# Overview

The game will be a microgame collection for mobile that utilises multiple sensors for varied and unique gameplay opportunities. Players will use touch, motion, microphone and proximity input to play through a large number of very short games, usually taking around five seconds each. The player has a certain number of lives which are lost upon failing games, when they’re all one it’s game over.

Ideally this project will act as a cumulation of all the techniques I have learnt throughout the course. It will act mostly as a portfolio piece so that I have more diverse projects to show off.

The target number of games is 80. They can be played in three modes: endless, which cycles through all games of a set difficulty until the player either quits or runs out of lives; challenge, which features increasing difficulties; and practice mode, where a specified game can be played as much as the player wants. Playing the game will earn a currency which can be used to purchase games and difficulties for practice mode as well as some items to change up the gameplay.

The game will be made in Unity, using GitHub for source control and Trello for project management. It will be released for free on Android on the Google Play Store as well as Itch.io. This release version will be of a high level of polish with no known bugs.

Research would include getting the various sensors to work, especially the microphone and proximity sensor as I have had no prior experience with those. The rest of the project will be utilising the game development skills I have learnt throughout the course to create a wide range of games spanning many gameplay styles.

# Aims and objectives

# Progress so far

## Research

As I already have a lot of experience in developing with Unity, I can easily come up with solutions myself so there isn’t a necessary need for very much additional research. However, I will have to research into the development and input techniques that I will be using throughout the project. I have looked into the various topics I need, but no formal research has been conducted so far. The input techniques I will have to learn more about are various forms of motion detection, microphone input and making use of the proximity sensor.

I will be looking at WarioWare and Mario Party minigames to see what similar games have done and use them as inspiration to create some of my own. I will take note of the various methods that they use to create a framework of rules that I will follow to develop my games. I will also be looking at other games I like in a variety of genres, to see what types of gameplay I can incorporate into a 5-15 second microgame.

## Prototype

Most of my time on the project so far has been spent on the prototype. At the time writing, I have designed 84 games, finished programming 26 games, using 10 unique input methods. Progress on creating these games has admittedly been slow, but I am developing them in such a way that I can reuse assets and scripts in order to eventually be able to drag-and-drop features to make games, so the development time for each game will decrease throughout this project.

## Weekly progress (Preparation weeks)

These weeks were all prior to the semester starting, so work on the project is much slower.

**Week 1 (25/05/18)**

* **Goal:** Very basic prototype of the microgame loading system
* **End result:** Very basic prototype, finished programming one game. Created some reusable art assets.
* **Reflection:** More progress was made than expected, game loading could do with more work but it is functional

**Week 2 (22/06/18)**

* **Goal:** Create any amount of microgames
* **End result:** 1 game was created, but is buggy
* **Reflection:** Not enough progress was made, especially considering the original plan was to create two microgames per week throughout the summer. Hard to get motivation to work over the break, but at least some work was done rather than nothing at all.

**Week 3 (03/08/18)**

* **Goal:** Create any amount of microgames
* **End result:** Fixed microgame started in previous working week
* **Reflection:** Again, not enough progress has been made. Hard to get motivation on the project at this point. Even harder now that I’m busy with other university work.

**Week 4 (30/10/18)**

* **Goal:** Create any amount of microgames
* **End result:** Two games were fully programmed
* **Reflection:** Managed to get back into doing more work on the project, but still could do with more.

**Week 5 (16/11/18)**

* **Goal:** Polish areas of the game made in the prototype
* **End result:** Polished the hint screen and started working on the practice mode
* **Reflection:** Changed the focus of the goal to improving older features, which was mostly accomplished. Could have also done work on creating more games

**Week 6 (16/12/18)**

* **Goal:** Create any amount of microgames
* **End result:** 1 game was fully programmed
* **Reflection:** Again, much more work should have been done on the project and it’s hard to focus on it considering all the other projects being worked on at the same time.

**Week 7 (28/12/18)**

* **Goal:** Create any amount of microgames
* **End result:** 1 game was fully programmed.
* **Reflection:** Again, more work should have been done

**Week 8 (25/01/19)**

* **Goal:** Create any amount of microgames, using a new sensor
* **End result:** 1 game was created, using the microphone
* **Reflection:** Managed to branch out to other sensor types but as always, more work could have been done

# Weekly progress (Project weeks)

This is when the semester started and the project started picking up the pace

**Week 1 (01/02/19)**

* **Goal:** Create any amount of microgames, using a new sensor
* **End result:** 2 games were created, both using the gyroscope
* **Reflection:** More games created than usual, but still far from ideal

**Week 2 (10/02/19)**

* **Goal:** Create 5 games, stretch goal of 10
* **End result:** 5 games were created, older games were polished as well as the menu and some bugfixes were done
* **Reflection:** Much bigger improvement than usual, managed to reach a goal as well as polishing up other areas of the project

**Week 3 (17/02/19)**

* **Goal:** Create 5 games, stretch goal of 10
* **End result:** 5 games were created, implemented the 4th and final sensor
* **Reflection:** Managed to maintain the momentum from last week and implemented the proximity sensor, which is the one which is no already handled by Unity, so it was quite a challenge.

