CROSS-PLATFORM DEVELOPMENT TECHNICAL DESIGN DOCUMENT

ASSESSMENT DESCRIPTION

For this assessment you are required to create a game using either Unity 3D or Unreal Engine 4, either individually or as part of a team, and build it for multiple platforms. Which platforms you support will depend on your available devices, but your teacher will advise you as to which platforms would be best supported. At a minimum, your game must perform on:

- At least two different web browsers, and
- At least two different digital devices

The game does not have to be a detailed game project but must at least make use of loaded assets and a basic Graphical User Interface (GUI) demonstrating industry best practice. You must also demonstrate the use of your target platform's specific input devices. For example, on a mobile device you would need to demonstrate *touch-screen input* whereas on a PC the project would utilize *controller input or keyboard.*

The project involves several stages of development, outlined below.

1.0 Development Environment

Below is the Schedule, as well as a provided link to the Project Management [link]

Week 1
> Research Game (*concepts, styles, mechanics, designs*)
> Create TDD & Research document
> Find assets and scripts
> Install Unity with platform builds
> Set up version control
> Install all relevant software (*drivers, JDK, bluestacks*)

Week 2
> Implement basics (*controls/movement, objects, assets*)
> Create basic UI (*Pause menu, Main menu*)
> World design & generation (*Procedural ground generation, Obstacles*)

Week 3
> Apply assets and animations to objects
> Collectibles & Scoring
> Redesign all UI (*Animations, transitions, designs, functionality*)
> Implement mobile compatibility (*Touch input & screen sizing*)

Week 4
> Implement Audio
> Bug Fixes, redesigns, apply feedback, touch ups
> Cross platform testing and debugging
> Review documentation and documents

Week 5
> Review and finalise documents

> Evaluate against client brief
> Create platform builds
> Compile documentation and evidence

