KINGA KLOSKOWSKA

CM3133 - GAMES DEVELOPMENT

GAME DESIGN DOCUMENTATION

MODULE LEADER: YANG JIANG

WORD COUNT: 835

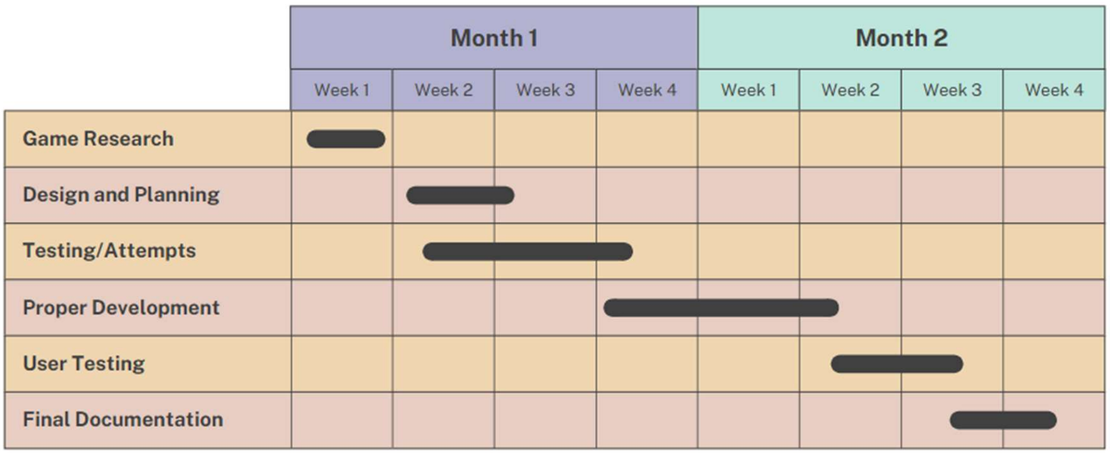
Page of Contents

Project Timeline/Gantt Chart

The project was set out in a way to take me just under two months (October – December), I have planned out each stage beforehand and carefully explained how those certain stages came to be afterwards. The below information briefly represents my goals and directions in which I aimed to head in before starting the project.

- **Week 1 – 2 Game Research**, looking into interesting mobile/small PC games, checking out their mechanisms, components and possible assets using source codes on GitHub.
- **Week 2 – 3 Design and Planning**, following the assessment brief, designing the possible wireframes and general interface of the possible game.
- **Week 2 – 4 Testing/Attempts**, testing the game idea to see whether it's possible to make in a short timeline. Seeing what works and what might need to be changed.
- **Week 4 – 6 Proper Development**, developing the game according to the outlined plan, based on previous testing and attempts.
- **Week 6 – 7 User Testing**, testing the game out for bugs and letting other people around me play the game and provide me with feedback.
- **Week 7 - 8 Final Documentation**, making sure that all documents regarding the game are complete, making last changes to the game and recording the final game presentation. Uploading necessary parts of the assessment to GitHub ready for submission.

Project Timeline



Project Timeline - Explained

Game Research

The first week was fully focused on researching the types of games that seem possible to recreate and run easily on Unity. I have picked a couple of games I played before such as Alto's Adventure, Geometry Dash and finally, Flappy Bird. I chose the last option because of the intriguing simplicity of the game as well as addictive factors.

Design and Planning

The design and planning of the document took around 2 weeks, I have complied the previously gathered information from the research and began creating mood and story boards alongside wireframes to help visualise how the game might look like. I have also looked into example Flappy Bird game source codes, to extract the vital requirements and useful tips on how to run this type of game properly. In this stage I have also took my time drawing my own character and designing the obstacles.

Testing/Attempts

During the testing stage, I was testing what works and what doesn't. Trying to put together and incorporate the vital parts of the game to make it usable. I have managed to do test out different characters, learn the proper usage of physics, rigidbodies and components in Unity 2D.

Proper Development

After testing and trying out new things, I had to finally start my own project. Finally combining the knowledge learned both in lectures and successfully retaining knowledge from useful tutorials from YouTube. I have started to develop Fly Ball. Steep learning curve, however in the end it all worked out.

