# Inertia

Design document for the video game called Inertia

By Jonathan Berkeley & Mark Hurley

Contents

[Introduction 3](#_Toc56975101)

[Program Design 3](#_Toc56975102)

[Technologies 3](#_Toc56975103)

[Structure of Unity 5](#_Toc56975104)

[Design Patterns 6](#_Toc56975105)

[Application architecture 7](#_Toc56975106)

[Database design 10](#_Toc56975107)

[User interface design 11](#_Toc56975108)

[Wireframe 11](#_Toc56975109)

[User Flow Diagram 12](#_Toc56975110)

[Style guide 13](#_Toc56975111)

[Paper Prototype 15](#_Toc56975112)

[Storyboard 16](#_Toc56975113)

[Level Design 17](#_Toc56975114)

[Environment 18](#_Toc56975115)

[Conclusion 19](#_Toc56975116)

## Introduction

The product is a 3D movement-based physics video game. The player navigates through levels using a grappling hook projectile gun to reach a location within the level. The player assumes control of an unnamed test subject who is armed only with a grappling gun. The grappling gun shoots a projectile which pulls the player towards the surface the hook strikes. The player needs to use momentum gained from this action to progress through the level. This unnamed test subject has the objective of escaping the laboratory they are trapped in by solving various physics-based puzzles using the grappling hook gun.

The game is beginner-friendly but can also be a challenge. It is designed to be easy to pick up but difficult to master. A timer is added into the game for those skilled players who are interested in beating the game in the fastest time possible. The game requires the player to make their way from one end of the level to the other while jumping across and around platforms. It will be a relatively short game with the core focus being on gameplay and level design.

This game is being developed in the Unity game engine. The Pro version of Unity is available to students free of charge, so the Pro version is being used. Unity comes with a useful tool called Unity Collaborate, a version control system, which syncs the project as changes are made, making development as a team much easier. Github is used periodically to log major changes to the project, Github is also for logging major issues that arise. The game scripts are coded in the C# programming language.

## Program Design

### Technologies

#### Unity:

Unity is a game engine used to massively simplify video game creation, and it is free. It is very accessible and easy to learn as there is a lot of documentation for it available free online. Unity will be used to create and edit the game’s assets and setting up various scenes the game will have.

Unity is the core tool that handles rendering, game memory, input/outputs, physics calculations, and much more for the game. Unity also has a collaboration version control system feature allowing development to sync changes seamlessly within the program without having to do any major downloads.

#### Affinity:

Affinity Photo is one of the best programs available for creating graphics and textures for the game. This is used in Inertia to create textures and graphics for the game such as the Main Menu screen, Game Over screen, and textures/colours for the various assets.

#### Visual Studio Code:

Visual Studio Code is used to edit and create the C# scripts that make up the code for the game. Visual Studio Code is useful as it has plenty of plugins and extensions as well as having a very modern interface and it is easy to use and easy to manage.

#### Visual Studio 2019:

Visual Studio 2019 is also used for creating and editing C# scripts. It is similar to Visual Studio Code but is a lot more heavyweight, containing a lot more functionality out of the box.

#### Github:

Unity Collaborate is the main tool used for version control in this project, though Github is used for periodic versions to store a history of changes in case something goes wrong during development. Github is where issues are logged and addressed as well, in the issues tab of the project.

#### Trello:

Trello is a list application used for planning development through a series of cards. It is very useful for coordinating efforts of development on the game, especially for planning what needs to be done for sprints. As well as logging what features have been finished and what still needs to be done.

#### Discord:

Discord is a program for communications originally designed for gamers to talk to each other. It is evolved into a multipurpose chatting program with support for embedded code with syntax highlighting. This is used for communicating and coordinating on the project.

#### Alternatives:

Alternatives to these were considered. Blender was considered for level and model development, though Probuilder plugin in Unity took over in this area, as it was easier to learn and faster to develop in for Unity. Photoshop was considered for graphic development as well but was less suited for video game asset development than other software. Developing a mobile game was also considered, as Unity has support for creating mobile games, though this idea was scrapped as it was not practical to make or play this game on a mobile device.

### Structure of Unity

A game created in Unity has a specific number of folders and files in it that all exist in 2 specific but different hierarchies.

The first one is known as the Project hierarchy, this contains all the assets and plugins ready for use and to be placed in the game’s scene, an example is seen in figure 1 below.

