SDD Major Project

Problem Statement:

My project will offer a short gaming experience for a user to take a break after studying. As there are currently a lack of games which are enjoyable and easy to use. Some games take too long to get going (loading of maps, story development/ cutscenes) and when you are trying to take small gaps in-between studying it can be hard to find something to play. My game will solve that problem by giving the players a simple game concept, which has varying content through either random spawning location, map creation etc, to make a player experience new and refreshing experiences with each playthrough. The game will be short to fit in as a gap for students who want to take a short break from studying. Each game would be approximately three to four minutes in length in order to distract the user for a smaller amount of time. My concept will allow users to play a short but enjoyable game with ever-changing content as well as different play styles to encourage irregular gameplay. This problem was chose for my project as I wanted to make a enjoyable short gaming experience, and also because I enjoy making and playing games.

Initial Ideas:

Windows Game (Made in Unity): Short arcade style top-down shooter 2D, which has progressively spawning enemies and obstacles. It will also involve a portal like mechanic as seen in the Valve Portal series which allows the player to enjoy a fun arcade style game with a refreshing concept.

Features/ Functionality:

* Arcade style shooter
* 2D, made in Unity and C# code
* Portal mechanics
* Automatically enemy spawning mechanics
* Scoring system based on coins collected as well as enemies destroyed (allows a trade-off from playing to passively)
* Simple pixel art 2d graphics
* Basic sound FX and music
* Multiple enemy types

Windows Game (Made in Unity): A 2D top down game, a singular dodgeball player has to clear stages of progressively harder levels of enemy players. Replication of the ideas of dodgeball but in a basic minimalistic level with only black and white colours.

Features/ Functionality:

* Pixel art graphics
* 2D, made in Unity and C# code
* Shooting mechanics
* Hand crafted levels, with random enemy spawns per room or auto generated
* Randomly spawning obstacles
* Scoring system based on time, how enemies were painted and hidden objects
* Basic sound FX and music
* Fluid movement system
* Enemy features:
  + Player detection based on user’s line of sight
  + Last player detection
  + Patrols

Studying timer/ planner app (JavaScript + Electron): A study planner which takes your desired amount of time to study for a subject and can be broken down into different sections for example summaries, definitions and past papers/ questions. The app will be able to give an audio cue on when you should try something else instead of working in the same area for a long duration of time. The app will also be able to keep track of what needs to be completed for each subject.

Features/ Functionality:

* Simple and interactive UI
* Ability to store subjects and homework material in a database
* Customisable timer and audio cues
* Customisable themes
* Analog and digital clock
* Ability to set up study breaks with time:
  + For example, 2hs broken into 40min revision + 20min break +60min past papers

Feasibility Study:

Windows Game (Arcade T.D.S):

* Unity Software is free to develop with the Personal Edition
* Previous experience with Unity so it will be easier to start the stages of development as well as prior knowledge with the C# programming language.
* Art will be created in an application called Aesprite and Tilesetter (both paid applications but bought previously.)
* Game Idea is simple making it easy to expand upon (new level, different characters etc.)
* The games core mechanics are simple and have been done before making it easy for new players of the application to pick up and enjoy the content available.
* Will be playable on a PC with i5 7th gen recommended processor, 8GB RAM and does not require a dedicated graphics card

Windows Game (Dodgeball):

* Unity Software is free to develop with the Personal Edition
* Previous experience with Unity so it will be easier to start the stages of development as well as prior knowledge with the C# programming language.
* Art will be created in an application called Aesprite and Tilesetter
* Game Idea can be expanded upon with new map ideas, enemy types, weapon types, etc.
* Game idea is simple to pick up
* Will be playable on a PC with i5 7th gen recommended processor, 8GB RAM and does not require a high-end graphics card.

Launching of either game (if):

* Overall development cost of application is free (present)
* Using the personal license of Unity but if annual revenues where to exceed 100k the licence would need to upgrade to either plus ($25 per month annual) or pro ($125 per month) also in prices in USD
* The application could also be launched on Steam for a $100 fee which can possibly attract more attention for the game.
  + Launching on steam will also require knowledge of the steamworks API.
  + The Steamworks API people say it is hard to use and understand in the beginning so scheduling would have to possibly take into account the learning of the steamworks API.
  + The steamworks API also creates new work that can be done in relation to the application to make it seem more polished and to possibly get more users of the software.
    - In game achievements to encourage players to reach specific goals
    - Store page and screenshots to show players what they would be playing/ purchasing.
* The application can be launched for free on Itch.io or other indie game stores.
* Application can be launched on Steam but can also generate steam key, with steamworks and sell to external sources such as Gog, and humble bundle.

Studying timer/ planner app (JavaScript + Electron):

* Development has no cost associated with the project.
* Have very little knowledge of JavaScript and Electron so part of the schedule will be spent developing knowledge of the tools.
* The app would be mostly developed with myself as the end user as current products on the marketplace don't do what I want, other could also use the program but there is no information whether someone else would use such and application.
* Application if launched publicly would cost $20 on the Microsoft store for an individual license which is cheaper than launching a game on steam or the google play store.
* The application can fit easily with high school student habits
* Can be played on low-end systems with a PC with an i3 6th gen or later processor recommended, and at least 4GB RAM.

Idea to be developed:

The idea I have chosen to develop is the Unity dodgeball game. I will be developing this project for the following reasons:

* Already knowledgeable with C# language which is beneficial
* Have worked with Unity 2D and 3D with previous projects.
* There are lots of questions and answers on Unity Forums which helps to solve hard problems.
* Lots of tutorials available on YouTube and other websites
* Unity has well written documentation and many users making forums easier to find answers.
* There are many ways to expand upon the game such as:
  + New maps, characters, etc
  + Music and SFX
  + Stealth mechanics
  + Varying assets and obstacles
  + New AI mechanics
  + Paint splatters
  + Visual effects

Consideration Of Issues:

Ergonomics:

By considering ergonomics I can develop a game which is friendly for consumers to use and is intuitive which allows for users to have a good experience with the application.

* User friendly - easy to pick up and including tutorial on how to play the game
* Quick load times
* Friendly colours which won't be hard to look at
* Consistency of button placements and controls

Inclusivity:

Cultural:

* All assets in this project will have no existing resemblance of anything real, if by some chance it is, the any negative intentions are not meant. The language of the game will be in English because it is the only language I know, and the target audience is English-speaking users.

Economic:

* The game if was to be released would be released for cheap or for free, allowing for many players.

Gender:

* My game will not discriminate against any gender intentionally.

Disabled:

* Due to the project deadline there will unlikely be any disability options for the application. This would include colour-blind settings, high contrast buttons and larger text.

Privacy/ security

* No personal data will be stored within the application. If the game is further developed and released only high scores and achievements would be stored.

Intellectual property and software license agreements.

In the development of my project I will ensure all my code is to be fair use or credited if not my own. This is to ensure that there are no possible legal issues that can occur with releasing my application in the future.

Programming Language Selection:

I will be using the C# language to be used for creating my game. This is because it is the best documented programming language to work alongside unity and is easier to find answers on forums rather than JavaScript. I chose this rather than using C++ and GML because I have lacking experience using C++ in Unreal Engine and GML is only exclusive on the game maker studio suite of applications which require expensive licensing for developing which is not ideal for a project. The other main language I'm experienced using is Python but it is not an ideal choice as I will have to use a library named *pygame* and it will not be able to work as well with my project. In addition to this, there is less documentation on it than tools such as Game Maker, Godot and Unity.

Initial Design Specifications:

Core mechanics:

