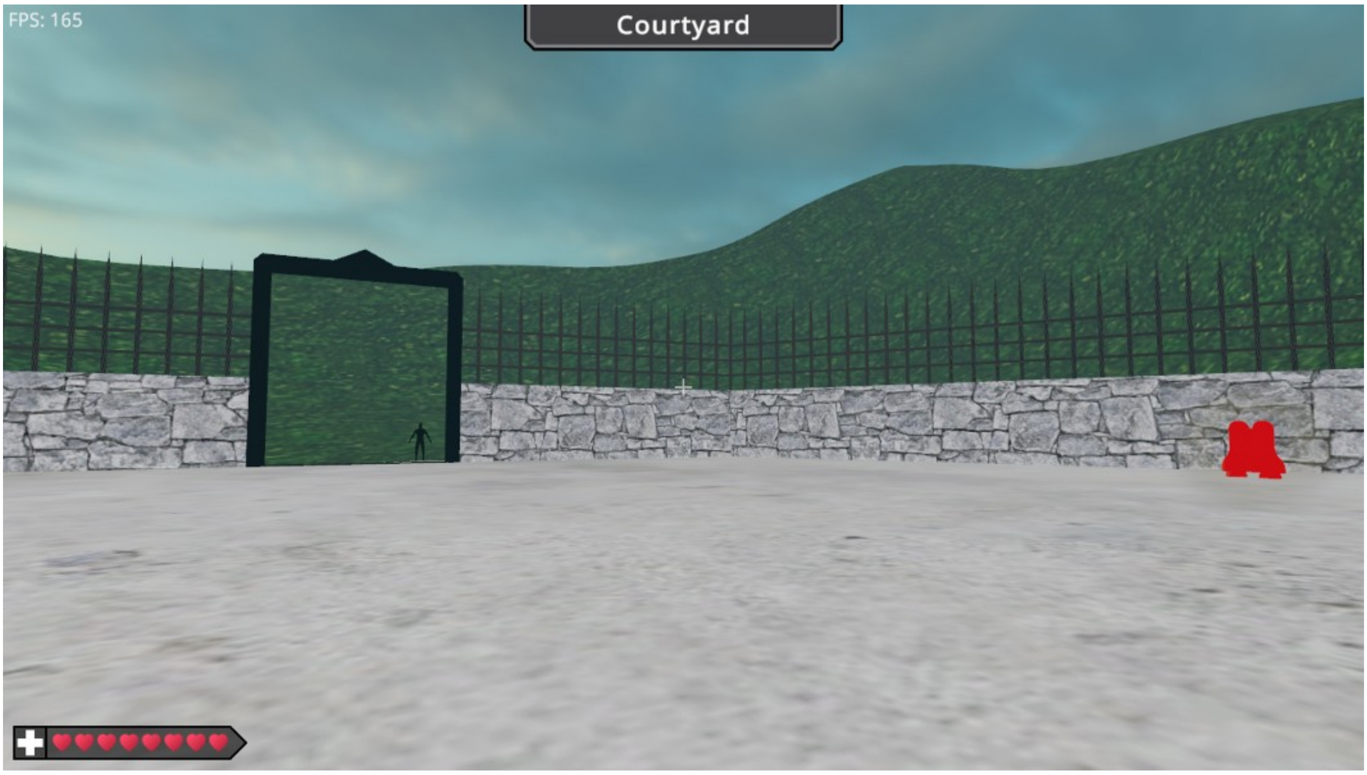
Video Game and Console

A report by Calvin Hogg

Major Projects 2022
Semester 1, Assignment 3

Table of Contents

Overview



The game's courtyard area.

When I initially pitched a video game for Major Projects earlier in the year, I wanted to create something that would serve as a (for lack of a better word) physical representation of my previous works. Visual and musical elements would work to generate certain feelings that tie in with the corresponding work and create atmosphere.

The overall style of the game would be reminiscent of 90s video games, when 3D graphics were still fairly new.

I had planned to learn the inner workings of the Raspberry Pi hardware, and use that knowledge to create both a customized operating system and game console that would feature it. Due to time limits however, the console never got far into development, and most of what was shown in assignment 2 reflects its current state.

The player would've encountered signs and non-playable characters (NPCs) throughout the game that would provide tips and allude to the terrifying beast in the dungeon. Additionally, a graveyard area was planned for behind the castle, with various statues and easter eggs of the real world (for instance, photogrammed people from Applied Media).