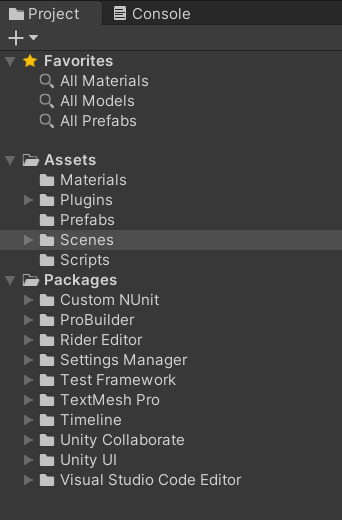


Figure 1 - Unity Project Hierarchy

There are 2 main folders here The Assets folder and the Packages folder. Within the Assets folder, there are the individual files such as scripts, game assets, and textures which we keep here and drag into the game scene as needed. It is also easy to manage to organise as you can put folders within folders to micromanage everything. E.g. In the Scripts folder, there will be the scripts created for this project. See Figure 2 below.

Graphical user interface, application

Description automatically generated

Figure 2 - Scripts in the Scripts folder

The second main folder in Unity is the Packages folder. This folder contains various extensions that the user may need for their game, such as the Probuilder extension. This package allows the user to create a level in a matter of minutes. See the layout in figure 3 below.

A picture containing graphical user interface

Description automatically generated

Figure 3 - Files in Probuilder Folder

The other hierarchy in Unity is in the scene, this hierarchy is everything that is currently in the game scene that appears on the screen. To add something to this scene, you simply drag the file you want to add from the assets folder into the scene hierarchy. Everything in here is from the previous hierarchy that contains the assets and packages. E.g. Figure 4 below.

Text

Description automatically generated

Figure 4 - Scene Hierarchy

Those are the main structures available in Unity. It makes it easier to track assets and the various components of the game.

### Design Patterns

Unity does not have a set software design pattern. However, this project aims to be consistent across scripts, with stylistic consistency with Unity’s coding conventions. In general, the coding style should be consistent with C#’s outlined coding conventions, only differing when Unity’s specific style also deviates (https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions).

### Application architecture

The game's architecture is heavily reliant on the Unity game engine. The code for the game is compiled when exported. Unity makes use of a technology called IL2CPP to convert C# code into C++ code to generate code that works on more platforms. This process can also improve game performance.

The application has an initial main menu that greats the user when they start the game, this menu gives them the opportunity to modify settings such as graphical settings, controls, and volume. On the main menu there is a button to begin playing the game. When the user clicks this, they will be brought into the game. The main menu acts as a control area for the player, it obeys the typical convention for video games that experienced gamers will expect. Similarly, there is a pause menu that can be brought up during the game, this menu has similar functionality to the main menu. It allows the player to modify their settings but also allows the player to leave the game.

All specific architecture aspects for this project are controlled by the Unity game engine. Such as input, calculations, game memory, physics, rendering, etc. An issue that arises with this for other games developed in Unity is that cheats can rapidly be developed, as existing video game cheating software already exists for other Unity games. This is less of a problem for this project though, as cheating only really causes damage to multiplayer games. This is one of the drawbacks of Unity though, it will be easier for a hacker to crack the game and release it for piracy.