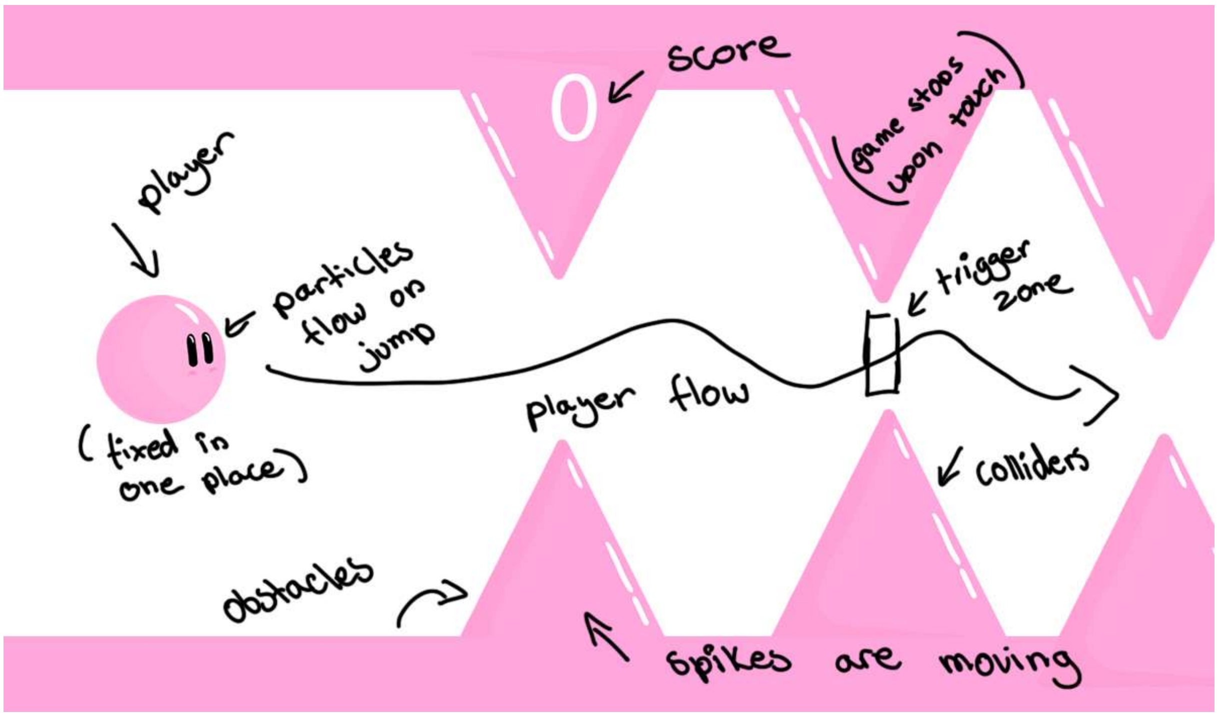
User Testing

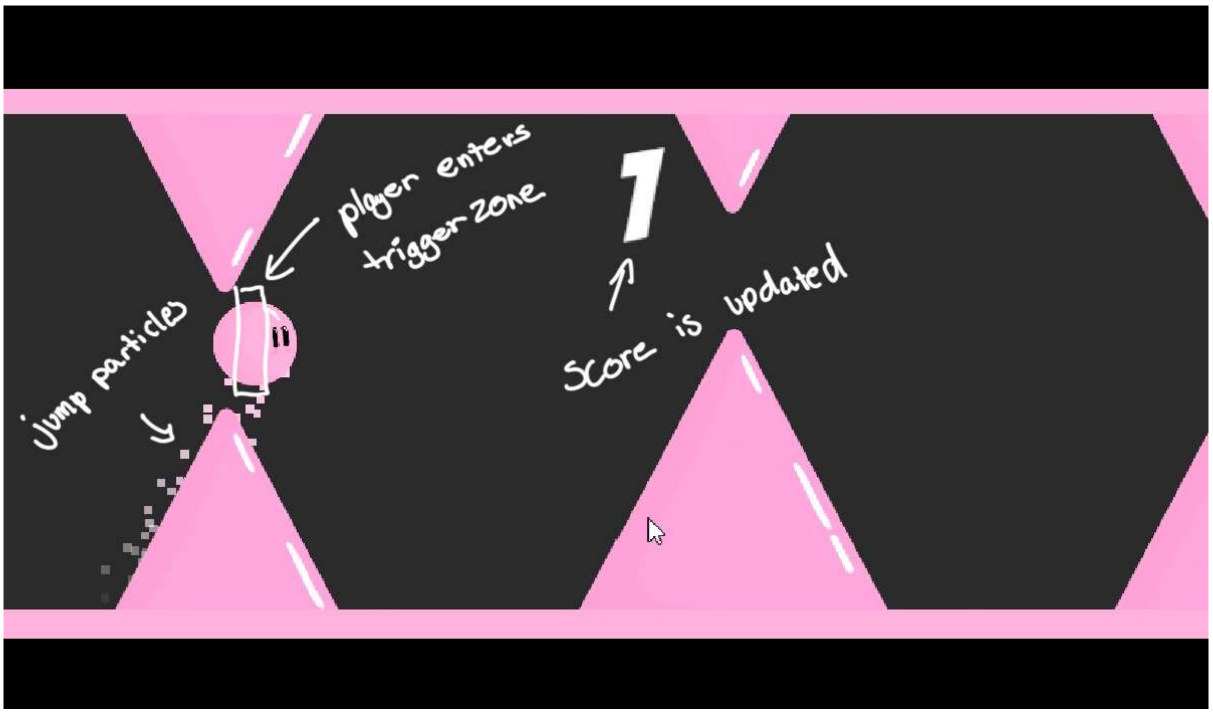
The game was tested throughout the developmental stage, with the feedback I have received from my friends as well my own adapted criticism. I have decided to tweak a couple of in-game's assets. I have increased spike spawn rate as well as speed as well as the possible "opening" to gain a point.