Many other parts of the game are unfinished. Most textures are placeholders, pickups don't always work, and the game lacks narrative. Progress on the console and it's operating system were stalled around assignment 2's due date, so the content shown in the corresponding workbook is an accurate representation of it's current state.

Research and Contextual Influences

Music

I have been researching Koji Kondo and his musical work at Nintendo for Context and Practice this semester. Again, due to time limitations I was unable to compose any background music for the game or reach out to arrange collaborative work, let alone music inspired by Kondo's works.

The only notable piece of music in the game is in the dungeon level. This track was sourced from a free-to-use video on YouTube which I have since lost the link to.

The ominous cutscene that plays upon entering the pit at the end of the dungeon corridor was intentionally designed to confuse the player, and takes the format of a full-motion video or FMV, something often seen in early 3D games. This pit initially led to a scrapped level, but was changed for certain reasons. The audio heard during this cutscene is an edited sample from a Portal 2's An Accent Beyond, composed by Mike Morasky.

Visuals

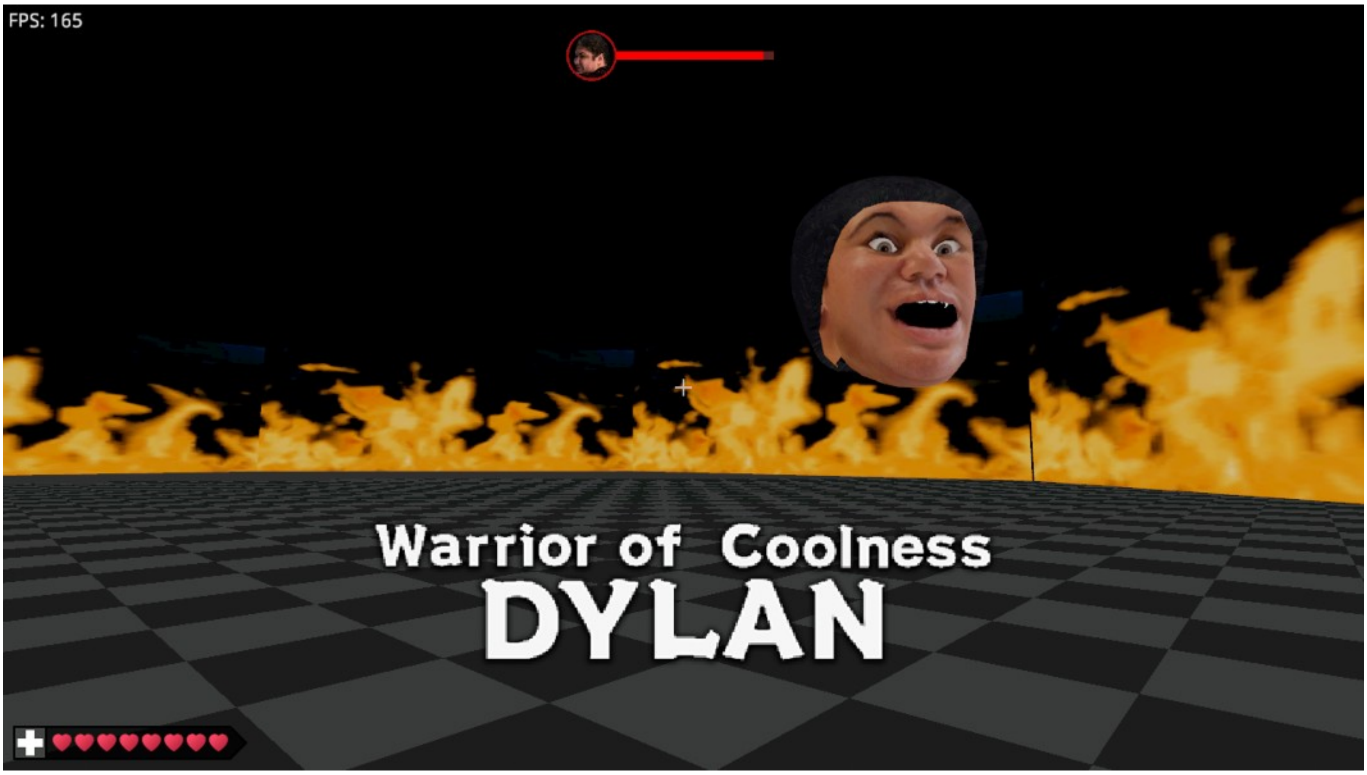
I tried to recreate the visual style of 90s games partly through the use of baking lighting into the texture maps themselves, and using low-poly geometry, to name a few. I'm sure these techniques would've helped the game run on low-end hardware such as the planned Raspberry Pi-based console, keeping framerates at a high enough level, and temperatures slightly lower.