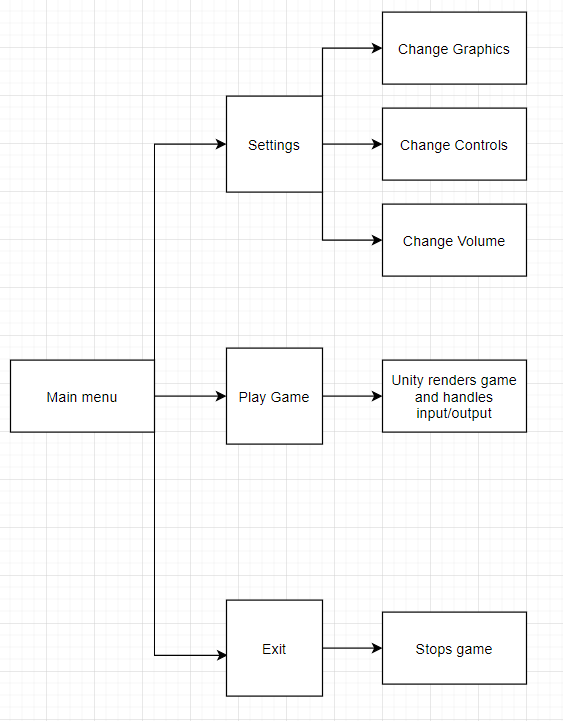


Figure - Block diagram for Inertia

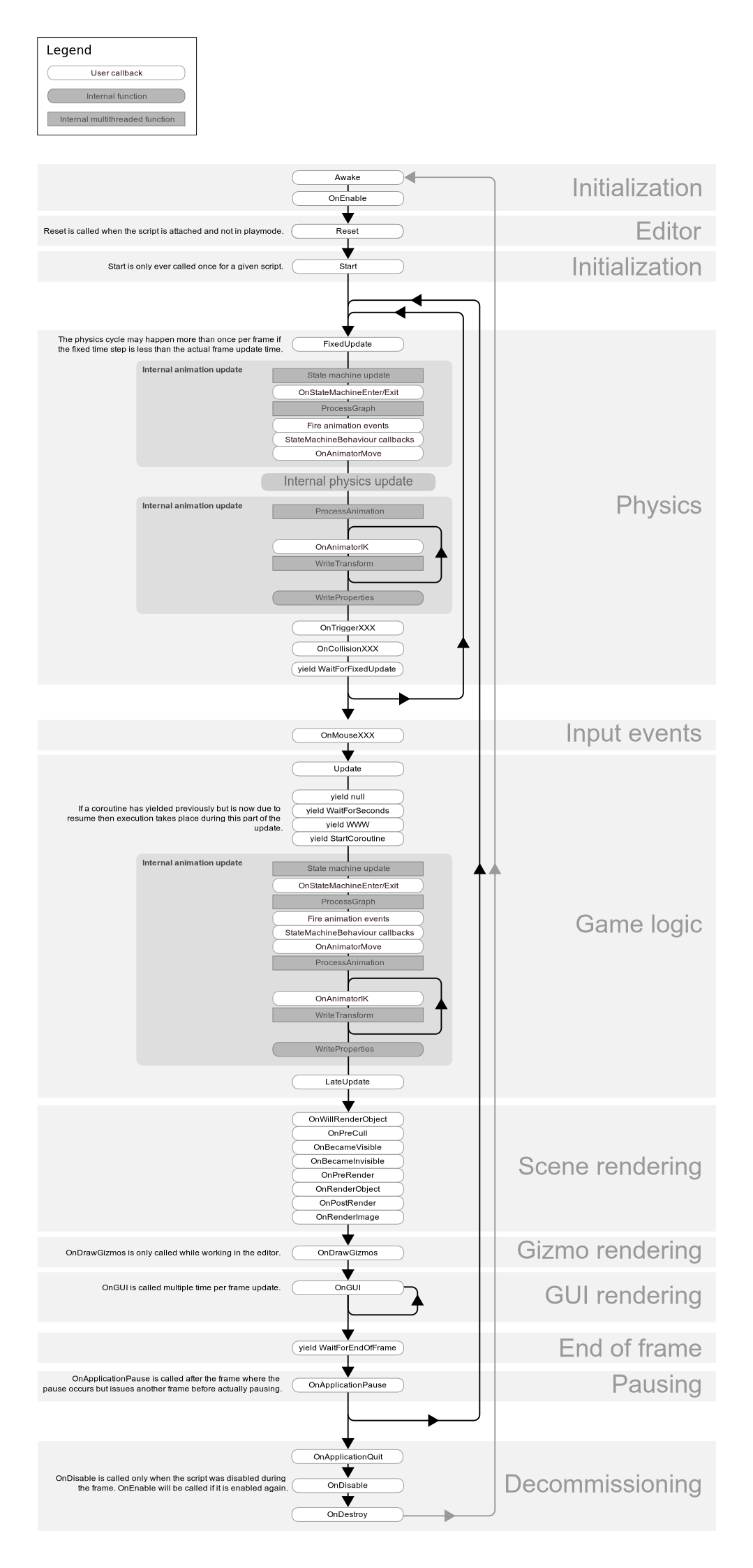


Figure 6 - Unity Mono application architecture, from https://docs.unity3d.com/uploads/Main/monobehaviour\_flowchart.svg

### Database design

Inertia is single-player and therefore does not have any database, so this section does not apply. All online interaction for the sale of the game would be handled by a third-party game launcher such as Steam.

