Gesture Based UI

By: Rachel McClelland

ID: G00337231

And

Niall McCann

ID: G00344474

GitHub Link: https://github.com/rachelmcclelland/GestureBasedUI-Project

Lecturer: Damien Costello

Table of Contents

[Purpose of the application 3](#_Toc37193996)

[Menu Screen 3](#_Toc37193997)

[Options Screen 3](#_Toc37193998)

[Game Screen 4](#_Toc37193999)

[Pause Screen 4](#_Toc37194000)

[Game Over Screen 5](#_Toc37194001)

[Gestures identified as appropriate for this application 5](#_Toc37194002)

[Hardware used in creating the application 6](#_Toc37194003)

[Architecture for the solution 7](#_Toc37194004)

[UnityEngine.Windows.Speech 7](#_Toc37194005)

[Windows.Kinect 7](#_Toc37194006)

[Conclusion & Recommendations 8](#_Toc37194007)

[Rachel 8](#_Toc37194008)

[Niall 8](#_Toc37194009)

[References 9](#_Toc37194010)

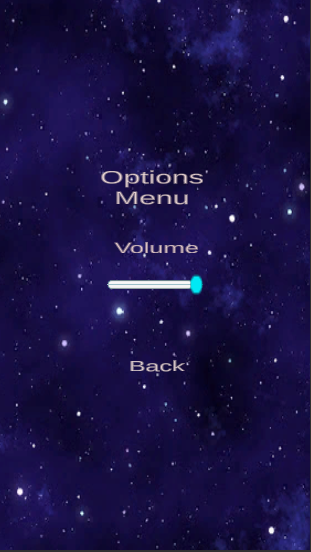
[Test Plan 10](#_Toc37194011)

# Purpose of the application

The purpose of this application was to create a unity game using gesture-based technology that was playable by a user using a hardware component suitable for the game that was designed. We decided on a game where the user can control a spaceman character to dodge oncoming enemies by moving side by side. The user can move the character left and right with their own body movements using the Xbox Kinect Camera Version 2. We have also designed this application with the use of voice recognition through the microphone on the laptop or PC being used to play the game. The user will be able to use speech to control the main menu screen and the user will also be able to pause and un-pause the game using speech also.

## Menu Screen

This is the first screen the user will see when the game starts. It contains the title of the game and three buttons. The Start button brings the user to the first level of the game. The Options button has the option to change the volume level of the game and the Quit button allows the user to end the game.

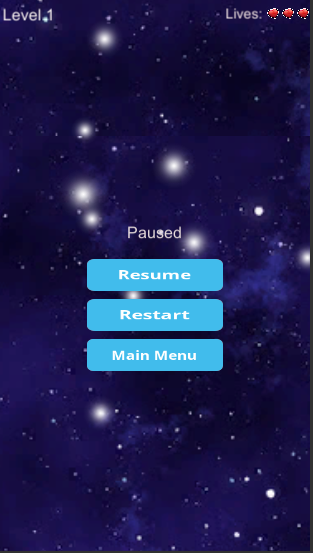
Options Screen

This screen allows the user to change what the volume level of the audio playing in the game will be. They can use their mouse to slide the slider to the left to lower the sound or to the right or increase it. After deciding what level, the user would like the volume to be at, the can then press the Back button to go back to the Menu Screen.



