BSc in Software Development

Year 4

Gesture Based UI Development

Road Fighter

Christopher Madden (G00214065)

Andrej Lavrinovic (G00196984)

Gary McHugh (G00308668)

Table of Contents

Team Members:

Student Names:

Andrej Lavrinovic (G00196984)

Christopher Madden (G00214065)

Gary McHugh (G00308668)

GitHub Link:

https://github.com/Fourth-Year-Gesture-UI

Project Idea:

Our project idea is to create a two-player fighting game from scratch using the Kinect v2, the game will be able to track 2 separate users. We are going to use Unity with Kinect v2 to create avatars/characters to represent the user's. When the user's punch gesture is recognised we will play an animation on the avatar. We will then determine whether there has been a collision between a user's gesture and their opponents body in game. This will be done through colliders on the avatars within Unity. If a collision has been detected between the user's gesture and the opposing players body, the user will then score a point/ take health off their opponent. We will use health bars to represent the player's health. The winner is the person who takes away the most health from the other player. We may have a timer within the game to limit how long the fight will last. The gestures that we intend to use are listed under "Potential gestures".

Purpose of Application:

The purpose of our application is to create a 2-player fighting game similar to Street Fighter using the Kinect and Unity. We want to experiment with various gestures and see how well we can train the Kinect to track various gestures. We want to see what gestures will be recognised and what will not. What gestures can be distinguished from others? An example of this would be the different types of punches like a straight punch and a hook punch. We want to create a two-player game where each player will control a Unity character. We will detect each user's gestures and play animations when the defined gestures have been detected. If a collision occurs between the users punch/kick against the opponent's body, health will be deducted from the opposing player's health bar.

The health system we intend to use is shown below:

Gesture's	Damage dealt
Total Health	**100**
Left Straight Punch	**7**
Right Straight Punch	**5**
Left Kick	**12**
Right Kick	**3**

Things worth highlighting

- Custom built environment using external 3D assets
- All fighting gestures were built by us using Kinect Studio from scratch
- Used Kinects inbuilt hand gestures along with our own
- Music on menu page and in game
- Sound effects for exiting main menu and for hit feedback
- Fully working health system with dynamic health bar
- Different end states (red win, blue win and draw)
- External animations integrated into the project
- Tied to real life gestures
- Visual feedback when a player hits another

Technologies Used:

Hardware:

- Kinect V1 (research)
- Kinect V2

IDE:

- Visual Studio 2015

Programming Language:

- C#

Game Engine:

- Unity

Operating System:

- Windows 10

Hosting Site:

- GitHub

Additional:

- GitHub Issues
- Zube (project management, Kanban boards)
- Slack (Github notifications and messaging phone app)
- Google drive (document sharing)
- Screencast-O-Matic (screen recording software)

Research

Kinect V1

We started out using the Kinect V1 as we did not have access to Kinect V2 at that point. We used it as a way to research how a Kinect works and how to hook up a Kinect to Unity. We researched the differences between the Kinect V1 and Kinect V2 in terms of the documentation available, resources available online, each versions ability to integrate into Unity. Finally, we researched what parts of the body each Kinect could track and how many players it could track.

Kinect V2

Having researched both Kinects we decided to use the Kinect V2, this was mainly due to its easy integration with Unity as well as the availability of resources online for the Kinect V2. We found an API that enabled us to track users using the Kinect V2 using the Tracking ID. We used the code from the labs along with this API to determine when a user enters the frame. We hooked up the Kinect with Unity and ran some sample projects to ensure we had everything working correctly. We were now able to use the Kinect V2 for our project.

Potential Gestures

- Straight Right Punch
- Straight Left Punch
- Left & Right Uppercut
- Block (two arms raised to face)
- Hand up (menu Navigation/pausing game)
- Left kick
- Right kick
- Win gesture

Implemented Gestures

- Straight Right Punch
- Straight Left Punch
- Block (two arms raised to face)
- Left Kick
- Right Kick

Straight Punch Gesture:

Due to Kinects inability to distinguish between a straight punch and a hook we were unable to implement different types of punches. However, we feel that the single type of punch implemented that we have selected will be enough for this project. We wanted to make sure that we selected a definitive gesture so that it could be easily detected. The Kinect had no trouble detecting the punch gesture as it contains a lot of upper body movement as well as arm movement. This made it easy to detect this gesture form various types of user. The Kinect was able to detect this gesture with a high level of confidence.