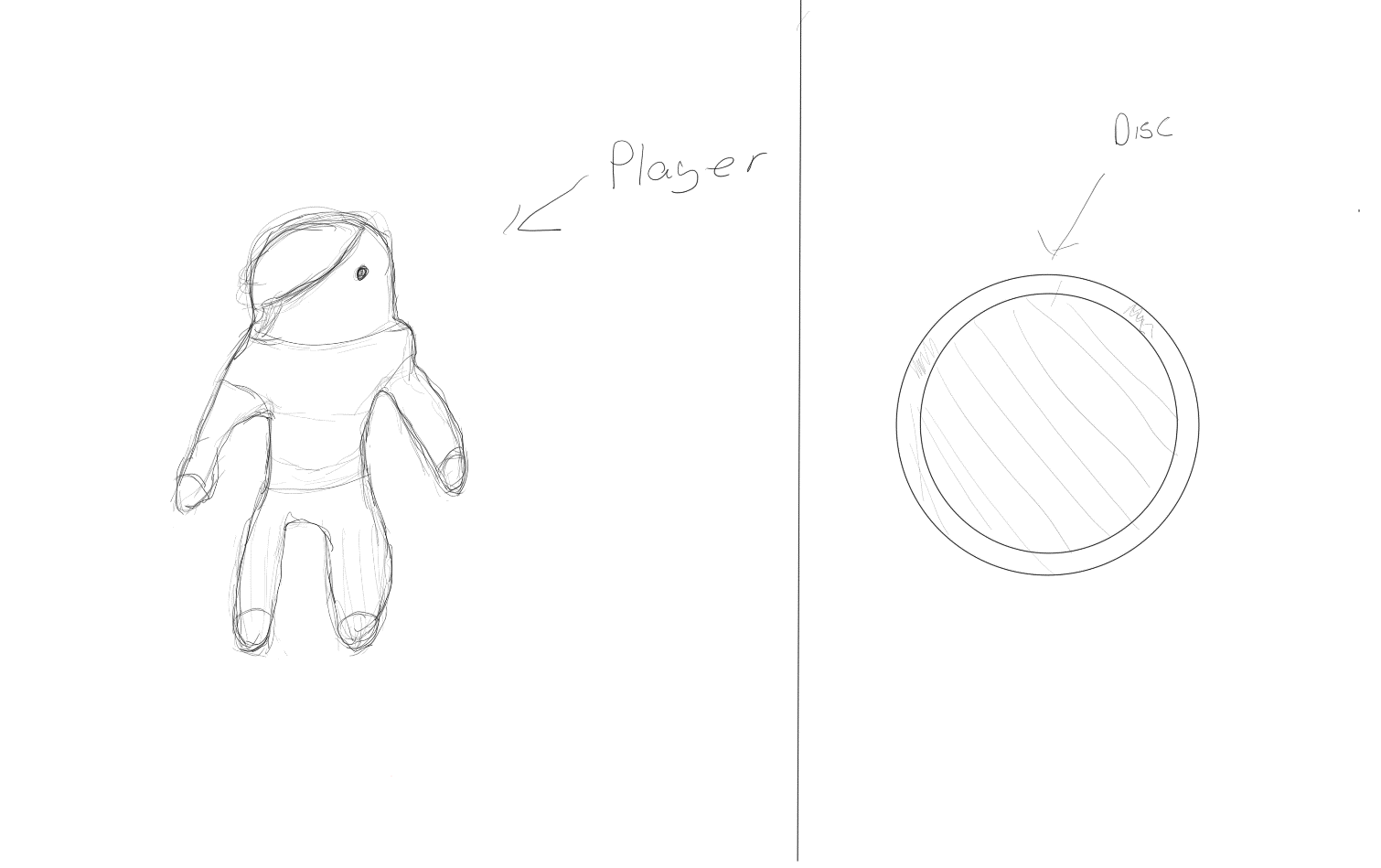
* Movement
  + Top Down movement
  + Dodge mechanic
  + Sneak mechanic
  + Free camera
* Enemy
  + Player detection
  + Shooting
  + Patrol
* Throwing
  + Ball recovery
  + Bouncing
* Map
  + Enemy patrols
  + Randomly spawning goal/ flag
  + Multiple rooms
  + At least 3 levels
* Basic UI
  + Timer
  + Score (at end of the game)
  + Pause menu
  + Start menu
* Player
  + Animation controller
  + Weapon sound effects
* Basic lighting system
* Sharers
* Animation controller for player

Extras:

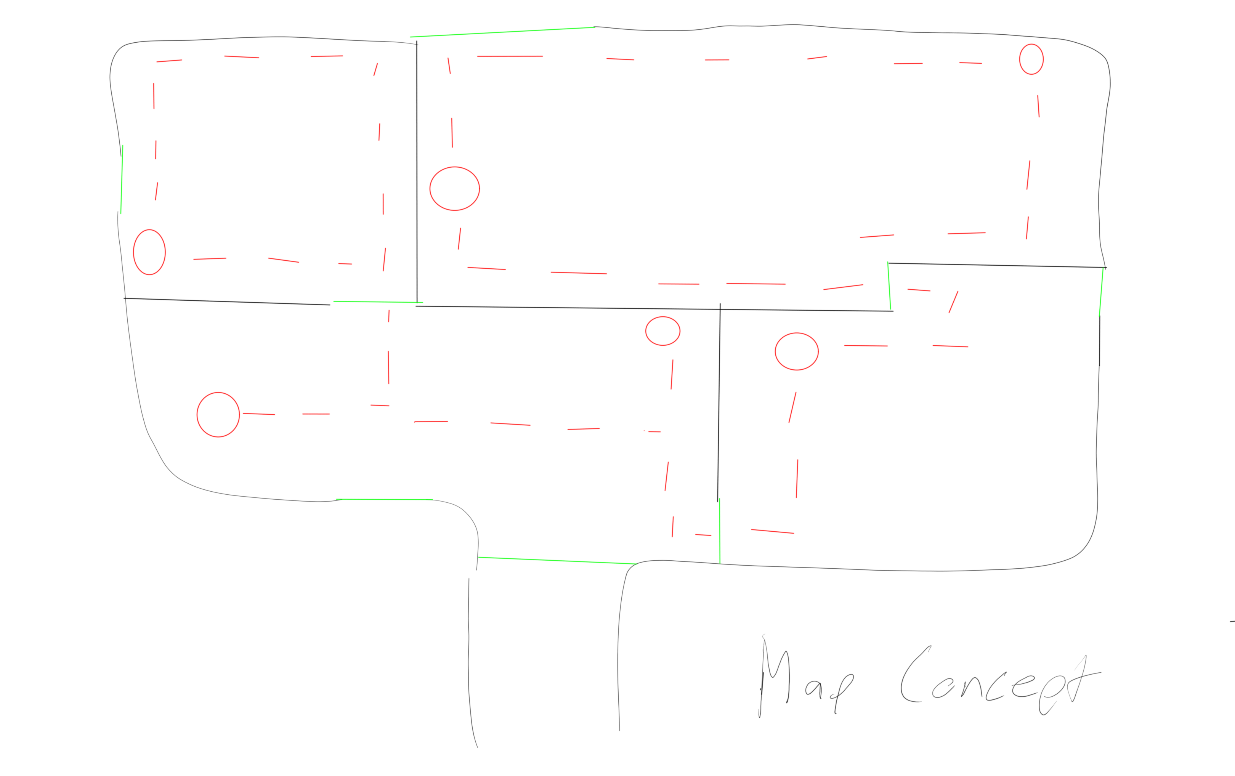
* Movement
  + Multi directional dodge
* Enemy
  + Player last location before line of sight was broken
  + Work's with other enemy's in order to stop player routes
* Map
  + Auto generation
* Main game music
* Visual effects
  + Particle FX
* Stealth mechanics and rewards
* Restart game mechanic
* Multiple weapons and characters
* Weather effects
* Varying levels of throwing power
* Main Menu, Level Select, Settings
* Difficulties - Increase enemy's fov, speed, etc
* 8 different animations for player movement
* Animation controller for enemy
* Time rewinding
* Charged throws

