Solent University

SCHOOL OF MEDIA ARTS AND TECHNOLOGY

**BSc Computer Games (Software Development)**

**2019**

**Lewis Rosier**

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

Supervisor : Kostas Dokos

Date of submission : March 2019

Contents

[Table of figures 2](#_Toc2312271)

[Overview 3](#_Toc2312272)

[Progress so far 4](#_Toc2312273)

[Research 4](#_Toc2312274)

[Prototype 4](#_Toc2312275)

[Weekly progress (Preparation weeks) 4](#_Toc2312276)

[Weekly progress (Project weeks) 5](#_Toc2312277)

[Project specification 7](#_Toc2312278)

[Potential solutions 8](#_Toc2312279)

[Tools and techniques 9](#_Toc2312280)

[Methodology 11](#_Toc2312281)

[Management 13](#_Toc2312282)

[Resources 14](#_Toc2312283)

[High-level overview 15](#_Toc2312284)

[Flowchart 16](#_Toc2312285)

[Reading List 17](#_Toc2312286)

[Literature Review 17](#_Toc2312287)

[Appendix 17](#_Toc2312288)

[1. Microgame design in microgame collections 17](#_Toc2312289)

[2. Microphone input 18](#_Toc2312290)

[3. Proximity input 18](#_Toc2312291)

[Bibliography 18](#_Toc2312292)

# Table of figures

[Figure 1 - How the different classes interact, only showing those that are involved with every microgame 10](#_Toc2127799)

[Figure 2 - The logic used by the game manager to select which game to load 11](#_Toc2127800)

# 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

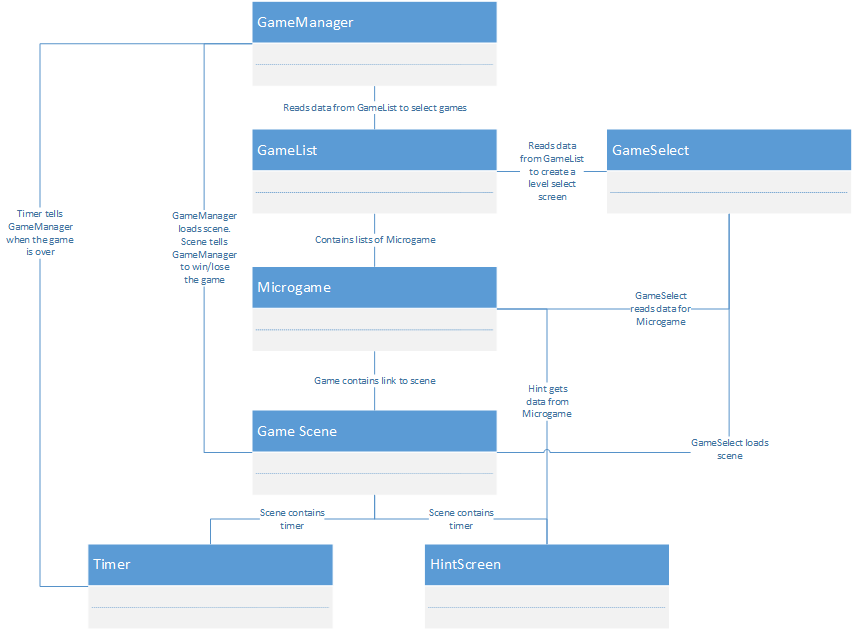


Figure 1 - How the different classes interact, only showing those that are involved with every microgame

# Flowchart

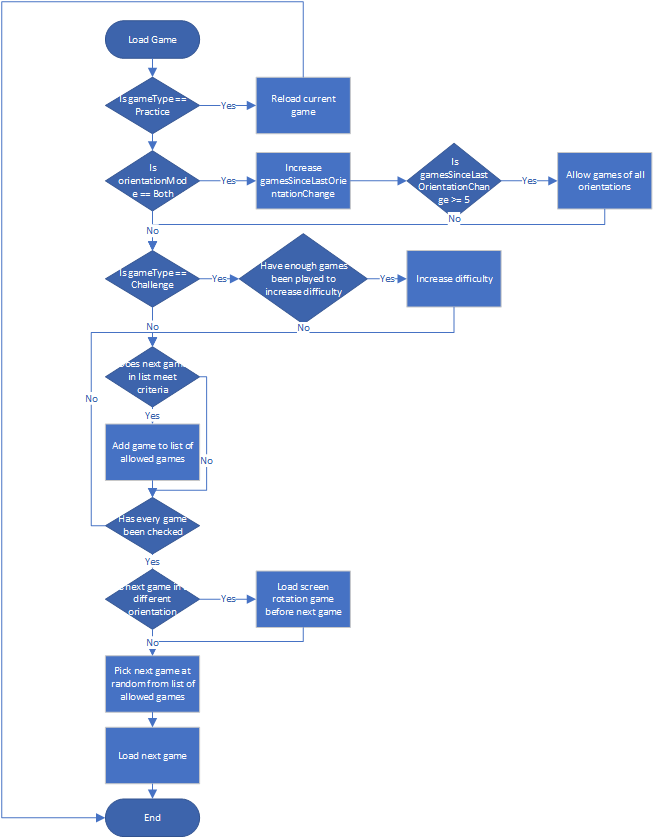


Figure 2 - The logic used by the game manager to select which game to load

# 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.

# 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].