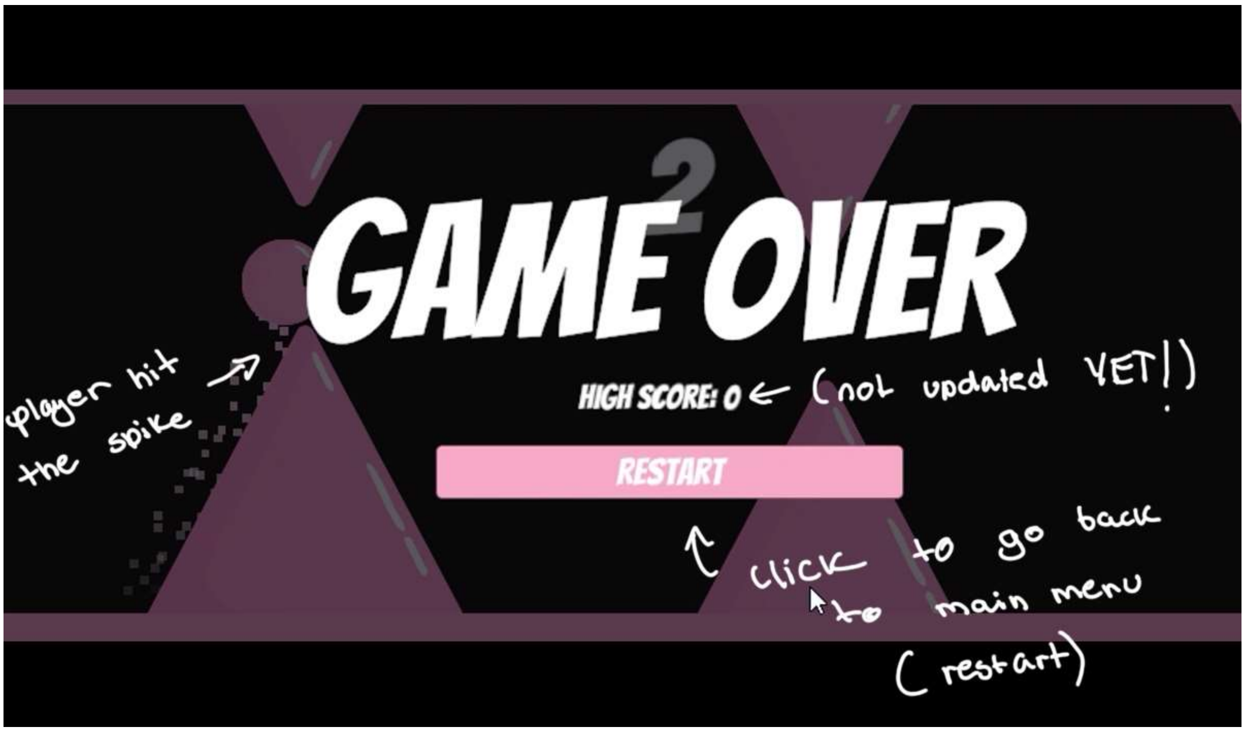
Final Documentation

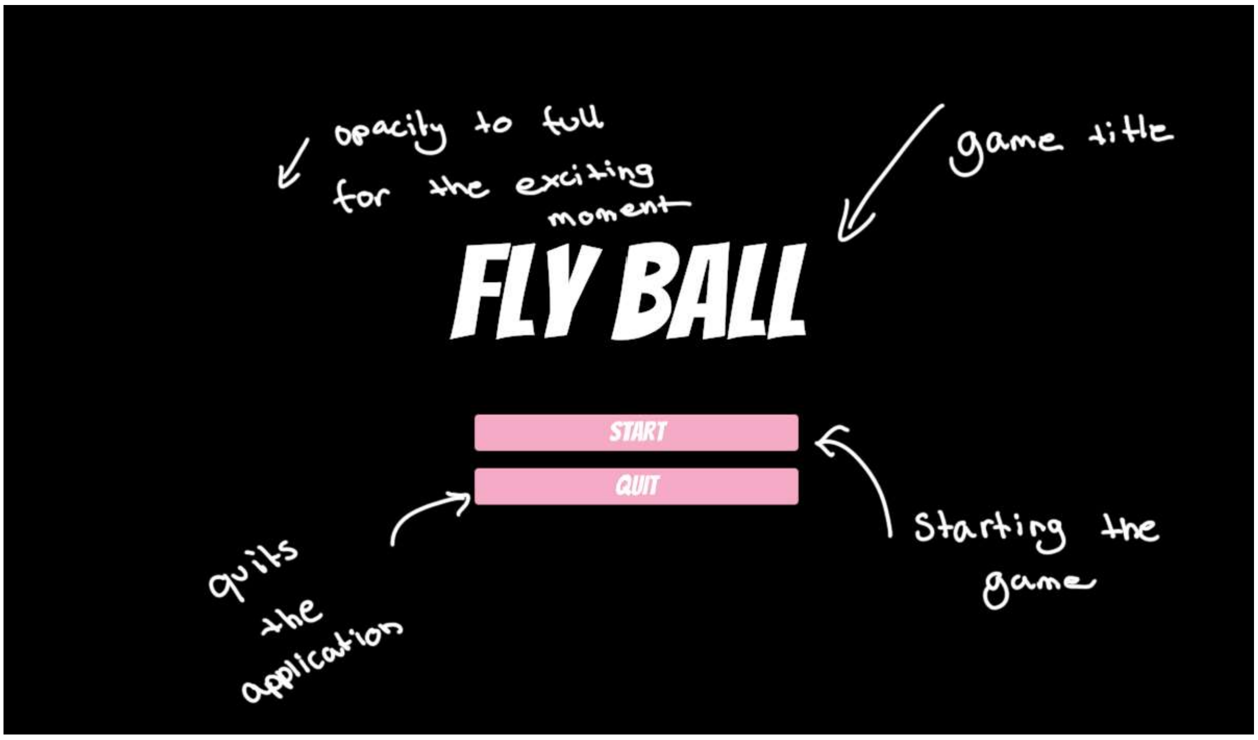
Checking over the final version of documentation regarding the project, making sure all of the boxes are ticked, recording a presentation video for the game.

