Assignment 2

Video Game Progress

The courtyard level, like the one in the proposal

The Cyberthug Core, successfully added with collision and lighting

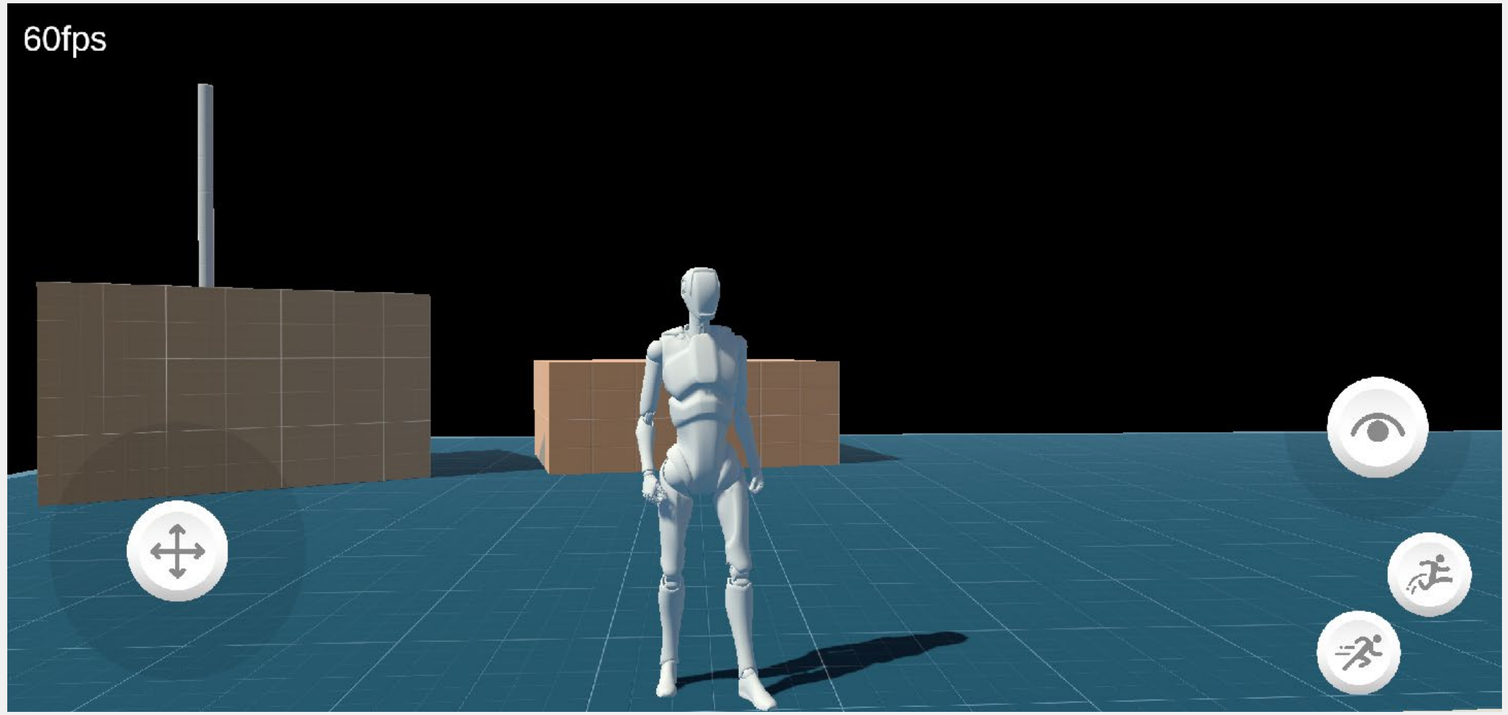
Assignment 2: Video Game Progress

Summary and Gameplay Mechanics

Progress on the video game is going well. In the alpha stage, I've prioritized my effort on creating the basic level geometry and gameplay mechanics, while detail is still lacking for now, and any sort of plot or narrative is currently absent.