The courtyard level resembles the starting level in Super Mario 64, where in both instances the castle is bigger on the inside and serves as the hub to other levels. In my game, the interior of the castle cuts away to a space-like void with an absent elevator in the middle that would've taken the player to the gallery, Cyberthug Core and dungeon. An early mock-up of this elevator can be found in the Cyberthug Core level, albeit with the UV guide left in by mistake.

The boss fight is introduced by the large white text describing the boss, a reference to the gameplay in the Nintendo 64 Zelda games, which Dylan himself is very familiar with.

Unfinished Content

Boss fight

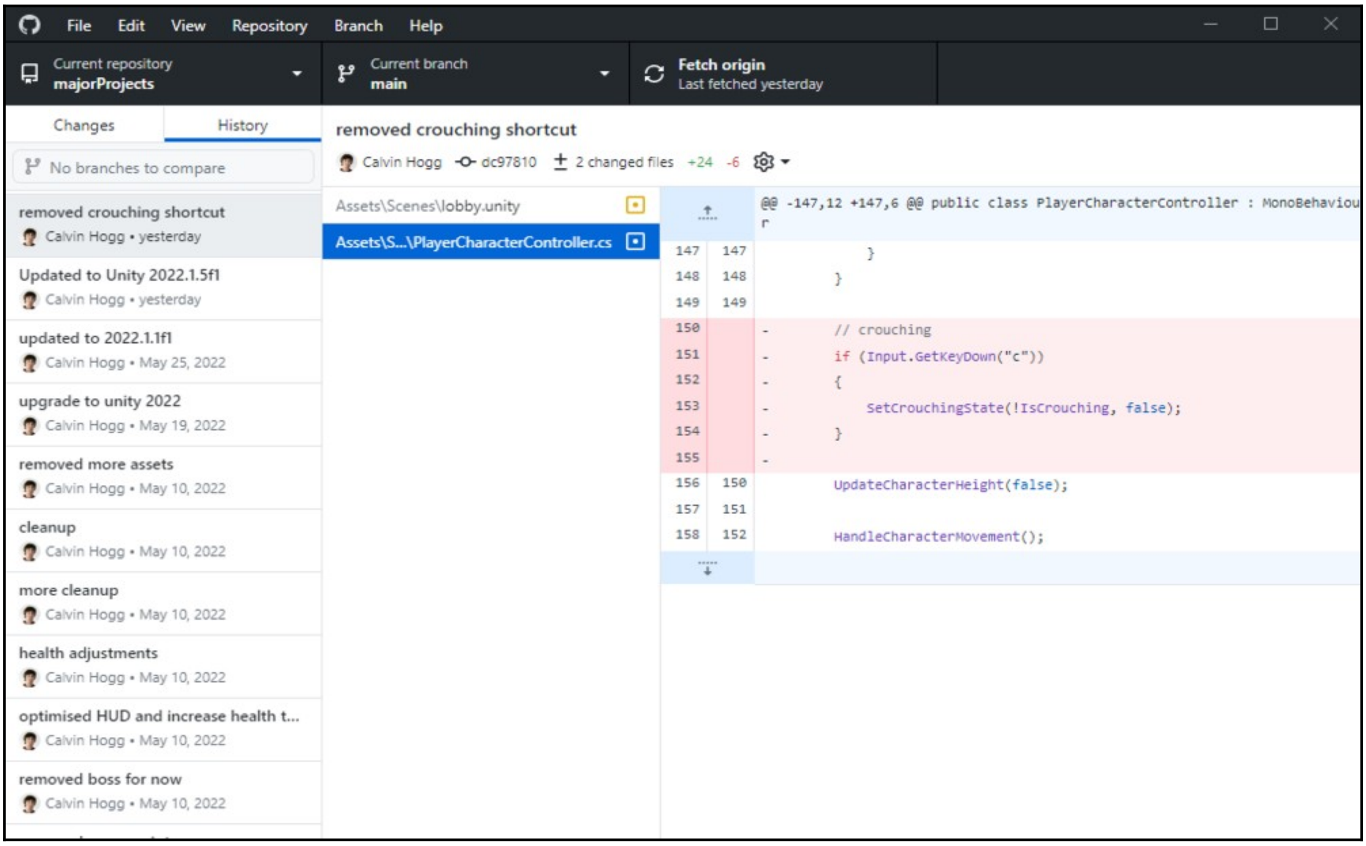


The boss fight area.