**Week 4 (24/02/19)**

* **Goal:** Create 5 games, stretch goal of 10
* **End result:** 7 games were created, reworked game loading to use scriptable objects rather than an XML file for storing game data. Added saving and loading unlocked games
* **Reflection:** A big improvement again, loads of games added and important features were implemented.

# Project specification

Final project will include:

* Minimum of 50 microgames
* Stretch goal: 80 microgames
* Varied use of four different mobile sensors
  + Touch (tap, multitap, hold, swipe, drag, virtual joystick)
  + Motion (accelerometer, gyroscope, orientation)
  + Exotic sensors (Microphone, proximity)
* Different gameplay modes
  + Practice mode
  + Challenge mode
  + Endless mode
* In-game store (using in-game currency)
* Settings menu
  + Allowed orientations
  + Allowed input methods
  + Accessibility features
  + Clear data
  + Language settings
* Released on Google Play Store

# Potential solutions

Problem: A game does not end up being fun  
Solutions: Remove the game, change the game, leave it.

I believe that the best option here would be to make enough small tweaks to make the minigame be as fun as it can be. Failing that, I would leave the game as it is because I wouldn’t want to remove content in a project based on making a lot of content efficiently.

# Tools and techniques

|  |  |
| --- | --- |
| Tool (choice in bold) | Use |
| **Unity**, Unreal | Game engine |
| **Visual Studio** | IDE |
| **Photoshop**, Krita | 2D art assets |
| **3DS Max**, Blender | 3D art assets |
| **Github**, GitKraken, GitLab | Source control |
| **Trello**, Hacknplan | Project management |

### Game engine

|  |  |  |
| --- | --- | --- |
| Criteria | Unity | Unreal |
| Familiarity and experience | ✓ |  |
| No additional cost | ✓ | ✓ |
| Can build to mobile | ✓ | ✓ |
| Supports touch controls | ✓ | ✓ |
| Supports motion controls | ✓ | ✓ |
| Supports microphone input | ✓ |  |
| Supports proximity input | ✓ |  |

I decided to go with Unity for this because in addition to being the engine I am by far the most familiar with, which was the most important metric, but it also has much better support for the types of games that I want to create.

### 2D art assets

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Photoshop | Krita | GIMP |
| Familiarity and experience | ✓ |  | ✓ |
| No additional cost | ✓ | ✓ | ✓ |

My only metric for this area is familiarity as the project will not be judged on the quality of its art so I don’t want to spend more time than I need to learning potentially better pieces of software.

### 3D art assets

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | 3DS Max | Blender | Maya |
| Familiarity and experience | ✓ |  |  |
| No additional cost | ✓ | ✓ |  |

Again, the only metric here is my own personal experience with the software, because learning new art tools is not part of the project’s scope.

### Source control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | GitHub | GitKraken | GitLab | Bitbucket |
| Familiarity and experience | ✓ |  |  |  |
| No additional cost | ✓ | ✓ | ✓ | ✓ |
| Simple to set up and manage repository | ✓ | ✓ | ✓ | ✓ |

These source control tools all have very similar features and could all be appropriate choices for the project, however I decided to go with GitHub as I have by far the most experience with it and learning new software wouldn’t have any additional benefits.

### Project management

|  |  |  |
| --- | --- | --- |
| Criteria | Trello | Hacknplan |
| Familiarity and experience | ✓ | ✓ |
| No additional cost | ✓ | ✓ |
| Simple to manage | ✓ |  |
| Wide range of project management features |  | ✓ |

I originally started the early stages of the project using Hacknplan as it provided more project management features, but the overall process in Trello is much simpler and is easier to take a glance at and understand how the project has progressed and what areas should be improved on next. Because of this, I decided to switch to Trello.

I have decided on all of these pieces of software for the same reason: I have years of experience with each of them which will allow me to produce work as efficiently as I am able, which is very important for large-scale projects such as this one. I have also previously looked into alternatives numerous times, none of which provide enough benefits to justify learning new software. In short, I know the chosen tools significantly more than others and the years of experience have taught me that I work very well with them, which maximises the efficiency in this project.

# Methodology

The project briefly started back in May where a prototype of the game loading system was created. I had planned on working on the project throughout the summer, but it was difficult to motivate myself for that.

I will be working on as many games as I can manage without burning myself out, aiming for at least one game per day. I don’t have any specific plans for which order games will be implemented as I will be creating whichever game I have the most motivation to work on, to ensure that I work as efficiently as possible.

Fortunately, the project plan is very open, so I can consider each game before implementing it and perform tweaks whenever necessary. Progress will be re-evaluated every week during meetings with the supervisor, where we discuss how the project should continue to move forward and readjust goals and features to be more realistic and viable.