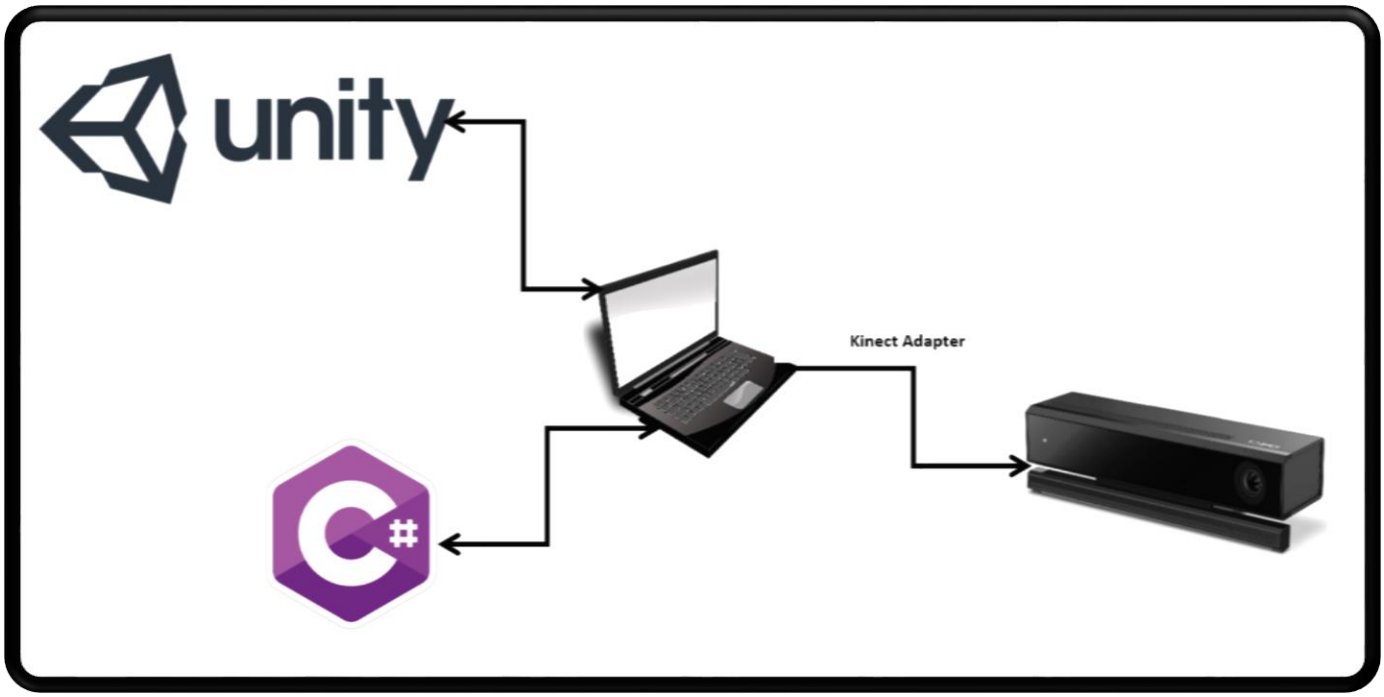
Kick Gesture:

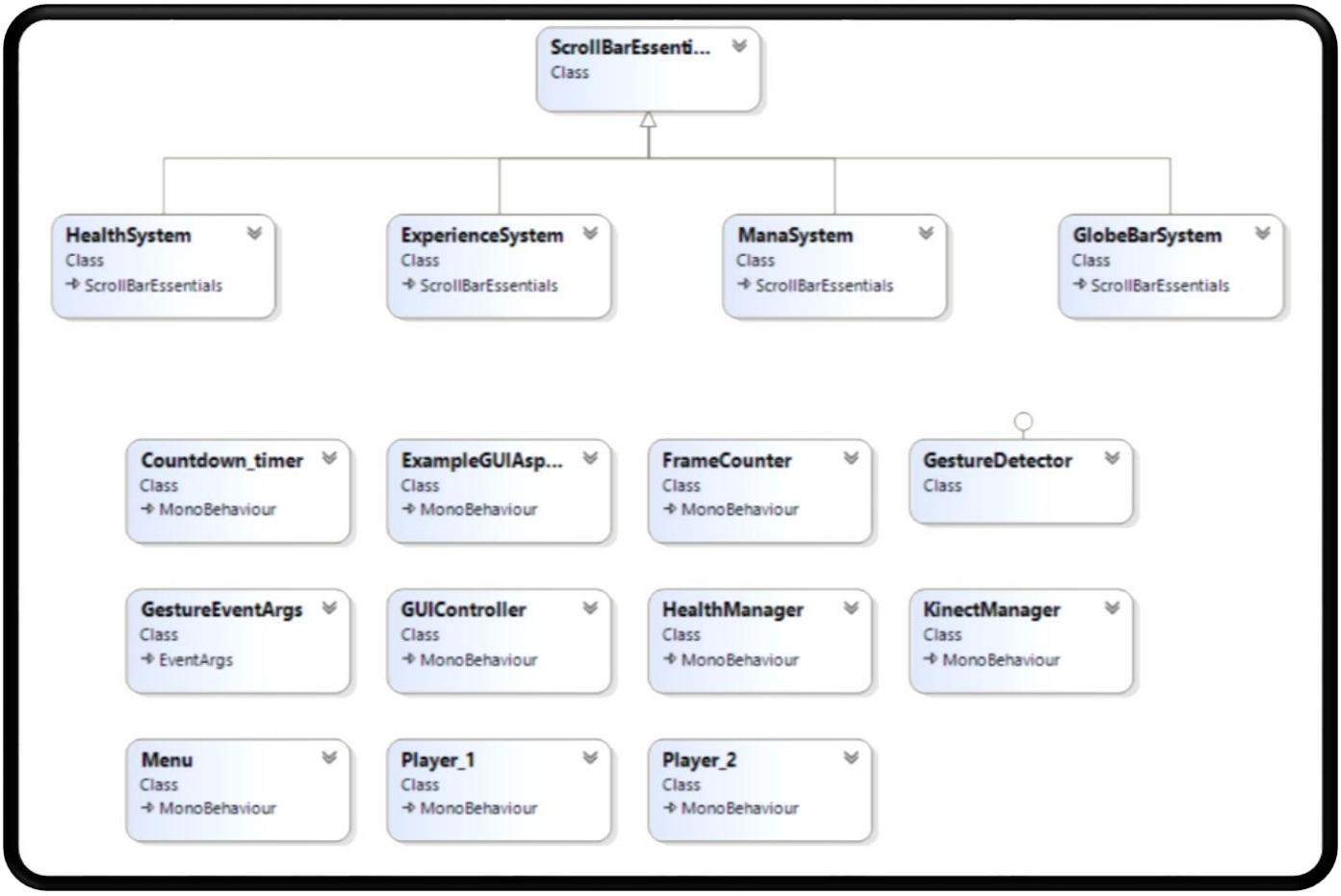
This gesture was also very easy to track as it was the only gesture that contained lower body movement. A kick is also a definitive gesture making it easy for the Kinect to detect this gesture with a high level of confidence. This is due to the large level of movement if the lower body (Kinect likes lots of movement for its gesture). Having trained it with different bodies, the Kinect was able to detect this gesture with a high level of confidence.

Block Gesture:

This was the hardest gesture to implement as it contained the least movement. Having trained the Kinect to recognise this gesture many times, it detected it with a low level of confidence. To overcome this issue, we set the confidence level for this gesture to be detected to be about half of the other Punch and Kick gestures. We tried to make this gesture as realistic as possible so that it would be intuitive to the user. Although we found it difficult to detect initially we are happy with how it turned out. The movement of the arms is being detected with an acceptable level of confidence.