Initial Design Sketches

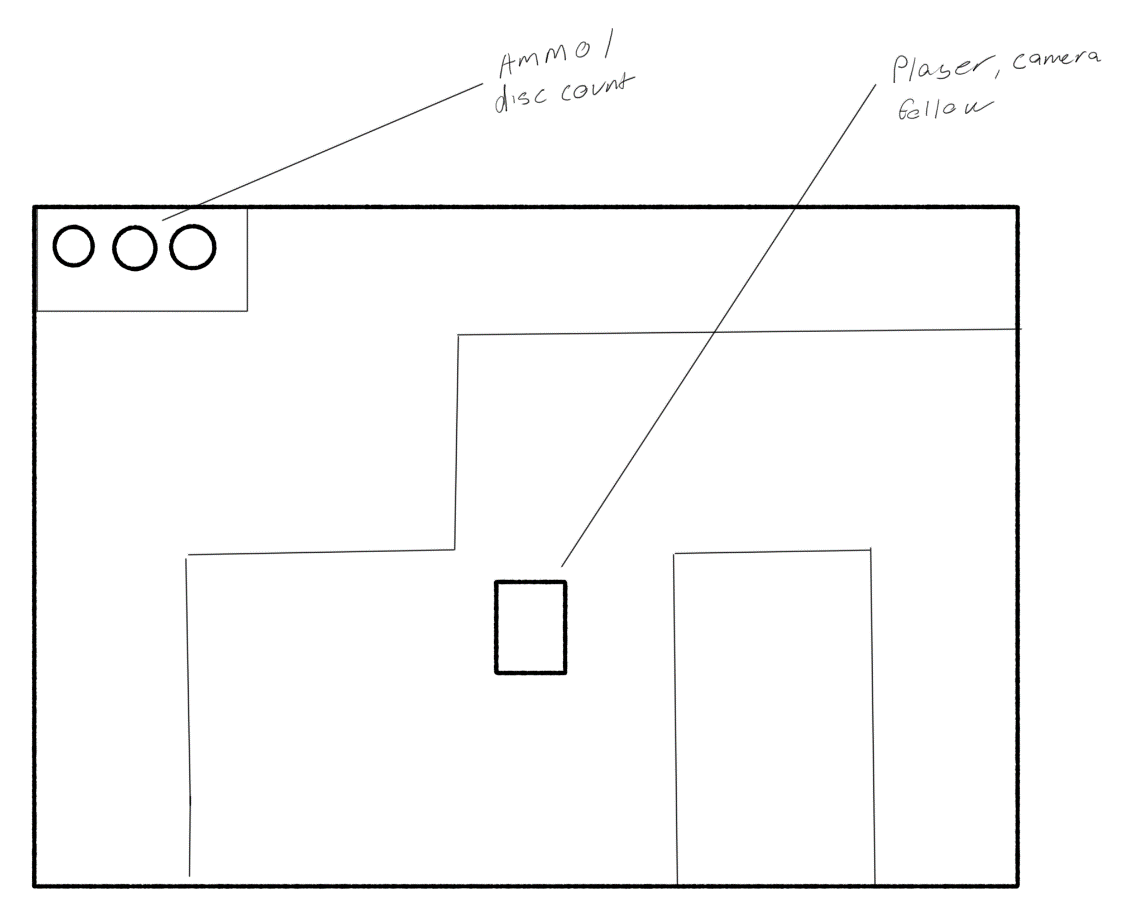
Player and Disc

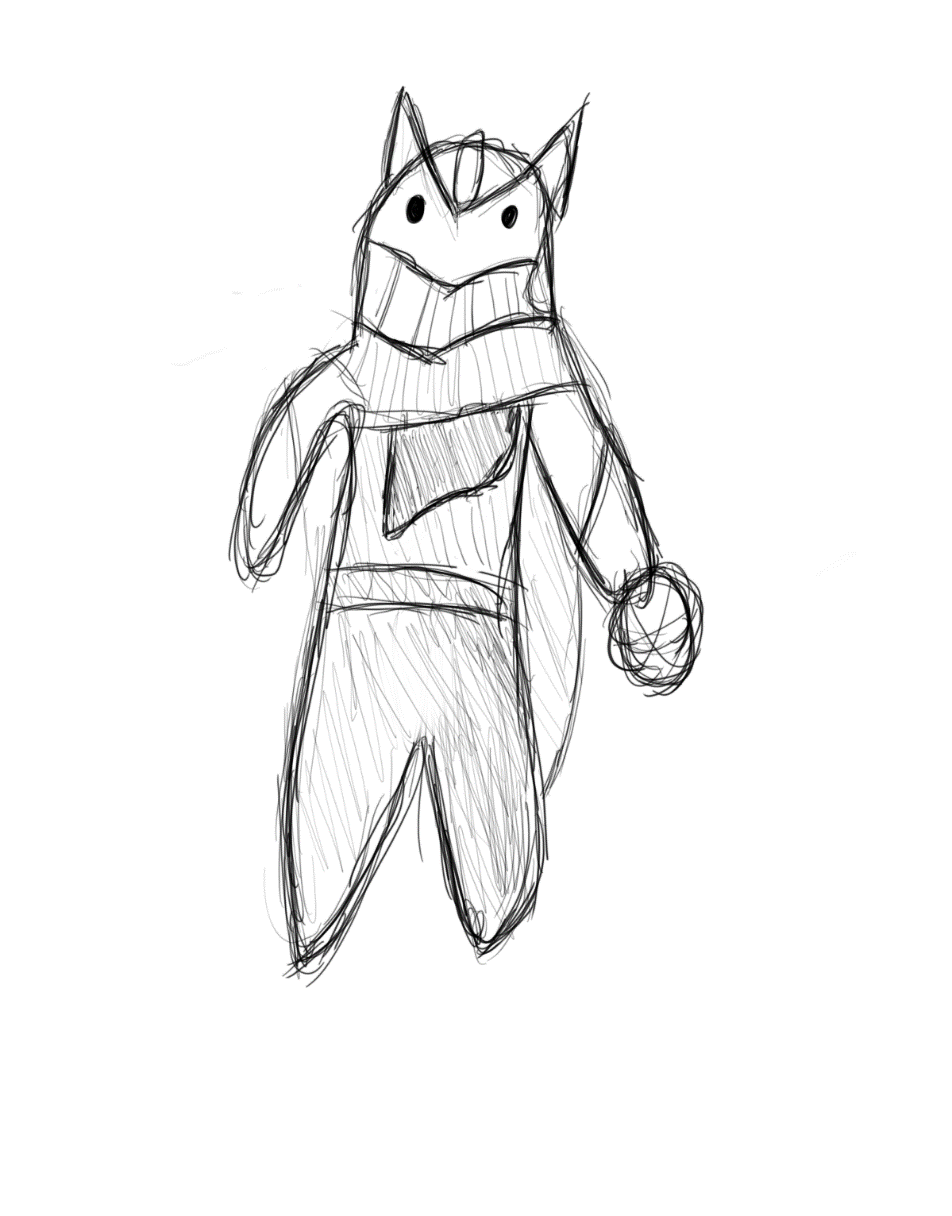


Map

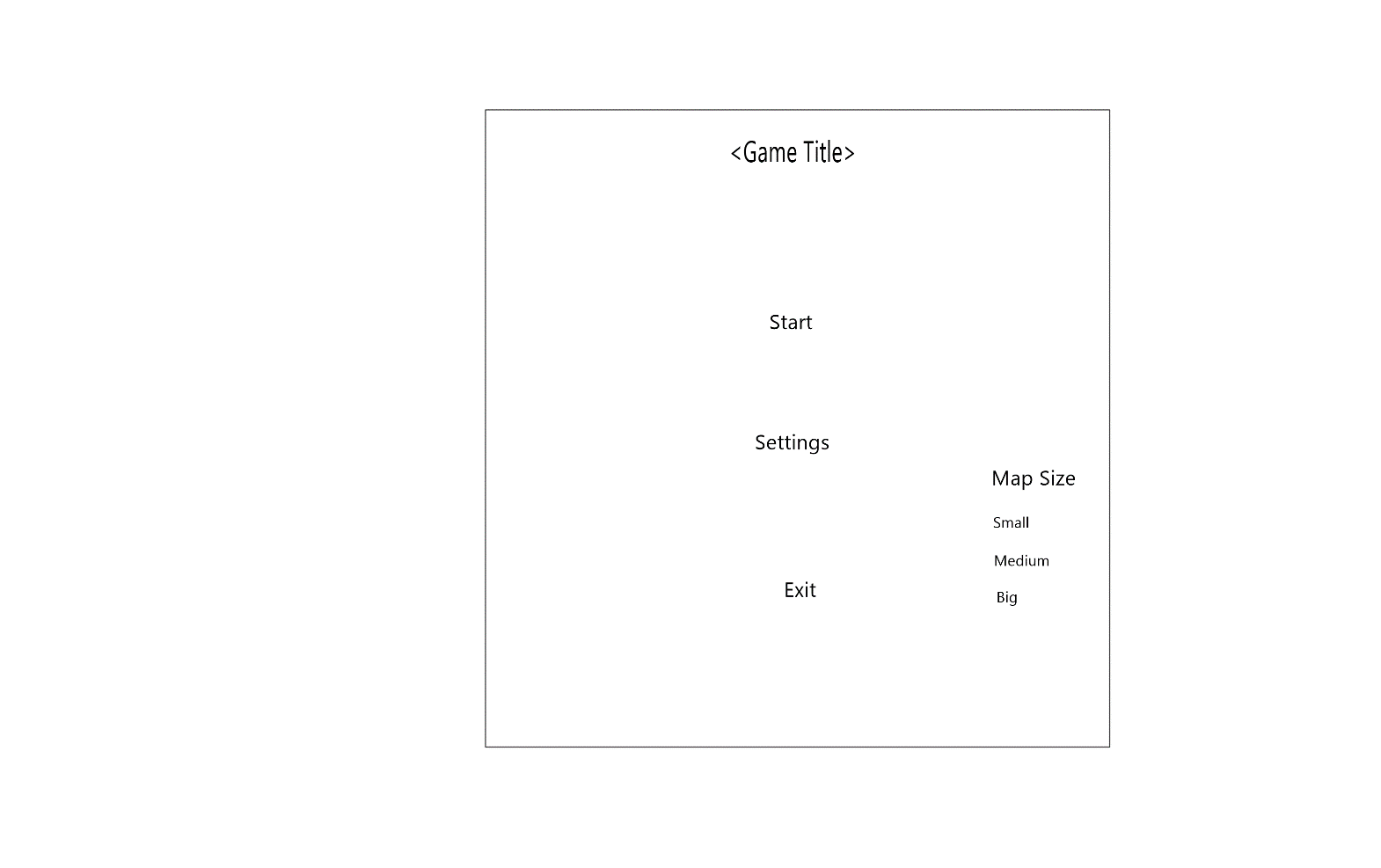


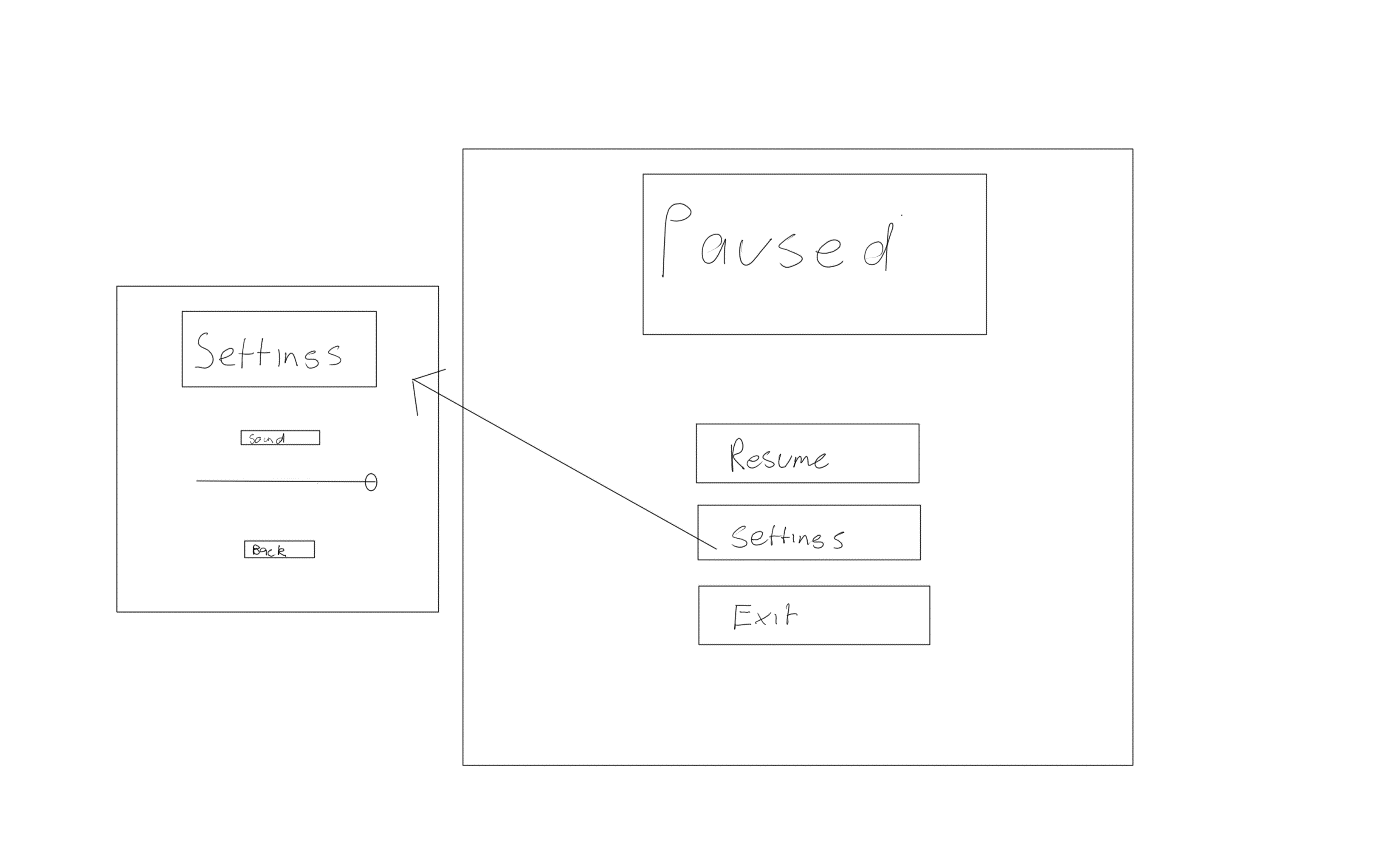
UI Sketch



Player Sketch 2

Main Menu Concept



Game Pause + Settings

New Idea to be worked on:

Initial idea new / to be developed

This will be a 2d Platformer where the player has to run and complete levels. The character will have multiple ways to quickly get through each level and avoid obstacles. The game will be a fast-paced platformer to keep the player engage it will have multiple varying levels and introduce progressive new mechanics overtime.

Designspecifications

* Movement
  + Left Right movement
  + Wall jumping
  + Animations
* Time slow down
* Basic UI/ main menu
* Traps/ obstacles for player
  + Spikes
  + Laser
  + Projectiles
* Map
  + Custom made levels with increasing difficulty

Feasibility Study

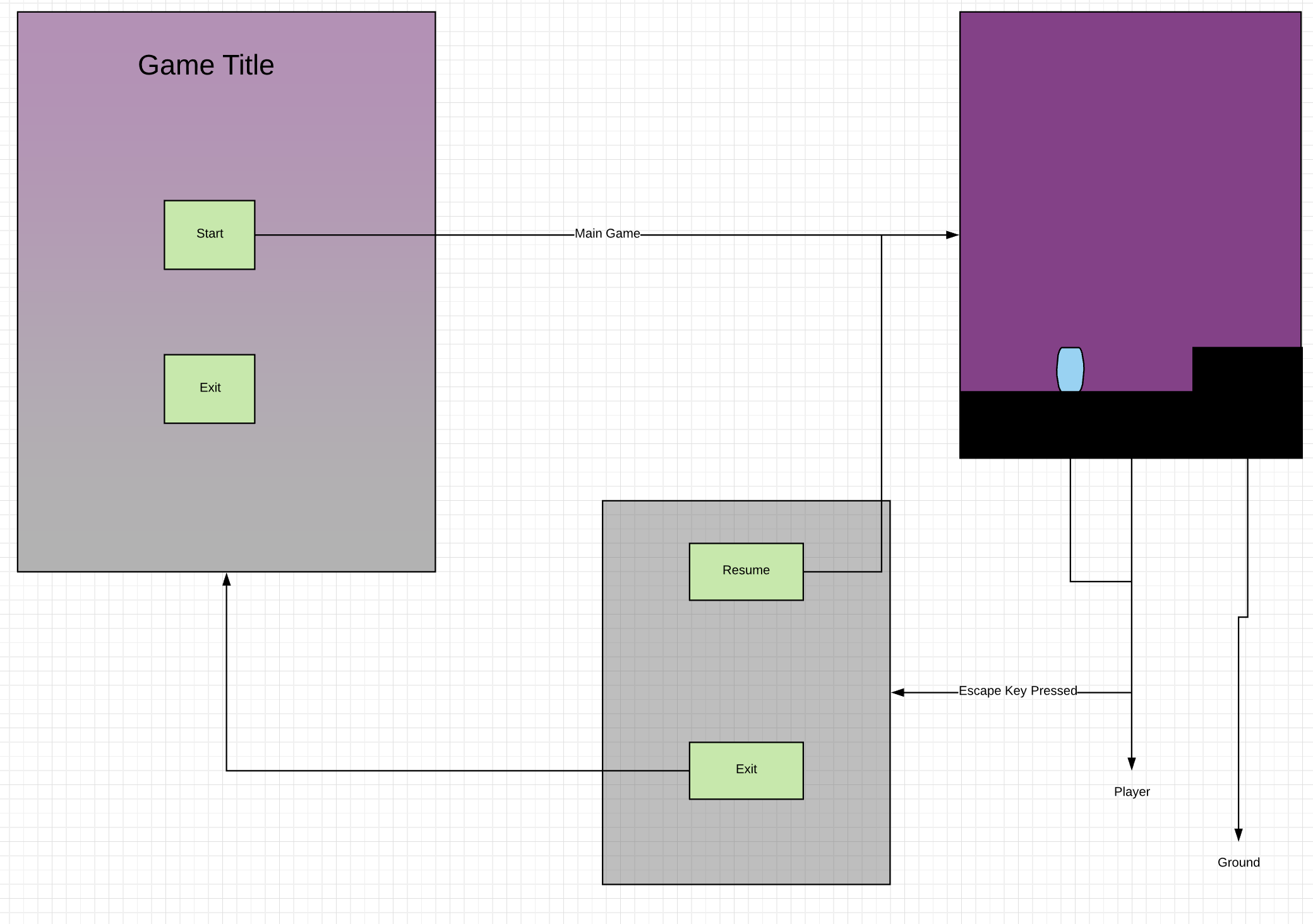
* Godot is an open source free to use game engine with no costs associated
* The game will require low specs to run, with a low amount of storage
* The language which will be used will be GDScript, this is a language similar to python and is the main language of Godot, because this is the chosen language it will have to be learnt which will take more time as I’m unfamiliar with it.
* Art will be created in an application called Aesprite and Tilesetter
* Game Idea can be expanded upon with new map ideas, enemy types, weapon types, etc.
* Game idea is simple to pick up and has depth

Programming language

GDScript, I chose to use GDScript as it is the preferred and best supported language of my chosen game engine Godot. Originally, I was going to use Unity to create my project but due to some network issues and the unity launcher I decided I would work faster in Godot using GDScript. I could have used C# but due to it not being the normal language of the engine forum answers are usually not written in it. Everything else mentioned in the original mention of programming language.

Part A – Planning and Designing checkpoint

Screen Designs and discussion of UI design



Discussion:

The main game screen will have a simple gradient with the elements in the centre. Elements include the game title, start and exit as shown above. When the start button is pressed the user is taken to the main game, which includes no UI. When the escape key is pressed in game the player will be taken to a pause menu which include 2 buttons with an opaque black colour which allows the player to exit or resume. This style of simplistic UI was chosen in order to not overcrowd the user and allow them access to the basic navigation expected of a game.

Input Process Output table for one module – (Jump mechanics) function

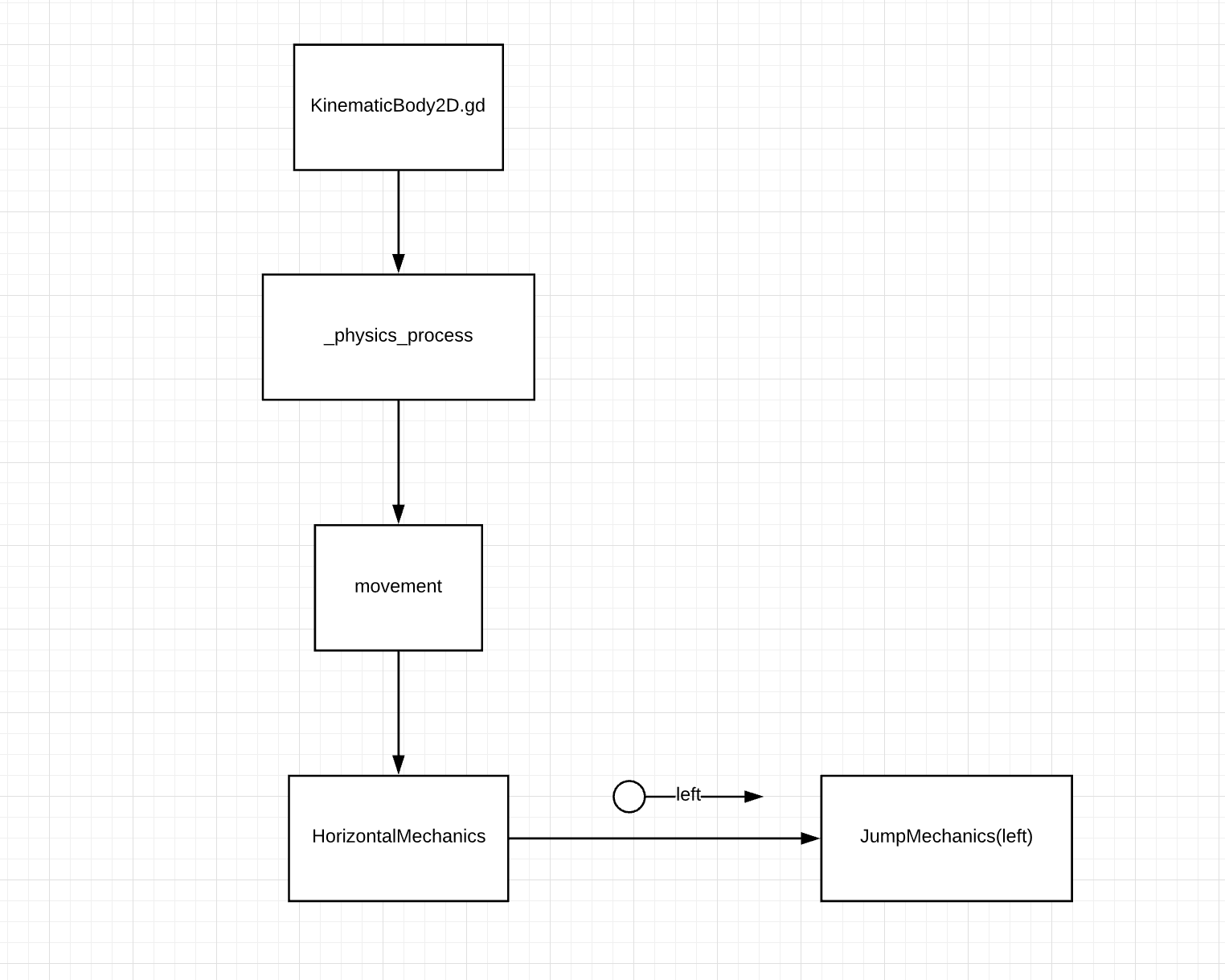
|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| The direction that the player is facing known as left in the code  Whether the Space key is pressed | Checks whether player is on the floor and whether they press the Jump key  If player is on the wall, checks whether the player is pressing against it and whether they press the Jump key | The player Jumps if they are on the ground or if they are touching the wall, If the player is sliding down the wall it slows the gravity showing a sliding effect. |

