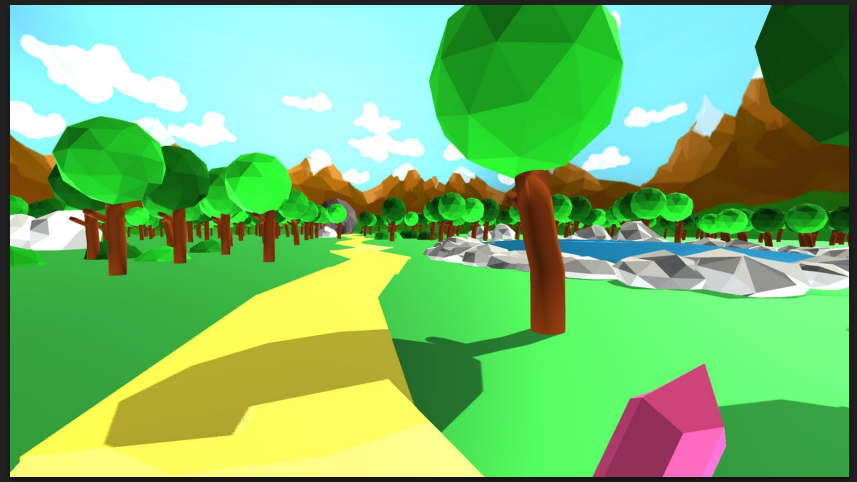
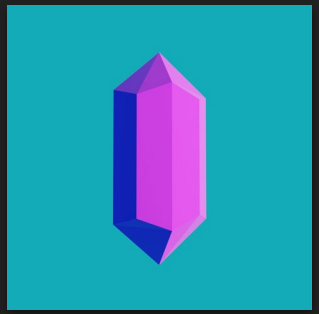
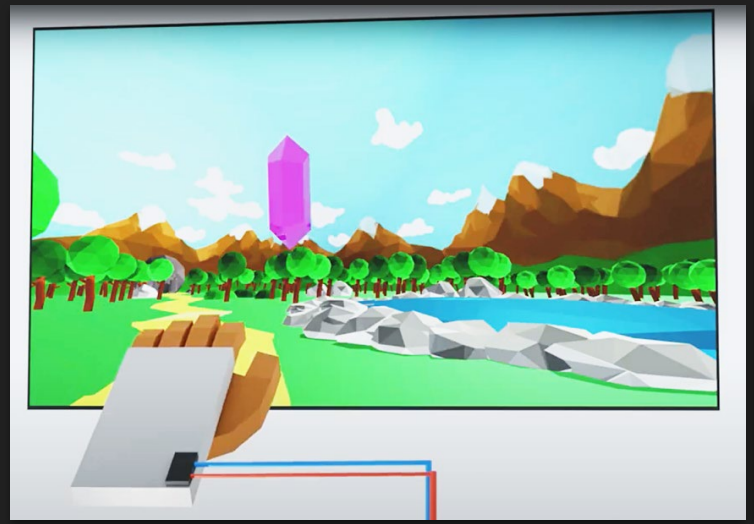