Architecture of solution





Project Diary

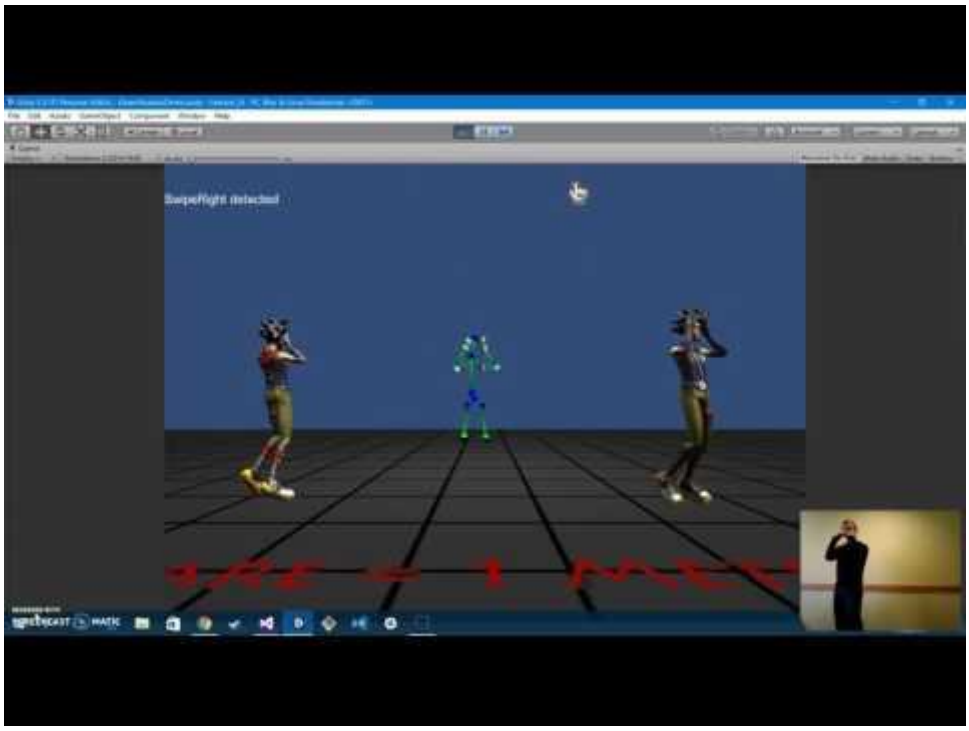
Week 1 (6th Feb):

- Decided on technologies we wanted to use (raspberry pi or Kinect)
- Researched and compared those technologies
- Decided to use Kinect
- Researched Kinect and Unity
- Came up with an idea (2 player fighting game with Kinect and Unity)

Week 2 (13th Feb):

Meeting 1

- Set up environment (Github, Slack, Google Drive, Unity etc)
- We were going to use Kinect v1 however, we then found out that we would have access Kinect v2's.
- We compared the Kinect v1 and v2 by doing research in terms of support, documentation and ability to hook it up to unity.
- We decided to use the Kinect v2 as there were more resources available on this as well as easier integration to Unity.
- Got Kinect V1 working with unity as part of our research while waiting for the Kinect v2's to be delivered.
- Tried other various projects that we found online which used Kinect and Unity
- **Issue:** tracking the user gestures from a side profile



Meeting 2

- Worked more with the unity package with the MS-development kit that we downloaded for Kinect. Worked out how the program was recognising gestures and looked deeper into the code that controlled it
- Downloaded another unity package called "Fighting motions" that has a "human like" model with fighting animations.
- We discussed how we can use the gestures from the unity MS-development kit to control the model from the fighting motions package
- This led us to realising that the MS-Development kit did not have gestures for punching, dodging, kicking etc. We would have to code up all new animations which would be extremely time consuming and a difficult task.
- We decided to check up if there is a better way to solve this problem with the Kinect V2 development kit. As it turns out there is and there are Microsoft tutorials about recording our own gestures and saving them in '.gbd' files. Not only that but the tutorials even demonstrate how to integrate these files into a unity project and what unity plugins are needed.
- With this new-found information, we decided that Kinect V2 is the device we want to use for our project and when we get it our next step is to learn how to record our own gestures and get them to trigger a fighting animation. This will with our first prototype for the game

Week 3 (20th Feb):

- Today we received the Kinect V2 and had a group meeting a few hours later to get ourselves familiar with it. There is a different SDK for it and we all downloaded it and set it up on our machines.
- We were then able to connect the Kinect to our laptops which was a smooth process
- Once this was done, we looked through some of the examples that came with the SDK such as the voice and depth sensing tools. On inspection, we found the 2.0 SDK was very similar to the 1.0 version but it seemed to have more examples of its functionality.
- At this point, we discussed what our next step would be and decided that we needed to learn how we are going to record the gestures needed for our game.
- We found the 'Kinect Studio' software which is used for recording gestures and we practiced making a punching gesture. This was done on the fly and we found it saved the gesture in a file in the Kinects default directory
- We realised we were going to need more information about how to use this file. After a bit of research, we found a helpful tutorial.

