Solent University

SCHOOL OF MEDIA ARTS AND TECHNOLOGY

**BSc Computer Games (Software Development)**

**2019**

**Lewis Rosier**

***Final Major Project: Final Report***

Supervisor : Kostas Dokos

Date of submission : May 2019

Contents

[Table of figures 2](#_Toc7462932)

[Overview 3](#_Toc7462933)

[Progress so far 4](#_Toc7462934)

[Research 4](#_Toc7462935)

[Prototype 4](#_Toc7462936)

[Weekly progress (Preparation weeks) 4](#_Toc7462937)

[Weekly progress (Project weeks) 5](#_Toc7462938)

[Project specification 7](#_Toc7462939)

[Potential solutions 8](#_Toc7462940)

[Tools and techniques 9](#_Toc7462941)

[Methodology 11](#_Toc7462942)

[Management 13](#_Toc7462943)

[Resources 14](#_Toc7462944)

[High-level overview 15](#_Toc7462945)

[Flowchart 16](#_Toc7462946)

[Reading List 17](#_Toc7462947)

[Literature Review 17](#_Toc7462948)

[Implementation 18](#_Toc7462949)

[Preparation weeks 19](#_Toc7462950)

[Weekly progress (Project weeks) 22](#_Toc7462951)

[Evaluations, Reflections and Future Development 28](#_Toc7462952)

[Evaluation 28](#_Toc7462953)

[Reflection 28](#_Toc7462954)

[Future development 29](#_Toc7462955)

[Appendix 30](#_Toc7462956)

[1. Microgame design in microgame collections 30](#_Toc7462957)

[2. Microphone input 30](#_Toc7462958)

[3. Proximity input 30](#_Toc7462959)

[Bibliography 31](#_Toc7462960)

# Table of figures

[Figure 1 - How the different classes interact, only showing those that are involved with every microgame 15](#_Toc7305850)

[Figure 2 - The logic used by the game manager to select which game to load 16](#_Toc7305851)

[Figure 3 - "Jumper" sprite sheet 19](#_Toc7305852)

# 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.

# 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.

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Remove game | Change game | Leave it |
| Game as a whole is fun | ✓ | ✓ |  |
| Game as a whole features a large number of microgames |  | ✓ | ✓ |
| Time was spent efficiently |  | ✓ | ✓ |
| No additional time spent on microgame | ✓ |  | ✓ |

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, which would allow me to improve the game, while not cutting down on the number of games. 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. Sometimes a microgame might not fit in very well and wouldn’t work regardless of how it’s changed, so removing the game could still be a good solution to maintain a certain level of quality.

# Tools and techniques

|  |  |
| --- | --- |
| Tool (choice in bold) | Use |
| **Unity** (Unity Technologies, n.d.), Unreal (Epic Games, n.d.) | Game engine |
| **Visual Studio** (Microsoft, n.d.) | IDE |
| **Photoshop** (Adobe, n.d.), Krita (KDE, n.d.), GIMP (GIMP, n.d.) | 2D art assets |
| **3DS Max** (Autodesk, n.d.), Blender (Blender, n.d.), Maya (Autodesk, n.d.) | 3D art assets |
| **Github** (Github, n.d.), GitKraken (Axosoft, n.d.), GitLab (GitLab, n.d.), Bitbucket (Atlassian, n.d.) | Source control |
| **Trello** (Atlassian, n.d.), Hacknplan (Hacknplan, n.d.) | 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.

# 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.

I will be going through the same software development methodology stages for every microgame on its own, as well as the full game as a whole:

**Whole project:**

* **Planning**
  + Consider the project specification
  + How many microgames should there be?
  + What should these microgames be like?
  + How should the game flow from microgame to microgame?
  + What input methods will be used?
  + What game modes should be available?
  + What will keep the players coming back for more?
* **Analysis**
  + Consider what solutions will have to be created
  + Consider how those solutions will be performed
  + Consider how those solutions should interface with the rest of the project
  + How will microgames be loaded?
  + How will the different methods of input be read and handled?
* **Design**
  + Brainstorm a small sample of microgames to act as proof of concepts
  + How will the player navigate the game’s menus?
  + How should the player’s progress be displayed?
  + What additional features could be added to improve the user experience?
* **Implementation**
  + Create game loading system
  + Create foundations for new microgames to easily be added into the project safely and efficiently
  + Start the development cycle for each microgame
* **Testing**
  + Playtest the game to find bugs
  + Playtest the game to come up with new ideas
  + Have others test the game for feedback and improvements
  + Figure out what does and doesn’t work
* **Shipping**
  + Release the game onto the Google Play Store
  + Advertise through social media?
* **Maintenance**
  + Will additional games be added after release?
  + How will these updates work?
  + How frequently will the game be updated?