Storyboard

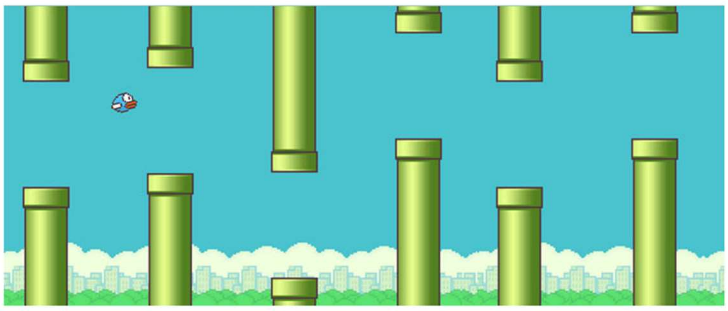

score
0
(game stops upon touch)
player
particles flow on jump
trigger zone
(fixed in one place)
player flow
colliders
obstacles
spikes are moving

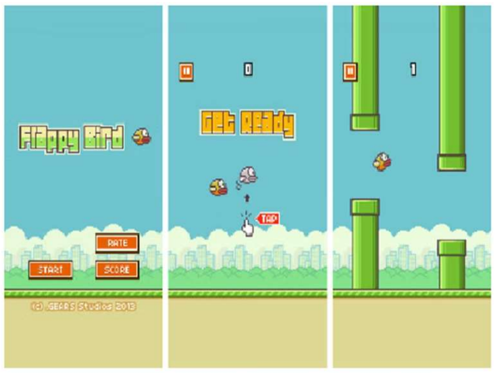

player enters trigger zone
1
score is updated
jump particles

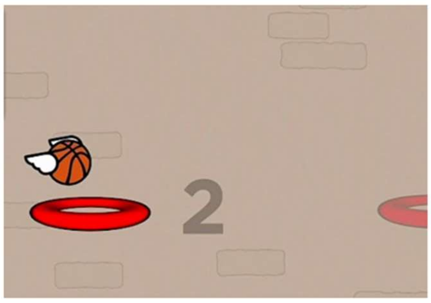

2
GAME OVER
HIGH SCORE: 0
RESTART
player hit the spike
(not updated YET!)
click to go back to main menu (restart)


opacity to full for the exciting moment
game title
FLY BALL
START
QUIT
quits the application
Starting the game