Identify and describe all key functions (module)

Player controller (KinematicBody2D.gd)

|  |  |
| --- | --- |
| **Function** | **Application** |
| \_physics\_process(delta) | Called during the physics processing step of the main loop. Physics processing means that the frame rate is synced to the physics, i.e. the delta variable should be constant.  Function calls Movement() |
| Movement() | Calls the Functions below and controls the gravity mechanic as well as using the in built move\_and\_slide function for player movement |
| HorizontalMechanics() | Controls are values to do with movement on the x axis as well as determining the direction of the player |
| JumpMechanics(left) | Controls all values to do with the Y value including wall jumping |
| on\_UpGravity\_body\_shape\_entered(body\_id, body, body\_shape, area\_shape) | Detects whether an object enters the body, than makes the flips the gravity |
| \_on\_UpGravity\_body\_shape\_exited(body\_id, body, body\_shape, area\_shape) | Detects when an object exits the bod, then restores gravity |

Structure chart



Pseudocode (Horizontal Movement function)

BEGIN

REM 🡪 Where min returns the lowest of 2 values and max returns the largest of 2 values

motion = Vector2

LET ACCELERATION = 30

LET MAX\_SPEED = 450

left = FALSE

Sprint =1

IF SHIFT key is PRESEED THEN

Sprint = 1.3

ENDIF SHIFT key is RELEASED THEN

Sprint = 1

ENDIF

IF RIGHT ARROW or D is PRESSED THEN

motion.x = min(motion.x + ACCELERATION , MAX\_SPEED \* Sprint)

PLAY run animation

ELSE IF LEFT ARROW or S is PRESSED THEN

motion.x = max(motion.x - ACCELERATION, -MAX\_SPEED \* Sprint)

FLIP SPRITE

PLAY run animation

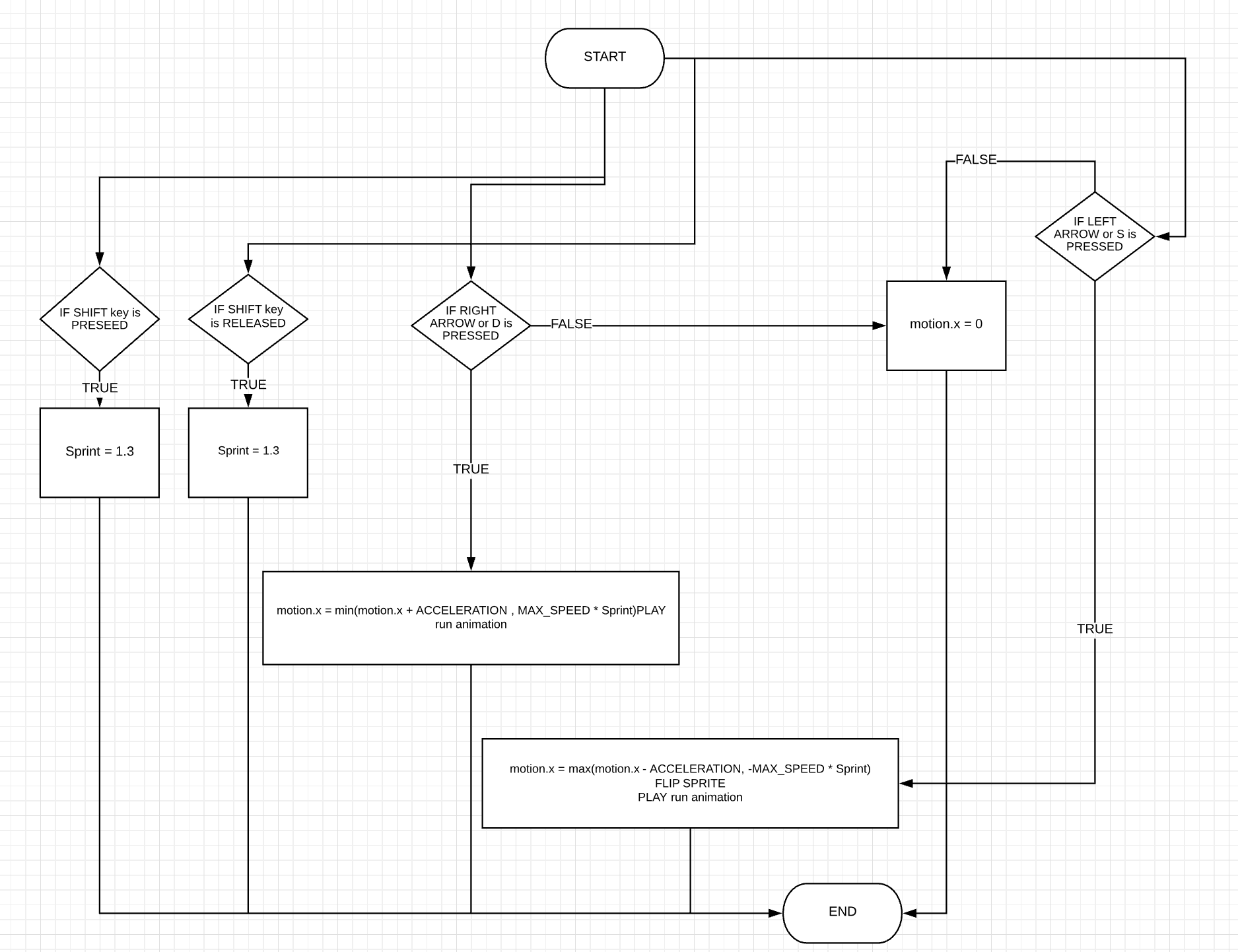
ELSE

motion.x = 0

ENDIF

END

Program flowchart (1 module)



Data dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable Name** | **Data Type** | **Description of Purpose** | **Format of data values** | **Expected values** | **Data Validation** |
| FLOOR | Vector 2 | Makes sure the player doesn’t snap to the ground when using built in function move\_and\_slide | X,Y | Const (0, -1) | Const (0, -1) |
| ACCELERATION | Int | How fast the player accelerates |  | Const (30) | Const (30) |
| MAX\_SPEED | Int | The max speed of the player |  | Const (450) | Const (450) |
| JUMP\_HEIGHT | Int | How high the player can jump |  | Const (-380) | Const (-380) |
| Sprint | Int | Controls the sprint – acts as a bool |  | 1  1.3 | 1 or 1.3 |
| Gravity | Int | Controls the downwards force the player experiences |  | 10  -0.5 | N/A |
| wallPush | Int | The force the player experiences when jumping off the wall |  | 525 | N/A |
| Motion | Vector2 | Controls all of the motion in the x and y direction for use in the move\_and\_slide function | X,Y | Motion.x  ± (30, 60, 90, 120,150,180,  210, 240, 270, 300, 330, 360, 390, 410, 440, 450+  Motion.y  Any value including decimals | N/A |
| gravityFlipped | Int | Controls the gravity flipping in order to make the player move upwards when in the dedicated areas |  | 1,  -0.5 | N/A |
| Left | Bool | Detects whether the player is facing the left direction |  | True  False | N/A |
| world\_scene | String | Store the scene file, so when the player enters the portal it switches to the correct level |  | World.tscn  World1.tscn  … | N/A |

Test data (1 module) - HorizontalMechanics()

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected Outcome** | **Outcome** |
| **Shift Key** | When Pressed  Player moves faster  When Released  Player returns to normal speed | When Pressed  Player moves faster  When Released  Player returns to normal speed |
| **Right Key or D** | Player moves towards the right | Player moves towards the right |
| **Left Key or S** | Player moves towards the left | Player moves towards the left |
| **No Input** | Player doesn’t move | Player doesn’t move |

Desk check results (1 module) - JumpMechanics(left):

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Condition** | **Expected Outcome** | **Outcome** |
| **Left = True** | On Floor | Player jumps | Player jumps |
| On Wall && On Floor | Player jumps and pushes himself towards the right | Player jumps and pushes himself towards the right |
| **Left = False** | On Floor | Player jumps | Player jumps |
| On Wall && On Floor | Player jumps and pushes himself towards the left | Player jumps and pushes himself towards the left |