The game's ending would've taken the form of a boss fight, in similar fashion to many video games. Only an early version of this made it into the final release due to time limits. The player would've battled against the disembodied head of Dylan Nathaniel-Jones, a classmate in Major Projects. I had noted a few ideas for the boss fight, such as knocking him off the platform into the void below or shooting him with a spud gun.

In the final release however, Dylan merely flies around the platform laughing and chomping while a health bar sits inactive in the GUI.

Development Process and Rationale



removed crouching shortcut
Calvin Hogg • yesterday

Updated to Unity 2022.1.5f1
Calvin Hogg • yesterday

updated to 2022.1.1f1
Calvin Hogg • May 25, 2022

upgrade to unity 2022
Calvin Hogg • May 19, 2022

removed more assets
Calvin Hogg • May 10, 2022

cleanup
Calvin Hogg • May 10, 2022

more cleanup
Calvin Hogg • May 10, 2022

health adjustments
Calvin Hogg • May 10, 2022

optimised HUD and increase health t...
Calvin Hogg • May 10, 2022

removed boss for now
Calvin Hogg • May 10, 2022

removed crouching shortcut
Calvin Hogg dc97810 2 changed files +24 -6

Assets\Scenes\lobby.unity
Assets\S...\PlayerCharacterController.cs

```
@@ -147,12 +147,6 @@ public class PlayerCharacterController : MonoBehaviour
                }
            }
-           // crouching
-           if (Input.GetKeyDown("c"))
-           {
-               SetCrouchingState(!IsCrouching, false);
-           }
-
            UpdateCharacterHeight(false);

            HandleCharacterMovement();
```

The GitHub commit history, showing every change made to the game's source code since May 10th.

Code management

Throughout the lifecycle of this project, I've used GitHub to regularly update my code changes to an online repository. This allows me to keep track of changes, and rollback the code if something breaks. I tried to maintain good practice by limiting each commit to a specific change. Before transferring the code from an old GitHub account, I had created releases of both the source code and built executables at regular points, along with accompanying changelogs describing new features.

Finalization

I had initially planned to lock everything down after loading the game onto the console, to prevent end-user tinkering. My reasoning for this was simply to think how a big company like Nintendo or Sony might think, and come up with various strategies for preventing piracy or copyright infringement etc. Not that my game is worth those efforts of course! Some strategies include glueing the Raspberry Pi's USB ports or constructing the case out of metal, which would discourage extraction of the game's and operating system's files.