1.1 Game Engine
(*Proprietary/Unreal/Unity and version*)
Unity Engine 2020.3 / Unity Engine 2021.1

1.2 IDE
Visual Studio 2019 w/ Visual Code

1.3 Source Control Procedures
Github w/ Github Desktop

1.4 Third Party Libraries
N/A

1.5 Other Software
(*2d art assets, audio, 3d modelling etc*)
N/A

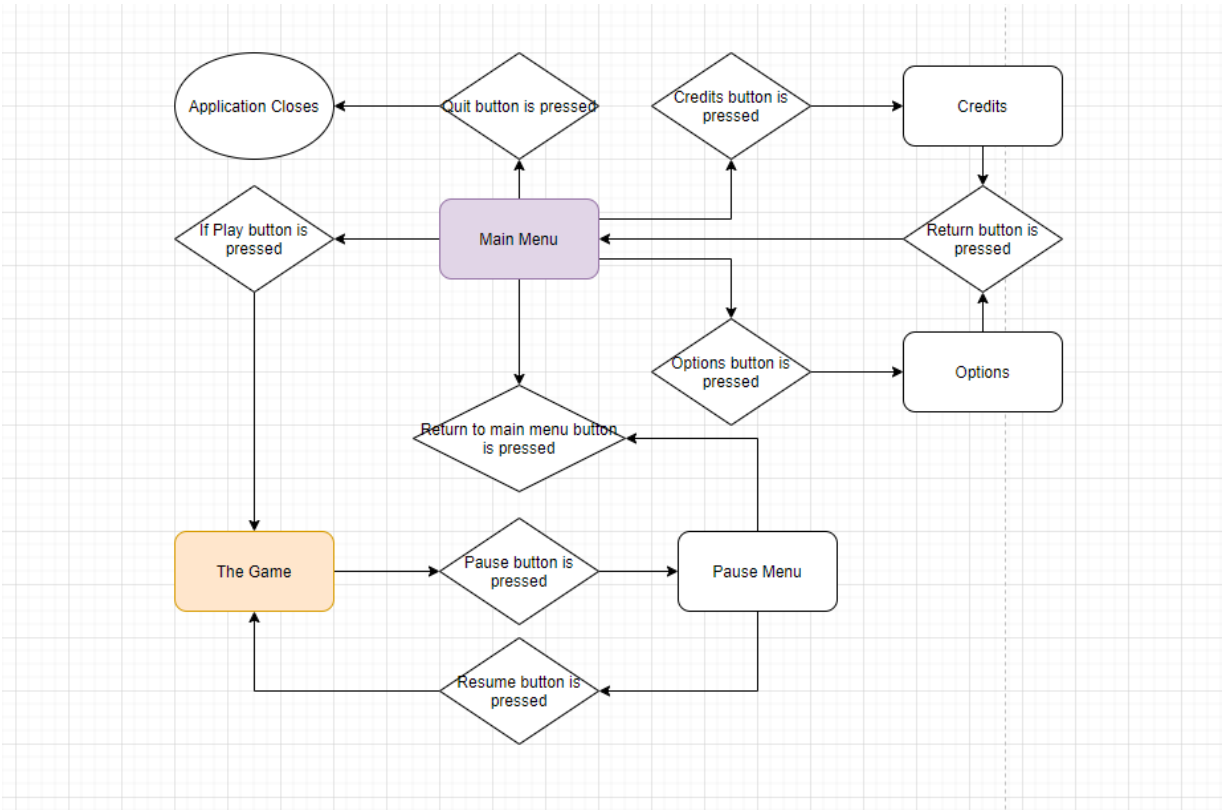
2.0 Game Overview

2.1 Technical Goals
(*3d graphics, 60fps, Challenging AI etc.*)
3D graphics with minimal animation, maintain a steady fps.

2.2 Game Objects and Logic
(*A list of logical elements in the game, i.e. door, button, pistol, ammo, light, bullet, wall, character etc. and description of their behaviour and purpose.*)
GUI Buttons and sliders will be found within the GUI main, options, and pause menus.
World objects such as the directional light, audio and scoring managers, effects manager.
Scene objects such as the player, obstacles, ground, and collectibles.

2.3 Game Flow
(*description of what the player can do (actions) from the start menu to playing the game, through to hitting quit. Include how to win, how to lose, how the player is moved, and what programmer things might need to be considered*)
From the Main Menu, the player will be able to access the options menu to control sound, or start the game. From here the game will start.
Within the game, the player will be able to use touch screen movements (mobile) or arrow keys (PC) to control the left and right movements of the player. They will have to maneuver obstacles in order to travel as far as possible (directly proportional to their score).
The game ends/resets when the player collides with an object, sending them to the beginning. There is no end to the game.

<Flow Diagram depicting the flow of the interactable GUI>



3.0 Mechanics

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

The core mechanics in the game are simple: The player object will move at a constant force to which the player will have to control the left and right movements (and clamped forward/backward movement) to avoid obstacles. Player power ups will be able to be collected to provide the player with boosts (such as speed and invulnerability)

4.0 Graphics

(*Describe graphics features here. I.e., is your game top-down 2D?*)

3D Low-poly, linear camera angle.

5.0 Artificial Intelligence

(*Describe how AI works, i.e. state machine, fuzzy logic, GOAP. Describe the various behaviours and how they change behaviour, how do the 'creatures' in the game evaluate the world. Include diagrams/flowcharts showing decision making processes*)

N/A The game will not have agents/objects requiring AI.

6.0 Physics

(*What engine are you using, what features from it (spring? Colliders?) How will physics be handled for objects? (box or sphere collider for objects, capsule for player) need to record specific locations for any reason? Potential slowdowns and how to mitigate.*)

Unity Engine provides physics features such as rigidbodies, 3D colliders and mesh colliders. Rigid All of which shall be used to allow the player to interact easily with the environment. The low-poly art-style will reduce complexity for colliders to allow simpler and easier collisions.