That said, the foundation is there, and with the screenshots shown you can see the game will feature health and jetpack meters, as well as a map. The text in the top-left is for debugging purposes and won't be visible in the final release in June.

As mentioned in assignment 1, the game will serve as a "physical manifestation" of my artwork, with thematically-relevant areas such as the life-size Cyberthug Core shown to the left, and an art gallery featuring photographic work.



A sample third-person template provided by the Unity devs, showing a placeholder robot model, some objects and touchscreen controls.

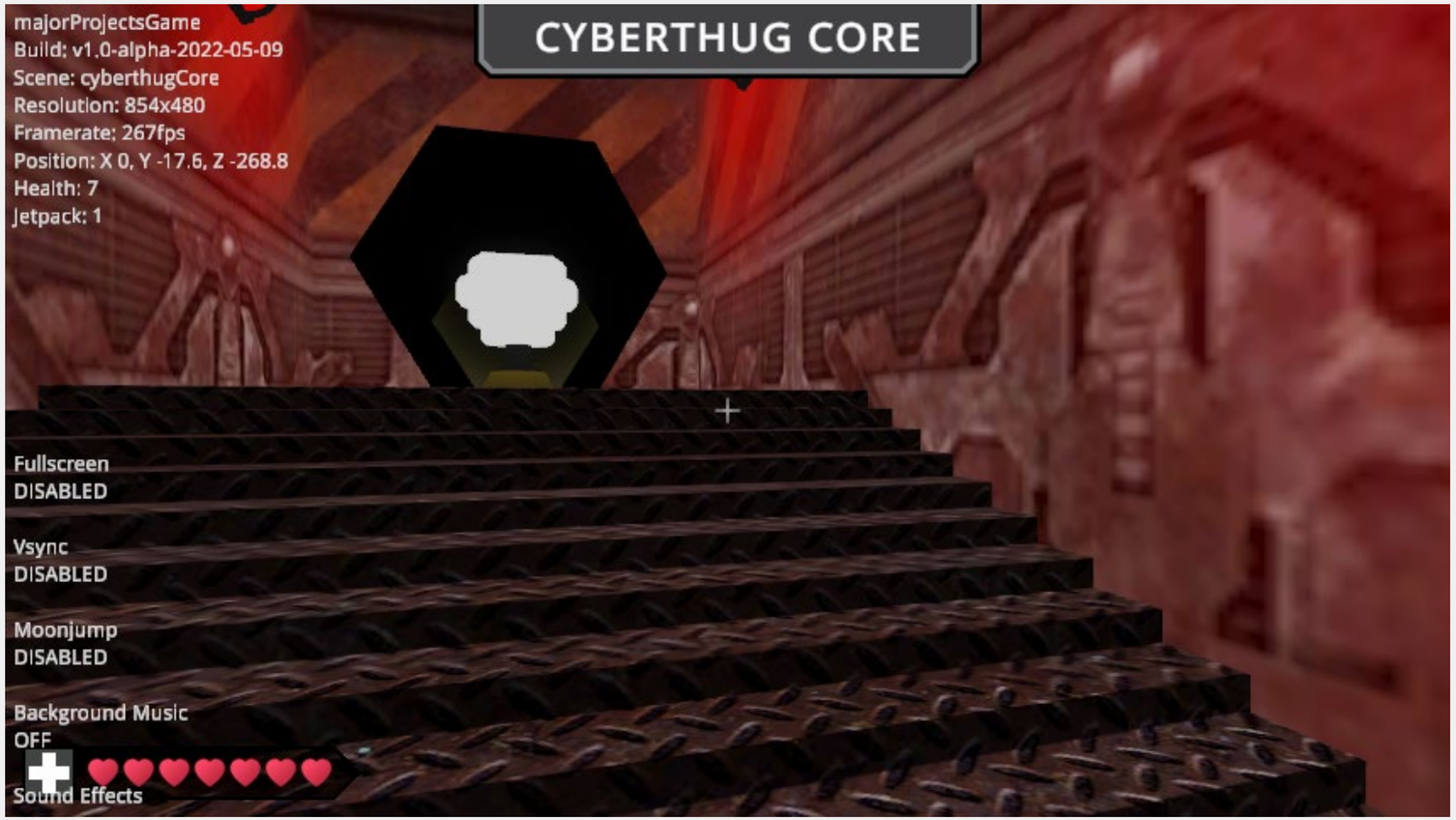
Assignment 2: Video Game Progress

Third-Person Perspective Test

One of the early ideas that I wanted to implement in the game was having the Cyberthug characters make an appearance in some way. Other than using them as non-playable characters (NPCs), I didn't have many options, as a first-person perspective would hide the player's model.