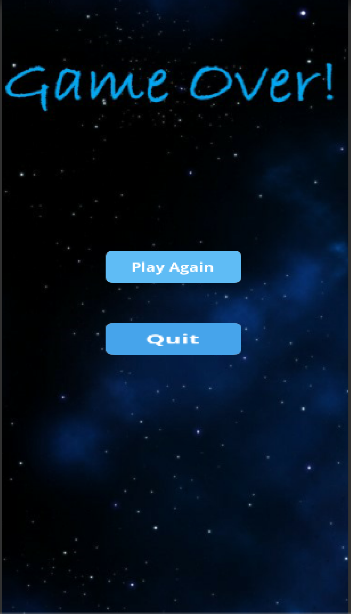
Game Screen

This screenshot shows one of the levels on the game. The other level is quite similar, but the falling enemies have been sped up and appear quicker in the second level. Up the top left-hand corner shows the level the user is on. The top-right shows the amount of lives the user has left. At the start of each level the user has 3 lives and once they lose these lives, the game is over. The game can be paused. We decided to only allow the user to be able to pause the game with speech and not use a key on the keyboard. The reason behind this is because while playing the game, the user will not be able to get to the keyboard in time to pause the game as they may lose a life doing so. The Spaceman will appear at the bottom of the window and be controlled by the movement of the user using the Kinect Camera. They will only be able to move left or right. The enemies the user will have to dodge will be falling from the top of the screen. This includes ship ships and aliens. They fall at random places along the top and at a random time.



## Pause Screen

The pause screen will be displayed when the user says one of the following commands: “Stop”, “Pause”, “Pause game”. This will stop all movement in the game. It will display three buttons also. The Resume button allows the user to go back to where they were and continue with the game. The user can resume by either clicking on the resume button of saying “Resume” or “Play”. The Restart button allows the user to restart the current level that they are on. This is like resuming the game but to restart the user can say either “Restart” or “Reload”. The Main Menu button brings the user back to the Menu screen. This can also be done by saying “Main menu”.

Game Over Screen

The Game Over screen will be displayed when the user loses all three lives. When the user reaches this point, they can either Play Again or Quit the game. This can be done by either clicking on the buttons, or again by using the speech recognition inbuilt into the game by using the commands “Play” or “Play Again” to restart the game, or by using the command “Quit” or “End Game” to stop the Application.

# Gestures identified as appropriate for this application

For the gestures that this application uses, the way in which the spaceman follows your movements, is by using the body joint of the spine of the person that is standing in front of the camera of the Kinect camera. This allows the user to move within the area of the camera, and the spaceman will follow either left or right with those corresponding movements. The user is not able to move any other direction. We decided that they did not need any other movement type in this game and added code in that prevents the user from moving the player object in any other direction.

The speech recognition is detected from the microphone on the laptop. It is used to control the menus of the game. The game will only recognize certain commands that are said. We did not want to have just one command for each action as there are different ways of saying start or pause. For example: to start the game the user can say, “play”, “play game” or “start”. This different ways of starting the game all start the game the same way. Allowing the user to pause the game was done similarly, we decided to allow the user to pause by saying, “pause”, “pause game” or “stop”.

# Hardware used in creating the application

The hardware used in this game for the main movements of the user was the Kinect Camera to locate the central spine joint of the user. Once the joint was detected, the user’s object which is a spaceman image appears onto the screen.

We decided to use a Kinect Camera for this project, as Rachel already owned a Kinect Camera Version 1. After many attempts to use this camera, we figured out that the version one camera is no longer supported by Windows, unless your operating system is Windows 8 or below. We were able to get the Kinect running on the computer by changing the Compatibility mode in the Compatibility tab in Properties when right clicking on the downloaded SDK. This needed to be done before installing it. After doing it, we can get the camera on the Kinect opening and running on the laptop. However, when we began developing the game, the code associated with the Kinect Version 1 would not work with Visual Studio 2017 or 2019 as it was outdated. This caused issues as we knew we would need to be able to integrate the Kinect into the game but had come to an unexpected halt at this point. We resorted to borrowing a Kinect Version 2 camera from Damien.

We used the microphone on the laptop to allow the user to control and navigate through the different menus. We allowed the microphone access to the Unity game and created methods that listen for certain keywords depending on what scene they were in.

The Myo armband was a piece of hardware that at the start we had considered using. We had tested the armband in the labs and concluded that is was easy enough to refractor the already gathered code to have it useable in our game. However, since Rachel already had a Kinect and we wanted to challenge ourselves more by having to research how to add the Kinect into a game, we felt it was an ideal solution to use the Kinect in the game.

# Architecture for the solution

## UnityEngine.Windows.Speech

This library was used in the game to detect what the user is saying and what to do with that information. The class that we used from this library is KeywordRecognizer. This class allows us to use the microphone to listen for inputs made by the user. It is then able to match this input to a list of registered keywords that we decided on. Each of these keywords have their own method which is called as soon the game recognised a keyword.