Spine is Fine

Aditya Sadavarte, Samyukta SJ,
Malay Vasa, Ishika Prasad

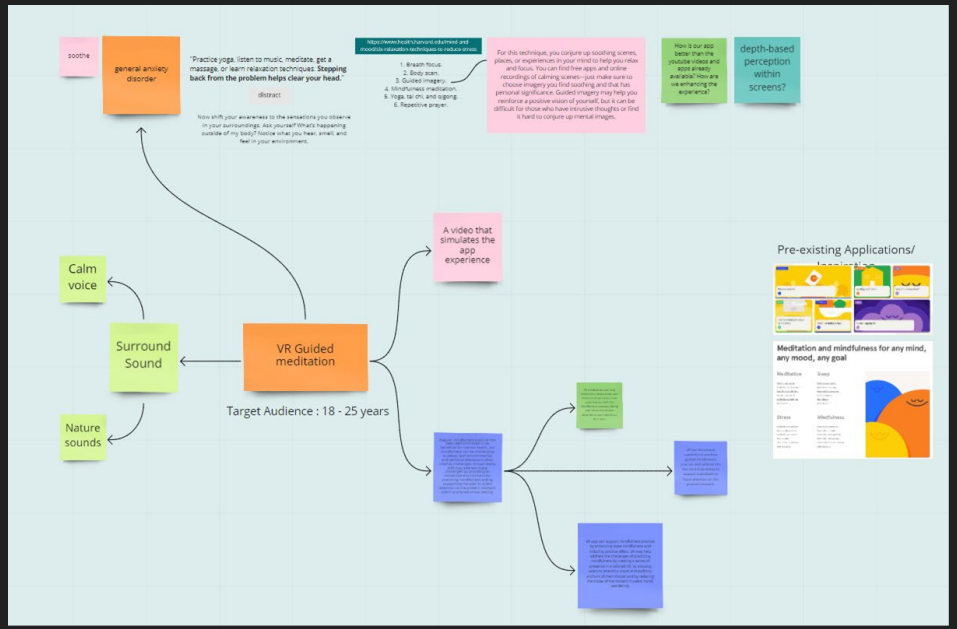


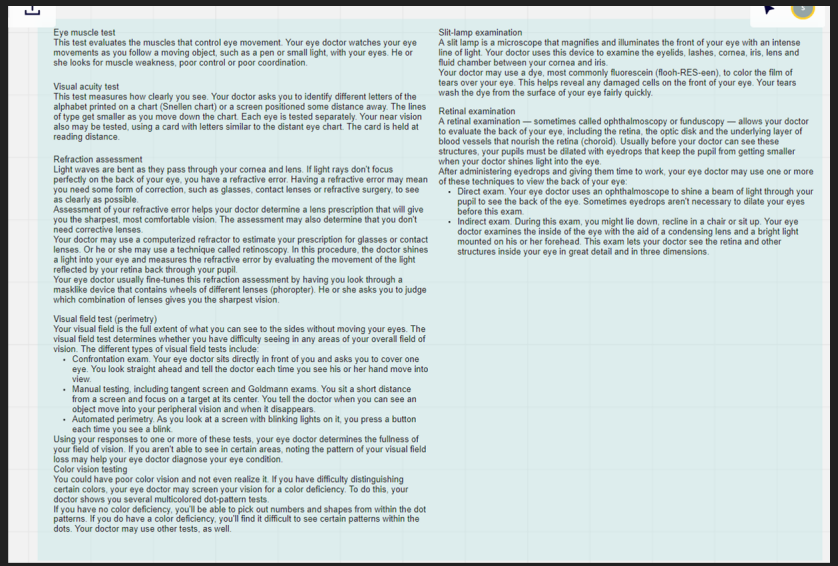
Index

Project Brief

The aim of our project was to gamify physiotherapy. The end stages of physio rehab usually comprises of monotonous exercises, that can leave the patient feeling unmotivated and can often hinder with the rehabilitation process. Our goal for this project, was to create an interactive and engaging game-based rehabilitation tool, that would provide some variation and appeal for patients undergoing physiotherapy, and thereby facilitate the recovery of their motor and cognitive functions. The VR Game that we have designed requires simple physical movement in order to complete the tasks. The tasks are structured and assessed according to the specifications of the *Berg Balance Scale.* The current prototype consists of 2 levels. The first level tests their hand-eye coordination and makes sure that the patient moves their muscles with whatever little residual motor function they have. The second level focuses on walking and the movement and synchronization of the entire body as a unit. After each 20-minute session, the patient will get a report on his/her phone linked to our app, that will give them details on the progress made as well as the report for each session based on the Berg balance scale test that can then be sent to a professional for assessment.

Iterations





Eye muscle test
This test evaluates the muscles that control eye movement. Your eye doctor watches your eye movements as you follow a moving object, such as a pen or small light, with your eyes. He or she looks for muscle weakness, poor control or poor coordination.

Visual acuity test
This test measures how clearly you see. Your doctor asks you to identify different letters of the alphabet printed on a chart (Snellen chart) or a screen positioned some distance away. The lines of type get smaller as you move down the chart. Each eye is tested separately. Your near vision also may be tested, using a card with letters similar to the distant eye chart. The card is held at reading distance.

