Duck Catching Game using Kinect V1 and Speech Recognition

<u>Authors</u>

Stoyan Rizov (G00313180)

Kamila Michel (G00340498)

Cristina Florina Nita (G00332787)

This game is based on the old Duck Game released for the NES (Nintendo Entertainment System) in which the player had to kill all the ducks in the game to win.

Table of Contents

Table of Figures

1. Purpose of the game

1.1 Game Concept

The main character is an anime-based human Fig. 3 Game Character that needs to collect the ducks Fig. 2 Duck to increase the score all the while kicking the snakes Fig. 1 Snake before being killed by them. The game ends when the health bar is at its lowest and the user dies.

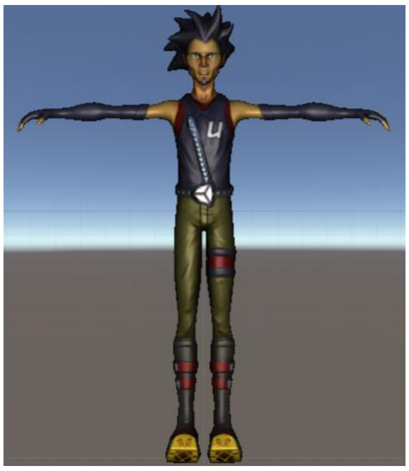


Fig. 3 Game Character

Fig. 2 Duck

Fig. 1 Snake

1.2 Game Genre

This is a shooter-type game based on the original Duck Hunt game Fig. 6 Duck Hunt that was first released in Japan in 1984, on the Nintendo Entertainment System (NES) Fig. 5 NES video game console (1), the 2rd game console ever released by Nintendo after Color TV-Game Fig. 4 Color TV-Game, “a series of five dedicated home consoles” (2)

Fig. 5 NES

Fig. 4 Color TV-Game

Fig. 6 Duck Hunt

1.3 Objective

The main goal of this project was to create a Duck Hunt based game, with the Kinect Hardware. In this game, the player's body is displayed as the in-game character. The player will have to use their body parts according to the game instructions to collect the ducks and destroy the enemies.

The game's goal is to collect as many ducks as possible to level up the score before being bitten by the snakes, which are the enemies that need to be destroyed. Being an endless game, the ducks spawn continuously until the game ends making it very competitive and fun to play as the difficulty increases – the number of snakes in the game is incremented by 1 every 30 seconds.

1.4 Front-End

This will include a Main Menu Fig. 7 Main Menu. that takes the player to the actual gameplay. When the word "Play" is spoken or "Play" button is chosen using the hand cursor.

The menu has four buttons – Play, High Score, Instructions and Quit – each carrying different functions in relation to the functionality of the game.

As any other game, the Play button is used to start the game and it takes the user to the scenery, which is that of a field with grass and trees, specific to the environment in which both ducks and snakes can be found. To access the high score as well you can press the "High Score" button or say, "High Score" Fig. 8 High Score**Error! Reference source not found.**. The same applies to quit the game entirely and to view the Instructions Fig. 9 Instructions– which is used to let the user know how to play the game.

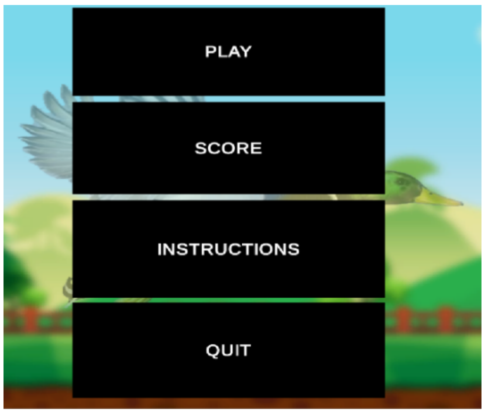
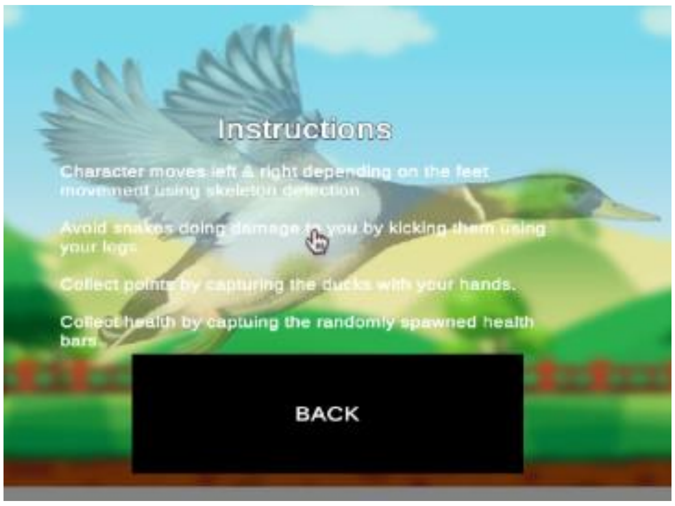


Fig. 7 Main Menu

Fig. 8 High Score



Instructions

Character moves left & right depending on the feet movement using skeleton detection

Avoid snakes doing damage to you by kicking them using your legs

Collect points by capturing the ducks with your hands.

