Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

GAME NAME  
Cross Platform Development

Author: Thomas Dufresne

# Change Log

Updates made to the document should be described below.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date of change | Description |
| 0.0.0 | AIE | 31/08/2020 | Initial Template created |
| 0.1.0 | Thomas D | 8/09/2020 | Begin writing design document.  Add game overview, controls, game flow, items, and progress report for Friday 4th. |
| 0.2.0 | Thomas D | 10/09/2020 | Add mechanics, graphics, audio, AI, and interface. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Change Log 1](#_Toc50664961)

[1.0 Development Environment 3](#_Toc50664962)

[1.1 Game Engine 3](#_Toc50664963)

[1.2 Source Control 3](#_Toc50664964)

[1.3 Third-Party Libraries / assets 3](#_Toc50664965)

[2.0 Game Overview 3](#_Toc50664966)

[2.1 Genre 3](#_Toc50664967)

[2.2 Camera Perspective and Movement 3](#_Toc50664968)

[2.3 Platform 3](#_Toc50664969)

[2.4 Technical Goals 4](#_Toc50664970)

[2.5 Game Objects and Logic 4](#_Toc50664971)

[3.0 Controls 4](#_Toc50664972)

[3.1 Windows / Web 4](#_Toc50664973)

[3.2 Console / Xbox 4](#_Toc50664974)

[3.1 Android / Touch 4](#_Toc50664975)

[4.0 Mechanics 5](#_Toc50664976)

[4.1 Obstacles 5](#_Toc50664977)

[4.2 Items / Collectables 5](#_Toc50664978)

[5.0 Graphics 5](#_Toc50664979)

[6.0 Audio 5](#_Toc50664980)

[7.0 Artificial Intelligence 5](#_Toc50664981)

[8.0 Game Flow 6](#_Toc50664982)

[8.1 ‘Mission’ / ‘Level’ structure 6](#_Toc50664983)

[8.2 Objectives/Goal 6](#_Toc50664984)

[9.0 Levels 6](#_Toc50664985)

[10.0 Items 6](#_Toc50664986)

[11.0 Interface 6](#_Toc50664987)

[11.1 Menu 6](#_Toc50664988)

[11.2 UI/HUD 6](#_Toc50664989)

[12.0 Progress report and feedback Meeting Minutes 6](#_Toc50664990)

[Friday 4th September 6](#_Toc50664991)

[Wednesday 9th September 7](#_Toc50664992)

[Thursday 10th September 7](#_Toc50664993)

[Friday 11th September 7](#_Toc50664994)

# Development Environment

## Game Engine

Unity version 2019.3.6. This is used as it is the version currently installed on campus computers.

## Source Control

Link to GitHub repo: https://github.com/TheTastyGravy/Six-DoF-Game

## Third-Party Libraries / assets

*<State and explain the reason of use for any third-party libraries, assets from engine specific market places (Approval will be required) or packages.****All chosen third party libraries must be reviewed before adding to project by supervising teacher and licenses must be checked****>*

|  |  |  |
| --- | --- | --- |
| Asset Name  License | Url | Reason for use |
|  |  |  |
|  |  |  |
|  |  |  |

# Game Overview

## Genre

6DoF FPS Action game.

## Camera Perspective and Movement

Controlling a drone in first person, with 6 degrees of freedom, i.e. able to move vertically and roll the camera.

## Platform

PC, iOS, android, Xbox, web. Unity makes deploying to multiple platforms very easy, with everything just working with very few exceptions. Through development, any such components that are not cross platform will be avoided.

## Technical Goals

* Options menu using PlayerPrefs
* Saving and loading

## Game Objects and Logic

* The **player** has 2 ‘currencies’: health and ammo. Ammo is used to fire their weapon, and health is reduced upon taking damage, and when it reaches 0, the player dies, reaching a game over state. Both of these currencies are increased by collecting pickups, described later. The player takes user input to be able to move, rotate, and fire their weapon. The weapon consists of a single script with a Shoot function, called by the player. The script creates a bullet and applies a force to it in the forward relative to the player. Upon entering a trigger with the tag ‘Pickup’, it will call a function from its script, then destroy the game object.
* **Pickups** are simple objects that only have a trigger and a pickup script. The player will call a function in the script upon entering its trigger. This function will give the player ammo and health equal to its preset values.
* **Bullets** are another simple object. They have a collider and a bullet script, which will call a deal damage function on any object it collides with, before destroying itself.
* **Shields** contain no logic, but are instead used by bullets. They only have a trigger, but when a bullet enters a shields trigger, it will destroy itself. This results in an object that players and enemies can pass through, but not bullets.

# Controls

A 6DoF game naturally requires 6 degrees of input. This is trivial for mouse and keyboard, and doable on a controller. Touch controls for mobile, however, is more difficult. Input is limited to the screen space, so the player will only be using their thumbs to play the game, reducing the possible degrees of input to 4. Because of this, the mobile version of the game will not feature vertical movement or rolling the camera.

## 3.1 Windows / Web

Mouse and keyboard using WASD and space/shift for movement, mouse and Q/E for camera control. Esc to pause, left click to fire, F to interact.

