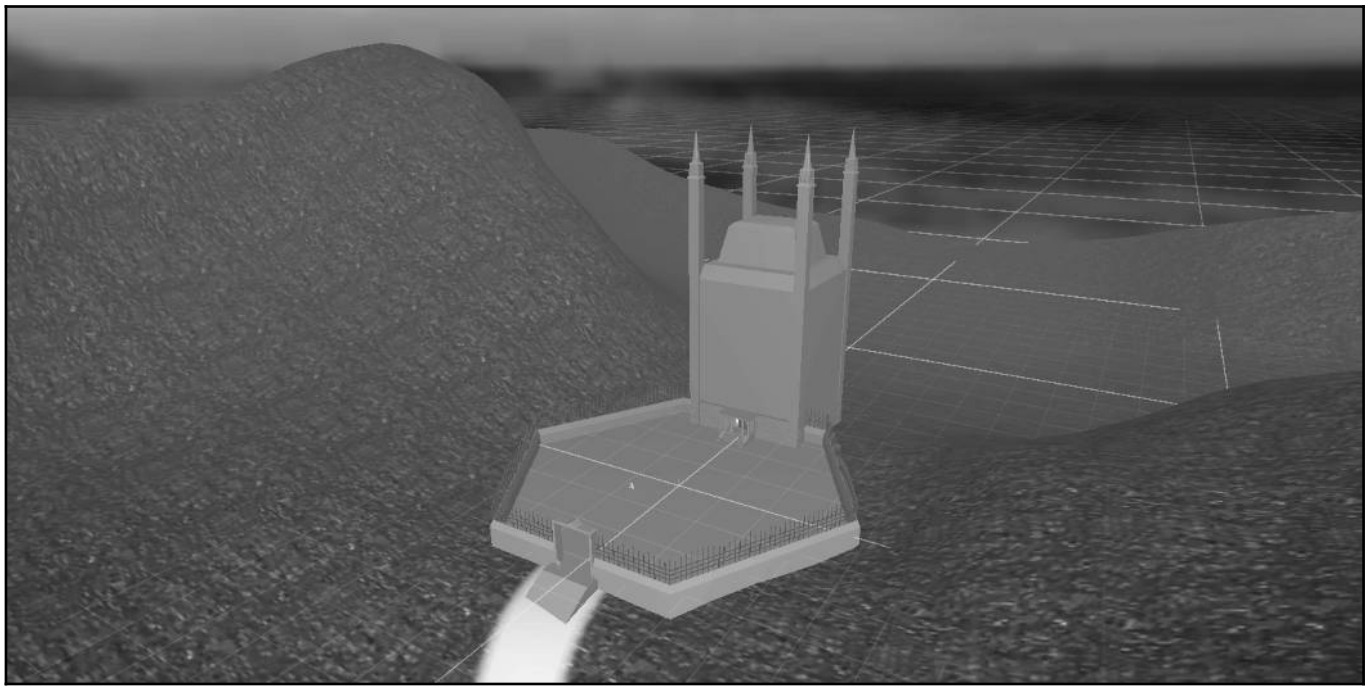
Video Game and Console

A proposal by Calvin Hogg

Major Projects 2022
Semester 1, Assignment 1

Table of Contents



An early build of the game showing the courtyard scene.

Introduction and Methodology

I plan to create a video game for Major Projects Sem1 2022. It will serve mainly as a showcase of my creative work thus far, displayed in various thematically-relevant ways (such as an art gallery with framed professional photos and a walk-in Cyberthug Core). There might be a plot/story added later, but this is largely dependent on the amount of time I have.

The game's genre will be a first-person puzzle/adventure, similar to Valve's Portal series.

I've had a tendency to adhere to a certain amount of realism with my previous work, which in retrospect I think limited my capabilities as an artist. This project will attempt to break through that with more abstract elements, things that wouldn't necessarily make logical sense in a material environment.

I also plan to build and program a console for the game to run on both as a challenge and an opportunity to include more examples of skills in the overall project. The console would be small, ideally a handheld device similar to the Nintendo Switch if things go to plan. I would use a Raspberry Pi as the core, with modules for the screen, speakers and controls. The console would run a bare-bones derivative of Debian Linux or Android, with just a launcher and settings menu.

The reason I chose this particular project for the first semester of Major Projects was to further develop skills in the fields of game development, small-scale construction, and programming. Admittedly game development isn't my main area of interest, but I figured it was good to get out of my comfort zone and have a basic understanding in that area.