A third-person perspective, such as the one shown here, would allow the Cyberthug player to be seen in all his glory, along with animations for walking, running and jumping.

Trimming the first-person template to it's core took some time back in January, so whether I go ahead with this is largely time-dependent.



The Cyberthug Core stage with unlit textures.

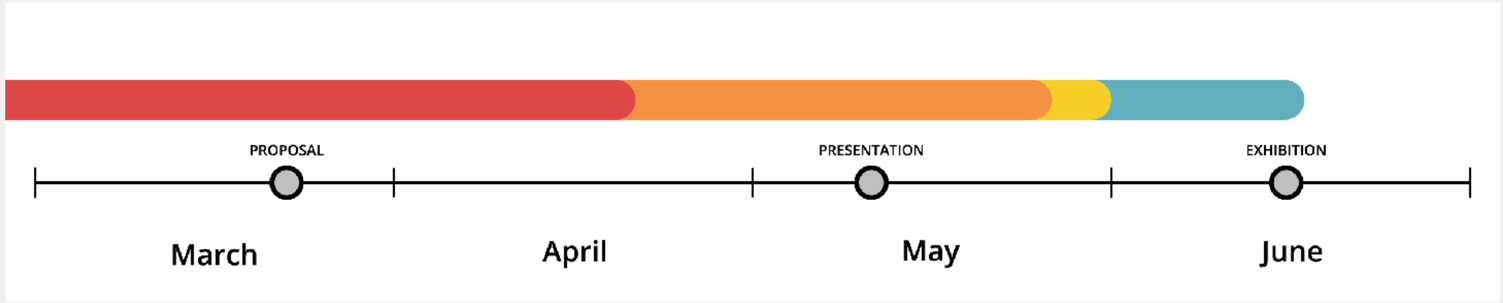
Assignment 2: Video Game Progress

Thematic Considerations

At some point I decided to ditch Unity's lighting system and give the appearance of lighting through texture maps.

I figured this might reduce the performance impact on the console's hardware while giving the game a Nintendo 64-era style.

There's something mysterious about old low-quality graphics in video games compared to the realistic graphics nowadays.



The original timeline created at the start of the semester

Revised Timeline

Despite my best efforts, work on the project isn't quite as far ahead as the original timeline.