Refraction assessment
Light waves are bent as they pass through your cornea and lens. If light rays don't focus perfectly on the back of your eye, you have a refractive error. Having a refractive error may mean you need some form of correction, such as glasses, contact lenses or refractive surgery, to see as clearly as possible.
Assessment of your refractive error helps your doctor determine a lens prescription that will give you the sharpest, most comfortable vision. The assessment may also determine that you don't need corrective lenses.
Your doctor may use a computerized refractor to estimate your prescription for glasses or contact lenses. Or he or she may use a technique called retinoscopy. In this procedure, the doctor shines a light into your eye and measures the refractive error by evaluating the movement of the light reflected by your retina back through your pupil.
Your eye doctor usually fine-tunes this refraction assessment by having you look through a masklike device that contains wheels of different lenses (phoropter). He or she asks you to judge which combination of lenses gives you the sharpest vision.

Visual field test (perimetry)
Your visual field is the full extent of what you can see to the sides without moving your eyes. The visual field test determines whether you have difficulty seeing in any areas of your overall field of vision. The different types of visual field tests include:

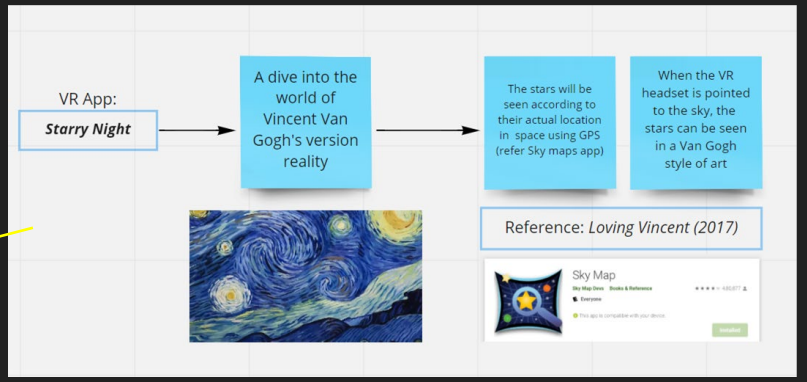
- Confrontation exam. Your eye doctor sits directly in front of you and asks you to cover one eye. You look straight ahead and tell the doctor each time you see his or her hand move into view.
- Manual testing, including tangent screen and Goldmann exams. You sit a short distance from a screen and focus on a target at its center. You tell the doctor when you can see an object move into your peripheral vision and when it disappears.
- Automated perimetry. As you look at a screen with blinking lights on it, you press a button each time you see a blink.

Using your responses to one or more of these tests, your eye doctor determines the fullness of your field of vision. If you aren't able to see in certain areas, noting the pattern of your visual field loss may help your eye doctor diagnose your eye condition.
Color vision testing
You could have poor color vision and not even realize it. If you have difficulty distinguishing certain colors, your eye doctor may screen your vision for a color deficiency. To do this, your doctor shows you several multicolored dot-pattern tests.
If you have no color deficiency, you'll be able to pick out numbers and shapes from within the dot patterns. If you do have a color deficiency, you'll find it difficult to see certain patterns within the dots. Your doctor may use other tests, as well.