Practitioner Research and Inspiration

Nintendo circa 1997 (Mario 64 and Zelda: Ocarina of Time)

I chose to research Nintendo as they're well known for their Mario and Zelda franchises. Two personal favourites are The Legend of Zelda: Ocarina of Time and Super Mario 64. These titles both came out in the mid-90s for the Nintendo 64, and were the first 3D titles for each franchise. The improvements Nintendo made graphics-wise are apparent when compared to their recent counterparts, Super Mario Odyssey and Breath Of The Wild.

The gameplay in SM64 is very open-world, and is fairly flexible when it comes to how the player chooses to progress in the game. That said, each level is largely the same aside from the visuals, and the plot is only mentioned occasionally throughout the game. The plot is that the protagonist has to save Princess Peach from the evil Bowser, although this is only brought up occasionally. The disjointed levels combined with the lack of characters make the game as a whole feel like an experiment, like a whole load of test maps. Almost as if Nintendo is using this as a prototype for future 3D Mario games.

Ocarina of Time on the other hand, is much more story-driven with a clearer plot. It's considered a favourite by many Zelda fans. It follows a boy named Link who is called upon to save the land of Hyrule from the evil forces of Ganon. Every stage in the game is connected which results in a believable integrated world. The addition of unique characters and plot-related cutscenes make this a far more realistic experience compared to SM64.

Valve circa 2009 (Portal franchise)

I also chose to research Valve and their Portal series, mainly due to the atmosphere of the games. Both games' gimmick is the use of placing two portals strategically to progress, keeping concepts like physics and momentum in mind while doing so.

The key difference between them is that the original is noticeably puzzle-oriented while the sequel is comparatively more story-driven, with added characters and dialogue. While the sequel showed more of the fictional world of Aperture Science with the old locations showing signs of deterioration, the original's dull atmosphere managed to do a better job of instilling a sense of loneliness and surveillance, and the long depressing ambient soundtrack further captures this.

Video Game

Research Objectives

A big part of the work behind the game is modelling real-life objects. My workflow for this involves visiting the locations, taking photos from different angles to get a good idea of the various proportions and details, as well as some clean shots of textures that could be used. Due to the fact that I'm making a game and not a film, the textures don't have to be perfect. I had considered putting photogrammetry to use here, but decided it wouldn't be suitable for a game due to the high poly count and messy texturing. Most of the objects are hard-surface models and are therefore easy enough to create manually, as well as being more practical. Later in the game's development, focus would shift to performance optimizations where possible in order to make gameplay smoother on less capable hardware, such as the console.

Software and Development

I initially planned to code the game entirely from scratch using a language like C++ or C#. I thought this would allow me to have more control over how the game works, and avoid the Unity splash screen that shows up in the free edition of the editor. This idea didn't last long, as it seemed like a lot of work that would've been more trouble than it was worth, so I decided to keep using the Unity engine.

The main programs I would use for the creation of the game are Photoshop and Illustrator for the images and textures, Autodesk Maya for modelling, GitHub Desktop for uploading changes, Unity for making the game itself and Visual Studio for writing the scripts.

Project Management

A personal Trello board will be used to keep track of bugs, tasks, ideas for new content and other stuff relating to the development of the game. I figured this was a good choice as it works well on my laptop and phone, and other school stuff uses the same service.

The source code for the game will be frequently uploaded to a private GitHub repository. This will serve as the "digital history" of the project, with milestones at various points and the ability to easily revert the code if something breaks. The repository's releases page will contain several builds created at regular points throughout the course of development, each following a specific version convention (for example, v1.0-alpha-2022-03-21)

Version 1.0 marks the first stable release, which is what this project is building towards. Each build can be in one of four stages that identify its place in development, listed below.

Alpha: Development focuses on adding new content, which may be rough around the edges. Some assets may be placeholders and are subject to change.

Beta: Development focuses on refining new content and fixing bugs. Performance optimizations are a big focus here.

Candidate: Version is ready for release unless significant bugs are found. Extensive playtesting and gathering feedback will typically happen during this stage.

Release: Version is ready to be distributed.

Console

Research Objectives

Considering the Raspberry Pi is a modular single board computer, it makes sense that the game and the console's operating system would need to be small and light enough to run smoothly without encountering issues like over-heating or lagging. I would need to research the different versions of Raspberry Pi available, compare their hardware capabilities and be aware of any potential limitations ahead of time.