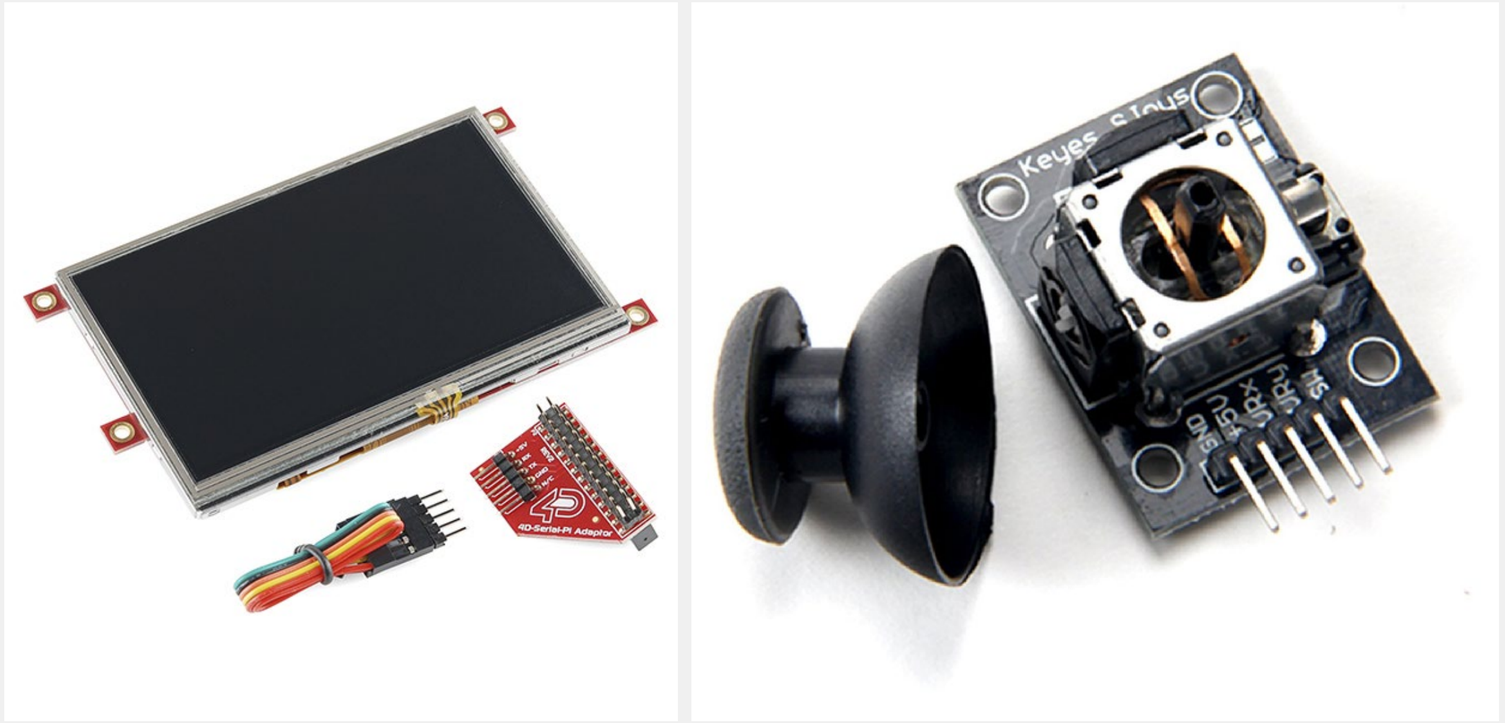
I had planned to move the game into the beta stage in mid-April, but a lot of content is still missing, leaving the end result still unclear at this point.

I think a big reason for this is focusing on refining various elements too early, such as making the HUD pixel perfect for the target resolution. I need to remember the alpha stage is for creating placeholder assets and basic functionality, which is to be refined later.

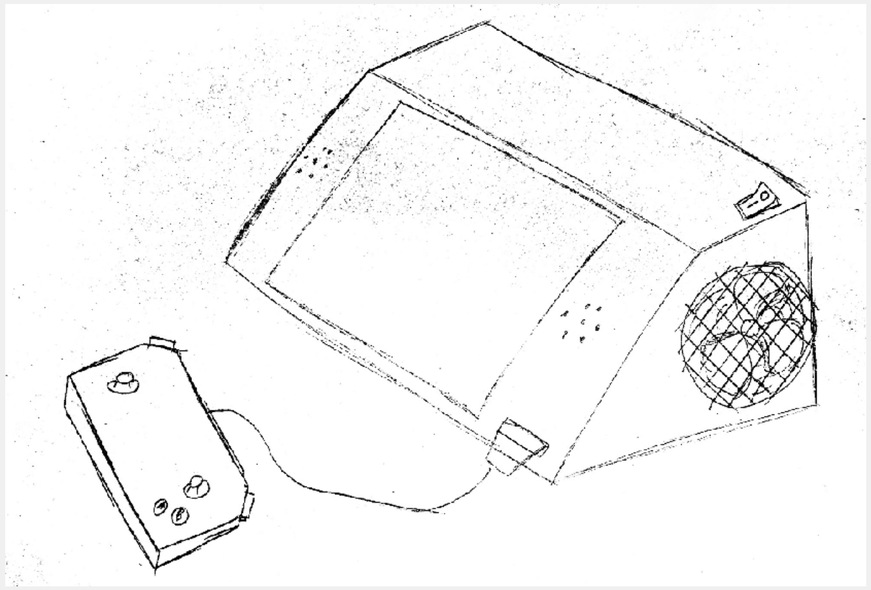
With all that said, I think the game could use an extra week or two in the alpha stage.