Collect health by captuing the randomly spawned health bars.

Fig. 9 Instructions

1.5 In-Game Menus

A Pause Menu will be included which will allow the user to resume the game, restart the game from the beginning and navigate to the main menu. All this will be done via speech recognition making it easier for the player to access the menu while away from the computer.

As the game ends when the character dies, there is a Game Over Menu Fig. 10 Game Over included which shows the high score of the player.

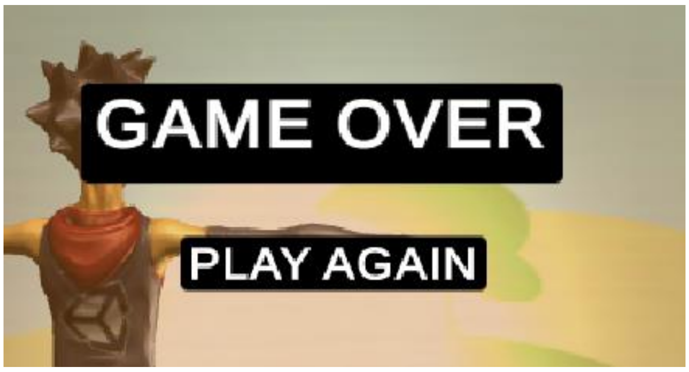


Fig. 10 Game Over

1.6 Control Mechanisms

To control the game the player will be using their bodies, specifically their hands to collect the ducks, their feet to kick the enemies as well their heads to collect the health bonuses which give a health boost and increase the lifespan of the character.

The menus will be controlled with a hand cursor recognized by the right hand, or via speech.

1.7 The Game

The player will be using their hands to catch the ducks – as supposed to using a gun to shoot them in the original game. Score is increased by one with every duck caught and it keeps increasing until defeated.

Snakes were included as enemies that try to harm the in-game character and can be destroyed when kicked, otherwise they will eat away the health bar until it is at its lowest and that is when the game ends. At the end of the game, the highest score is displayed on the screen and can be reset at will.

In the game, the ducks are spawned continuously until the character dies and this makes for a fun and competitive never-ending game to play as well as a good fitness game for those who don't have time or the energy to attend the gym since it involves a continuous hand and feet movement.

2. Gestures identified as appropriate for this application

2.1 Myo Armband

Fig. 11 Myo Armband

Originally, the idea was to create a shooter game using a gun-like hand gesture as per Fig. 12 Gun-like hand gesture to shoot the ducks in the game. Although the idea was good, we believed that it was not complex enough for the amount of people in our team so we have decided to consider other hardware options that seemed to involve a much more significant amount of research and complexity for a group of three people.

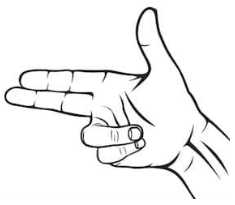


Fig. 12 Gun-like hand gesture

2.2 Leap Motion Controllers

Fig. 13 Leap Motion

The Leap Motion Controllers seemed like another decent choice for the game as the thought of using them inside the game with an in-game gun seemed like a good idea. The plan was to use the hands from inside the game to the weapons and fire at the ducks. Eventually we came to the conclusion that using the player's full body would be a better option for a much more fun experience, than just using the hands and unfortunately the leap motion controllers are not capable of recording anything outside the boundary of hand gesture so we have excluded this option also.

2.2 Kinect V2

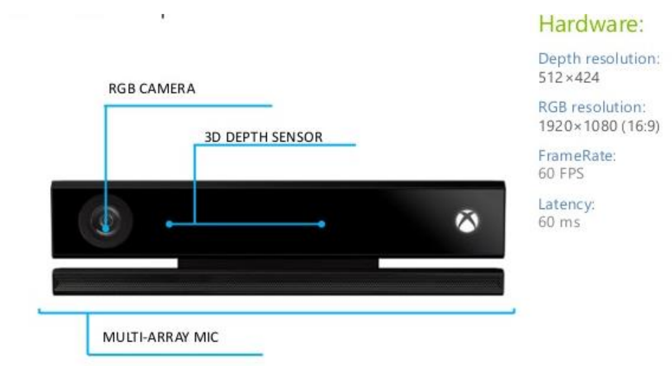


Fig. 15 Kinect V2

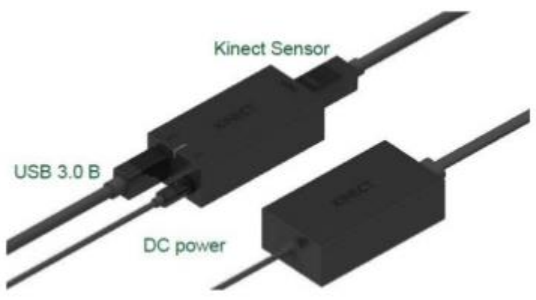


Fig. 14 Kinect V2 Adapter

We have decided to use the Kinect Hardware as we believe it is the best option as it can use both the hands and the player's body movements. The original thought of shooting the ducks was changed into catching the ducks and kicking the enemies as it gives a more unique and interesting experience for everyone to have playing as it is a fitness friendly and comedic gameplay.