Log book/ GitHub updates

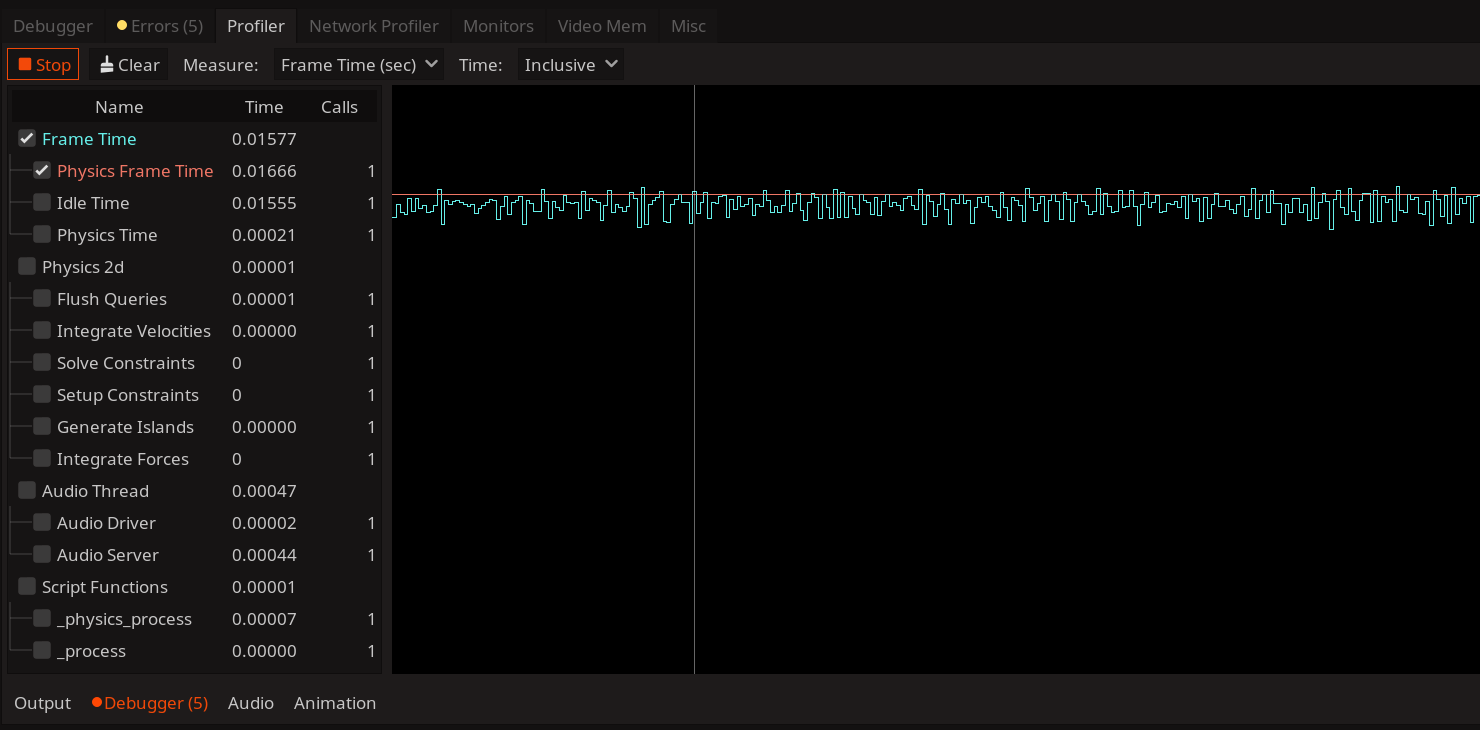
<https://github.com/mitchellk1401/SDD-Major-Work> - Added as collaborator to project

Folio Part 2B

Discussion of errors and testing strategies

Testing strategies used:

* Print statements: To understand what functions of code was running and how often it was happening.
* Breakpoints: As GDScript is an interpreted language upon running the application through the Godot Engine it places a breakpoint where an error happens which made debugging easy.
* The profiler built into the engine was used in order to find functions which caused drops in FPS. Screenshot of profiler below showing frame times. It also looked at the response of function called in the screenshot below it shows the main menu’s response time.



Errors encountered while working on project

Most of the errors encountered whilst working on the project was due to logic and syntax errors. For example, the most recent one was the timer not working between scripts. This was a logic error which happened due to a script being called at the beginning of the application but also being called multiple times when game scenes changes as the object was instantiated with the player object. This errors was fixed by making a separate script that loads when the executable starts and the text for the timer is updated by getting the time value from the separate script in order to stop clashing values in the project.

Detailed test report

Horizontal mechanics

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected Outcome** | **Outcome** |
| **Shift Key** | When Pressed  Player moves faster  When Released  Player returns to normal speed | When Pressed  Player moves faster  When Released  Player returns to normal speed |
| **Right Key or D** | Player moves towards the right | Player moves towards the right |
| **Left Key or S** | Player moves towards the left | Player moves towards the left |
| **No Input** | Player doesn’t move | Player doesn’t move |

**Comments on function**

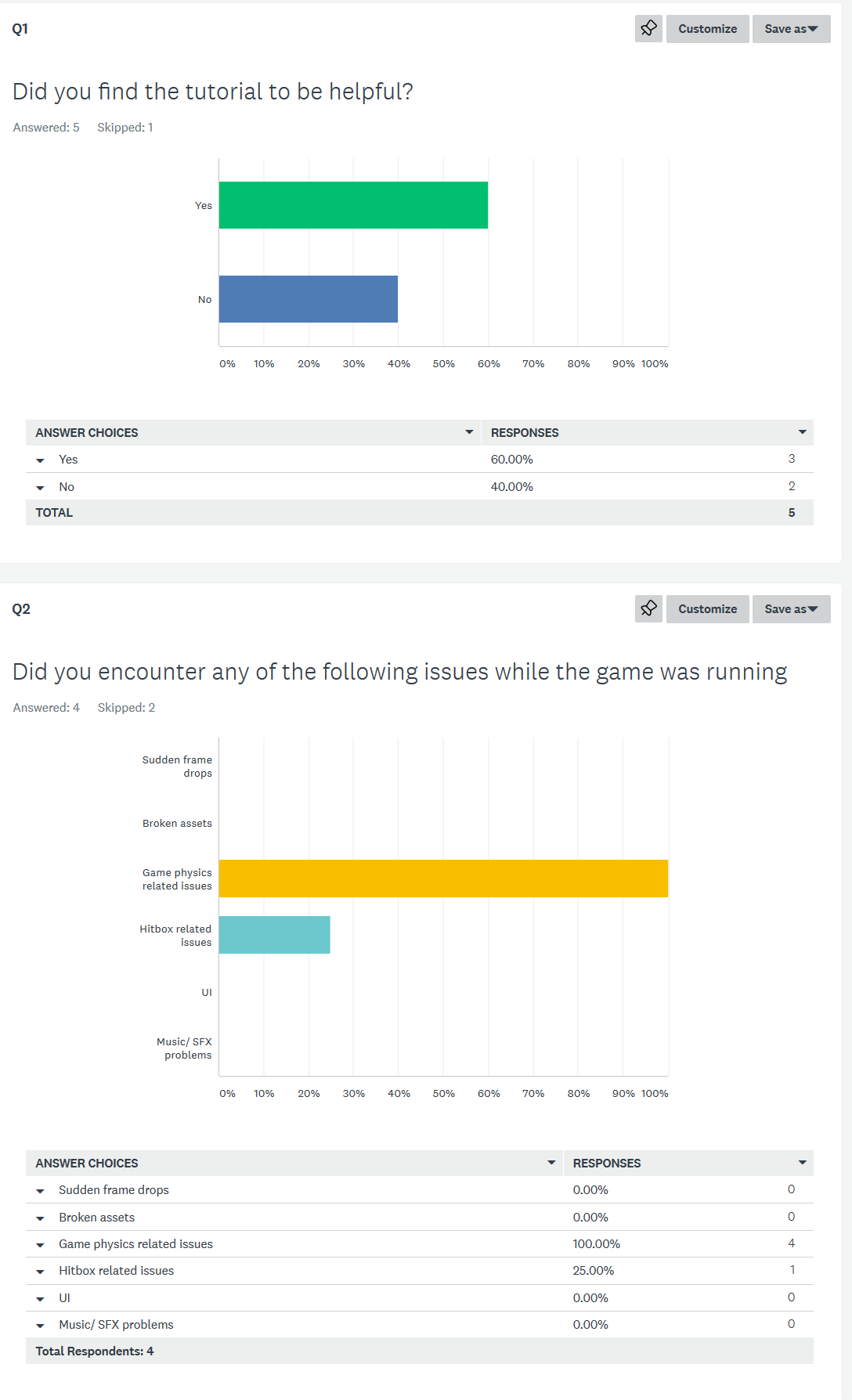
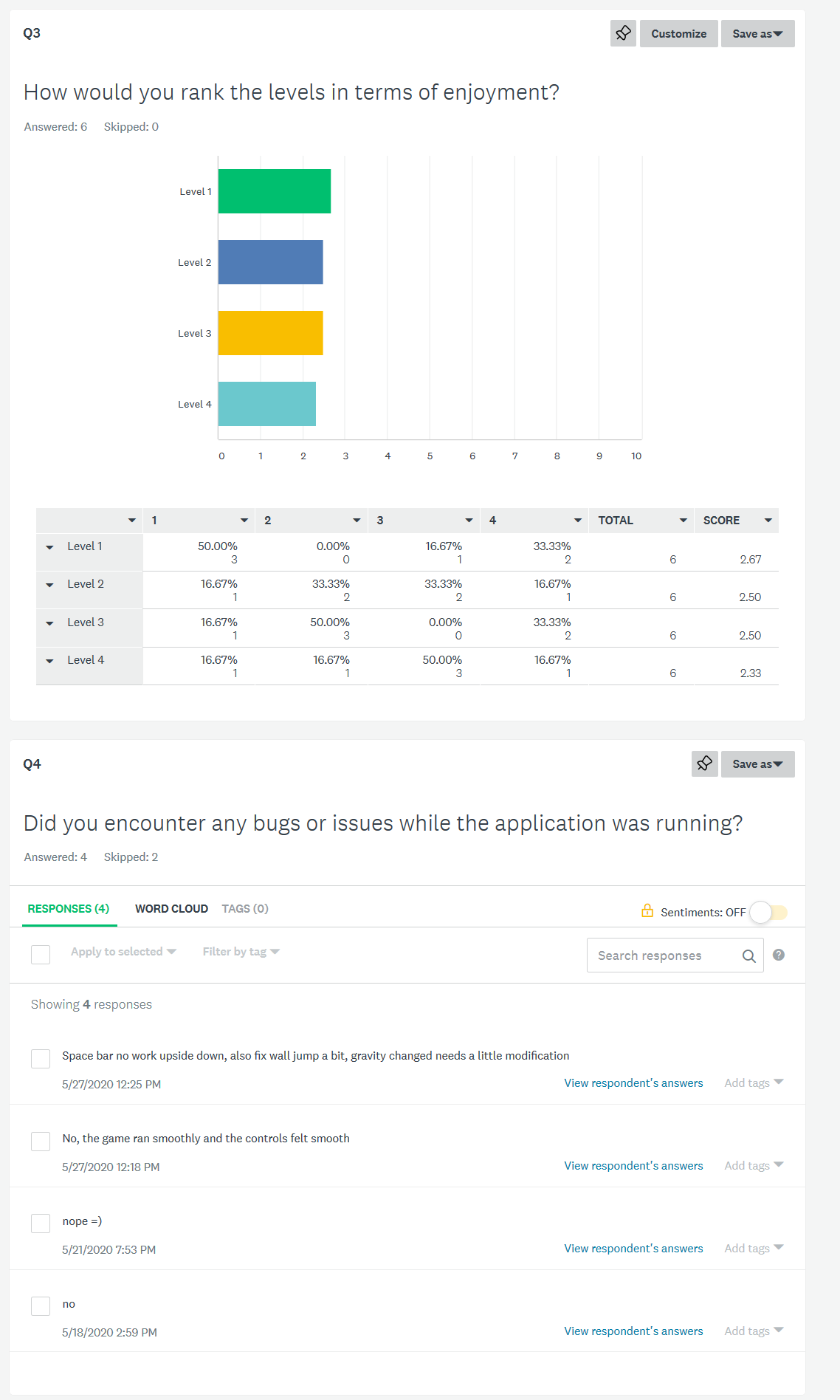
Function works accordingly for the player moving and does not cause any problems if other keys are pressed. The algorithm performs how you would expect a character to move around the scene with no lag visible to the user. Feedback from users about this section was the character felt slow in terms of speed which has been updated in a newer version and there was no lag noticed by any users who used the application.