## User interface design

The UI for this game is designed to be straightforward and clear to the player. The menus obey typical conventions for video games as previously mentioned, so it should be easy to adopt for experienced gamers. The interface is designed to be intuitive for people new to gaming as well, this is the reason plain language and structure is used in favour of technical language or a complicated menu structure.

### Wireframe

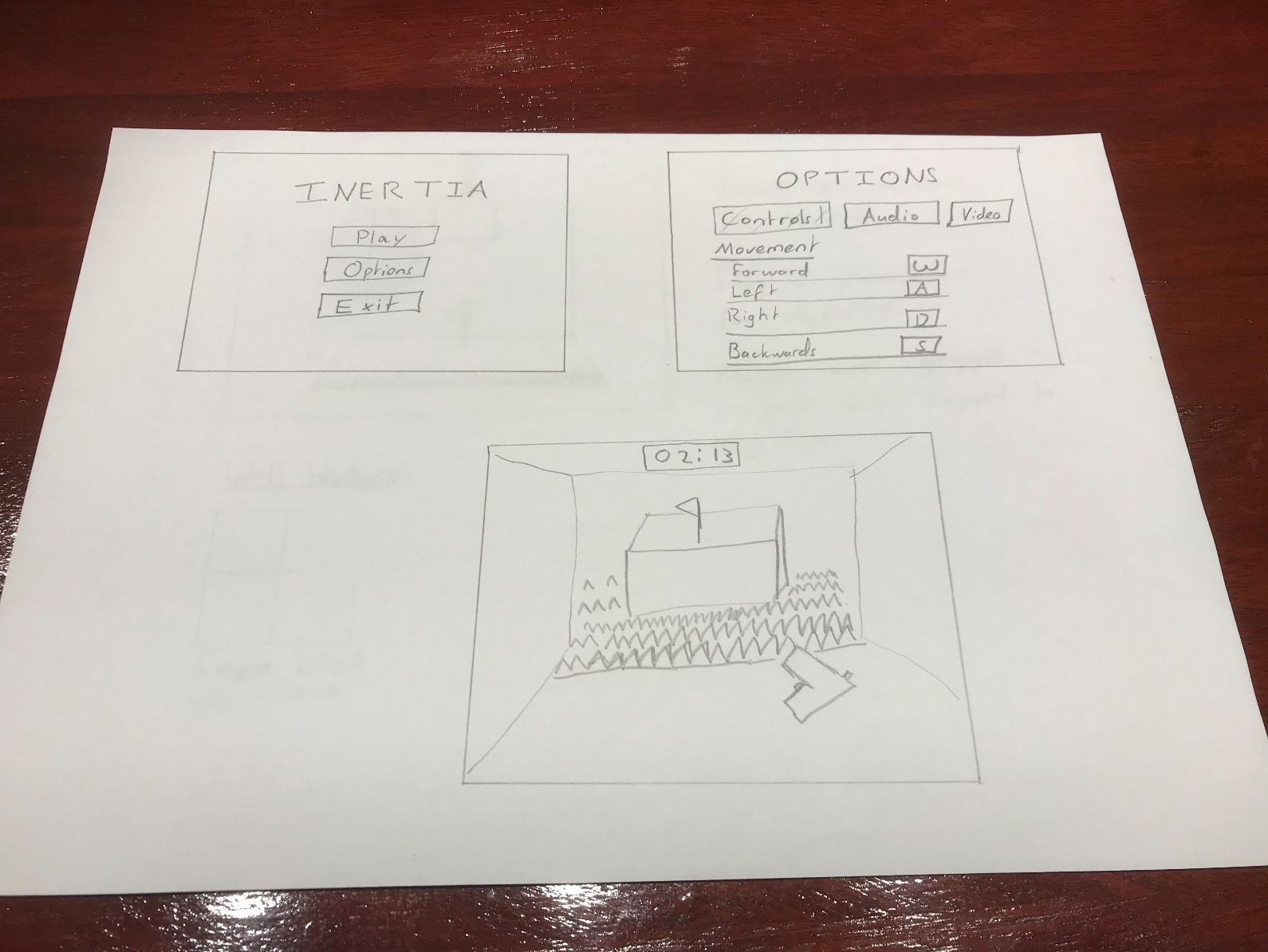


Figure 7 - Game sketch

A traditional wireframe is not applicable to this game. Instead, this is a representation of the main menu, the options menu, as well as the play state. The flag in the final frame represents the objective, the spikes represent hazards. The player has a grapple gun in hand to navigate over the hazards to get to the objective platform.