- This explained how we can get the software to learn the gesture and even improve its recognition of the gesture by giving it multiple different files of the same gesture. We learned here that to have a very accurate gesture it would need to be recorded using people of different heights and sizes.
- When we finished with the gestures we discussed what gestures we will need for the basic game and what extra gestures can be put into the game if time is on our side.
- We moved on then to looking further into some of the fighting animations we had already imported into Unity. We needed to know if that would be suitable for our base game and we found they would.
- The last thing we did in the meeting was to make a project plan for the next few weeks. We had already set up a sprint a week ago, which was for getting the needed gestures recorded. We had given ourselves 2 weeks to do that but nearly a week of that was taken up with waiting to get the Kinect V2, however, we still feel like we can get the required work done for that sprint in time.
- The next sprint after will involve getting the gestures to trigger the animations in Unity. We didn't want to plan too much further as we need to see if all the work gets finished in time but if it is then the final 2 weeks will be making the game around the working gestures and animations.

Week 4 (27th Feb):

- We recorded a right punch gesture using Kinect Studio contained within the Kinect SDK 2.0.
- We then built those gestures in the Visual Gesture Builder marking the appropriate points of the recorded gesture.
- We then built the database so that we could train the application to detect this gesture from other users.
- This was done using the built-in machine learning in Kinects Visual Gesture Builder.
- We used different body types and punching styles and trained the application multiple times with each body type.
- We then recorded gestures to ensure that the punching gesture was being recognised at a high percentage of confidence.

- Tested different types of punches against our created punch's (i.e. right hook and uppercut) to see if the Kinect could differentiate between the types of punches.
- We determined that the Right hook may not be possible as it is detecting a 30% confidence of a straight right punch during a right hook, meaning these separate gestures would be difficult to distinguish.
- Looked at hooking Unity up to the created gestures so we could play a punch animation when the right punch was detected.
- Defined what other gestures we will need (left punch, uppercut (left and right) block)
- Started to record the left punch gesture.

Week 5 (6th Mar):

Meeting 1:

- Researched how to use Kinect and Unity together using C#
- Connecting the Kinect to Unity by using C# code to get a handle on the Kinect Sensor and give feedback when connected
- Trying to get the Kinect to recognise a person in the camera and display output to the Unity console when this occurs.
- Lots of debugging

Meeting 2:

- Getting a Tracking ID for each person who is in the camera
- Trying to get Kinect to recognise our punch gesture and give feedback when this occurs
- This included a lot of trial and error with the code we were using and took a while to understand
- A major issue we ran into was Unity crashing randomly especially when adding breakpoints to the C# code at certain points
- Another issue we ran into was the name of the Gesture not matching the Gesture ID in the code. This meant that although we were recognising a gesture, we were never getting to the gesture in the if statement
- This was due to unclear/no documentation on the Kinect leading to confusion and debugging
- Updates the Github repository with code

Week 6 (13th Mar):

Meeting 1:

- Researched how to track 2 separate players when using the Kinect and the various methods of tracking users (i.e. closest player, first player tracked, most active).
- Researched how to connect the C# Punch gesture to the animation we have for a punch in Unity.
- Added code to Github and updated our Issues.
- Researched the various ways of tracking 2 users (Tracking ID, Skeleton ID and Player ID).
- The API we are using doesn't allow us to get a handle on the skeleton ID meaning we have to use the Tracking ID
- Got the punch gesture working with the Unity animation.
- We ran into an issue where each punch would result in the Unity character punching twice.
- This was due to the animation method being called for each frame of the punch that was above a certain confidence level.
- We solved the issue by adjusting the timing of the animation to be slower meaning that the animation would play once regardless of all the method calls.