# Management

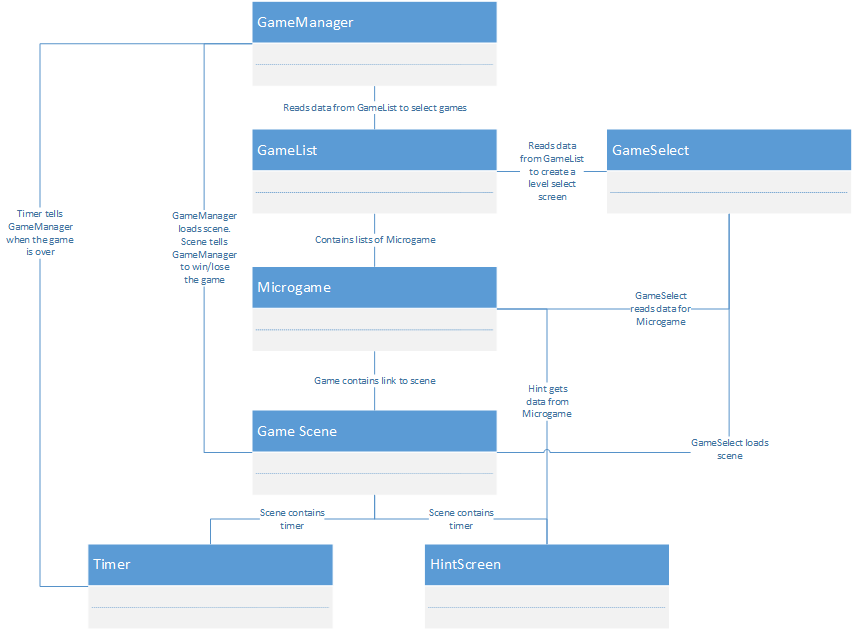
I am using Trello to keep track of the project, where I have a list of all games which have checkboxes to measure their progress. To do this, I am tracking whether each game is functionally complete, complete art assets, complete sound assets, is polished with clear win or loss events and supports the game’s multiple difficulty levels. These conditions will be tested before being marked as complete.

My primary metric is how many games with mostly unique systems have been implemented to a high degree of polish. I have decided on this because my goal is to showcase a wide variety of different gameplay styles and input methods, so while simply having a large number of games is impressive on its own, the game would be much more impressive and enjoyable if each of those games is unique in its own way. Tangentially related, my second most important metric is the entertainment value as I would like the game to be enjoyed by others, so if I or testers don’t find something fun, it should be changed. I have no plans or interest in monetisation, so the game’s market appeal is not something that I will be considering.

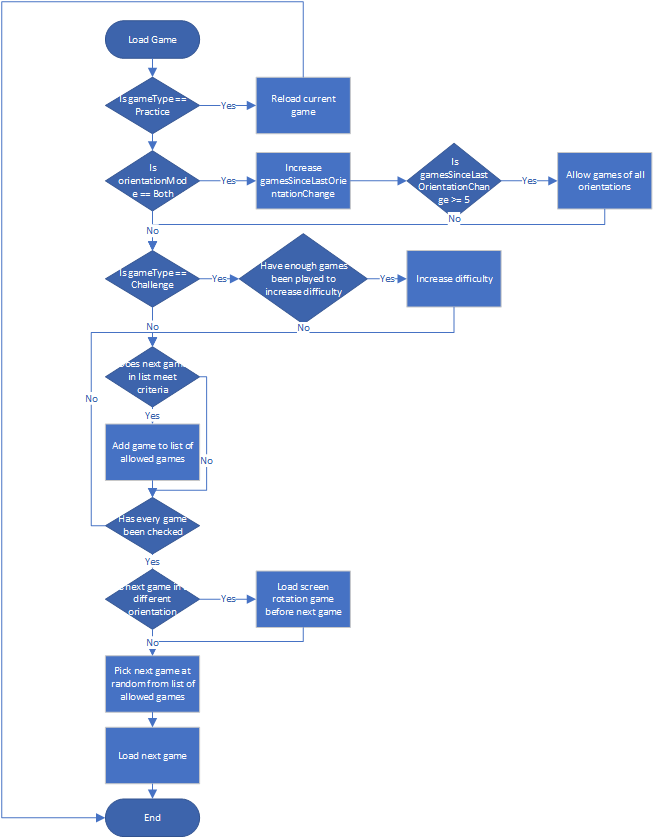
# Resources

The project does not require any equipment that is not already available to me or additional resources beyond people who would be testing the game. To have other people involved with the project, I would need to go through the ethics board for permission.

# High-level overview

  
*Diagram showing how the different classes interact, only showing those that are involved with every microgame.*

# Flowchart

  
*Diagram showing the logic used by the game manager to select which game to load*