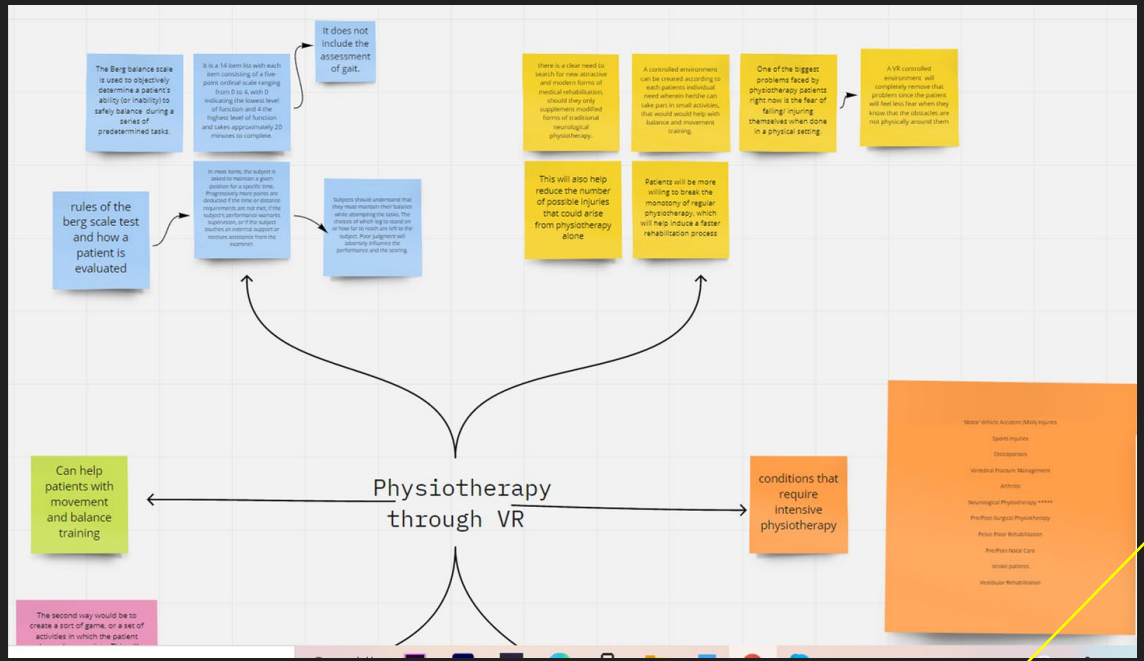
Slit-lamp examination
A slit lamp is a microscope that magnifies and illuminates the front of your eye with an intense line of light. Your doctor uses this device to examine the eyelids, lashes, cornea, iris, lens and fluid chamber between your cornea and iris.
Your doctor may use a dye, most commonly fluorescein (flooh-RES-een), to color the film of tears over your eye. This helps reveal any damaged cells on the front of your eye. Your tears wash the dye from the surface of your eye fairly quickly.

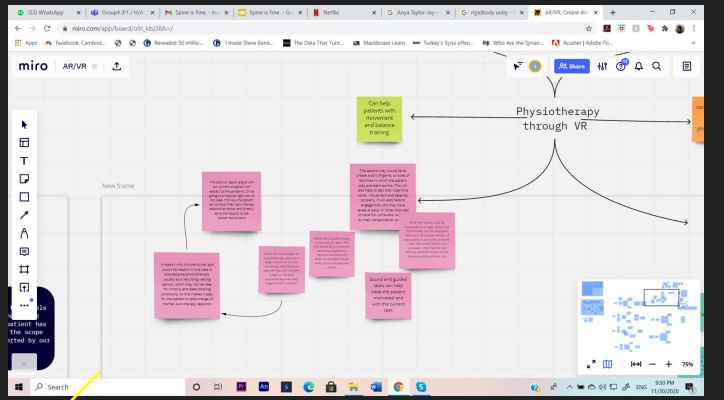
Retinal examination
A retinal examination — sometimes called ophthalmoscopy or funduscopy — allows your doctor to evaluate the back of your eye, including the retina, the optic disk and the underlying layer of blood vessels that nourish the retina (choroid). Usually before your doctor can see these structures, your pupils must be dilated with eyedrops that keep the pupil from getting smaller when your doctor shines light into the eye.
After administering eyedrops and giving them time to work, your eye doctor may use one or more of these techniques to view the back of your eye:

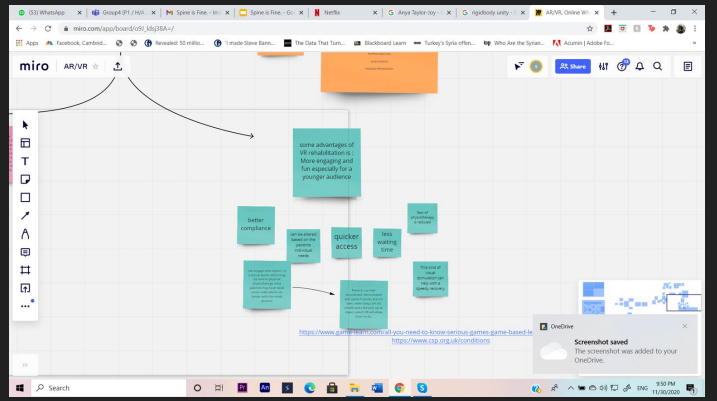
- Direct exam. Your eye doctor uses an ophthalmoscope to shine a beam of light through your pupil to see the back of the eye. Sometimes eyedrops aren't necessary to dilate your eyes before this exam.
- Indirect exam. During this exam, you might lie down, recline in a chair or sit up. Your eye doctor examines the inside of the eye with the aid of a condensing lens and a bright light mounted on his or her forehead. This exam lets your doctor see the retina and other structures inside your eye in great detail and in three dimensions.