### User Flow Diagram

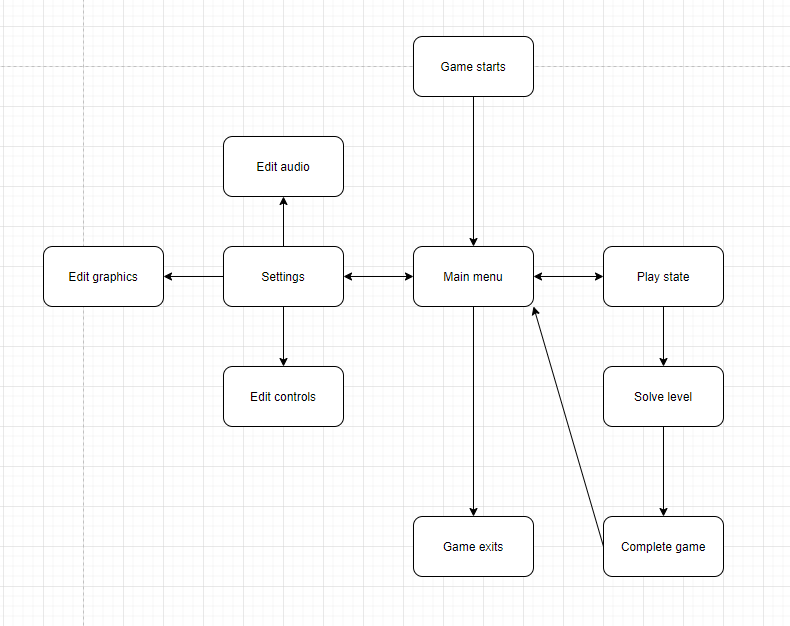


Figure 8 - User flow diagram for Inertia

This diagram represents the various actions a player can choose within the game, in the order that they can be chosen. Double-sided arrows represent direct navigation being possible between both states.

### Style guide

**Colour Palette**

Our game takes place in a lab so we will be using more muted and plain colours to match the aesthetics

Chart, treemap chart

Description automatically generated

Figure 9 - Colour Palette

**Typeface**

The font we will be mainly using in this game is Liberation Sans it is very simple and straight forward and fits the theme of our game which is set in a lab

Graphical user interface, text, application

Description automatically generated

Figure 10 Liberation Sans

**Menu’s**

Every menu in the game will look similar to this, main menu, options menu, pause menu, etc.



