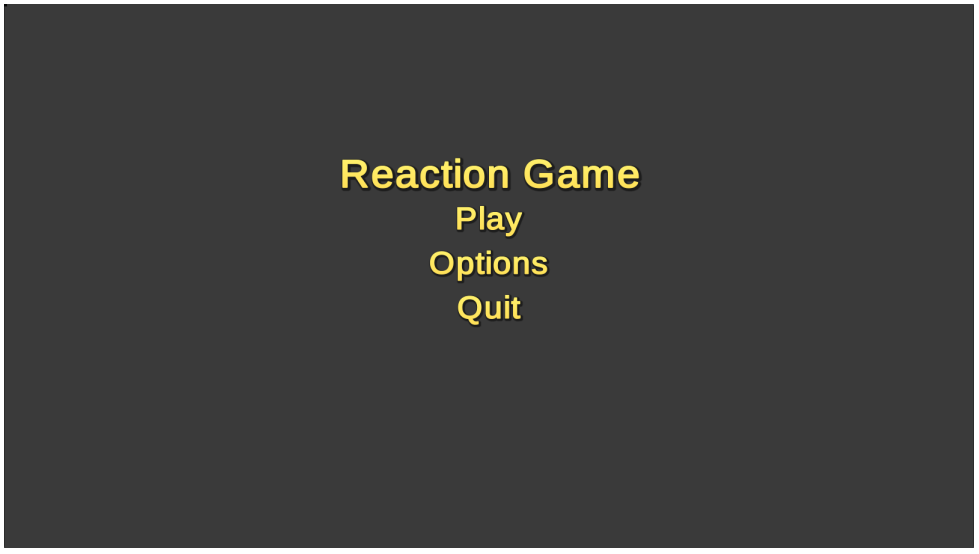
Gesture Based User Interface Development
Report

Reaction Based Kinect Game

Submitted By :
Colm Woodlock G00341460



Contents

1 Purpose of the application

The purpose of this application is pretty simple, what I wanted to do was create a reaction based game similar to the one pictured below. As you can see with the one picture below it is a physical machine and is quite large, I wanted to create a more portable version of this where the only hardware needed was a Kinect 2.0 camera and a computer. My main goals with this was to create an adaptable game space that could be configured to your surroundings and to get accurate readings on when the users hands entered the scoring zones.

2 Research Phase

During the research phase, I researched real versions of this reaction game and also virtual examples. These games are designed to test concentration levels, hand-eye co-ordination, peripheral vision and reaction times of athletes. One such example I found during my research was the use of the Dynavison D2 at the IMG Academy, a very popular training academy for athletes in the US. Below is a screen grab from a YouTube video they have showing their Dynavison machines.

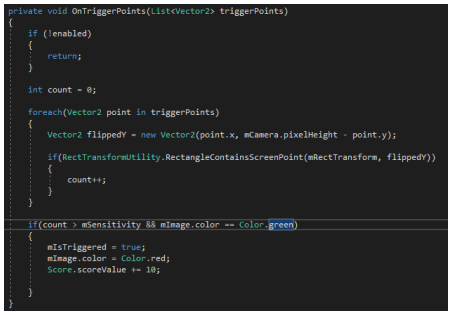
As you can see in the image the athlete is presented with a large array of physical buttons which light up in sequence and the goal is to push as many buttons as possible in an allotted time.

Another example of a reaction wall that I found online was one called the EyePlay Wall. This was a screen which was projected onto a wall and could be used by children to play games. Unlike the Dynavision D2 this was not designed for athletes but rather aimed a children to get them active. Below is an image of the EyePlay Wall.

In this image you can see the child is bursting balloons on the projected surface on the wall as opposed to physical buttons. This leads to a more visual pleasing experience.