After a lot of research we, as a team, found out that to build a game as such with the second version of Kinect would be very difficult as all the libraries and tutorials (3) for the movement of the body parts were either outdated, hidden, paid or deleted since the both the Kinect V2 and V1 Hardware are not used anymore (4) in the gaming industry for 3 main reasons:

1. People in general didn't want to play Xbox One games for Kinect since they were not considered the best
2. Unfortunately, Kinect v2 is a "more closed ecosystem than Kinect v1 and this has not been that good for makers" (4)
3. The sensor has too many cables and it requires a desk of their own which is very inefficient for people with small homes

2.3 Kinect V1

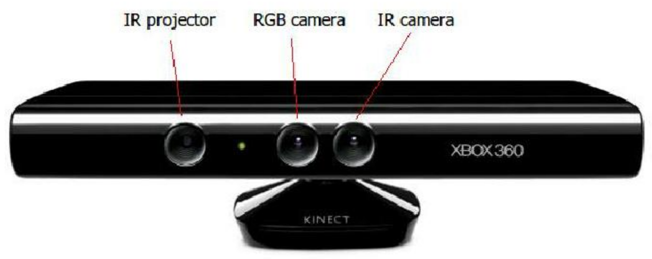


Fig. 17 Kinect V1

Fig. 16 Kinect V1 Adapter

Having both Kinects researched, we eventually decided to use Kinect V1 instead of Kinect V2 as both Hardware are seemingly similar in terms of the overall use, the main difference being that Kinect version two includes 25 joints as supposed to version one which includes 20. Although the switch from the Kinect with a higher number of joints to the one with less sounds like a terrible idea, it wouldn't affect our gameplay as the only extra joints are fingers and toes which are not required. Therefore, there would be no need to use the newest version.

2.4 Speech Recognition

At the start, we have thought of using the Microphone from the Kinect Hardware, but after a lot of research we have found out that there isn't much documentation about it for the Unity Platform and being new to this technology we wouldn't be able to figure out how to implement it without some information available to us, at least not for version one. The newest Kinect version had more documentation available about Speech Recognition, but it wasn't possible to implement it to version one as both Kinects require different SDK's and connect differently.

After a long, throughout research, we have decided to turn to Unity Speech Recognition as there are a lot more tutorials and explanations on how to implement it as well as solutions for issues that might be encountered.

3. Hardware used in creating the application

3.1 Kinect

Formerly known as Project Natal, this hardware is a device used as a “controller for Microsoft’s Xbox 360 and Xbox One gaming systems” (5). It uses video data collected by the Kinect Camera and audio data from the Kinect Microphone to interact with games, movies and other entertainment programmes available.

The capability of this hardware to “distinguishing the user's gestures, voice commands, facial characteristics, skeletal data and full body motions” (5) allows the programmes using it to identify individual users and the names connected to them which plays a significant role in researching fields such as “IT health, education, home automation and in health technology” (5).

An example of research project that has been implemented with the Kinect is translating sign language into spoken as well as written language. Researchers from China have developed a the “Kinect Sign Language Translator” (6), with Microsoft Research that makes it easier for people who are unable to use their vocal chords, to communicate with others. Just like there are different languages spoken in different countries, “there are more than 300 sign languages” (7) in the world, as surprisingly as that sounds. Kinect not only allows communication with people who do not understand it, but also with those who have learnt a different one, to communicate with each other.

3.1.1 About Kinect V1

Putting it into simple words, Kinect V1 is a Smart Camera for Xbox 360 – released on Xbox 360 in November 2010 and on Microsoft Windows in February 2012 – that uses body gestures or speech commands as controllers. Its camera uses infrared light Fig. 18 Kinect IR which accounts for a great skeleton tracking, even at night, and it can track 20 joints in the body for one player in the Default mode, 10 joints in the Seated mode Fig. 19 Kinect Tracking Mode and as many as 48 joints for up to two players at the same time (8). We will only be using the Default mode for a single player.