7.0 Items

(*List of items you can pick up that can affect the player, and what they will affect, like 'picking up the hammer (refer collisions above) adds 5 to the players attack attribute'. Include details on how items influence gameplay or AI logic.*)

ITEM TYPE	NAME	DESCRIPTION
Power Up	***Boost***	Provides invulnerability and boosts the player forward at an insane speed for a brief time.
Power Up	***Invulnerability***	The player becomes invincible for a short amount of time.
Power Up	***Jump***	Give the player the ability to jump once.
PowerUp	***Resurrection***	Provides the player with an additional life, where upon death they will be given invincibility for 3 seconds and continue playing.

8.0 Game Flow

8.1 Level Structure

(***Are all levels stored in memory? what data is saved across levels, are levels loaded synchronously to prevent pauses?***)

I don't intend on adding levels to begin with, however if time permits, I would like to add a check-point level system, if the player reaches a significant distance, they will be provided the opportunity to collect some powerups to prepare them for the upcoming level/area.

8.2 Objectives

(***What does the player try to accomplish on each level/mission? How is the player's progress evaluated?***)

The player's progress is evaluated through their score, where their goal is to try to surpass their previous score and strive to reach further and further.

9.0 Levels

(*If any of the Levels require specific behaviours, describe those here*)

N/A

10.0 Interface

10.1 Menu

(***What are the menu options and what do they do?***)

The main menu will have three prompts: Play, Options, and Exit.
Options menu: Music Slider, SFX Slider. (*Accessible from the pause menu as well.*)

Pause menu: Return, Restart, Quit.

10.2 Camera

(***Describe the camera, how it moves, perspective/orthographic, can it switch? How? Does it need to render-to-texture? does it prevent itself going through walls, use flowcharts to document behaviour***)

The camera will be following the player from a fixed distance behind them, matching the player's speed.

10.3 Controls

(***Keyboard, tablet touch/swipe/tilt, joystick, mouse etc. record double taps, multi touch, use mouse smoothing/ scale mouse for aiming etc.***)

For any touch-screen compatible device (*primarily handheld, in this case, for android phones*), the screen will register when the left and right sides have been tapped in order to control the player movement. Similarly, on PC (*Application and webGL*), the user will be able to control the player vid arrow keys.

Below is a pseudocode example of player controller movement:

```
| Add force to the player's Z-Axis
|
| IF (right trigger is pressed OR right touch input button is pressed)
| | Apply positive force on the players X-Axis
|
| IF (left trigger is pressed OR left touch input button is pressed)
| | Apply negative force on the players X-Axis
|
| IF the player's coordinates are Out Of Bounds OR player collides with an obstacle
| | Enable the players death animation
| | Enable the players death sound
| | Destroy the player object
| | Restart the game/scene
```

11.0 Asset List

(*List all files needed, along with known attributes*)

Many of the files to be imported seem to be .FBX files, which is the recommended file format for Unity and is supported internally. [Unity-Manual: Model File Format]

CraftPix.Net

https://craftpix.net/

I have taken 3 Total asset packs from CraftPix in order to decorate my games world space. (DesertPlants, Stones, and TropicalPalmTrees)

Unity Asset Store

Low Poly Bird: Ultimate Pack

https://assetstore.unity.com/packages/3d/characters/animals/birds/low-poly-bird-ultimate-pack-200559

I have purchased this pack in order to provide some well-made and fully rigged models.

Low Poly Environment - Nature Pack

---

https://assetstore.unity.com/packages/3d/environments/lowpoly-environment-nature-pack-free-187052
Several environment items have been used to create obstacles and the main menu.

Low Poly Flowers
https://assetstore.unity.com/packages/3d/vegetation/plants/lowpoly-flowers-47083
One asset was used from this pack, within my project.

12.0 Technical Risks

(*if you want your game to be a 1000 player pvp battle royale with 4k 120fps graphics, you need to say if this is doable and how you intend to do it*)

The only technical risk I can think of is ensuring performance is consistent for each platform and that graphics are compatible through different specs.

13.0 Revision History

(*As you revise the document, list what was changed and when it was changed*)

VERSION	DESCRIPTION
0.1	Initial Document
0.2	Edited to accommodate for newly added assets
0.3	Edited to add final missed fields
0.4	Edited to finalise and touch up for submission