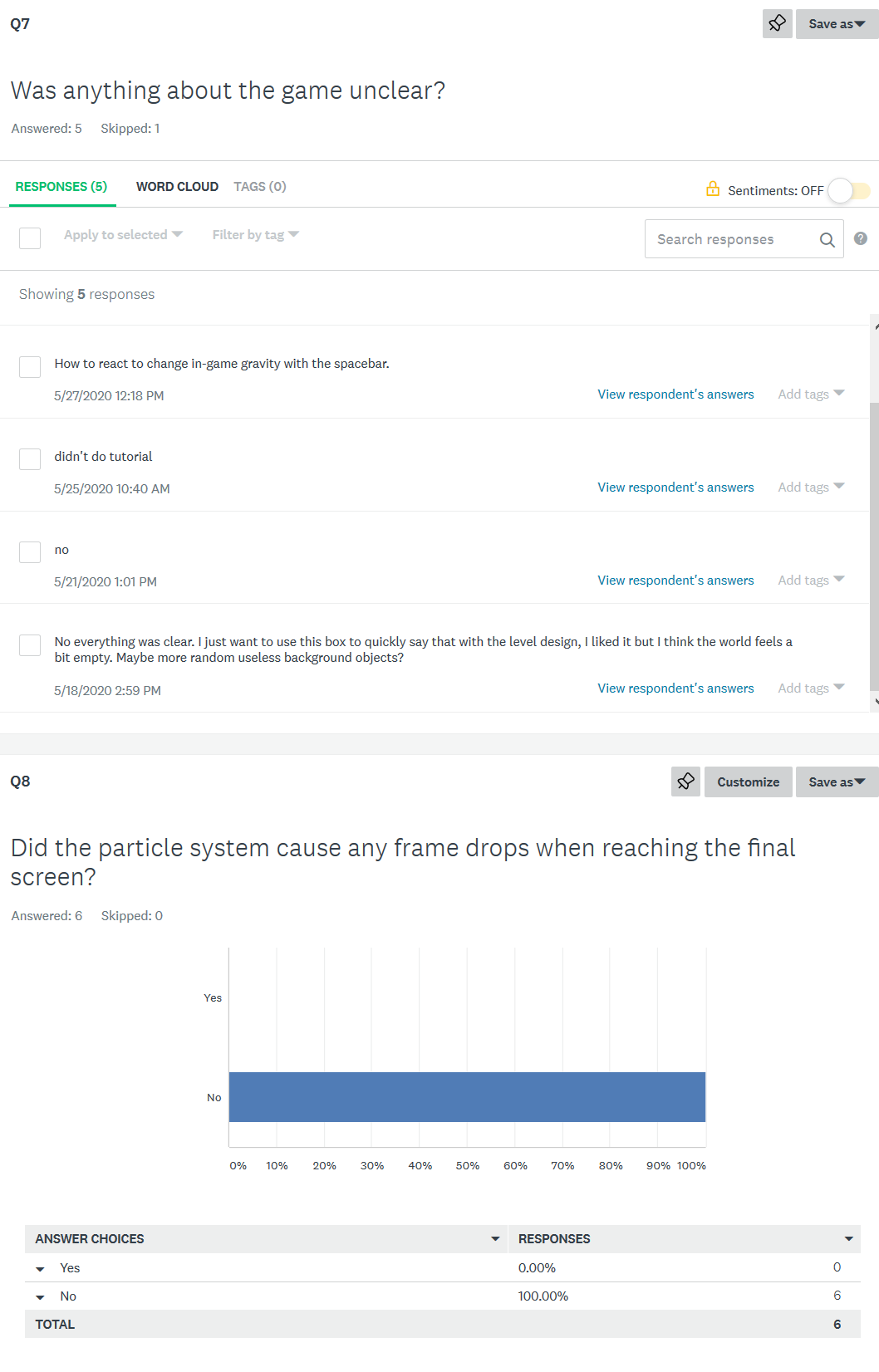
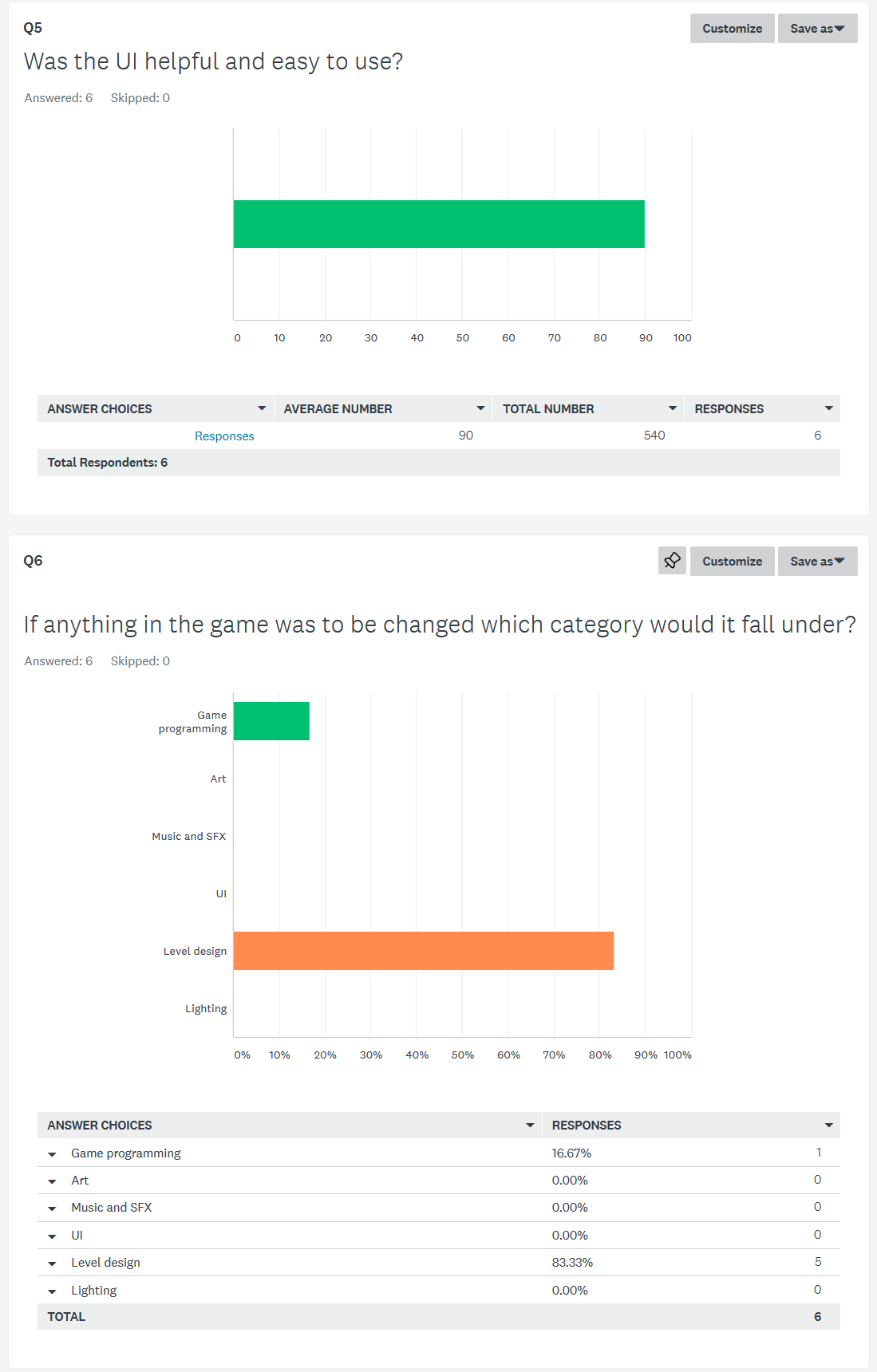
Peer evaluation and acceptance testing