Meeting 2:

- Worked on getting a second players punch being recognised
- Added a second unity character
- Determining which player would control which character and what we would do if we lost the tracking ID of a player (i.e. they step out of the camera)
- Got second players punch being recognised and mapped to the character
- Both characters are now controlled by users and punch animations are played when the user punches
- Researched how to add Collison to characters in Unity and detecting collisions against the character (i.e. a punch)
- Defined the scoring system of the game and the scene we would use
- An Issue we had was trying to access one of our created unity scripts as we wanted to get a reference to it
- unity has its own way of getting access to those scripts (code snippet here) we were trying to get a handle on the script, we needed to get a handle on the game object and then the script

Week 7 (20th Mar):

Meeting 1:

- Added Unity scene for the UI
- Added collision to characters
- Brought in new assets for the environment
- Displayed a message when collision is detected between the players
- Mesh collider wasn't tracking his hand, it was tracking his entire body, when the character extended his arm to punch the mesh collider wasn't extending out with the hand, it was stuck to the body.
- This was solved by using a sphere collider and attaching it to the hand of the character
- Got one player punching from one person (i.e. one player could only control one person when two people are in frame)
- Started work on having to players, no solution as of yet

Meeting 2:

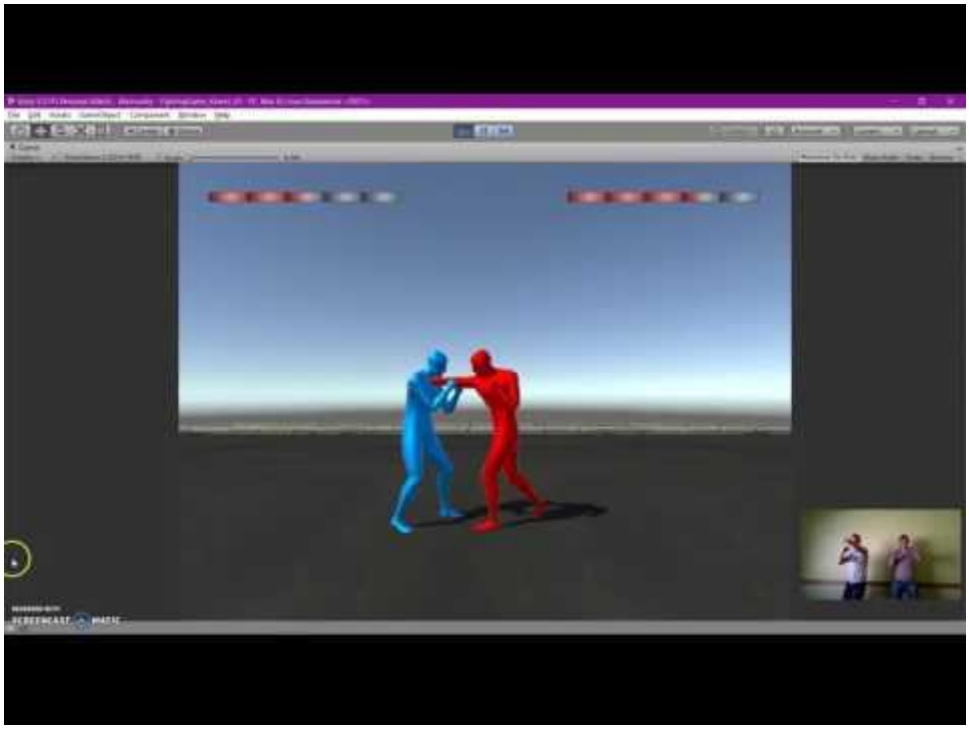
- Got 2 separate players punches being track and recorded a screen cast to show this
- Discussed general Game design and how we would go about it
- We defined how many points each type of punch would take off (i.e. straight punch 5 points, upper cut 7 points, may need to be re-balanced later on)
- We defined what the game would look like, how the health bar would look, what the backdrop would look like, how the menu would work etc.
- Defined plan for development before deadline
- Set up new sprint for final week and a half and added Github Issues for this
- Updated GitHub code
- Created left punch gesture in Kinect Studio, trained it and tested it to ensure we could detect the gesture with high confidence (used 3 different people)
- Created block gesture in Kinect Studio, trained it and tested it to ensure we could detect the gesture with high confidence (used 3 different people)
- Need to add health system to the game where each fighter starts with 100 health and different punches take off different amounts of health
- We need to create more gestures like Upper cut and a way to navigate the menu's
- Work on game design/functionality
- We discussed what a block gesture would look like, what would be a distinct gesture that we could detect that would make sense to the user.
- We then defined what this gesture would be
- We wanted the gesture to be intuitive to the user, we also followed real boxing style, a boxer would block in this way. This gesture was very distinct from the punches, this was important to not confuse the Kinect with gestures