## 3.2 Console / Xbox

Controller using left stick and bumpers (vertical) for movement, right stick and D-pad (roll) for camera control. Right trigger to fire, start to pause, A to interact.

## 3.1 Android / Touch

Custom touch joysticks and buttons are necessary for mobile input. Due to the lack of controls available, the mobile version will lack vertical movement and the ability to roll the camera, but the game will remain playable.

Left joystick for movement, right joystick for camera control. Left button to fire, upper-right UI button to pause, right button to interact.

# Mechanics

*A list of intended core game mechanics. I.e., what the player can do and how they achieve this, and what this will trigger in the game. For example, shooting enemies is a core mechanic in an FPS.*

* **Shooting**

When the player fires, a bullet is created in front of them moving in the direction they’re facing at a speed slightly faster than they can move, preventing them from being hit by their own bullet. Bullets cannot collide with each other, rather ignoring such collisions. Bullets will, however, collide with shields when hitting them from the outside, but not from within, allowing the player a safe area to fire from.

Upon hitting something, the bullet will attempt to deal damage to the object it has hit, then instantiate an explosion prefab at the position it hit. The explosion has no colliders, and destroys itself shortly after.

When the player shoots, they will also lose ammo. When they run out of ammo, they are unable to shoot, and must collect an ammo pickup to start firing again.

* **Interaction**

When within range of an intractable object, the player can press the interact button to perform whatever action the object does, e.g., a save point will save the game.

## Obstacles

The primary obstacles the player will come across are enemy turrets that will fire at the player, which will be scattered throughout the level, and the maze-like geometry of the level itself, being likely to disorientate players.

## Items / Collectables

There are two pickups: health and ammo. These will be scattered throughout the level and may be dropped by enemies upon death. They are collected by entering their trigger radius

# Graphics

The game will feature very minimalistic graphics, using generic shapes rather than 3D models. The reason for this is it’s easier.

# Audio

To match the minimalistic nature of the game, no audio will be included in the game.

# Artificial Intelligence

Due to the nature of the game, pathfinding is not possible, as in order to navigate the 3D geometry, a volumetric navigation grid would be necessary, and Unity only provides 2D navigation meshes. Consequently, the only enemies the game will feature will be stationary turrets. These will cast a ray towards the player, and if it hits, it will start shooting at them, otherwise it remains inactive.

# Game Flow

## ‘Mission’ / ‘Level’ structure

Different levels will be in separate scenes, with the appropriate scene loaded upon entering the level. Data will be saved using the [binary formatter](https://answers.unity.com/questions/1320236/what-is-binaryformatter-and-how-to-use-it-and-how.html) to load a [serialized](https://docs.unity3d.com/Manual/script-Serialization.html) class from file, containing the player’s health, ammo, checkpoint and level. The level’s scene will be loaded, and the player will be initialized with the health and ammo at the designated checkpoint. Upon reaching the end, the checkpoint will be reset and the level index increased, with the player’s health and ammo updated.

## Objectives/Goal

The player simply moves through the level, taking out enemies, until they reach the exit.

# Levels

All levels will feature the same mechanics, with the level itself being made using unity’s pro builder package.

# Items

The only items will be pickups for health and ammo, which will be placed throughout the level and possibly dropped by enemies.

# Interface

All UI elements are properly anchored with text elements set to best fit, resulting in everything being scalable with resolution. Because of this, the same UI will work on mobile, PC and TVs without changes.

## Menu

Thanks to the event system, menu systems are easy to implement. Touches are automatically detected as click events for UI elements, and menus can be navigated using the horizontal and vertical input axes for controllers. Because of this, no extra code has to be written to make menus accessible on other platforms.

In total, there are three menus: the main menu, pause menu, and options menu. The main menu simply goes to the game and the options menu, with a third option to exit the game. The pause menu allows the player to resume (unpause), respawn (last save state), restart the level, and return to the main menu. Upon winning a level, respawn is replaced by next level, and in any game over scenario, the resume button is removed.

The options menu allows settings such as sensitivity and invert Y axis to be changed via a slider or input field. It will also display the games controls, with each control scheme able to be selected by pressing the appropriately labeled button.

**WIREFRAME**

## UI/HUD

The HUD only needs to provide basic information: health, ammo, and a crosshair. The former two are displayed in an info panel in the upper left corner, while the crosshair is naturally in the center of the screen. In addition to these, a textbox exists at the bottom of the screen that is used to display when the game is saving, and a button in the top right corner to open the pause menu when on mobile.

**MOCK UP**

# Progress report and feedback Meeting Minutes

## Friday 4th September

Describe state of project

* Prototype with most of its core features working.

Feedback from teacher and peers:

* A good base without much substance.

Action Items:

* Work out the details of what the game will be.

## Wednesday 9th September

Describe state of project

* Prototype finished

Feedback from teacher and peers:

* Describe

Action Items:

* Move on to implementing features in final game

## Thursday 10th September

Describe state of project

* Starting on final game

Feedback from teacher and peers:

* Describe

Action Items:

* Finish project

## Friday 11th September

Describe what has been done since last time

* Thing
* Thing

Feedback from teacher and peers:

* Describe
* Describe
* Describe

Action Items:

* Describe
* Describe
* Describe