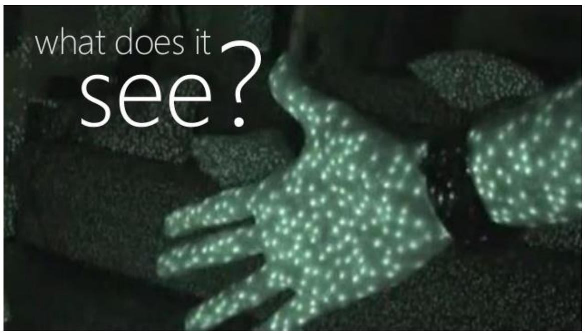
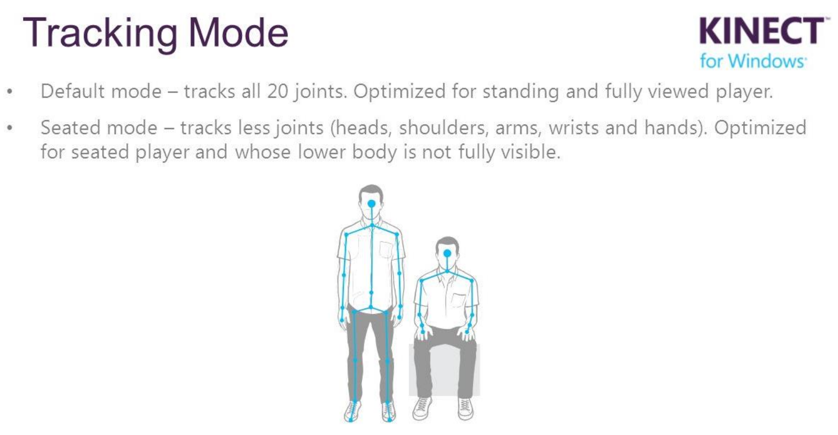


Fig. 18 Kinect IR


Tracking Mode

KINECT for Windows

- Default mode – tracks all 20 joints. Optimized for standing and fully viewed player.
- Seated mode – tracks less joints (heads, shoulders, arms, wrists and hands). Optimized for seated player and whose lower body is not fully visible.

Fig. 19 Kinect Tracking Mode

3.1.2 Reason of Choice

Our idea involved using our body parts such as hands and legs to interact with the game. Kinect V1 would have been one of the best options for the gameplay as it allows you not only to use your hands, but also the rest of your body. This hardware and the Kinect V2 were two of the best choices, but we have decided on the first version as the second one contained features we did not need to include in our game.

Apart from that, the hardware we chose seemed to have a tint of complexity to it. It was very challenging to work with and interesting to investigate.

3.1.3 Comparing Kinect V1 and Kinect V2

As we have worked with both, not only have we managed to gain an insight on the Kinect hardware; but we also have experienced the difference in working with them in relation to finding information online as well as the difference in coding on both hardware.

They are both similar in terms of coding and functionality, but they both require different SDK's to work with:

Kinect V2:	*Kinect V1:*
• Kinect for Windows SDK 2.0	• Kinect for Windows SDK 1.8 • Kinect for Windows Developer Toolkit v1.8

In terms of joints, both Kinects recognize a specific amount. However, even with version two having the advantage of recognizing more, it still does not affect the way our game works as the only extra joints are the fingers. There is no requirement for finger movement to catch the ducks or kick the snakes. Our movements were purely based on hand and feet movements which can be done with both anyways.