Moodboard





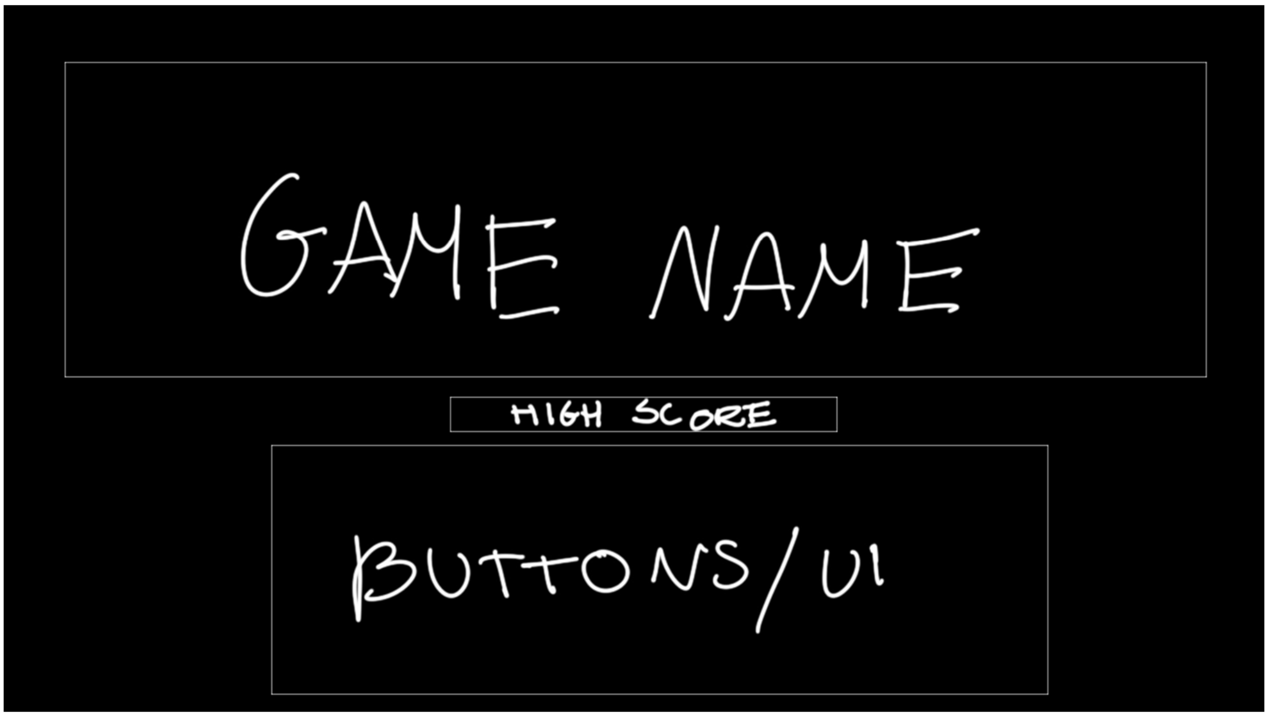


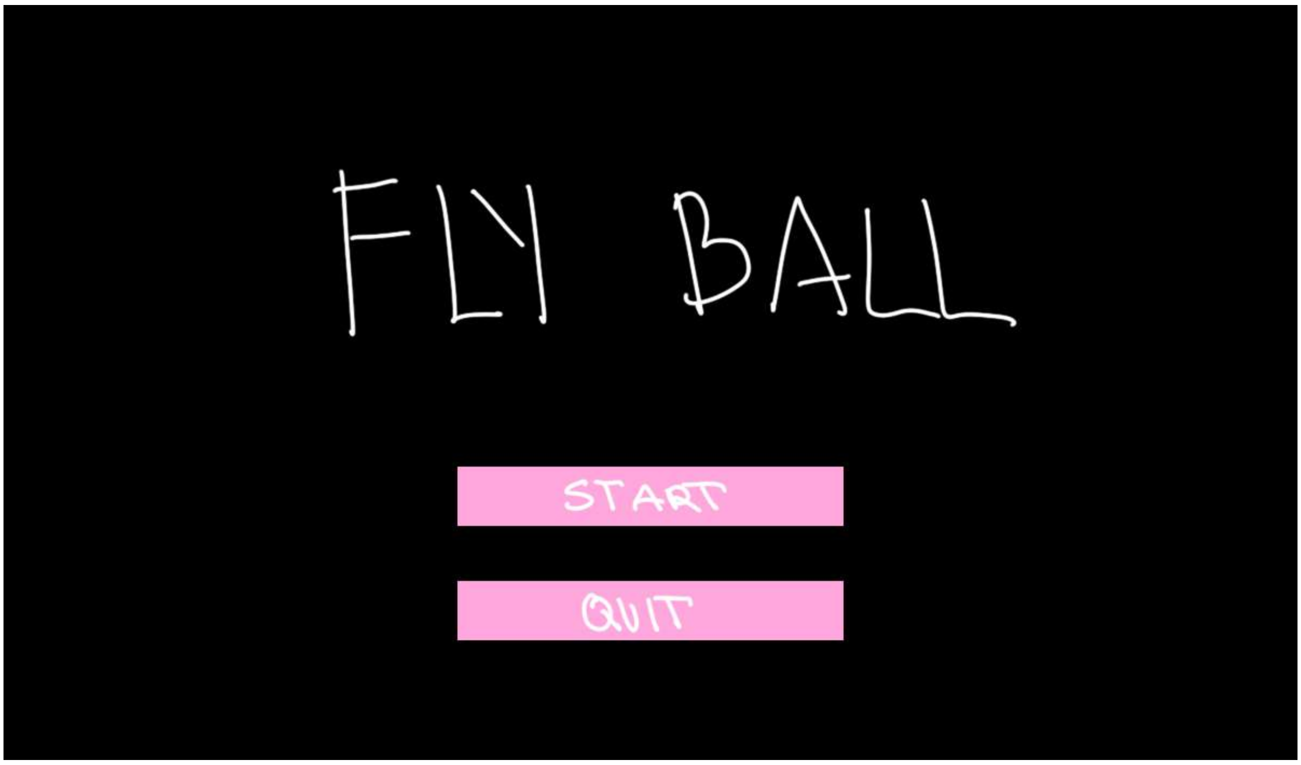