**Individual microgame:**

* **Planning**
  + What mechanic will the game revolve around?
  + What will the objective be?
  + What input methods will be used?
* **Analysis**
  + What solutions will have to be created?
  + How can the mechanic be implemented?
  + Does the design have to be changed to better fit the scope?
  + Does the design have to be changed to fit into the time limit?
* **Design**
  + How should the game look?
  + How should the objective be clearly signified?
  + How should the mechanic be clearly signified?
  + How should the input be clearly signified?
* **Implementation**
  + Development of the microgame
* **Testing**
  + Playtest the microgame to find bugs
  + Playtest the microgame to find improvements and tweaks
  + Have others test the game for feedback and improvements
  + Consider if the microgame is fit to be included in the project
* **Shipping**
  + Add the microgame to the microgame list
  + Integration testing
* **Maintenance**
  + Does the microgame still need adjustments later down the line?
  + Polish the game if necessary

# 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.

Progress will be measured by how many unique games have been completed, as well as how close to completion they are. Because I am aiming for 50 microgames, each game would be a total of 2% of completion in this area. 50% of each game will be the programming, as that is what takes the longest and has the biggest impact on the end product. The remaining 50% will be divided evenly between the other areas of art, sound, juice (tweaks that add to the game’s presentation to make it look more interesting but don’t change gameplay) and difficulty levels. Everything in the game’s menus will also be part of the metrics, counting for around 10% of the game’s total completion. See below for a table detailing the metrics.

|  |  |
| --- | --- |
| Metric | Percentage (total) |
| Menus | 10% |
| Access all features | 50% (5%) |
| Shop | 15% (1.5%) |
| Settings | 20% (2%) |
| General settings | 50% (1%) |
| Accessibility settings | 50% (1%) |
| Polish | 15% (1.5%) |
| Microgames | 90% |
| Each microgame | 2% (1.8%) |
| Programming | 50% (0.9%) |
| Art | 12.5% (0.45%) |
| Sound | 12.5% (0.45%) |
| Juice | 12.5% (0.45%) |
| Difficulty | 12.5% (0.45%) |

# 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.

The development of this project will require a lot of manpower to achieve an ideal state of the game. As a one-man project, a significant amount of time will have to be dedicated to it each week, as well as maintaining enough motivation to work on it.

The amount of time taken to develop microgames can vary greatly, depending on how many new scripts or assets have to be created. Each game can take around 30 minutes to several hours, so it’s not possible to gauge how much time will be required to complete the project at this current time.

Resources have been planned around not having to spend any additional money to complete the project, so money will not be an issue. An exception to this is the one-time fee to be able to publish onto the Google Play Store, which will only cost $25 at the end of the project.

For hardware, there aren’t any additional resources that are required to complete the project. This includes a PC to develop on and an android device for testing. The game will be tested on various mobile and tablet devices in order to aid with optimisation, all of which are already readily and freely available, either at the university or by having testers use their own devices.

For software, a key metric was that it should ideally not cost any additional resources to use. All software chosen is already available.

# High-level overview

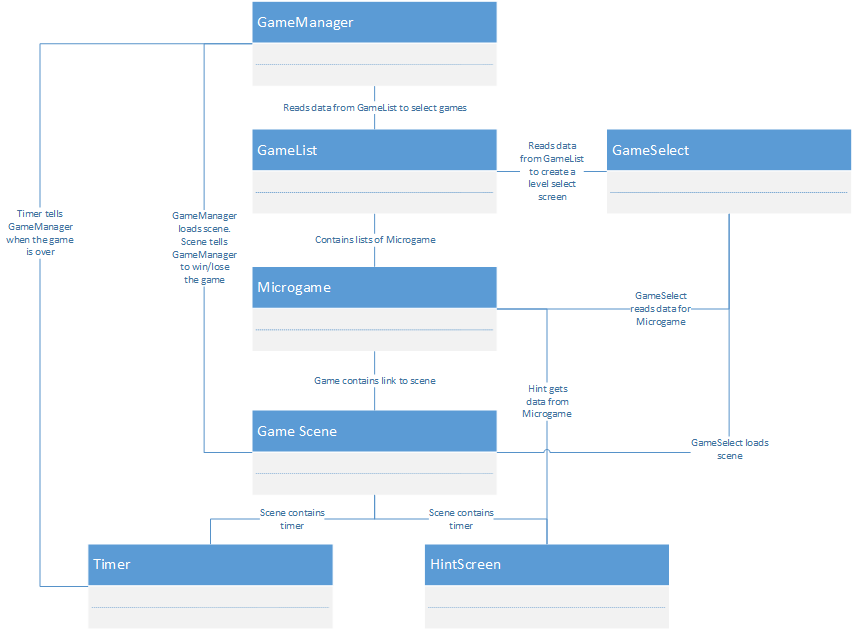


Figure 1 - How the different classes interact, only showing those that are involved with every microgame

# Flowchart

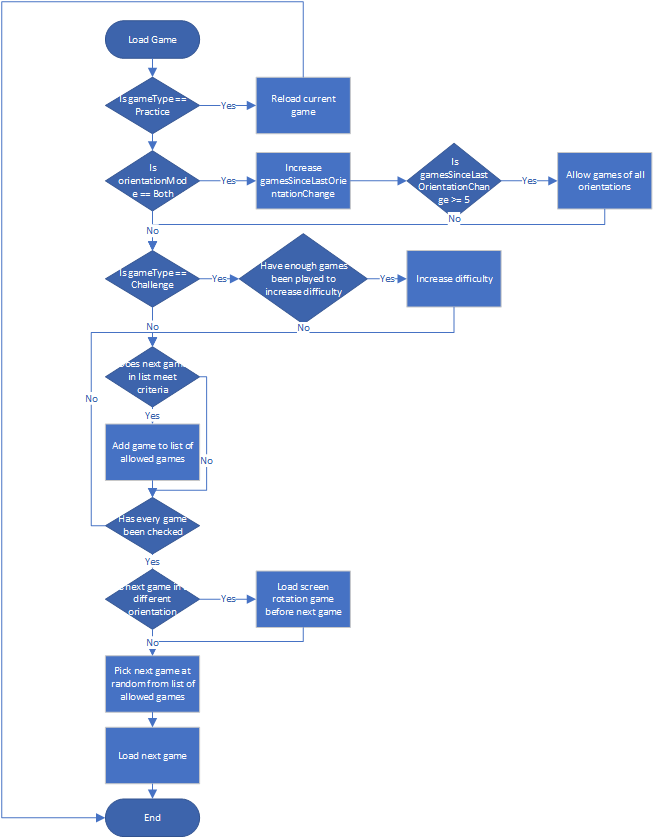


Figure 2 - The logic used by the game manager to select which game to load

# Reading List

* Games
  + Warioware
  + Dumb Ways to Die
  + Mario Artist Polygon Studio
* Techniques
  + Reading microphone input
  + Reading proximity sensor input