Kinect V2

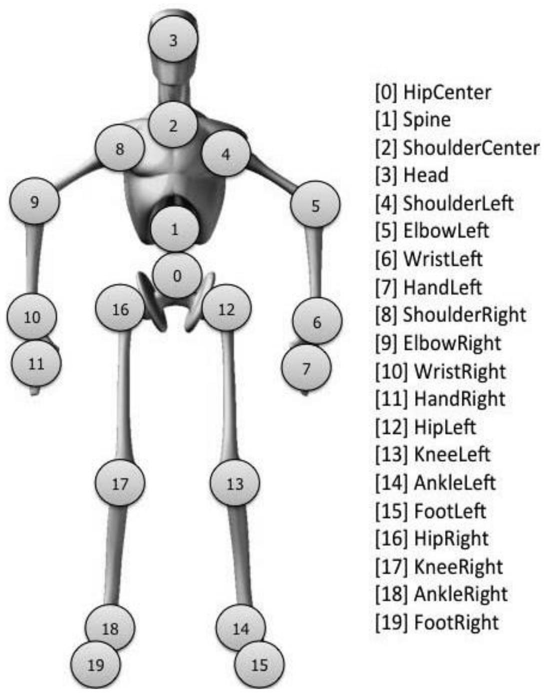


Fig. 20 Kinect V1 Joints

Kinect V1

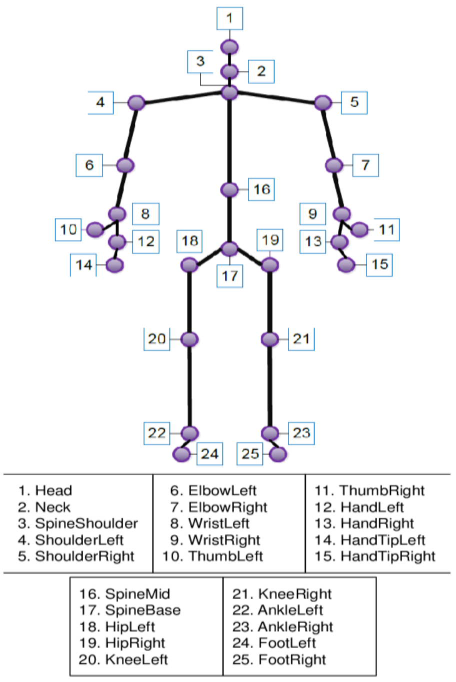


Fig. 21 Kinect V2 Joints

We found it very difficult to find information on linking a character in the game with the body skeleton of the player for the second version (3) and unfortunately, the Dynamic Link Library (DLL) (9) was out of date and all of the other options we came across had similar issues; if not outdated they would either be paid or they would have been removed.

Eventually, we had to turn to version one as we have heard connecting the skeleton to the game character is less complicated. Although this was the case, there were many other features that were easier to implement on the second version of Kinect, such as Speech Recognition and Character Jump.

As we have only swapped the hardware versions, we believed that we didn't have to make too many changes to our code – which was initially working on version two – to work on version one. We found it difficult to do that, so we concluded we should just restart the whole game instead of changing the connection to fit version one.

When it came to implement a jumping action to the character so that it would be able to catch the ducks higher up, we found plenty of information for the second version. We couldn't, however, find any information for the previous one. We have tried to figure out how the jump works on version two to try and see if we can implement it to the older version, but that didn't seem to work. After a long while of trying and failing, we decided to exclude the jump from our game.

We also tried to implement Speech Recognition on the Kinect – which will be discussed in more detail later. Unfortunately, we couldn't find any helpful material explaining how to put it into practice, at least not for Kinect V1, but we could find plenty for version two.

Both versions are still used in research for various fields; but are now dead for Microsoft Windows games development. Which might be the main reason as to why there is so little non-paid, up-to-date information about them.

3.2 Speech Recognition

Speech recognition is an input technology used to determine words and sentences in spoken languages. These identified phrases are used to trigger an action or event in machines (10). This type of input is generally very useful to those who cannot use their limbs to interact with technology and it is also an easier alternative for the same purpose as using your mouse or keyboard.

3.2.1 Issues

The big issue with speech recognition is that the accent differs from person to person, and this can affect its accuracy. Even for a native English speaker this can be a problem. But if using simple words instead of long sentences to interact with a game, this issue might not be encountered.

Another major problem would be not being able to have background music included as it can interfere with the speech input. To prevent that, the user must use a microphone that is capable of filtering background noise (11), which is an inefficient solution especially for those who cannot afford them. The only other option to avoid this problem would be to not include any background music and to warn the user to rid of any background noise.

3.2.2 About Unity Speech Recognition

Unity has a class in *UnityEngine.Windows.Speech* called *KeywordRecognizer* that takes in voice input and matches those phrases to a list of keywords from the Dictionary. This class "listens to specified keywords with the specified minimum confidence" (12) only. In our game, the confidence level is medium, so anything below that is ignored.

3.2.3 Reason of Choice

The main reason for using speech recognition would be that it would make it easier for the players to access the menus and play the game with the Kinect. For this hardware, the user must be between 1.8 and 2.5 meters (13) away from the camera to be able to use both the hands and feet. Having buttons that require the player to interact with the mouse would be inefficient, hence the use of speech input.

It is used to access the gameplay from the starting menu – as well as all the other pages whose names are visible on the buttons – and to access the game's main menu from both the pause and game over canvases.

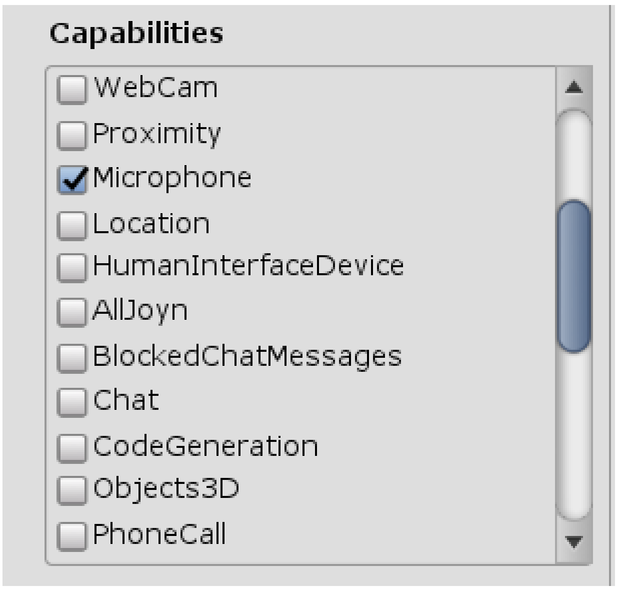
3.2.4 Comparing Kinect V1 Speech Recognition and Unity Speech Recognition

There is a lot more information available on this topic for Unity than there is for Kinect. Apart from that, Unity's only requirement is to change its settings to allow microphone input while the speech intake from the machinery requires three different SDK's.