GUI Wireframes/Custom Buttons

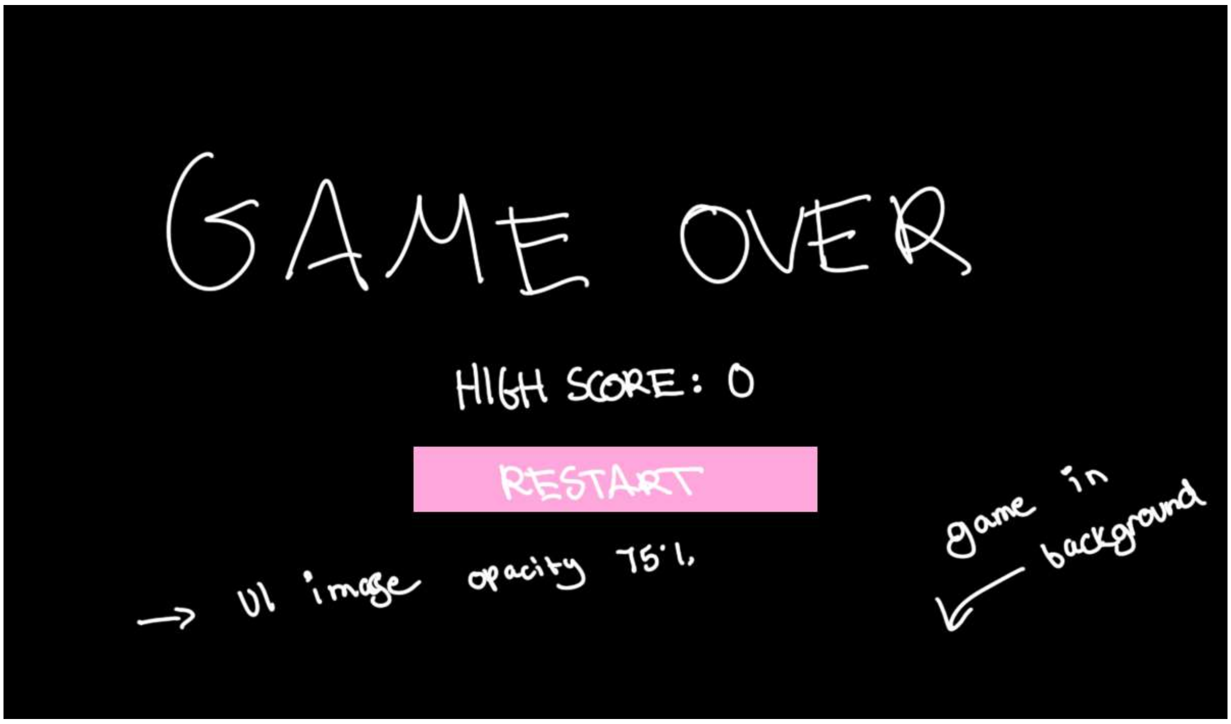
WELCOME TO...