# Literature Review

After researching how other microgame collections handle their games (Appendix 1), I found that the key to having an enjoyable experience is plenty of variety and games that are very easy to understand. Adding small details when games are won/lost also adds a lot to the experience.

I researched how to use the sensors that I have no prior experience with (Appendix 2, 3) and found that it is quite simple to get them both working well.

# Implementation

## Preparation weeks

These preparation weeks took place before the semester and, by extension, the project officially started. Work done during this time was significantly slower so that the design could have more room to take its shape and allow for more planning, rather than quickly solidifying the format which would make it difficult to make changes.

**Week 1 (Week beginning 21/05/18)**

Work on the project had started shortly after the end of the second year of the course. The objective of this sprint was to plan and develop a very basic prototype of the microgame loading system, which would be the basis for most of the game’s gameplay. This was achieved, creating a system that randomly selected scenes to load based on multiple variables that controlled which microgames could be loaded. These variables are the different options the player can select to determine which sensors and screen orientations they would like to use, as well as behind-the-scenes decisions such as whether the orientation switching games could be loaded.

If the player selects both orientations to be usable, then the loader will pick games based on the current game’s orientation, only selecting different orientations if enough games have been played since the last change, at which point it will load a short orientation switching microgame before the next proper microgame. The loader could also not select the same game twice in a row to prevent repetition.

Data for the microgames has been stored in a local XML file, which is read by the game manager to help it select and load microgames. The XML file is read and the various attributes are used to create a Game object.

Basic art assets for a character and some basic scenery was also created during this time. The character is a very simple sprite that is made in greyscale so that its colours can be changed freely in the engine. This character, the Jumper, is used heavily throughout the game due to its versatility.



Figure 3 - "Jumper" sprite sheet

The first microgame, Platformer, was also created here. In it, the player is tasked with controlling a 2D platformer game and must jump on the head of an enemy, an action seen in most 2D platformers. The game was fully programmed and graphically complete.

The timer bar and pop-up hint system was implemented. These are included in each game, displaying a short game-specific instruction like “Stomp!” in front of a background image for a short period before sliding away to reveal the game. The timer is at the top of the game screen and ticks down either to 0 or until the player completes the game, at which point the background image will slide back onto the screen to create a seamless transition between each game.

**IMAGE OF OLD HINT AND TIMER HERE**

The game manager was mostly used to keep track of options and handle game loading, but it also had a lives system which was used to give the player a certain number of chances at clearing the games before the game ends.

Finally, a very basic main menu was also implemented, only featuring a button to start playing the game, as well as a results screen to display how many games had been cleared, which would be loaded when the player runs out of lives. This menu also shows the player how much in-game currency was accumulated and add it to their total, which is saved to the device.

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

The main menu was added to include a series of options, as well as smooth scrolling transitions between the different menu screens.

Work on a second game started, Fight Monster, which has the player tap a monster to command a group of four adventurers to attack it. When the monster runs out of health, it dies and the player wins the game.

This game was created using a tool that was intended to speed up the development of tap input games by acting as a controller, reading inputs and then finding and calling appropriate functions in other scripts to pass through the data, with more features being added later as they’re required for new games. Ideally, this would have worked well as a simple drag-and-drop system for various methods of input, but the added layer of abstraction meant that it had less functionality and was slightly more resource-intensive than using bespoke code for each game, so the tool was only used in one other microgame.

This feature had some bugs which prevented it and the microgame itself from being finished.

**Week 3 (30/07/18)**

The monster fighting microgame and a portrait version was finished, including all graphics implemented.

**Week 4 (29/10/18)**

Two games were created during this sprint, Pet Dog and Divert Traffic.

Pet Dog was the other game to use the scrapped touch input system, with rubbing detection added for this game, having the player pet a dog quickly enough in the time frame. This game was completed, including full graphics and sound implementation.

Divert Traffic has characters that move constantly and bounce off walls and the player must move them to prevent them from bumping into each other. There was a problem where the player would “let go” of the characters if they move to fast, so the dragging was changed to move the character to the touch position until the touch ends, not just when the character itself is being touched. This game was completed with full graphics implemented.

2D water graphics were also implemented to be used in future games. This used a custom shader that was found on GitHub (**REFERENCE**), but the rest of the water was created by me.

**Week 5 (12/11/18)**

This sprint focused on polishing prototype features, rather than creating new game content.

The hint screen was changed to load hint data from the microgame XML file, rather than simply having the text be edited in the scene. This allowed the prefab to be changed without affecting the hint line, which would be a massive issue later on when there are significantly more microgames available.

The hint screen also loads the sensor data for the microgames to display which sensors the player would have to use in a new panel at the bottom of the screen.

