FINAL DESIGN DOCUMENT

MULTIMEDIA SYSTEMS CS6331.001

PHANTOM LIMB PROJECT

BY:

KIA KHADEM **kck130030**

SUMIN REDDY GUJJA **sxg172130**

Game development for phantom limb project

PROJECT STATEMENT:

The goal of this project is to develop an interactive and fun game to be used as a mechanism therapy for amputees who might be suffering from a missing limb pain known as the phantom limb pain and integrate it with the phantom limb project currently being developed at the lab.

PROJECT INTRODUCTION:

Phantom limb pain (PLP) is not an actual pain but the sensation of it which is generally caused by an amputation in a person, the persons brain still interprets and sends signals to a limb which is no longer there. Most of the amputees experience this pain commonly after they have had a surgery. Now it has been widely researched that PLP can be best alleviated by using non-medical methods. Keeping this in mind we have developed a unique game which serves the purpose of therapy when played by the patient.

IMPLEMENTATION:

The game was majorly developed upon the concept of mirroring, mirroring works very effectively because it creates a reflective illusion of an affected limb to trick the brain into thinking a movement in the game has occurred without pain.

Coming to the gaming environment itself which was created in unity with the help of 'C#' scripting it has a player environment where in the patient can be seated on a chair in a virtual room and step on small spherical molds on a platform which were created using the virtual environment. The objective of the player is to step on as many spheres as he can to increase his score the molds pop up in different colors and randomly in a grid space making the game more challenging and interactive.

The interactions of the actual human body with the game environment was achieved using a Kinect, Kinect helps us to capture the human motion by covering the room with constant, predetermined pattern of infrared dots it helps obtain a depth and color image which is then processed using a random forest method to obtain the skeletal image of the person standing in front of it. we have written a script to detect any skeletal collision on the virtual object and increase the score, we have also implemented a difficulty level adjustment along with the position adjustment maker to start the game.

MY ROLE:

I didn't know what to write I just mostly wrote that we both have done the work collectively and I wrote down the piano script in my part of the code that we will be explaining and in your part I have included the whole color master script.

//I have included your code and stuff I also put down the references
// you just need to add how we have worked out the project together and your role in it column
// write about the whole color master code and how you have done it using tags and the game object spheres used and collisions obtained with the human and the virtual environment.
// also just write about debugging and stuff, etc.

CODE:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class ColorMaster : MonoBehaviour
{

    public int tileColor;
    public float timeInterval = 2f;
    public float lengthCheck = 0;
    public GameObject[] tagFinder = new GameObject[12];
    int mouseCheck = 0;
    // Use this for initialization
    void Start()
    {
        //initialize random number generator, this will create a number that
corresponds to tile numbers.
        tileColor = Random.Range(1, 6);

    }

    // Update is called once per frame
    void Update()
    {
        //the game will begin once the left mouse button is clicked.
        if(Input.GetMouseButton(0))
        {
            mouseCheck = 1;
        }
        if(mouseCheck == 1)
        {
            if (Time.time >= lengthCheck)
            {
                //print("the current time is: " + Time.time);
                //print("tileColor is: " + tileColor);
                // print("LengthCheck's value is: " + lengthCheck);



                switch (tileColor)
                {
                    case 1:
                        tagFinder[0].SetActive(true);
                        tagFinder[0].GetComponent<Renderer>().material.color =
Color.red;
                        tagFinder[3].SetActive(true);
                        tagFinder[3].GetComponent<Renderer>().material.color =
Color.red;
                        tagFinder[1].SetActive(false);
                        tagFinder[2].SetActive(false);
                        tagFinder[4].SetActive(false);
                        tagFinder[5].SetActive(false);
                        tagFinder[6].SetActive(false);
                        tagFinder[7].SetActive(false);
                        tagFinder[8].SetActive(false);
                        tagFinder[9].SetActive(false);
                        tagFinder[10].SetActive(false);
                        tagFinder[11].SetActive(false);

                        /*tagFinder[4].GetComponent<Renderer>().material.color
= Color.clear;
                        tagFinder[5].GetComponent<Renderer>().material.color =
Color.clear;
                        tagFinder[6].GetComponent<Renderer>().material.color =
Color.clear;
                        tagFinder[7].GetComponent<Renderer>().material.color =
Color.clear;
```

```
                    tagFinder[8].GetComponent<Renderer>().material.color =
Color.clear;
                    tagFinder[9].GetComponent<Renderer>().material.color =
Color.clear;
                    tagFinder[10].GetComponent<Renderer>().material.color =
Color.clear;
                    tagFinder[11].GetComponent<Renderer>().material.color =
Color.clear;*/
                    break;
                case 3:
                    tagFinder[4].SetActive(true);
                    tagFinder[4].GetComponent<Renderer>().material.color =
Color.yellow;
                    tagFinder[7].SetActive(true);
                    tagFinder[7].GetComponent<Renderer>().material.color =
Color.yellow;
                    tagFinder[0].SetActive(false);
                    tagFinder[1].SetActive(false);
                    tagFinder[2].SetActive(false);
                    tagFinder[3].SetActive(false);
                    tagFinder[5].SetActive(false);
                    tagFinder[6].SetActive(false);
                    tagFinder[8].SetActive(false);
                    tagFinder[9].SetActive(false);
                    tagFinder[10].SetActive(false);
                    tagFinder[11].SetActive(false);