3 Gestures identified as appropriate for this application

The gestures I plan to use for this application are pretty simple. By using the raw colour and depth data from the Kinect I will be able to apply tracking points to the person standing in front of the camera. Using these tracking points I will be able to determine if a significant amount of the user's hand is inside the rectangle in this case that will act as the goal to "touch" in order to score a point. In the code snippet below it shows the function I used to check to see how many trigger points were inside the rectangle to cause it to trigger it as a successful score.



```
private void OnTriggerPoints(List<Vector2> triggerPoints)
{
    if (!enabled)
    {
        return;
    }

    int count = 0;

    foreach(Vector2 point in triggerPoints)
    {
        Vector2 flippedY = new Vector2(point.x, mCamera.pixelHeight - point.y);

        if(RectTransformUtility.RectangleContainsScreenPoint(mRectTransform, flippedY))
        {
            count++;
        }
    }

    if(count > mSensitivity && mImage.color == Color.green)
    {
        mIsTriggered = true;
        mImage.color = Color.red;
        Score.scoreValue += 10;

    }
}
```

There is an if statement which checks if the number of trigger points within the scoring zone is greater than the sensitivity, which is used to set the game space, and if it is the active scoring zone it then triggers a successful score and changes the colour and adds to the score.

4 Hardware used in creating the application

The hardware I used when creating this game included the Kinect 2.0. The Kinect is a motion sensor add on to the Xbox One. the Kinect 2.0 is capable of processing 2 gigabits of data per second this offers much better accuracy on movements and recognising gestures over the previous generation model. The camera also captures 1080p video which was a major point for my game, as I wanted to display the live environment that the user was playing in on screen at all times.

Another feature of the Kinect that I wanted to sue in my project was the depth sensor. The depth sensor. The depth sensor contains a monochrome CMOS sensor and an infrared projector that help create the 3D imagery in the room. It measure the distance of each point of the players body by transmitting the infrared light and measuring its time of flight after it reflects off the objects.

By combining the raw video feed and the depth data I was able to create an adaptable game space that can be controlled by the user to select a 3D area where they will be tracked.

I had considered other hardware that was available to me. One such item was the leap motion controller. The leap motion controller is a sensor device that supports hand a finger motions as input. It didn't take long for me to rule this out as an option as as I previously stated I wanted raw live video footage showing for the user and also the accuracy in quick movements was quite poor from some of the research I did on the device the game space was also quite small for what I wanted to achieve with the reaction game.

Another device that I considered was the Myo Armband. The Myo Armband is a device that lets you use the electrical activity in your muscles to wirelessly control your computer over a bluetooth connection. Again I ruled this piece of hardware out quite early in the research process as I found that it was not very responsive to quick jerky motions and again didnt provide any raw video footage.

5 Architecture for the solution

5.1 CountdownTimer

In this class I declare 3 variables, sceneToLoad, timer and timerSeconds, I also have two functions Start and Update.

This function is used to control the countdown timer that will be used in the main game scene. In the start function I am getting a Text component from UnityEngine.UI this will be the text that is shown on screen. In the Update function I am then assigning the deltaTime function to timer and then setting the deltaTime as a float to the Text object. Finally there is an if statement to trigger when the timer reaches 0 which then loads the new GameOver scene.

5.2 GameOver

The GameOver script is pretty simple it has 3 functions, RestartGame, LoadMenu and QuitGame. All of these functions will be used by OnClick events with buttons on the Game over scene.

5.3 MainMenu

The MainMenu script has two functions PlayGame and QuitGame. PlayGame starts the game by loading the Main scene which contains the main game and QuitGame ends the game, it also prints to the console for testing purposes to show the button has been pushed.

5.4 SpawnRects

SpawnRects has 5 variables that I declare at the top. A GameObject array to store the rectangle prefabs that will be used for scoring, and array of colours, randnum for storing a random number, seconds to define the wait period between changing the active scoring zone and a boolean value for the while loop.

It also has a function Start and a CoRoutine NumberGen. In Start we place the Scoring zones in the array on the screen. In the CoRoutine NumberGen we wait for the predefined amount of seconds before selecting a random scoring zone and changing it's colour to green which signifies it is an active scoring zone. It then

waits for a predefined amount of seconds before changing the colour to black to show it is an inactive scoring zone.