I expect that the smaller I attempt to make the console, the more difficult it would be to cram components in. I would need to learn strategies to lessen the demand on the hardware, ensuring smoother framerates and reduced heat output.

Operating System

The console would most likely run on either AOSP (Android Open Source Project) or Ubuntu/Debian (popular Linux distributions), with customizations made of course. There are two ways I could go about it.

The first option is that I download the AOSP or Debian source code and customize it, before building an image that can be flashed to the console's internal storage. The second option is that I customize an existing installation of AOSP/Debian, which I expect would be quicker and easier.

The Raspberry Pi Foundation recommends their own Debian-derivative called Raspberry Pi OS. Their website provides links to download the operating system for Raspberry Pi, as well as a handy variant for desktop computers.

I would have to do benchmarks of how the game performs on the Raspberry Pi under both Android and Linux operating systems, which would play a large part in deciding which platform I proceed to use. At this point in time, I'm more familiar with writing apps for Android, although I'm not sure it's officially supported for Pi.

The operating system could have packages like the web browser and calculator removed, freeing up space for a personalized "launcher" which would greet the user upon booting and allow him/her to change settings such as brightness and volume, as well as launching the game.

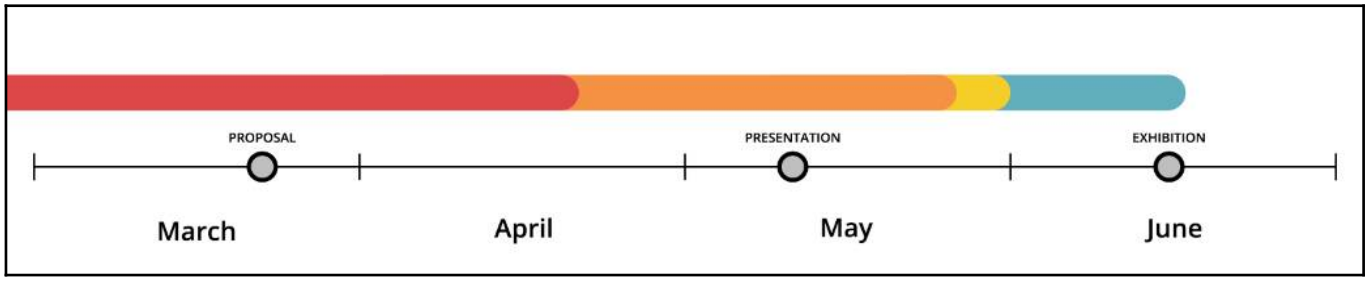
Desktop Environment

In the world of Linux, the desktop environments each have their own style. By default, for example, the latest version of Ubuntu runs the GNOME desktop out of the box, which is a good choice for most users. Considering once the game is launched, the desktop environment is largely out of sight, I decided that Xfce would be a better choice, due its balance between low system requirements and solid user interface.

Compared to those seen on Android phones and Ubuntu desktops, the console would only need a barebones experience used to launch the game and provide access to settings like brightness and volume. Something straightforward like the Nintendo DS launcher.

Timeline

I planned out a little timeline that shows how progress on the game and the console align with the assignment due dates.



Red: Game development alpha stage.

Orange: Game development beta stage.

Yellow: Game development candidate stage.

Teal: Priority is on the console.

Deliverables and Exhibition

The deliverables and exhibition options will largely depend on whether the console is completed in time. If all goes to plan, the final hand-in will include a video showing the game and the OS running on the console as well as an extensive workbook documenting everything. Any physical work relating to the console is solely my property however and won't directly be considered part of the submission.

If I'm unable to get the game working on the console for whatever reason, gameplay will be recorded on the Windows desktop platform (most likely running on my laptop).

The game's source code and executables will remain closed-source for now, though this may be reconsidered at a later date.

Bibliography

About Xfce. (n.d.). Retrieved from Xfce:
https://xfce.org/about

Brant, T. (2019, August 7). Raspberry Pi 4 Review. Retrieved from PCMag:
https://www.pcmag.com/reviews/raspberry-pi-4

Building Android. (n.d.). Retrieved from Android Open Source Project:
https://source.android.com/setup/build/building

Raspberry Pi. (n.d.). Retrieved from Wikipedia:
https://en.wikipedia.org/wiki/Raspberry_Pi

Raspberry Pi OS. (n.d.). Retrieved from Raspberry Pi:
https://www.raspberrypi.com/software/