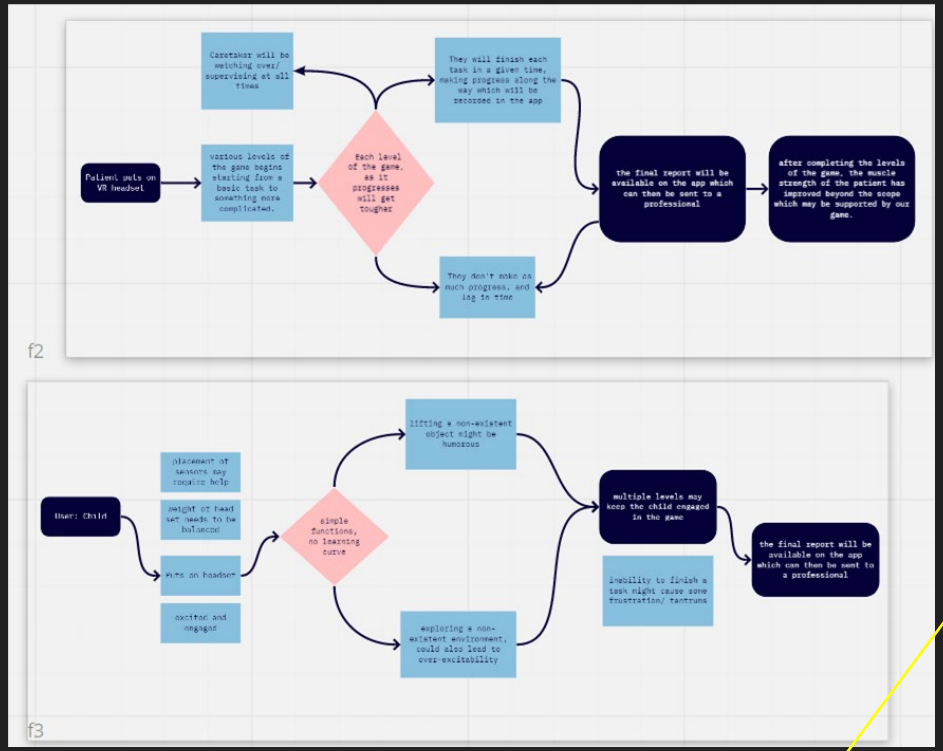
Iterative process to decide the concept for the project

