follow the adventurer and maybe even help in the fight with the final boss.

# Part 4: Discussion

In summary, I have hit the majority of the requirements for this task. The game presents a creative and different solution to the intro and option screens and has three complex primitives which vary greatly from one another. The primitives are texture mapped appropriately and are scaled and rotated to fit the scene. Furthermore, there is significantly more audio present than the requirements specify, and they are accurately placed in the relevant places with the game’s theme in mind. In addition, the HUD not only contains a custom font (Creepster – open font) showing the monster’s HP but the player’s HP is linked to a custom, visual heart graphic. The main weakness of Part 1 is that due to the aggressive triangles on the torch primitive, it is very difficult to apply texture coordinates that make the overall texture look seamless upon closer inspection.

In addition to this, the wobble technique adds an additional element to a first-person motion and fits well within the game genre and provides a significant change to the template code. The mesh-based objects are individual, correctly rotated/translated/scaled and provide great quality to the game. The use of all three lighting types in this game is very much relevant beyond the need for checking a requirement. The background directional light provides ambience while the point light albeit trivial supports the scene. The spotlight also plays onto the theme of the game with its presence on the main menu. The special effects go beyond the requirements and have been altered to completely cater to the game, which is why the lightning bolts were reduced in sized and the crossfade used in more than one way. Overall, this was also a strong section and with more time, I would work towards adding more animated/rigged models to my game, alongside special effects such as fog.

Furthermore, the use of game physics is present throughout the whole game and is something the game relies on. The collision, bounces and interactions improve the game’s feel and look.

Notes:

Talk about variable difficulty gameplay through using the speedup buff()