Practice mode was implemented with a new section of the menu that displays all microgames as a list of icons that could be clicked to repeatedly play a game until the player loses once. These icons were loaded using the game’s name to generate a path in the Resources folder.

**IMAGE OF OLD PRACTICE MODE HERE**

**Week 6 (10/12/18)**

The Roll Toilet Paper microgame was created, which had players repeatedly swipe down to unroll some toilet paper. This game was fully programmed and all graphics were implemented.

Work on a second game started, Platformer Worlds, which had players jump between planets that the character orbits around. This game was scrapped because the gameplay didn’t fit with the structure of the game, but the code remains so it can later be redesigned if needed.

**Week 7 (24/12/18)**

Two games were created during this sprint, Make it Rain and Fill Glass.

Make it Rain is similar to Roll Toilet Paper, but the player swipes up to throw money. This game was fully programmed, but not including graphics.

Fill Glass uses the water that was created in week 4 and is the first game to use a different sensor: the gyroscope. Unity has gyroscope support built in, so nothing extra was required to use it here. This game has the player rotate the phone to control a glass so they can collect drops of water.

**Week 8 (21/01/19)**

Created Blow Boat, which has players blow into the microphone to push a paper boat towards a goal.

While Unity does has microphone support by default, using it as inputs for games required some additional steps. First, the default recording device is found and a recording starts for a specified amount of time. Then on every frame, the current position of the recording is found and then the volume of that point is stored to a public variable. This point is between silence at 0 and peaking at 1. I created a tool that does this on its own and can be easily accessed by other game elements to use the current volume of the microphone as a controller for games.

## Weekly progress (Project weeks)

These sprints take place during the semester where this unit takes place. As such, work is done at a much higher rate and supervisor feedback is provided and taken into consideration to refine the game.

**Week 1 (28/01/19)**

Two games were created in this sprint: Combine Robots and Balance Plates. Both games make use of a generic steering script which uses movement and rotation values for very basic steering-like gameplay. Like before, this makes use of Unity’s built-in gyroscope tools.

Combine Robots has players steer a ship into three moving robot parts to connect to them, winning if they touch all three and exploding if they fail. Balance Plates has players control a stick that balances a stack of plates with a ball on top and have to use physics to prevent the ball from falling.

In this first supervisor meeting, a plan was created that, if followed, would allow for enough time in the last month of the project to polish all of the games to a high standard while also being able to work on extra features and the report. All 50 microgames must be fully programmed before the week beginning 25/03/19, then they must be polished by the week beginning 15/04/19, then the rest of the time will be spent heavily testing the game and working on the report.

Throughout the earlier sprints, the schedule changed to be increasingly busier in order to complete each milestone in time. Later on when speed picked up, the schedule stopped changing as enough work was being done to satisfy them. Below is a table detailing this final weekly schedule:

|  |  |  |
| --- | --- | --- |
| Day | Scheduled work (development milestone) | Scheduled work (polishing milestone) |
| Monday | 1 game created | 2 games polished |
| Tuesday | 2 games created | 4 games polished |
| Wednesday | 1 game created | 2 games polished |
| Thursday | 2 games created | 4 games polished |
| Friday | 2 games created | 4 games polished |
| Saturday | Catch up if needed, work on refining other features | |
| Sunday |

The third milestone did not have a strict schedule as the amount of work varied greatly as new bugs and changes were found, so they were dealt with as soon as possible so that there was enough time to allow for more potential work.

**Week 2 (4/02/19)**

Four games were created in this sprint: Plug Into Mains, Draw Path, Herd Cattle and Draw Platform. All of these games make use of a “lock and key” system, which is used to mark games as complete when key and lock objects collide if being paired with the same ID. The two drawing games use a drawing system, which tracks touch positions to add points to an array and to a line renderer to simulate the act of drawing a shape on the phone with your finger. These points can be used to make a series of points for objects to follow or to give the line collision for objects to interact with.

Plug Into Mains has players match plugs and sockets that are randomly placed on the screen by dragging them. Draw Path shows players a series of randomly placed walls that they must draw a path through to guide a character to the other side. Herd Cattle has players drag waves of cattle as they move towards pens, failing the game if they let cattle into the wrong pen. Draw Platform has players draw a series of platforms for a ball to roll across to reach a randomly placed target.

The menu also saw updates. The menu received a new layout that uses tabs at the bottom of the screen to navigate and the practice mode screen now uses a popup dialog which shows information on the game such as its name, a screenshot and what sensors are used.

Tweaks were also made to previously finished games.

**Week 3 (11/02/19)**

Five games were created in this sprint: Wave to Person, Launch Person, Zoom to Find, Infinite Jumping and Infinite Falling. The first of these, Wave to Person, uses the phone’s proximity sensor to detect whether something is close enough to the phone to simulate very rudimentary gesture controls. Unity does not include support for this sensor, but it can be used via a plugin called PAProximity (**REFERENCE**), which I created a wrapper for so it can be more easily integrated into the microgames. This plugin works by creating an AndroidJavaObject which is used as a middleman between the game and the phone, which can send and receive calls to different parts of the phone, used here for the proximity sensor.