Speech Recognition Requirements

Unity

To enable the microphone, go to the Unity Editor and navigate to "File > Build Settings > Player Settings > Universal Windows Platform Settings > Capabilities" and thick the box next to the Microphone (14) as shown in Fig. 22 Enable Microphone Capability



Capabilities

- ☐ WebCam
- ☐ Proximity
- ✓ Microphone
- ☐ Location
- ☐ HumanInterfaceDevice
- ☐ AllJoyn
- ☐ BlockedChatMessages
- ☐ Chat
- ☐ CodeGeneration
- ☐ Objects3D
- ☐ PhoneCall

Fig. 22 Enable Microphone Capability

Kinect V1

There are three different installations required for the microphone to work as well as a visual scripting library which costs $65 if we wished to implement it.

Downloads

- Microsoft Speech Platform - Software Development Kit (SDK) (Version 11)
- Microsoft Speech Platform - Runtime (Version 11)
- Kinect for Windows Runtime Language Pack

Library

- Playmaker (15)

As we researched about speech recognition for Kinect V1, we found a lot of information for Kinect V2 on how to include it in Unity (16). The only information we could find for the old version are the ones involving the Playmaker Library (15) or the Windows Presentation Foundation (WPF) which is "Microsoft's latest approach to a GUI framework, used with the .NET framework" (17) . WPF is not for unity but includes the C# and XAML languages. XAML being developed by Microsoft and is "used for initializing structured values and objects" (18).

Therefore, we decided to use Unity for voice input instead which was doable. The only issue with it would be that you would need to speak slightly louder while being two meters away from the microphone.

4. Architecture for the solution

4.1 Assets and Libraries

4.1.1 Asset

For our game, we have used an Asset called “Kinect with MS-SDK” (19) from Unity to help us with our basic game mechanics which are hand and leg movement. This asset included the body movements as well ass the sensor connection required to use the Kinect.

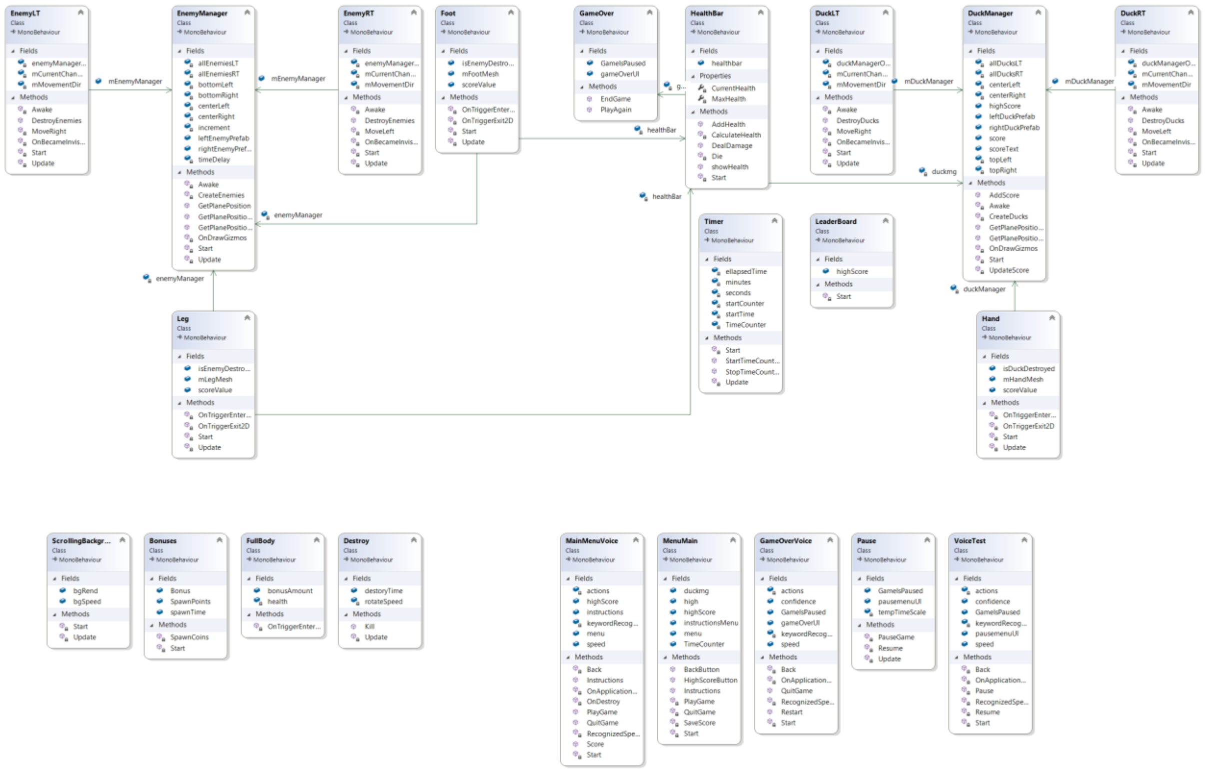
We also used this asset for the Hand Cursor that we implement in the Main Menu to help the user click on the buttons using their right hand which we tried to implement. Unfortunately, this feature caused high CPU usage as its script was running all the time and it had to be disabled.