### Paper Prototype

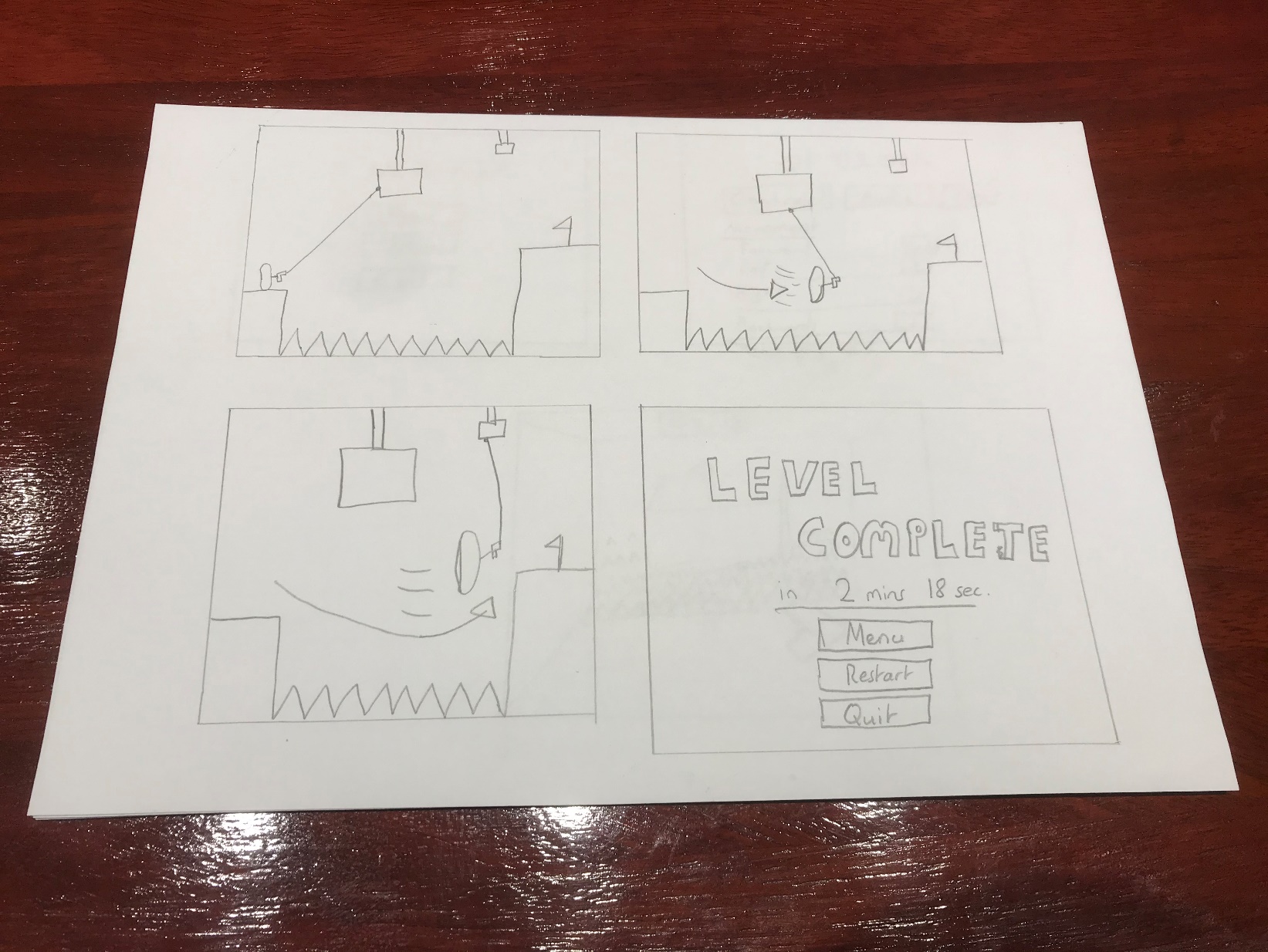


Figure 11 - A paper prototype for the game

This paper prototype shows a 2D representation of a 3D plane. The progression of this prototype is left to right. The player is represented by the capsule. The player uses a grapple gun to navigate from the starting platform to the goal platform (represented by the platform with a flag).

The grapple gun shoots a hook onto the cubes attached to the ceiling of the level, the player is pulled towards the cube, the player uses this momentum to swing over the hazard represented by the spikes that separate the starting platform from the goal platform. In the third panel, the player gains additional momentum by shooting the grapple gun again at another smaller cube attached to the ceiling.

In the fourth panel, the player has reached the goal platform, which completes the level. The time it took to complete the level is then displayed to the player.