Wave to Person uses the Boolean information from the proximity sensor to detect waving motions in front of the phone, so it can simulate the player waving to an in-game character. Launch Person has the player drag and drop a rock onto a seesaw, which if placed correctly will send the character standing on it flying off of the screen, completing the game. Zoom to Find generates a very simple planet with one character and many randomly scaled buildings, all given random positions, and the player must swipe to orbit, use two fingers to rotate and pinch to zoom so they can find the character within the time limit. Infinite Jumping and Infinite Falling both use the same code in different levels with different objectives to create similar, yet very different games. Players rotate their phone to steer a character who is constantly jumping so they can navigate platforms so they can reach the top or the bottom of the level respectively to win the game.

Tweaks were also made to previous games.

**Week 4 (18/02/19)**

In this sprint, the game loading was reworked to make use of scriptable objects rather than a single XML file. This is because the XML had to be edited in a different program and the increasing amount of games made edits somewhat difficult. Using scriptable objects instead allows the game data to be edited within Unity and for referencing variables such as icons to be added, instead of using hardcoded file addresses within the Resources folder. Games also now start off locked and must be completed in one of the other game modes to unlock. Saving and loading was also updated to include this, now using a binary formatter to store data in a single file.

Five games were made in this sprint: Build Bridge, Jump and Draw, Fire Cannons, Shush and Pick Nose. Build Bridge has players drag a small bridge over holes so a character can safely make it to the other side of the screen, otherwise they fall down a hole and lose the game. Jump and Draw is the first microgame to make use of two sensors, touch and motion. The gameplay is similar again to the two jumping games from the previous sprint, but also incorporates drawing so the player must create their own platforms while also trying to avoid spikes, which will kill them and end the game. The spikes were not originally included, but after testing the game was found to be far too easy with no obstacles to avoid.

Fire Cannons uses the microphone to command a ship, shouting a command to fire the cannons at an enemy ship. Shush uses both the touch screen and microphone to have players place their fingers against a talking character’s lips and then “shush” into the microphone. Pick Nose is very simple and has players drag a finger around to push it into a nostril.

Asteroid Shooter was also developed, which would use the gyroscope to steer a ship and shoots using the touch screen. The objective was to safely guide the ship through an asteroid field. This game was scrapped because it did not fit in with the scope of the microgames.

Another game, Dating Sim, was also started in this period. In it, the player would select the correct options in a visual novel style game to confess to another character. This was scrapped as the game was too text-heavy for the short time limit of each game and would also revolve a level of guesswork on part of the player, rather than skill.

**Week 5 (25/02/19)**

Only one game was created in this sprint as more time was spent writing the progress report. Input Code has players enter a code into a keypad. The code is randomly generated but is on-screen for the player to copy. Instead of using UI buttons, the player guides a finger to make it move to and push each button.

**Week 6 (04/03/19)**

Eight games were created in this sprint: Roll Ball, Tanning, Steer Car, Can Crush, Scare, Inflate Balloon, Peekaboo and Interrupt. During the supervisor meeting it was decided that the current selection of microgames was too similar input-wise so work on games that use only the touch sensor was put aside until there was a more diverse range of inputs. The final game will include 30 touch games, 10 motion games, 5 microphone games and 5 proximity games. This will give the game a fair balance of games based on their versatility as an input device.

Roll Ball has players tilt their phone to control a board that has a randomly placed hole and their objective is to use gravity to roll the ball into the hole. Like previous motion games, this uses the gravity of the gyroscope. Tanning has players block the sun with the proximity sensor to help a character achieve the perfect tan. Steer Car has players rotate their phone to steer a car to navigate around traffic, winning if they manage to not crash into another car. Can Crush has players crush am empty can using the proximity sensor, then toss it away with the touch screen.

Scare uses the accelerometer to create a camera that moves one-to-one with the player’s phone, simulating the player’s own perspective in the game. Players must look at moving characters and use the microphone to scare the character away. Inflate Balloon uses the proximity sensor to pump a balloon, inflating it gradually until it pops, completing the game. Peekaboo has the player use the proximity sensor to play peekaboo with a child, covering the phone and then uncovering it to win. Interrupt has players wait for a character to speak so they can use the microphone to interrupt the character, winning the game.

**Week 7 (11/03/19)**

Another eight games were created in this sprint: Cut Rope, Sneak and Steal, Plug Holes, Vocal Lesson, Stacking, Fly Swatting, Rock Paper Scissors and Skydive. Now that the game had a more diverse usage of sensors, we decided that the game should also involve wider usage of the touch sensor as most games currently used dragging inputs.

Cut rope has players cut a swinging rope to save a character, but they must cut it at the correct time otherwise the character will fall into the ocean. Sneak and Steal is similar to moving statues, having the player touch the screen to move a character towards a point, but will fail if they move while a guard is facing them. Plug Holes has randomly placed holes that the player must keep their finger on, with a new hole appearing when the player touches the newest hole. All holes must be held by the player at the same time to win. Vocal Lesson is a rhythm game where players hold the screen to sing and must match the notes that fly onto the screen.

Stacking involves moving a platformer character using on-screen buttons to jump on another character’s head, which will stack the characters into a tower. The player wins when all characters are in the stack. Fly Swatting has flies quickly moving around the screen and the player must tap them to swat, winning once all flies have been killed. Rock Paper Scissors places the player against an opponent who has thrown one of the three moves and the player must select the hand that will win the game. Skydive uses the gyroscope to move a skydiver left and right, with the objective being to pass through three rings.

**Week 8 (18/03/19)**

The last seven games were created in this sprint: Zoom and Enhance, Unlock Chest, Land Ship, Type on Keyboard, Pick Up Sushi, Draw Platformer Level and Shoot Ducks.