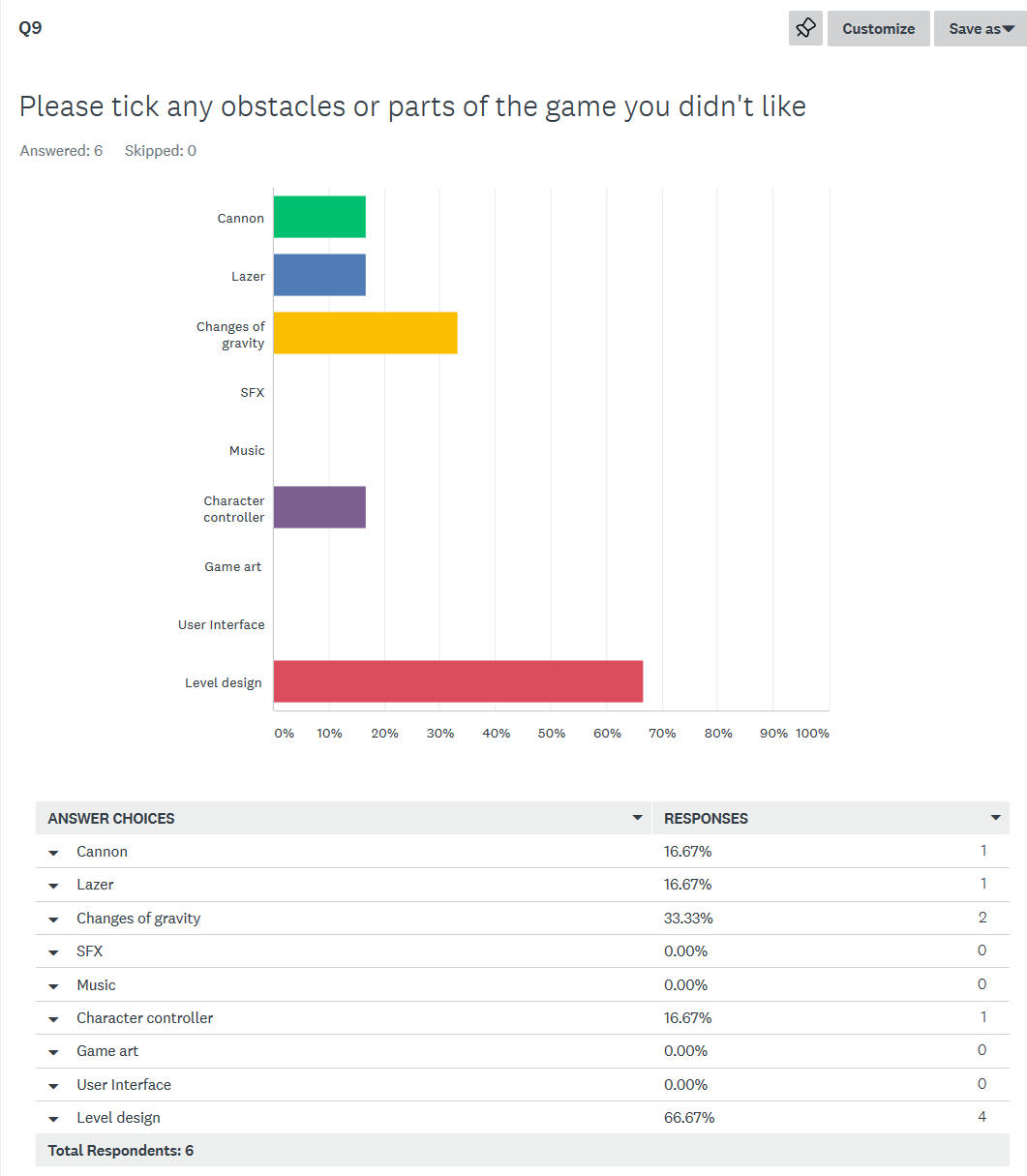
My project was tested by 9 different people where 6 people filled out a survey and the others gave vocal feedback on the project while they played.

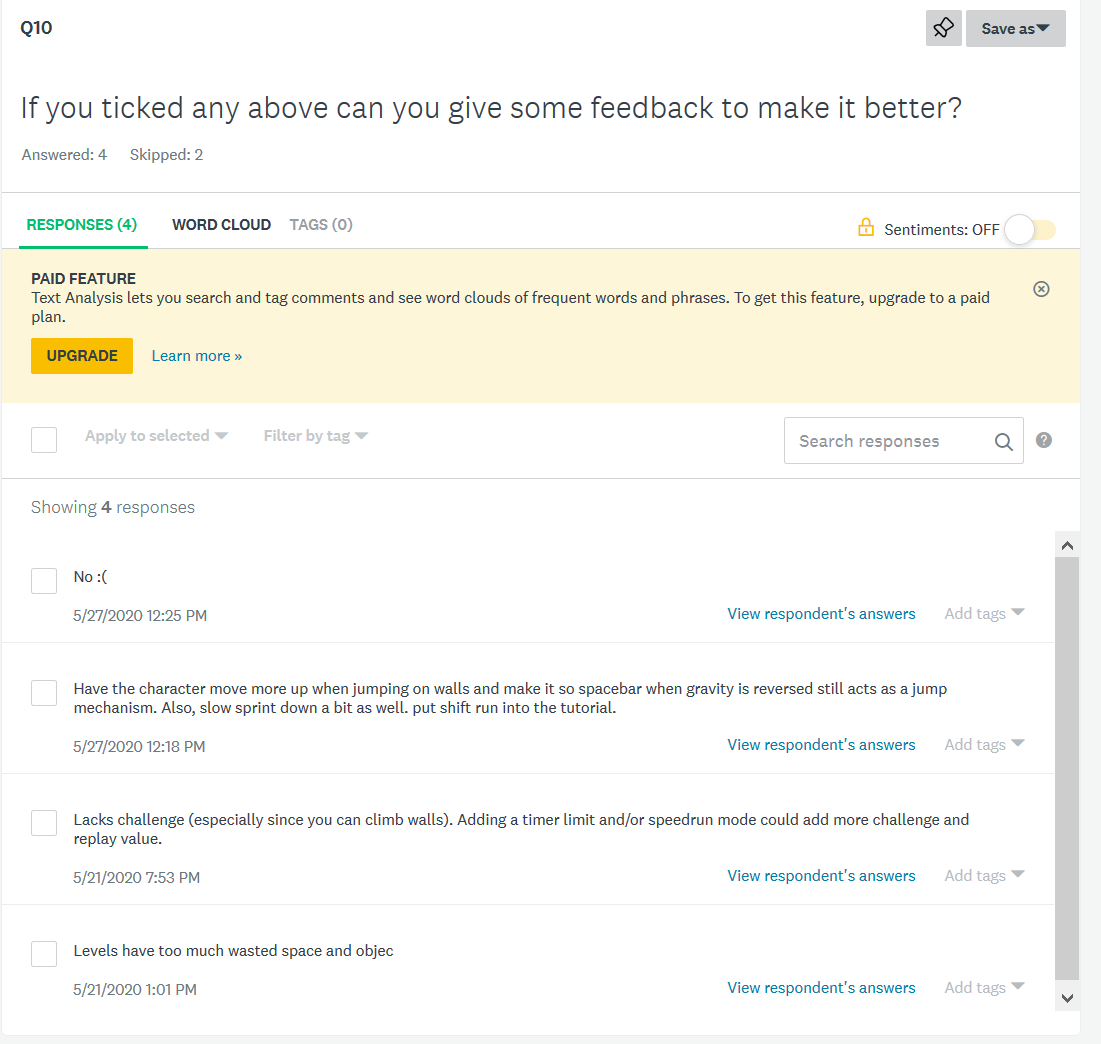
Non- Survey feedback

* Levels feels empty
* Character controller feels slow
* The 3rd level with the platforms was frustrating

The survey feedback included 10 questions: Results







Evaluation

User documentation and training

In-game tutorial + Website

mitchellk1401.github.io

Digital copy of source code/ compiled application

Compiled application on website + Copy of source code on GitHub

Mitchellk1401.github.io

GitHub updates

<https://github.com/mitchellk1401/SDD-Major-Work>

Bibliography

* <https://docs.godotengine.org/en/stable/tutorials/2d/particle_systems_2d.html>
* <https://docs.godotengine.org/en/stable/tutorials/2d/2d_movement.html>