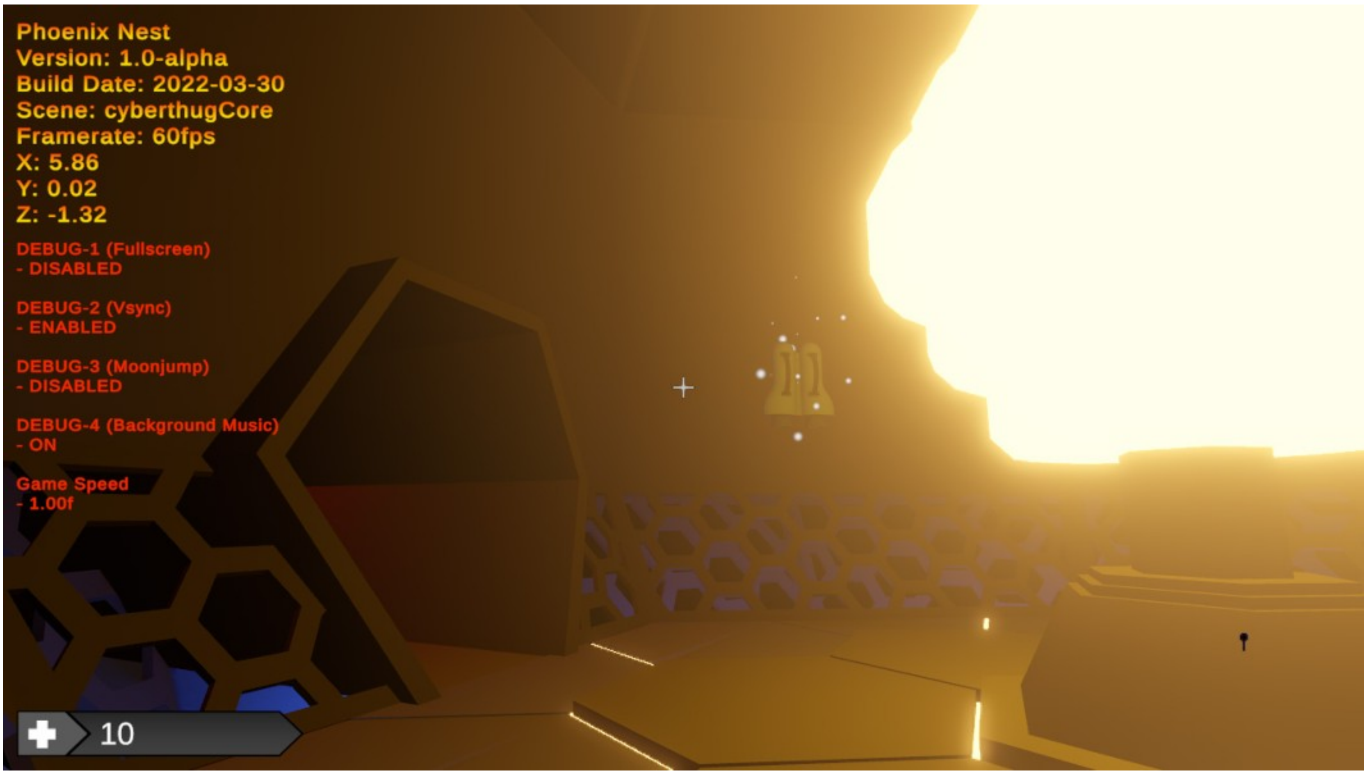
Choice of platform

It made sense to choose a variant of Linux for the console's hardware as it's open-source, which meant I could create my own OS. In assignment 2 I presented evidence of the operating system's development, namely a virtual machine running Ubuntu that had its desktop environment replaced with a simple window manager. This would've been built upon to create a graphical home-screen application, from which the player could launch the game or change various hardware settings such as the brightness or volume, assuming the console didn't feature physical sliders for adjusting said settings.

Gameplay

Had the console been completed, the game would've made use of its d-pad and joystick for interacting with the game. As this was not the case, I instead opted for simple mouse and keyboard controls similar to those found in most PC first-person shooter games. The WASD keys are used for movement, the spacebar for jumping and the mouse for looking around.