Zoom and Enhance uses pinch controls to zoom, then drag a slider to “enhance” the image, increasing the rendering resolution. This effect was achieved by updating and displaying a render texture, with the resolution being set based on the enhance percentage between a minimum and maximum resolution. Unlock Chest is another platformer type game where players must pick up a key and then run up to a chest to open it. Land Ship uses both touch and gyro controls to accelerate and steer a rocket so it can safely land on the landing zone. Type on Keyboard has players repeatedly tap a keyboard to type a text message. The keyboard is not functional and the messages are selected from a list of premade messages, so that the player doesn’t have to focus on typing a coherent message in five seconds.

Pick Up Sushi has a hand move side to side and an endless stream of sushi moving on a conveyor belt and the player must tap the screen to make an attempt at picking one up. Draw Platformer Level uses both the platformer character and the drawing tool to make players quickly draw a bridge between two distant platforms, then switch to the character controller to move them to the other side. Shoot Ducks has ducks fly out from behind grass at random angles and the player must quickly tap them to shoot all of the ducks.

As all 50 microgames have now been completed, the next three weeks will be spent polishing games by applying graphics and sound to all games, as well as adding appropriate visual touches where necessary.

Another game, Pick Up with Chopsticks, was started, where the player would use pinch inputs to pick up an object using chopsticks and put it in a container. Unfortunately, the act of pinching to pick something up was uncomfortable and inconsistent so the game was scrapped. The idea of pinching to interact was then reused in a game called Honk Horn, where the player would be shown a horn given a random rotation and had to pinch at the correct angle to honk it. Despite being a larger scale, the controls were still too uncomfortable so Zoom and Enhance was created as a way to incorporate more games that use pinch inputs but used in a more natural zooming motion.

**Week 9 (25/03/19)**

17 games were polished in this sprint: Build Bridge, Cut Rope, Draw Path, Divert Traffic, Draw Platform, Draw Platformer Level, Fight Monster Horizontal, Fight Monster Vertical, Crush Can, Blow Boat, Fly Swatting, Herd Cattle, Input Code, Launch Person, Make it Rain, Move Hole and Inflate Balloon.

Many art and sound assets for these games were retrieved from online sources and a full asset list can be found in **APPENDIX**.

**Week 10 (01/04/19)**

15 games were polished in this sprint: Fire Cannons, Balance Plates, Tanning, Infinite Jumper, Infinite Faller, Jump and Draw, Platformer, Stacking, Unlock Chest, Roll Ball, Steer Car, Roll Toilet Paper, Shoot Ducks, Sneak and Steal and Rock Paper Scissors.

The orientation switching scenes were also polished to include icons that represent the orientation to switch to as well as some text to better signify what the player has to do, while also being more visually appealing.

**Week 11 (08/04/18)**

The goal of this sprint beyond finishing polishing the microgames was to work on more additional features so the game can show off more advanced techniques that would not be applicable outside of microgames. Work on the shop system started, which allowed players to spend their in-game money on items which will adjust the way they play the game. These items are additional lives, game speed modifiers and earned money multipliers. Purchasing these items will add them to the player’s inventory, where they can then be used to modify the next playthrough, although this feature will be completed next week so more time can be spent finishing the microgames.

18 games were polished this week: Zoom to Find, Zoom and Enhance, Type on Keyboard, Pick Up Sushi, Wave to Person, Scare, Land Ship, Skydive, Combine Robots, Fill Glass, Vocal Training, Interrupt, Plug Into Mains, Input Code, Pick Nose, Shush, Plug Holes and Peekaboo.

Other microgames from previous sprints saw further adjustments to gameplay and polish, including bugfixes.

**Week 12 (15/04/18)**

The remainder of the project will be focused on implementing additional technical features to the game and testing the game with an additional tester to find every possible area where the game can be improved as well as testing the game on multiple devices. Another reason why the technical features are being developed now is because some of them require testing on multiple devices, so it is a better use of time to do all testing now.

This testing was carried out by simply playing the game and noting down any improvements that could be made, while also discussing possible solutions to these problems and other useful quality-of-life features.

The menu saw a graphical overhaul, replacing all default Unity fonts and sprites to give the game a more personalised look. The shop and inventory system was finished, including adding all the information to the user’s save file.

Online database functionality was added. The game connects to a remote MySQL server via PHP to securely transfer data. Users register when they start the game and have an account set up for them. They can use the profile edit window to edit their name, select an avatar and customise its colour, which will show in the global leaderboards. Users can browse through pages of the leaderboards to look at other players scores, with their own score and ranking information being shown at the top of the screen. This implementation is very simple but has a lot of room for expansion if needed in future versions of the game.

At first, the game would only connect to the database on a very inconsistent amount of devices. After some research, this problem was solved by changing the build system from the default Gradle to Unity’s internal build system as well as changing the way the server handles different user agents. After that change, however, there were no more issues with the online.

One of the suggestions from the tester was to include quick alert-style hints at the start of some microgames to draw the player’s attention to the key object of the game that they would have to interact with to win. These are very easily added into microgames and can be disabled in the options menu.

An earlier suggestion from the project supervisor was to include a tutorial of sorts to help ease players into using the more exotic sensors. This was also suggested by the tester. A small scene was created that can be accessed from the options menu which lets users play around with each of the sensors without a time limit or objective. From testing, we found that some devices don’t play microphone games as well due to the sensitivity of the microphone, so a calibrator was added to this scene which takes the loudest volume from the microphone and uses that as a multiplier for the microphone manager.