Meeting 3:

- Worked on block gesture in order to recognise the gesture better (higher confidence)
- Tested punches into blocks to ensure the user couldn't abuse the block feature
- Added left punch database to Unity
- Added code for left punch
- Added animation for left punch
- Added block database to Unity
- Added code for block
- Added animation for block
- Tested and changed left punch gesture
- Tested and changed block gesture
- Researched health bars in Unity
- Created Left Kick gesture
- Created Right Kick gesture
- Got block gesture working with both users
- Got left kick gesture working with both users
- Got right kick gesture working with both users

Meeting 4:

- Tested out health bar from a sample project
- Understood and added this code to our project
- Took out redundant code from the sample project
- Working out how to detect who hit who and the conditions that will have to be met to take health away from the players.
- Created plan for the document based on a class we had
- Got one gesture working with health (takes health when one punches the other)
- Have a bug where health bar wasn't filling up fully
- Needed to have the dimension values of the health bar to the power of 2
- Issue where player is being deducted health for punching their opponent
- Got health bar working with one players punch
- Issue where multiple hits occurring in one animation
- Added timer to fix the issue
- Now have 2 separate health bars working and collisions working with both punching gestures



Week 8 (27th Mar):

Meeting 1:

Today new assets were added into our project. We used these for the environment as the game was looking very bare without them. We now have set up the stage to look like an alley fight behind some buildings. Other props were also added to the scene.

The kick gestures were given damage completing all the gestures that are used to deal damage in a fight.

The camera has been readjusted to give the players a better view of the fight. We discussed the view the player should have and we took influence from other famous fighting games such as Street Fighter, Mortal Combat, Injustice and Killer Instinct. We also took into account the 3D models we have and how they fit into the scene. The camera angle we decided on is a side view that allows the players to have an equal view with the camera slightly zoomed out but no so much that the fighters are too small on screen. As the fighters are 3D models we've used 3D models for the environment and this have added depth to the scene which we feel is a nice fit for the game.

Meeting 2:

- Added a start screen to the application
- Added the appropriate lighting to the scene in Unity
- User can now raise hand to start the game
- Need to determine which character controls each player
- Need to fix positioning of health bars
- Glitch where lighting had weird shadows after application.loadLevel()
- Found a fix for this issue online that is linked in references

Meeting 3:

- Got 4 people of different heights from the class to come and test out the application, this was to ensure that the game worked with other users
- Confidence level had to be lowered to recognise other people's gestures
- Note from testing, Kinect has to be elevated to recognise taller people
- Added custom font to the start screen
- Added background Image to the start screen
- Added prompt to start to the start screen
- Added hand image to go with the prompt
- Main menu is now complete!!
- Added block feature where there is a chance that you won't take damage when blocking
- This means that the block cannot be abused
- Changed KinectManager.cs to better detect players
- **BUG: if gesture is held the animation is played repeatedly**

Meeting 4:

- Added music to the main menu and game
- Add sound effects to punches and kicks
- Improved health bar
- Added Damien as a member to our organisation
- Added a knockback gesture when the opponent gets hit
- Added colliders to the character's feet
- Created code to determine when a user wins
- Added knockdown animation
- Losing player now gets knocked down when their health bar depletes
- Added starting timer to the game to allow Kinect to open
- Disabled the relevant scripts until this was completed
- Issue: player could still punch after they had been knocked down
- We added a timer to disable the animator after a player had lost, this solved the issue
- Added draw condition, where when the timer hits 0, the game ends up with a draw
- Added gesture to get back to main menu

Meeting 5:

- Added text to show the user who won
- Issue where unity crashes after 5^{th} restart of the game
- Fixed the restart on draw and win as it was occurring randomly
- Stopped timer from running when someone wins
- Players fight randomly after restart
- Worked on tracking players who enter and leave the frame
- We tried to allocate the player a person regardless of them leaving and coming back
- This is a known issue with Kinect that we came close to fixing
- We just ran out of time at the end
- Created a win gesture
- Added the win gesture to the application
- Disabled punches and kicks when a player has won
- Balanced health system
- Took screenshots
- Created a class diagram
- Recorded video of application
- Created executable of application
- Added Boolean to prevent random menu skipping, player now has to do win gesture before they can proceed
- Coding is now completed for the project
- Took out redundant code

Conclusions and Recommendations

From this project, we can conclude that the Kinect does not contain sufficient documentation on the Kinect api for Unity. This led to days of research in trying to figure out how the api worked and how we could change it to do what we wanted. We noticed throughout the development of our project that the Kinect has trouble in determining certain gestures when the users stand to close to eachother. For example, a left and right kick when the users are standing side by side. An issue we came across during our testing is that our created gestures had trouble recognising new users. We had to lower the confidence level of the gesture to ensure that we could detect new users punches and kicks.

We noticed that the gestures that we created had to be definitive, meaning that small movements would not be picked up. This led to us having trouble with the block gesture as it is not as definitive as a kick or punch. With extensive training of the system we were able to detect the gesture. We ran into issues with Unity crashing when you try to close down the application. Finally, the tracking system when using the tracking ID made it very difficult to make multiplayer games. A new tracking ID is assigned to each person who enters the frame, making it very difficult to recognise when someone leaves and enters the frame again.

Overall the Kinect has good functionality and has the ability to track users all the way down to their hand movements. This is a technology that could be utilised in the future outside of the Kinect. If Microsoft can improve on this technology, we believe that it has the potential to have applications in people's day to day lives. An example of this would be the ability to control a car radio using a hand gesture, without having to take your eyes off the road. However, they do need to improve on the current Kinect and address the drawbacks that we have highlighted above.

The recommendations we would make for this project are to be definitive in the gestures that you make. Make it really easy for the Kinect to recognise the gesture that you create by having lots of movement in the gesture. We would also recommend that you train the system with many different body types (small and tall people). We would recommend that you set the confidence level to detect a gesture in your code to be low, this is to ensure that you can detect the different ways that people do different gestures. Finally, we would recommend that you store all gestures within the same database rather than storing each gesture in its own database.

User Interface

Main Menu

Start of the game

An example of a kicking gesture

End of game with a winner

Video of game

Below is a video demonstration of our final game

References

Unity Assets

Fighting Motions Vol 1.

- Magicpot Inc.
- https://www.assetstore.unity3d.com/en/#!/content/76699

Mobile Buildings

- https://www.assetstore.unity3d.com/en/#!/content/62960

Health Bar

- Game Bar Creation System
- https://www.assetstore.unity3d.com/en/#!/content/2806

Low Poly Street Pack

- https://www.assetstore.unity3d.com/en/#!/content/67475

Music and Sound Effects

- TS Sounds
- Tore Group
- https://www.assetstore.unity3d.com/en/#!/content/34453

Fonts

- Freedom
- http://www.fontspace.com/hxdes/freedom

Creating Kinect Gestures:

- https://channel9.msdn.com/Blogs/AdamTuliper/Custom-Gestures-with-the-Kinect-v2-for-Windows

Kinect and Unity

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

Useful Kinect Tutorials

- http://kinect.github.io/tutorial/

Multiple Collison Issue Solution

- http://answers.unity3d.com/questions/897505/preventing-multiple-collisions-on-same-target.html

Lighting Glitch Solution

- http://answers.unity3d.com/questions/919940/applicationloadlevel-changes-lighting-for-some-rea.html

Coloured text

- http://gamedev.stackexchange.com/questions/92149/changing-color-of-ui-text-in-unity-into-custom-values

Known bugs

- When gesture is held, the character keeps punching/kicking
- Block gesture turns off collisions
- Unity crashes after 5th restart of game
- The characters fight by themselves after the 3rd restart of game
- Game exits to main menu itself without using the users gesture occurring
- If the player who lost makes a gesture he will get up from the knockdown state

"Special Features"

- CPU vs CPU battles after 3 restarts
- Dynamic menu exiting
- Resurrection mode after death
- Training mode (collisions turned off)
- Dynamic player switching
- Incremental breaks after 5 games