Lighting system

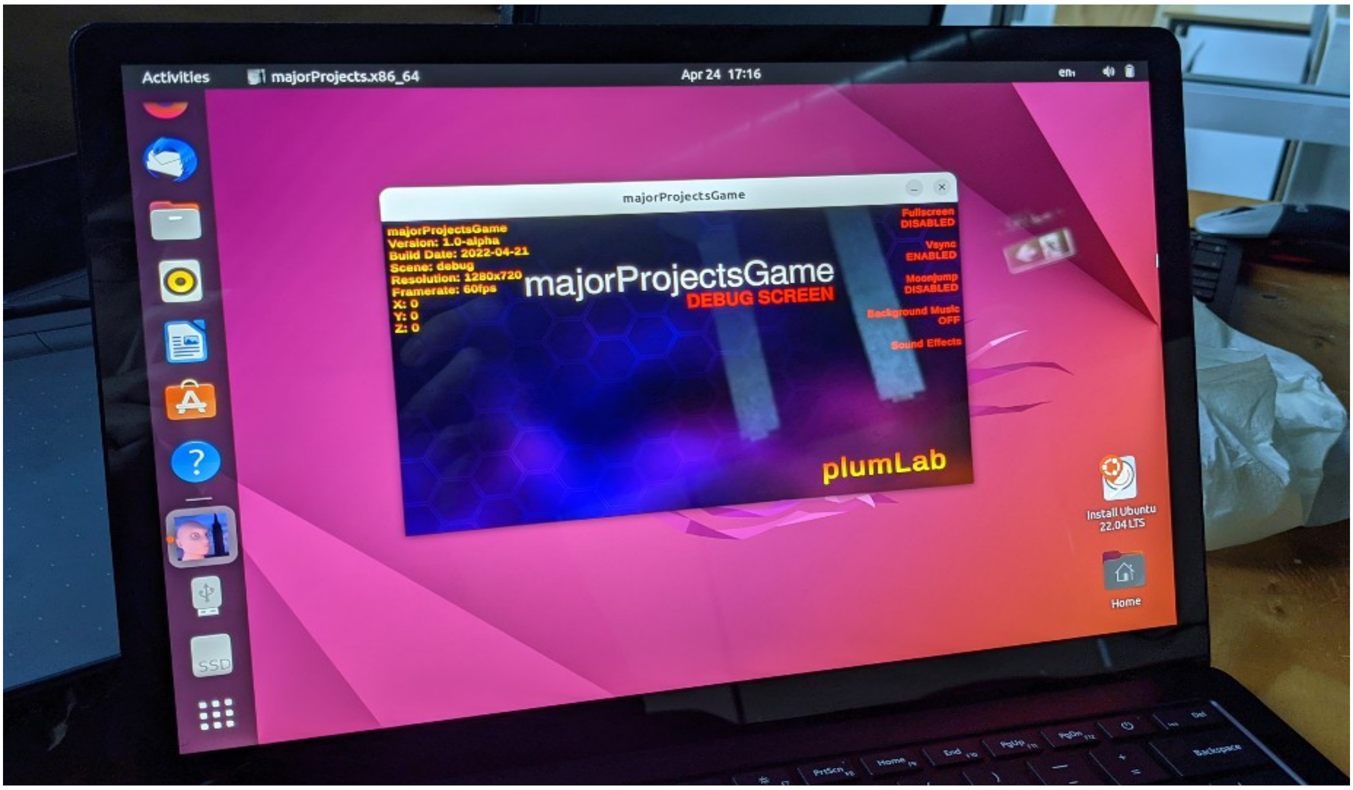


```
Phoenix Nest
Version: 1.0-alpha
Build Date: 2022-03-30
Scene: cyberthugCore
Framerate: 60fps
X: 5.86
Y: 0.02
Z: -1.32
DEBUG-1 (Fullscreen)
- DISABLED
DEBUG-2 (Vsync)
- ENABLED
DEBUG-3 (Moonjump)
- DISABLED
DEBUG-4 (Background Music)
- ON
Game Speed
- 1.00f
```

The Cyberthug Core with Unity's lighting system and post-processing effects.

Something you'll notice in the final game compared to the presentation (assignment 2) is the notable downgrade in graphical quality. All traces of Unity's lighting system were scrapped in favour of textures with lighting effects baked in. I had hoped that this would reduce strain on the Raspberry Pi's hardware, but as it was never completed, the differences in performance on decent desktop PCs are minimal.