                    /*tagFinder[0].GetComponent<Renderer>().material.color
= Color.grey;
                    tagFinder[1].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[2].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[3].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[4].GetComponent<Renderer>().material.color =
Color.red;
                    tagFinder[5].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[6].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[7].GetComponent<Renderer>().material.color =
Color.red;
                    tagFinder[8].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[9].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[10].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[11].GetComponent<Renderer>().material.color =
Color.grey;*/
                    break;
                case 4:
                    tagFinder[5].SetActive(true);
                    tagFinder[5].GetComponent<Renderer>().material.color =
Color.yellow;
                    tagFinder[6].SetActive(true);
                    tagFinder[6].GetComponent<Renderer>().material.color =
Color.yellow;
                    tagFinder[0].SetActive(false);
                    tagFinder[1].SetActive(false);
                    tagFinder[2].SetActive(false);
                    tagFinder[3].SetActive(false);
                    tagFinder[4].SetActive(false);
```

```
                    tagFinder[7].SetActive(false);
                    tagFinder[8].SetActive(false);
                    tagFinder[9].SetActive(false);
                    tagFinder[10].SetActive(false);
                    tagFinder[11].SetActive(false);
                    /*tagFinder[0].GetComponent<Renderer>().material.color
= Color.grey;
                    tagFinder[1].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[2].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[3].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[4].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[5].GetComponent<Renderer>().material.color =
Color.blue;
                    tagFinder[6].GetComponent<Renderer>().material.color =
Color.blue;
                    tagFinder[7].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[8].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[9].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[10].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[11].GetComponent<Renderer>().material.color =
Color.grey;
                    */
                    break;
                case 5:
                    tagFinder[8].SetActive(true);
                    tagFinder[8].GetComponent<Renderer>().material.color =
Color.blue;
                    tagFinder[11].SetActive(true);
                    tagFinder[11].GetComponent<Renderer>().material.color =
Color.blue;
                    tagFinder[0].SetActive(false);
                    tagFinder[1].SetActive(false);
                    tagFinder[2].SetActive(false);
                    tagFinder[3].SetActive(false);
                    tagFinder[4].SetActive(false);
                    tagFinder[5].SetActive(false);
                    tagFinder[6].SetActive(false);
                    tagFinder[7].SetActive(false);
                    tagFinder[9].SetActive(false);
                    tagFinder[10].SetActive(false);
                    /*
                    tagFinder[0].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[1].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[2].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[3].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[4].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[5].GetComponent<Renderer>().material.color =
Color.grey;
                    tagFinder[6].GetComponent<Renderer>().material.color =
Color.grey;
```

```
                        tagFinder[7].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[8].GetComponent<Renderer>().material.color =
Color.red;
                        tagFinder[9].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[10].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[11].GetComponent<Renderer>().material.color =
Color.red;
                        */
                        break;
                    case 6:
                        tagFinder[9].SetActive(true);
                        tagFinder[9].GetComponent<Renderer>().material.color =
Color.blue;
                        tagFinder[10].SetActive(true);
                        tagFinder[10].GetComponent<Renderer>().material.color =
Color.blue;
                        tagFinder[0].SetActive(false);
                        tagFinder[1].SetActive(false);
                        tagFinder[2].SetActive(false);
                        tagFinder[3].SetActive(false);
                        tagFinder[4].SetActive(false);
                        tagFinder[5].SetActive(false);
                        tagFinder[6].SetActive(false);
                        tagFinder[7].SetActive(false);
                        tagFinder[8].SetActive(false);
                        tagFinder[11].SetActive(false);
                        /*
                        tagFinder[0].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[1].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[2].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[3].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[4].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[5].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[6].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[7].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[8].GetComponent<Renderer>().material.color =
Color.grey;
                        tagFinder[9].GetComponent<Renderer>().material.color =
Color.blue;
                        tagFinder[10].GetComponent<Renderer>().material.color =
Color.blue;
                        tagFinder[11].GetComponent<Renderer>().material.color =
Color.grey;
                        */
                        break;
                }


                lengthCheck += timeInterval;
                tileColor = Random.Range(1, 6);
            }

            // quit the application once 5 minutes have passed.
```

```
            if (Time.time == 300.0f)
            {

                Application.Quit();
            }


        }
    }
}
```

REFERENCES:

- We have borrowed the 'HUNA' code from the lab (courtesy of multimedia systems and animations lab UT Dallas).
- https://docs.unity3d.com/Manual/index.html
- https://www.amputee-coalition.org/limb-loss-resource-center/resources-for-pain-management/managing-phantom-pain/
- https://en.wikipedia.org/wiki/Phantom_limb
- https://unity3d.com/learn/tutorials/s/scripting
- https://unity3d.com/learn/tutorials/topics/graphics/environment-details
- https://docs.unity3d.com/560/Documentation/Manual/PhysicsSection.html
- how to update every second: https://answers.unity.com/questions/122349/how-to-run-update-every-second.html
- Find with tag: https://docs.unity3d.com/ScriptReference/GameObject.FindGameObjectsWithTag.html