4.1.2 Library

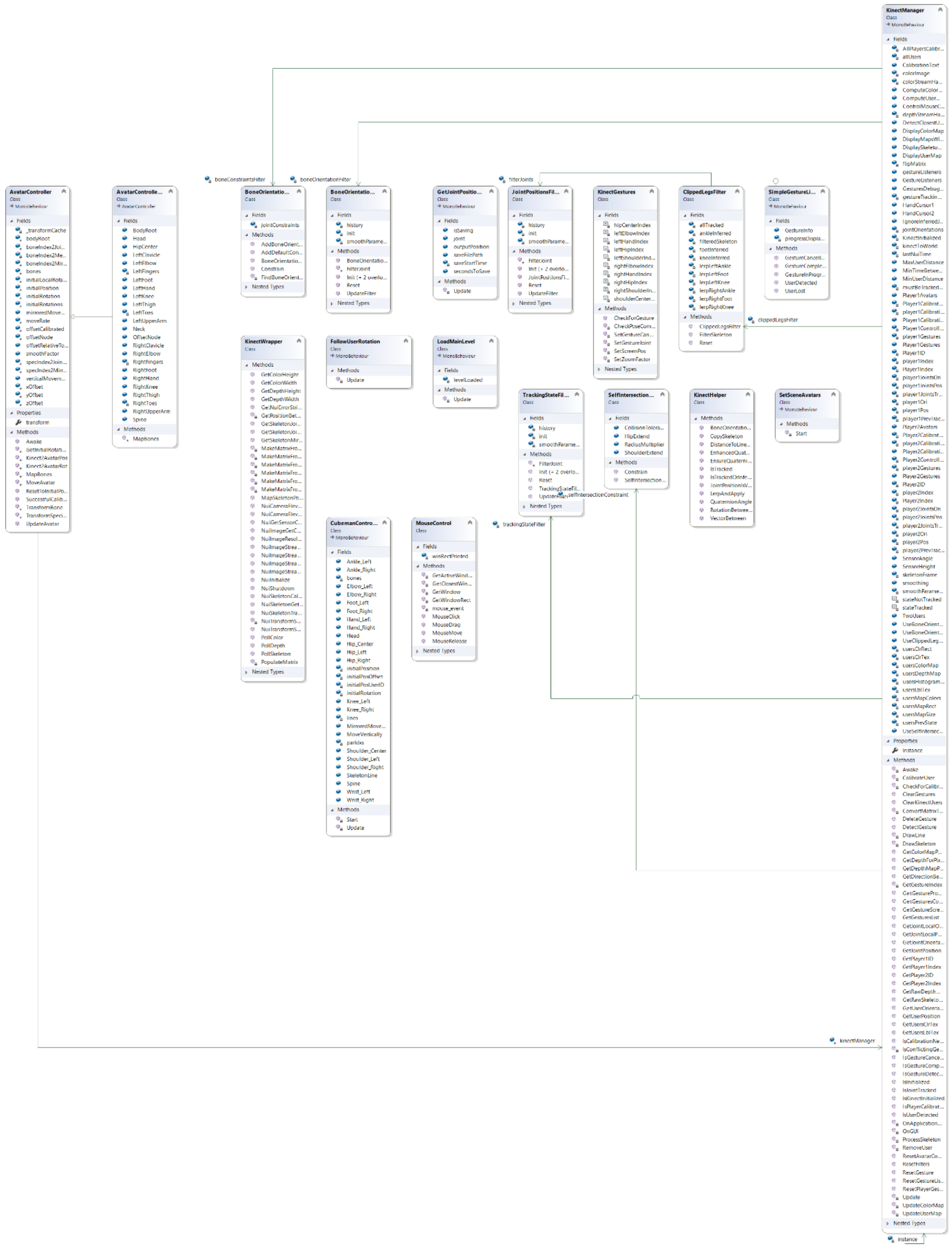
For the Speech we had to use KeywordRecognizer from Unity, which is a class that accepts audio input and uses it in comparison to a List of keywords.