## Windows.Kinect

This library is used to build a desktop application using the Microsoft Kinect. It provides the tools and APIs, both native and managed, that you need to develop any Kinect-enabled applications. The Kinect SDK has three different sets of APIs that are needed to create an application that works with a Kinect. A Windows Store application uses the Windows Runtime API. An WPF application created uses the .NET API and an application that requires native code uses a set of native APIs. This application that we have created uses the Windows Runtime API. One of the classes we use in this library is Windows.Kinect.Joint. We use this to get the position of a joint of a body. The joint we are using is the spine joint. JointType is a member of this class that contains all the types of joints available to use. This class also accesses each of the cameras in the Kinect, the infrared and colour. Other video data received from the Kinect is depth and body. These are used to create the skeleton of the user. This is essential as this controls the range of angles the camera can pick up, and how it can determine the joints of the body.

# Conclusion & Recommendations

## Rachel

For me, I found this project challenging while interesting. Using hardware in an application is not something I have done before which was why I felt it was interesting. The project was challenging at the start as we had originally started using the Microsoft Kinect Version 1 but the software available to download would only work on a computer that had Windows 8 or below. We started of watching tutorials and while following the code, it gave a lot of errors as it needed to be ran on older applications which we could not download. With that in mind, we were grateful enough to be able to borrow a Microsoft Kinect Version 2 to get our game working. After getting the Kinect working in the game, we did struggle with working on the rest of the game as we were not able to meet up and work on it together. Instead, we skyped when we were both free and worked on it together on the one unity application and used screensharing so that we could both see the code at the same time. I enjoyed working with Niall as we both knew what we wanted the game to look like at the end which allowed there to be no conflicts while creating the game. I do feel that this game could be improved on with more time. There could be lots more levels added where the user faces different challenges on different levels. Ones that could benefit the user or ones that could have the user end up back at the very start again. Due to one of us only being able to test with the Kinect, we added the speech recognition code into the game using the microphone on the laptop. This way it was easier to test when we were working independently. If I had more time, I would defiantly like to change that to be using the Kinect microphone instead.

## Niall

Overall, I found this a very enjoyable project to work on. I found it interesting to see how the software and hardware link into each other, and how tricky it can be to get the hardware working the way you would like it with the software. Once we discovered that the Kinect Version 1 was unable to connect to windows 10, and Damien was good enough to allow us to borrow a Kinect Version 2 we could really start to make headway into project. I really enjoyed working in a team with Rachel, she was extremely helpful, and when we couldn’t actual work side by side, we worked very well together through Skype videos and screensharing sessions. I really feel that even though we are going through unprecedented times, it really gave me a feel for how working with teams from all over the world may be like and was very insightful into working remotely as a team. If I were to undertake the project again, there are definitely some changes I would like to try incorporating into the game. I would like to try and get the MYO armband introduced to the game, where when the user clinched their fist, the spaceman would be able to shoot a projectile at the oncoming enemy, or make more levels for the user to go through, where they could meet a final boss after so many levels.

# References

Link for Processes to connect Kinect

<https://medium.com/@anran1015/unity-kinect-documentation-cf61a25f732>

Link for Kinect Setup in Unity Game

<https://www.youtube.com/watch?v=aHGlLxh6a88>

Link for Download and set up of Kinect

<https://channel9.msdn.com/Blogs/raw-tech/Making-your-body-the-controller-Kinect-Tutorial-for-Unity>

Link for Developing a unity game with Kinect

<https://andreasassetti.wordpress.com/2015/11/02/develop-a-game-using-unity3d-with-microsoft-kinect-v2/>

Kinect for Windows SDK

<https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271(v=ieb.10)>

Kinect API Overview

<https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn782033(v%3dieb.10)>

Link for Adding Pause Menu

<https://answers.unity.com/questions/1230216/a-proper-way-to-pause-a-game.html>

Link to Add Voice Recognition to a Unity Game

<https://www.youtube.com/watch?v=29vyEOgsW8s>

Link for Speech Recognition Tutorial

<https://www.youtube.com/watch?v=HwT6QyOA80E>

# Test Plan