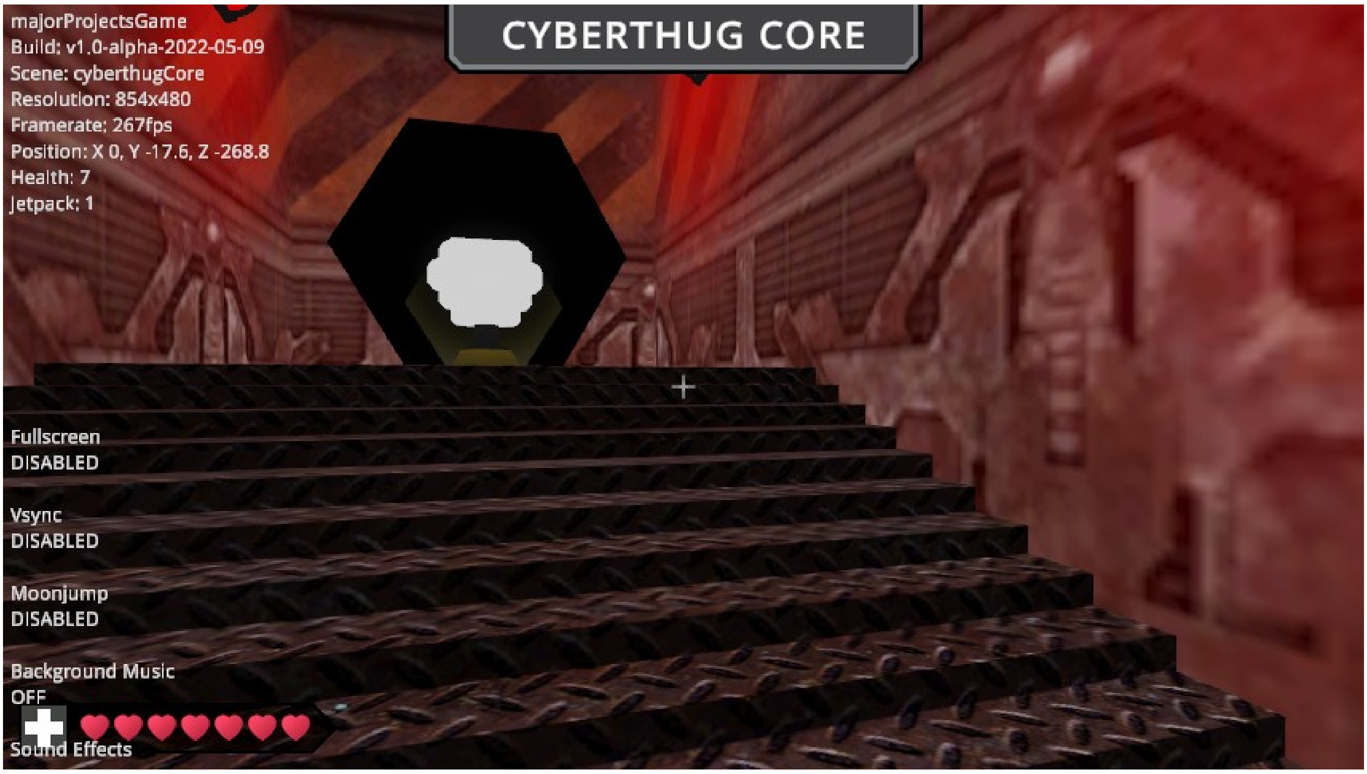
Visual Documentation



```
majorProjectsGame
Version: 1.0-alpha
Build Date: 2022-04-21
Scene: debug
Resolution: 1280x720
Framerate: 60fps
X: 0
Y: 0
Z: 0
```

```
Fullscreen
DISABLED

Vsync
ENABLED

Moonjump
DISABLED

Background Music
OFF

Sound Effects
```

Proof of the game running on Ubuntu Linux. The console would've run a trimmed down version of this OS.



```
majorProjectsGame
Build: v1.0-alpha-2022-05-09
Scene: cyberthugCore
Resolution: 854x480
Framerate: 267fps
Position: X 0, Y -17.6, Z -268.8
Health: 7
Jetpack: 1
```

```
Fullscreen
DISABLED

Vsync
DISABLED

Moonjump
DISABLED

Background Music
OFF

Sound Effects
```

The Cyberthug Core level, with statistics and debugging features visible.

Conclusion and Reflection

Overall, I can see that I've left a significant amount of work out of the final hand-in, mainly due to time limits, especially with regards to the console and operating system (although to be fair, that was always considered a bonus project). That being said, I've learnt a few things about the Unity game engine and coding in the C# language, as well as basic tearing-apart of the Ubuntu operating system.

With regards to the second semester of Major Projects, I think it's more important for me to be – for lack of better words – less experimental when it comes to the field of the chosen project. As mentioned in the proposal document, I had chosen game development for this semester in hopes of learning new skills and stepping outside my comfort zone, but I realize that at level 7 it's more important that I show what I'm actually good at. I plan to focus on computer-generated imagery (CGI) as it's my primary area of interest. You can expect to see a 3D render or possibly a CGI short film being produced by me in the second half of the year.

Bibliography

Gordon, Whitson (2019, August 30). *Beginner's Guide: How to Get Started With Raspberry Pi*. PCMag.

https://www.pcmag.com/how-to/beginners-guide-how-to-get-started-with-raspberry-pi

This article serves the simple purpose of explaining how the Raspberry Pi works, and what one needs to set up a rudimentary desktop experience. The guide carefully lists the hardware needed and provides a short tutorial for installing an operating system to its primary storage medium, the SD card.