A pause menu was added to the microgames, which allows players to quit the game whenever they want, instead of leaving the game alone to run out of lives. Some games didn’t pause properly, so they were all tested and fixed to prevent pause issues from occurring.

The ability to buy games was added. This uses a similar interface to the shop menu to allow players to purchase microgames that they haven’t yet unlocked from regular play.

Because difficulty modes were scrapped for reasons explained in the evaluation, the endless and challenge game modes were adjusted to accommodate. Endless now runs on a flat speed, while challenge mode increases in speed at a rate of 1.1x for every five games cleared.

In addition to these features, many microgames went through adjustments and bugfixes based on the feedback from the testing sessions. The main UI had also seen improvement to better support different aspect ratios.

**Week 13 (22/04/18)**

This sprint continues the same goals as the previous, adding extra features and implementing bugfixes.

A language setting was added to the options menu which allows the game to be translated into different languages and for the game to be able to dynamically switch between them. This works by reading lines from an XML file and using line IDs to find the correct text to display. Scripts attached to text objects have these IDs stored and can be called en-masse at any time by the language manager to update text to fit the newly selected language.

Font settings were added to help those who have difficulty reading the default fonts by letting them select fonts that have been designed with readability in mind, based on a list of fonts that are known to help with learning difficulties such as dyslexia. This works by, on level load or on command, finding all Text objects including those that are inactive and replacing the font and multiplying the font size by a scalar value, which is set up on a per-font basis to accommodate for the nuances of different fonts. Default values are also stored so the text can easily be set back to default.

Another accessibility feature added was the ability to change the duration of hints, which can also help those with difficulty reading for various causes. The options menu also saw tweaks to better accommodate for the now much larger list of settings and the possibility of more.

In addition to these features, more microgames and the UI had bugfixes and polish, as well as background music being added with one track for menus and another for gameplay.

**Week 14 (29/04/18)**

# Evaluations, Reflections and Future Development

## Evaluation

The project’s end goal has managed to be achieved. The game is a fully functional microgame collection that features a wide variety of different games using many different mobile sensors and can act as another portfolio piece in addition to the others created throughout the course.

This was verified by comparing the features of the game with those listed in the project specification, as well as feedback from players who said that the games were different enough from one another to stand out on their own.

Two objectives were scrapped from the original plan. Originally, each game was going to have three different difficulty levels, which challenged the player with harder alternatives of the game or simply a more challenging amount of time to complete them in. This feature was eventually scrapped as the games were made with a specific challenge in mind, meaning that the different difficulties would not add much more value to the game, as they would mostly just involve a value being changed. Implementing these additional difficulties for 50 games would also be very time consuming, especially during testing where three times the amount of games would have to be balanced, which would be a lot of time wasted that could have otherwise been spent on more interesting features or higher quality games.

Language settings were implemented, but only the menu was translated to showcase that the feature was working as intended because a full translation would take too much time that could be spent better elsewhere, as well as not being readable for the submission. The translation itself is also outside of this field, so it wasn’t appropriate to have a full translation.

## Reflection

Overall, I’d say that the project was a success. Most importantly, the target of 50 polished microgames was reached, which was by far the most daunting of the tasks. The games all work as intended and fit the scope of a microgame and are distinct. The tools created for the project, the language and font changing systems, work very well while being lightweight, so they could be uploaded to the Unity Asset Store on their own for others to use in their projects.

The biggest flaw I had in development was time usage. I started working on the project at the end of May 2018, several months before the next semester, and had planned to work on the game over the summer. Unfortunately, it was hard to motivate myself to work on the project so early, which lead to doing more work in a shorter amount of time. If I had managed to work over the summer, I could have reached the stretch goal of 80 microgames by the end of the project.

Throughout this project I learnt a lot more about mobile development. I was able to improve my development process and techniques to create a fully functional game, using techniques that were completely new to me, such as the various input methods, as well as those I knew before but was able to refine over the long duration of the project.

Some of the designs from earlier on in the project were changed slightly. Microgame data was stored in an XML document but was later changed to use scriptable objects. This was a smooth change, but the project could have been better planned from the start, with more research into development techniques that could be used in the game, rather than sticking to what I already know.

Going back to focus more on development tools would have also been very helpful. Given proper planning, design and implementation, a single system which handles mobile inputs could have been incredibly helpful. Upon retrospect, the issue with the touch input system which caused it to go unused could have been solved by passing through the Touch object to provide all the touch information that could have been needed. This was not done at the time as it was my first time creating a system that works like this so the thought never occurred, but if I had spent more time planning that feature then it could have seen significant use and sped up development time.

## Future development

Due to the open-ended nature of the game, it will be very easy to expand upon it after release. The most obvious would be to add new microgames, further extending the variety that the game offers. New modes can be added to spice up the game some more, such as a mode where the microgames are stripped of their time limit to offer a different type of challenge, such as seeing how long you can survive in the balancing game before failing or aiming to get points in each game. Some games like the platformer could even be expanded to include more content, by adding a mode where players go through various levels, turning it into a game in itself. These additions could also be included in the leaderboards, adding new areas for players to compete in. Creating more content for the game after release would not only be a great way of keeping players coming back for more, but can also be used as a platform for development prototypes that are created between larger projects, or simply ways to practice new ideas.