Assignment 2

Console Progress



Some of the various modules available for the Raspberry Pi, namely a display and joystick.



Assignment 2: Console Progress

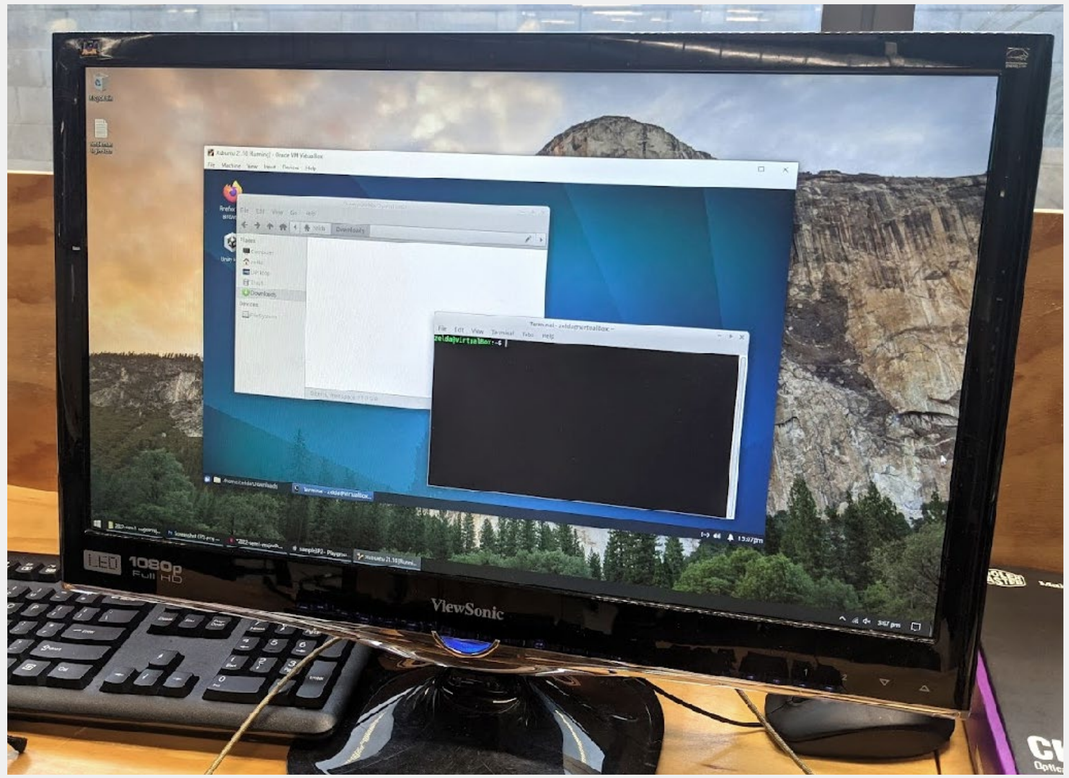
Console parts and construction

Pictured are modular components that can be used with the Raspberry Pi. The console would use common parts like a display, speaker and controls.