5.5 ReloadScene

ReloadScene is very simple it controls the button in the main scene which restarts the whole scene.

5.6 Score

Score is a script which controls the text which represents the score on the main screen. it has two variables declared at the start, scoreValue and score. It also has two functions Start and Update. In Start we get a handle on the text object that will be used for the score on screen and sit it to the variable score. Then in Update we add the predefined value for score onto the text object.

5.7 ImageViewer

In ImageViewer we have four variables mMultiSource, mMeasureDepth, mRawImage and mRawDepth. mMultisource is used to get access to the kinect packages that I downloaded from the Microsoft website. One of the files that was contained within the packages was the MultiSourceManager this will give me access to the raw image data and the raw depth data. mMeasureDepth will be used to measure the depth at which the subject is in the frame. mRawImage and mRawData are used to get the raw video feed and the raw depth data feed information.

There is one function and that is Update, in Update we are first outputting the raw image data from the Kinect to the screen and then doing the same with the depth data. If you were to run this as it is you would notice that the video is flipped upside down, this is because the Kinect by default shows it's live footage upside down so I had to create a custom texture to flip it.

5.8 MeasureDepth

The majority of the work is done in this function. There are many variables but I will only cover the important ones to the function of the class. I first declare trigger points these will act as the points that will cause a successful score in the game. I

have a number of arrays to store the tracking points from the depth data and from the raw image. I also have lists to store the number of active valid points within a scoring zone and the number of trigger points in a scene. I also initialise the mapper which is used to combine the depth data and the raw image data. The depth data by default is 512x424 and so I create that variable also.

I have an number of functions and will summarise what they are used for. In the Awake function we set up everything we need, this includes the camera, the mapper and the depth sensor. In the Update function we are building the game space where the tracking points will exist. This is done either invisibly or I added a key press that if you press the space bar it will show the active game space and the tracking points. The set functions are all used by sliders in the game to set the dimensions of the game space which can be sen by pressing space bar and also the depth at which the game space exists and also the sensitivity of the game space.

OnGui is a function that is mainly used in testing it displays the game space and the trigger points in the game. This gives the user a better understanding of what is going on.

DepthToColour is used to handle the mapping of the depth data and the raw image feed and also when I was having issues with performance and accuracy I decided to cut down the number of tracking points by only using every eight point. Down sampling was also used, this lowers the accuracy of the application but greatly improves the performance. for a simple game like this high accuracy was not needed.

FilterToTrigger is a function that takes all the points that are within the game space and turns them into trigger points to be used when trying to identify if they are within the score zone.

The next few functions such as CreateRect and ScreenToCamera are mainly used to give the user a visual representation on screen of what is happening in the other functions.

5.9 RectTrigger

In RectTrigger I set the sensitivity to what a successful amount of trigger points within the scoring zone is. I then have a boolean value to decide if the scoring zone was triggered or not. In the Awake function it creates the trigger points and then when they leave the game space the function OnDestroy, destroys them. The function OnTriggerPoints controls what happens with the trigger points and has an if statement that when the parameters are met adds to the score.

6 Conclusions Recommendations

6.1 Conclusions

In conclusion of this project I can definitley say I learned a lot working with the Kinect SDK and packages and trying to figure out how to do certain things without much documentation online as the Kinect has lost all support form Microsoft recently, not to mention the fact that not many people were using it in development either so there is hardly any resources available online.

I also had some issues where the amount of data points from both the colour space and the depth data was causing my laptop to crash but I figured that out with only rendering every eight data point and also adding the space bar to update the game space rather trying to update it in real time.

6.2 Recommendations

Some Recommendations I would have would be the addition of some extra levels to the game would be of benefit to the game.The addition of a pause menu might also be of use. And finally being able to identify trigger points from only the hands might also be a feature I would add to the game.