# Appendix

## Microgame design in microgame collections

### WarioWare (Nintendo, 2003)

Microgames in WarioWare are all very simple, easy to read and over in a flash. Each game starts by displaying a short hint on-screen, something that can be read and understood instantly, then the game is played for four seconds at the default speed.

Game speed is measured in beats as the game flows in tandem with the music. Games will last 8 beats, the real-time duration of which will decrease as the game and music gets faster.

Every game features one mechanic, with many being simple reskins of others. WarioWare also boasts a massive collection with the latest in the series, WarioWare Gold (Nintendo, 2018), featuring over 300 microgames, as the series has been around since 2003 and games tend to feature microgames from previous instalments. This allows the game to be familiar to fans of the series while also providing a lot of content for newer players, keeping the game enjoyable for a long time.

### Mario Artist Polygon Studio (Nintendo, 2000)

The WarioWare format can first be found in a Japan-only 64DD game called Polygon Studio, which has a section that contains a small handful of microgames, dubbed the Sound Bomber mode, that are presented in a formula that is identical to the later WarioWare series. It doesn’t have anything that can’t also be seen in these later titles, but the difference in difficulty and scope shows that the team focused on creating a large variety of games, all of which rely purely on skill and not just random chance, as is the case in two of Polygon Studio’s eight microgames.

### Dumb Ways to Die (J. Frost, S. Baird, 2013)

This game has a very similar format to WarioWare, but a key element of all of its microgames is how they will always end with an animation, showing the aftermath of a loss or a victory. The presentation is of a high quality, but the number of games is quite low, so it can quickly become repetitive.

## Microphone input

Unity has microphone support built in, although it’s not as straightforward as touch or motion input. However, it does all work with microphones on both the desktop and mobile, so no further code is required to work with different platforms.

Using the built-in *Microphone* class (Unity Technologies, n.d.), you can find devices using *Microphone.devices*, which returns a *string* array representing the device names of the connected microphones. You can then use *Microphone.Start()* to start recording from one of these devices, which can be stored as an *AudioClip*. From that, all information can be gathered from this clip in a similar fashion to any other *AudioClip*.

## Proximity input

Unity does not have any built-in support for the proximity sensor, so there is very little information available on getting it to work. It is possible, however, using the PA Proximity (Popup Asylum, n.d.) plugin and using the *PAProximity.Proximity* enum to determine whether or not something is within range of the proximity sensor, then simply using that information to create different games.

# Bibliography

Adobe, Adobe Photoshop CC. *Adobe Captivate-Unlock The Future of Smart eLearning Design*. Available at: https://www.adobe.com/uk/products/photoshop.html [Accessed February 27, 2019].

Atlassian, The Git solution for professional teams. *khansari / LASAHandwritingDataset - Bitbucket*. Available at: https://bitbucket.org/ [Accessed February 27, 2019].

Atlassian, Trello lets you work more collaboratively and get more done. *Feature Deep Dive | Getting Started with Trello*. Available at: https://trello.com/en [Accessed February 27, 2019].

Autodesk, 3D modelling, animation and rendering software. *Autodesk 2D and 3D Design and Engineering Software*. Available at: https://www.autodesk.co.uk/products/3ds-max/overview [Accessed February 27, 2019].

Autodesk, Make it with Maya computer animation software. *Autodesk 2D and 3D Design and Engineering Software*. Available at: https://www.autodesk.co.uk/products/maya/overview [Accessed February 27, 2019].

Axosoft, Git Client - Glo Boards | GitKraken. *GitKraken.com*. Available at: https://www.gitkraken.com/ [Accessed February 27, 2019].

Blender, Home of the Blender project - Free and Open 3D Creation Software. *blender.org*. Available at: https://www.blender.org/ [Accessed February 27, 2019].

Epic Games, Make Something Unreal. *What is Unreal Engine 4*. Available at: https://www.unrealengine.com/en-US/what-is-unreal-engine-4 [Accessed February 27, 2019].

GIMP. *GIMP*. Available at: https://www.gimp.org/ [Accessed February 27, 2019].

Github, Build software better, together. *GitHub*. Available at: https://github.com/ [Accessed February 27, 2019].

GitLab, The first single application for the entire DevOps lifecycle. *GitLab*. Available at: https://about.gitlab.com/ [Accessed February 27, 2019].

Hacknplan, Project management for game development. *HacknPlan*. Available at: https://hacknplan.com/ [Accessed February 27, 2019].

*J. Frost, S. Baird*. *Dumb Ways to Die* [video game]. Metro Trains Melbourne

KDE, *Krita History Comments*. Available at: https://krita.org/en/ [Accessed February 27, 2019].

Microsoft, Visual Studio IDE, Code Editor, VSTS, & App Center. *Visual Studio*. Available at: https://visualstudio.microsoft.com/ [Accessed February 27, 2019].

*Nintendo*, 2000. *Mario Artist Polygon Studio* [video game]. Nintendo

*Nintendo*, 2003. *WarioWare* [video game]. Nintendo

*Nintendo*, 2018. *WarioWare Gold* [video game]. Nintendo

Popup Asylum, PA Proximity. *Unity Asset Store - The Best Assets for Game Making*. Available at: https://assetstore.unity.com/packages/tools/integration/pa-proximity-25685 [Accessed February 28, 2019].

Unity Technologies. *Unity*. Available at: https://unity3d.com/ [Accessed February 27, 2019].

Unity Technologies, Microphone. *Unity - Manual: State Machine Basics*. Available at: https://docs.unity3d.com/ScriptReference/Microphone.html [Accessed February 28, 2019].