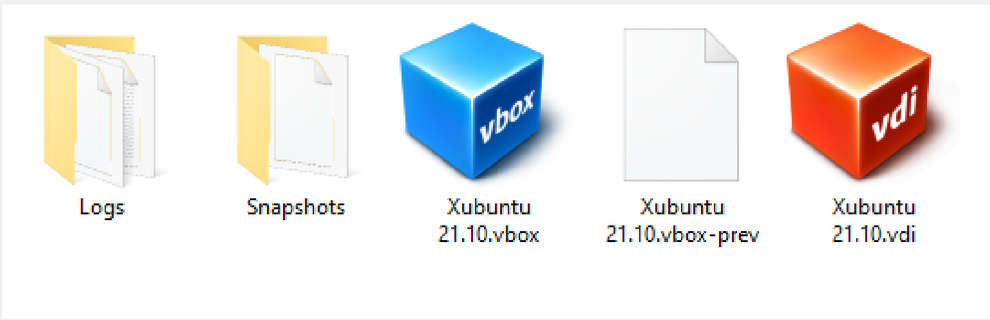
As for the outer shell, I'm familiar enough with laser-cutting that it could be constructed with MDF and cast acrylic. Alternatively, I could 3D print the casing.

Depending on the Raspberry Pi's capabilities, I could fit the whole thing into a single handheld device, or have the main bits contained in a small unit, with the controller separate.

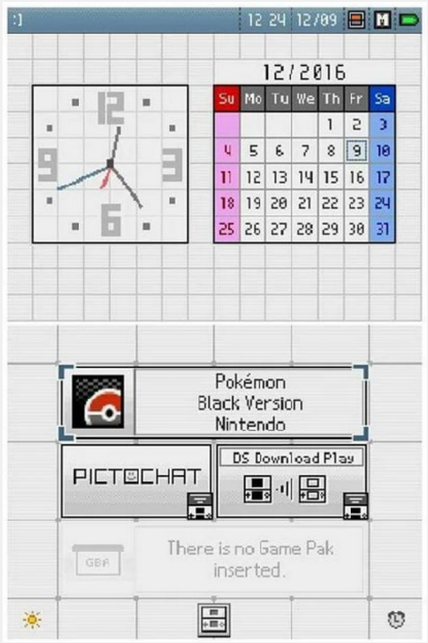
I happen to be building my own PC at the time of writing, so that knowledge would come in handy.



Ubuntu 21.10 running inside a virtual machine, with the file manager and terminal open. This installation uses the lightweight XFCE desktop environment.