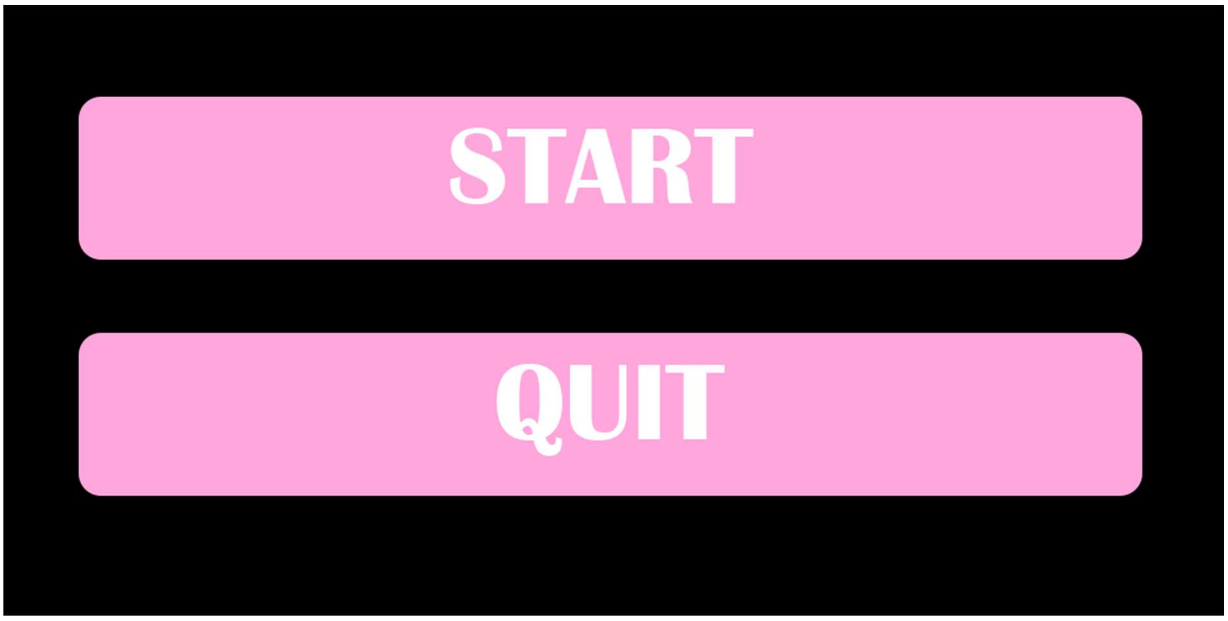
GAME NAME

START BUTTON


GAME NAME
HIGH SCORE
BUTTONS/UI


FLY BALL
START
QUIT


GAME OVER
HIGH SCORE: 0
RESTART
→ UI image opacity 75%
game in background


START
QUIT

Games Research

Flappy Bird is a simple 2D game that requires to player in this case a bird that is fixed to one position creating an illusion that it’s moving, to simply move up the screen as they pass through infinitely re-spawning green pipes with the goal of achieving the highest possible score and beating other players. The addictive factor lays just in this initiative, trying to beat players around the world. The game incorporates animation, the wings of the bird are moving and looping as well as parallax background and moving ground animation. This fascinating mechanism includes a working scoring and high score system that provides users with dopamine every time they beat their previous attempts.

For the comparison, “Dinosaur Game” very well known for Google users with poor internet connection, works similarly to the previously mentioned game. The dinosaur character is fixed to the left side of the screen, its only real job is to jump over the obstacles such as cacti and rocks spawned randomly in front of the player. The user receives the same satisfaction through the amount of distance travelled without touching the deadly obstacles.

Requirement Analysis

In-game functional vital assets

- **Character** the player, vital part of the game, the user controlled asset.
- **Spike Spawner Game Object** that takes care of re-spawning the spikes across the player “playground”, obstacles to be avoided. Script is added to change the randomized height of the spikes and how often they appear.
- **Spike Object** vital asses in order to create the spike spawner, positioned perfectly opposite to each other to make two spike obstacles.
- **Spike Movement Object** that makes sure that those two spikes relate to each other, script is later added to adjust the speed of the spikes. A **<u>trigger collider box</u>** is also added to the end of the spike “opening” to effectively record player’s score.
- **Jumping Particles** for the player to add some variety to the game, the player shows a trail of pink particles behind it as soon as it jumps. When it stops the particles begin to fade away.

- **Starting Button** activating the game, letting the player start playing the game.
- **Quit Button** shutting down the application with a simple Application.Quit script.
- **Restart Button** the button allows the player to restart the game, changing the scene back to main screen/starting a new game once again.
- **Score Text** with the use of simple scripts as well as trigger collider box placed in between of the spikes, the player score can be measured accurately.

In-game non-functional assets

- **Background images** for canvases of the UI, making the game look a little bit more aesthetically pleasing.
- **Game Over and Fly Ball Text** to introduce the game name as well as inform the player about their defeat.
- **Rain Sky Particles** adding more ambiance to the overall gameplay, working well with the rain audio.
- **Kinematic Rigidbody Collider Balls** moving around with the goal of satisfying the assignment's needs such as rigidbody colliding with a rigidbody.

User Testing Plan

The scope of the testing is to find out whether the game runs correctly and does what it's initially supposed to. The game will be tested amongst two different candidates and finally me, as the third game tester, with the goal of receiving feedback and some self-criticism. The game presentation will also be presented to all participants in total in order to receive more external feedback and second opinions after completion of the game. The document will also highlight the potential risks of the game and further recommendations. The feedback will be available in the User Testing Documentation.