A close up of text on a whiteboard

Description automatically generated

### Storyboard

Figure 12 - Storyboard

### Level Design

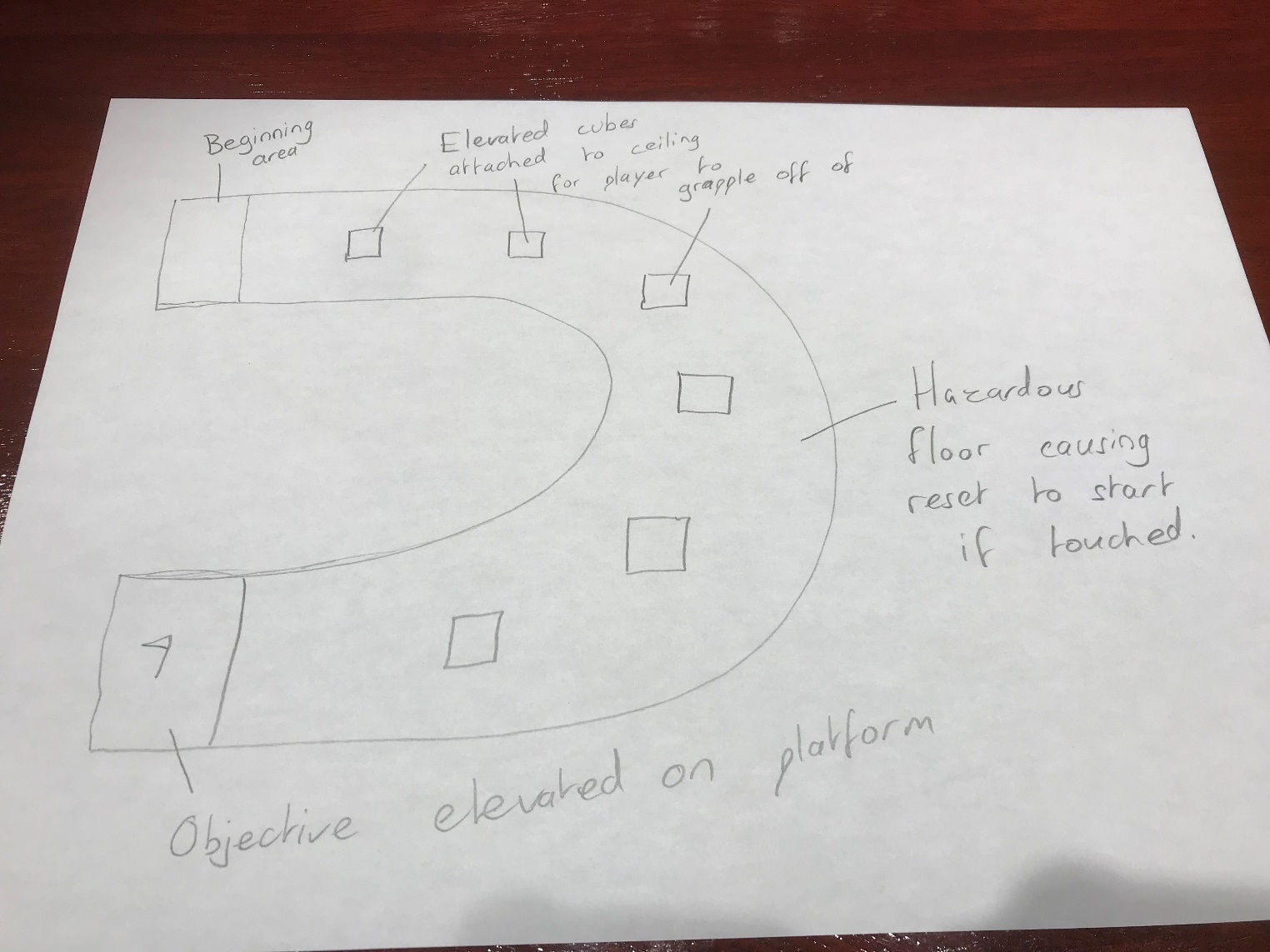


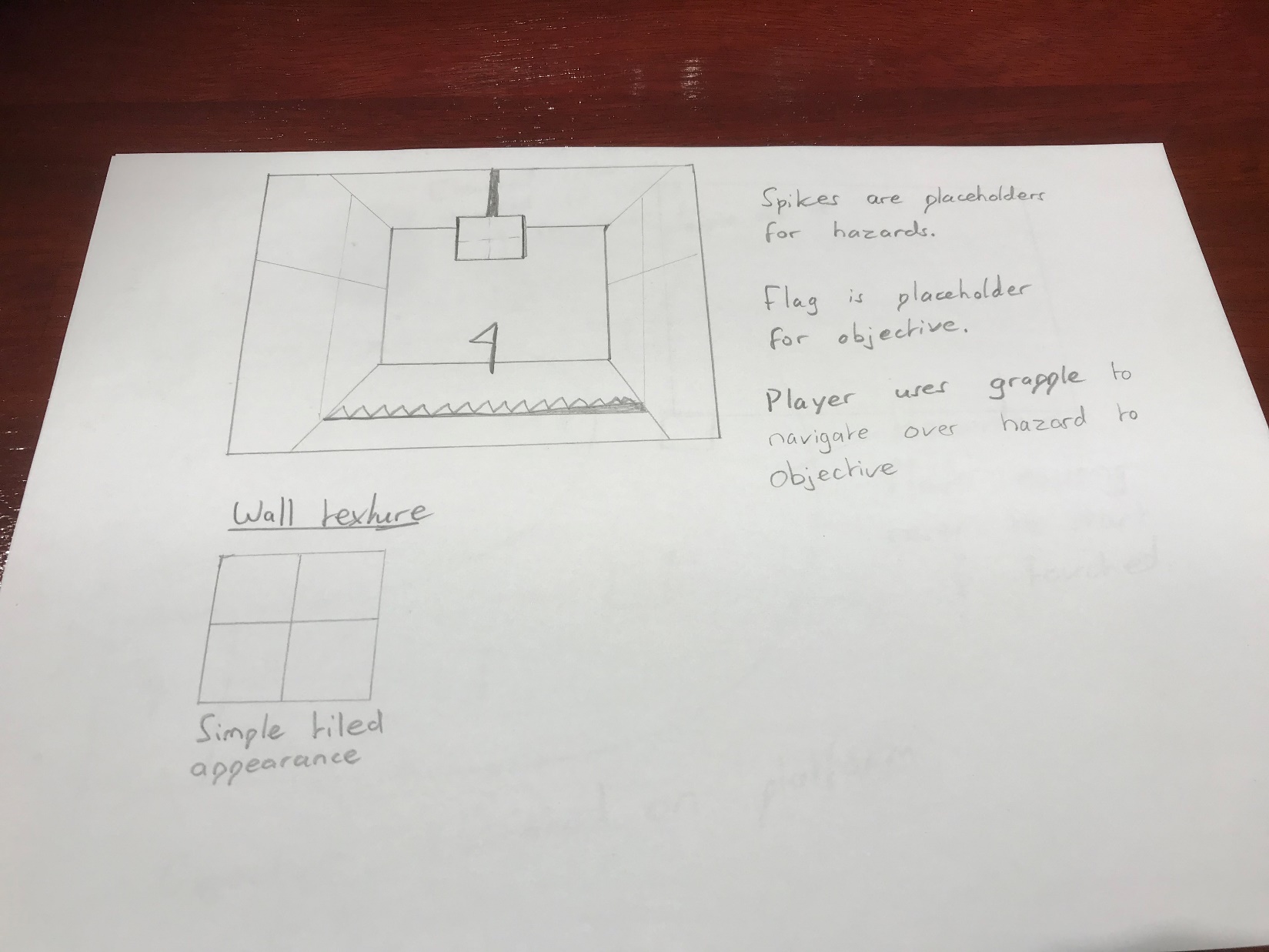
Figure 13 - An example of a level in Inertia

This is an example sketch in 2D drawn from a bird’s eye view of a level. The lines on the side represent walls, the walls curve around from the beginning platform to the goal platform. In this particular level, all of the floor is hazardous, if the player touches it, they will be reset to the start.

Their objective is to grapple from cube to cube (which are attached to the ceiling of the level) and build enough momentum to reach the elevated platform at the end. If they lose momentum, they will not have enough speed to reach the final platform and they will instead fall to the floor and be reset.

Upon reaching the final platform at the end, the player can move on to the next level. A timer shows them how long it took for them to complete the level.

### Environment



In the sketch at the top, there is a 3D representation of a level. The setting is indoors, with a focus on giving a lab feeling to the player, this is inspired by the highly successful Portal video game series games by VALVe (https://en.wikipedia.org/wiki/Portal\_(series)).

In the environment of this game, the texture for the surfaces is a plain tiled colour. As previously mentioned, this game takes place indoors, this makes it more obvious where the player is supposed to go as the level is linear in direction. The levels will feel purposely claustrophobic and restrictive, this again is inspired by the Portal series, where the player character is a test subject to scientific testing.

## Conclusion

This design chapter outlines the stylistic expectations for the video game. The User Interface and feel of the game are narrowed down. The style and aesthetics of the game are decided. The technologies involved in creating this game have been explored. The level design for the game has been developed upon, making it easier for the developers to create more levels with the same theme. The expectations of the game and the functionality required have been made clearer for the developers. Unity's structure, as well as Unity's internal architecture, was investigated, this should give the developers a closer understanding of the technologies involved in creating the game. User flow diagrams and sketches should also help with the development of the game as the desired user experience has been outlined.