The virtual machine's files as they appear to Windows



The Nintendo DS homescreen

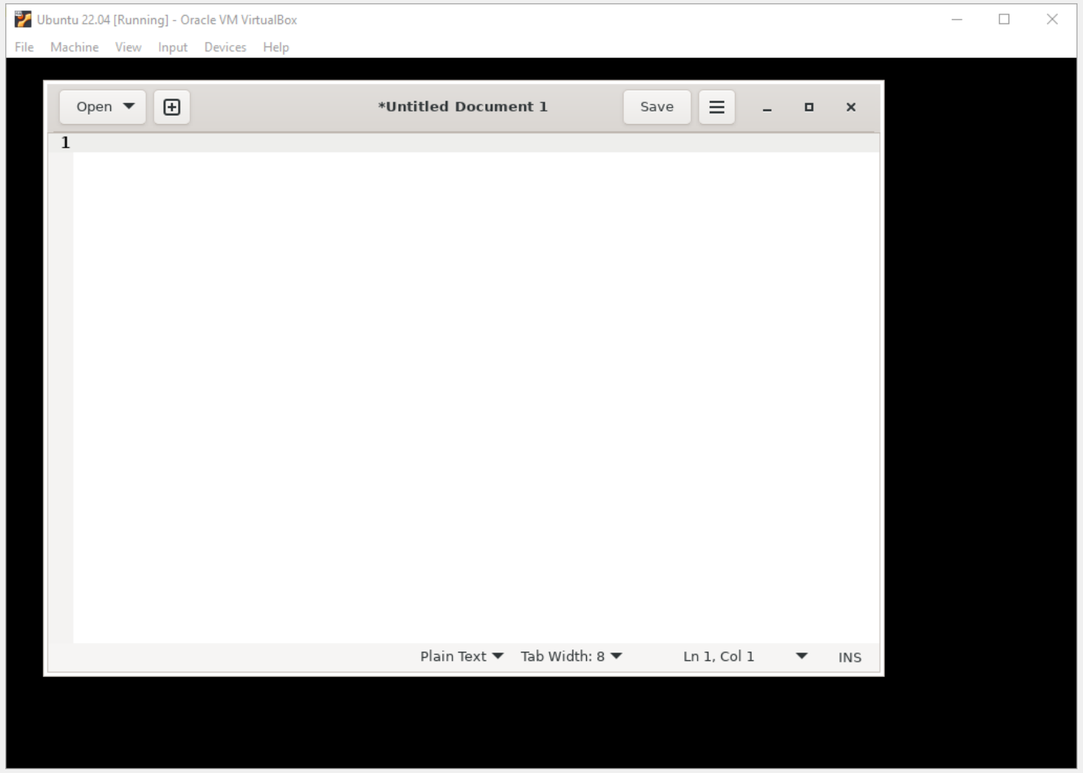
Assignment 2: Console Progress

Console OS

I figured due to time constraints trimming down an existing Linux installation would be quicker than learning how to build it from source.

By running Ubuntu inside a virtual machine, I can easily create snapshots to restore if I break something, and keep the OS on my external drive so I can work on it from different computers.

You can also see the homescreen of the Nintendo DS, which provides access to the inserted game cartridge and a few other useful options. This is what I had in mind to replace the desktop environment seen in the top-left. From a consumer standpoint, a full desktop environment would be both confusing and provide the option to hack/cheat the game, which would not be good.



The OpenBox window manager with Gedit running.

The game running on Ubuntu 22.04

Assignment 2: Console Progress

Console OS

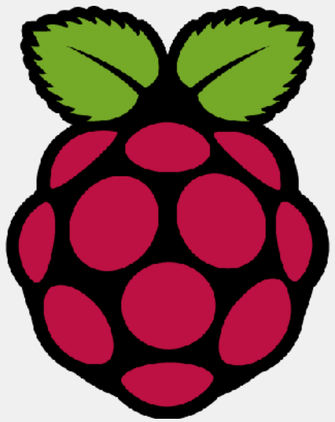
I installed the newly released Ubuntu 22.04 in a virtual machine and recklessly removed packages relating to the GNOME desktop environment. Rebooting resulted in Ubuntu showing the command line interface (CLI).

I installed a simple window manager called OpenBox and managed to make it run Gedit (a text editor) on startup.

When I tried to run the sample game, I got an error about a “segmentation fault”. It took a lot of trial and error, but I managed to get the game working by disabling 3D acceleration in VirtualBox, albeit with a pitiful 2 frames per second.

When I tested the build on a Ubuntu live USB, the framerate reached around 1000.

For reference, I’m aiming to reach a stable 60 FPS for the console, as this is considered the minimum for enjoyable gameplay these days.



Raspberry Pi

The Raspberry Pi typically has its operating system stored on a MicroSD card, which is inserted into the board itself.

Model 4B is powered by the Cortex-A72 ARM processor, running at 1.5 GHz, and the Broadcom VideoCore VI GPU, running at 500 MHz, and up to 8 GB of RAM.

The nature of the Raspberry Pi doesn't make it the best console for gaming, so I would need to take steps to optimize the game near the end.

Once I'm sure the game runs well on the console, I would probably solder the SD card in to prevent unauthorized access (I'm trying to think from the perspective of a successful game company with strict license terms).