4.2 UML Diagrams

4.2.1 Game Scripts



4.2.2 Kinect Scripts



5. Conclusions and Recommendations

5.1 Kinect is not used for Gaming anymore

Researchers are still benefiting from Kinect as an investigation approach in many fields, but in terms of entertainment, it’s not as popular. For this reason, we failed to find all the information required for the development of our application with both versions.

5.2 Unity Teams instead of GitHub

As we had issues with pulling the project from GitHub, it would have been a better practice to use the Unity Collaboration feature. The Unity Teams approach would have made it easier and faster than to save, store, share, secure and sync the project with all the members.

5.3 New Technology

The most enjoyable part of this project was using new technology to develop an application. We have improved our knowledge as we have never worked with Kinect before and that was a very entertaining process.

6. References

1. **Wikipedia contributors.** Duck Hunt. *Wikipedia, The Free Encyclopedia.* [Online] 29 March 2019. https://en.wikipedia.org/w/index.php?title=Duck_Hunt&oldid=889944827. 889944827.

2. —. Nintendo video game consoles. *Wikipedia, The Free Encyclopedia.* [Online] 20 March 2019. https://en.wikipedia.org/w/index.php?title=Nintendo_video_game_consoles&oldid=888713884. 888713884.

3. **nitta@tsuda.ac.jp.** How to make Unity's Humanoid take the same pose as Kinect V2's skeleton recognized in real time. *Kinect V2 Programming.* [Online] 2017. https://nw.tsuda.ac.jp/lec/kinect2/KinectV2_dll6/index-en.html.

4. **Skarredghost.** Is Kinect dead? *The Ghost Howls.* [Online] 31 October 2016. https://skarredghost.com/2016/10/31/is-kinect-dead/.

5. **Technopedia contributors.** Kinect. *Techopedia - Where IT and Business Meet.* [Online] https://www.techopedia.com/definition/10695/kinect.

6. **Microsoft blog editor.** Kinect Sign Language Translator – part 1. *Microsoft Research Blog.* [Online] 29 October 2013. https://www.microsoft.com/en-us/research/blog/kinect-sign-language-translator-part-1/?from=http%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fcollaboration%2Fstories%2Fkinect-sign-language-translator.aspx.

7. **Jamaluddin, Azwan.** 10 Creative And Innovative Uses Of Microsoft Kinect. *hongkiat.* [Online] 19 November 2017. https://www.hongkiat.com/blog/innovative-uses-kinect/.

8. **Wilson, Mark.** What Is Xbox 360 Kinect? *GIZMODO.* [Online] 14 June 2010. https://gizmodo.com/what-is-xbox-360-kinect-5563047.

9. **nitta@tsuda.ac.jp.** How to make Kinect V2 skeleton recognition as DLL and use it from Unity. *Kinect V2 Programming.* [Online] 2016.

10. **Rouse, Margaret and Kiwak, Karolina.** Speech recognition. *WhatIs.* [Online] December 2016. https://searchcustomerexperience.techtarget.com/definition/speech-recognition.

11. **Unuth, Nadeem.** What Is Speech Recognition? *Lifewire.* [Online] 19 December 2018. https://www.lifewire.com/what-is-speech-recognition-3426721.

12. **Unity3D Developers.** KeywordRecognizer. *Unity3D.* [Online] 26 March 2019. https://docs.unity3d.com/ScriptReference/Windows.Speech.KeywordRecognizer.html.

13. **Plunkett, Luke.** How Much Space Do You Need To Use The New Kinect? *KOTAKU.* [Online] 6 November 2013. https://kotaku.com/how-much-space-do-you-need-to-use-the-new-kinect-512746898.

14. **Zeller, Matt, et al.** Voice input in Unity. *Microsoft.* [Online] 21 March 2018. https://docs.microsoft.com/en-us/windows/mixed-reality/voice-input-in-unity.

15. **Chouls, Alex, et al.** Playmaker. *Unity Assetstore.* [Online] https://assetstore.unity.com/packages/tools/visual-scripting/playmaker-368.

16. **nitta@tsuda.ac.jp.** How to make Kinect V2 speech recognition as DLL and use it from Unity. *Kinect V2 Programming.* [Online] 2016; 2017. https://nw.tsuda.ac.jp/lec/kinect2/KinectV2_dll3/index-en.html.

17. **WPF Tutorial Developers.** What is WPF? *WPF Tutorial.* [Online] 2007. https://www.wpf-tutorial.com/about-wpf/what-is-wpf/.

18. **Wikipedia contributors.** Extensible Application Markup Language. *Wikipedia, The Free Encyclopedia.* [Online] 5 January 2019. https://en.wikipedia.org/w/index.php?title=Extensible_Application_Markup_Language&oldid=876939643. 876939643.

19. **F, Rumen.** Kinect with MS-SDK. *Unity Assets Store.* [Online] 10 August 2015. https://assetstore.unity.com/packages/tools/kinect-with-ms-sdk-7747.