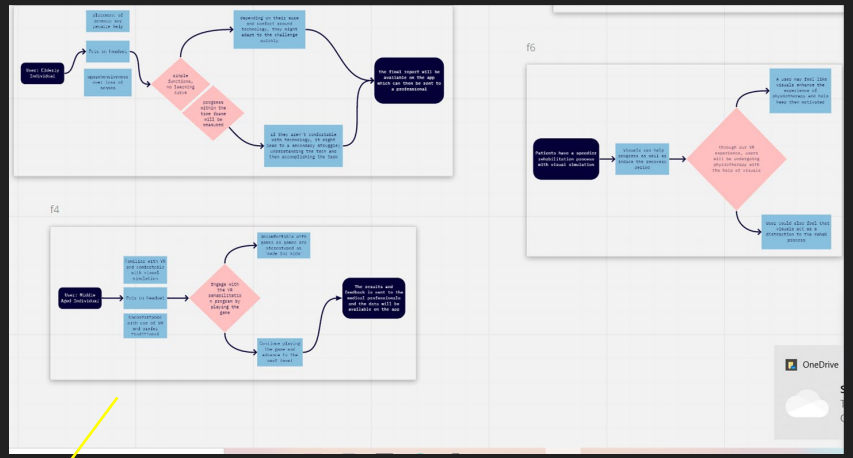


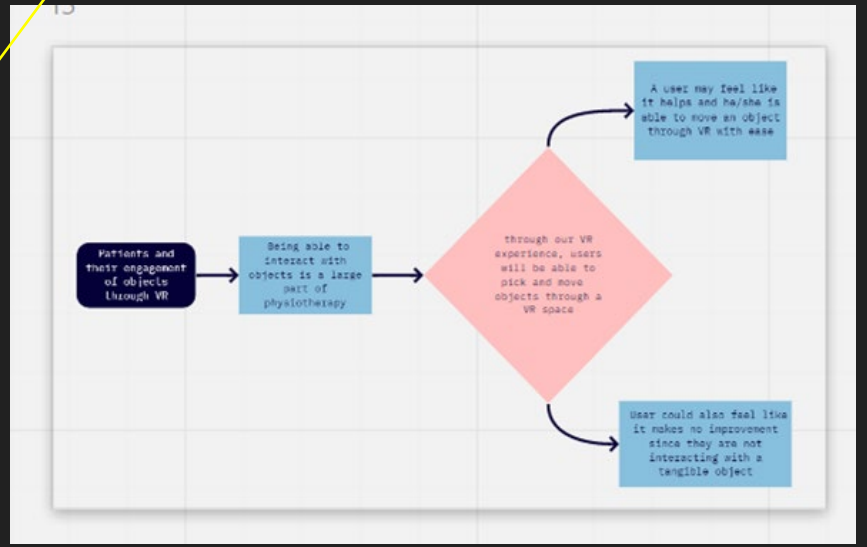




MIND MAP FOR THE CONCEPT WE WENT AHEAD WITH



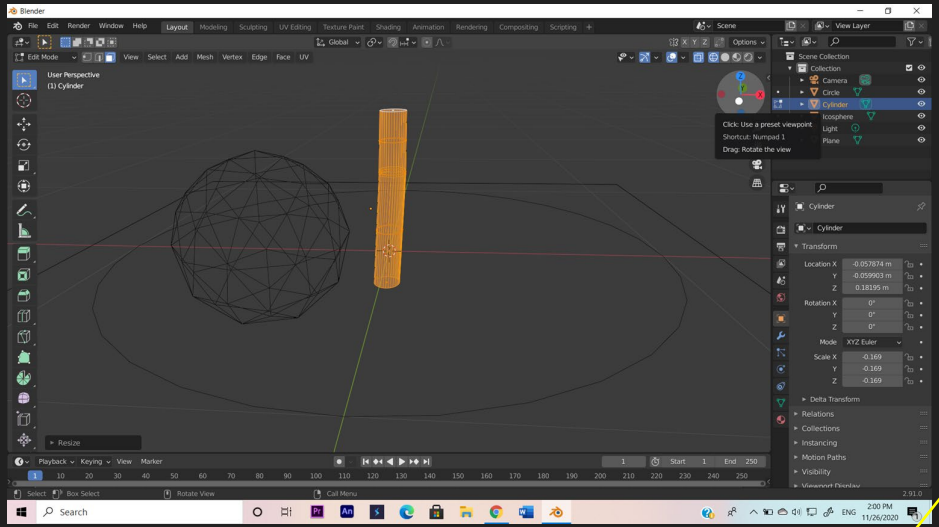


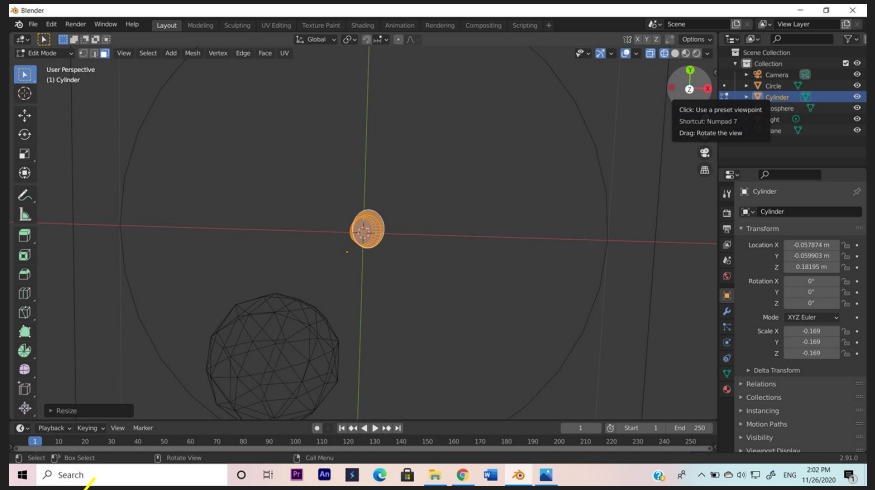


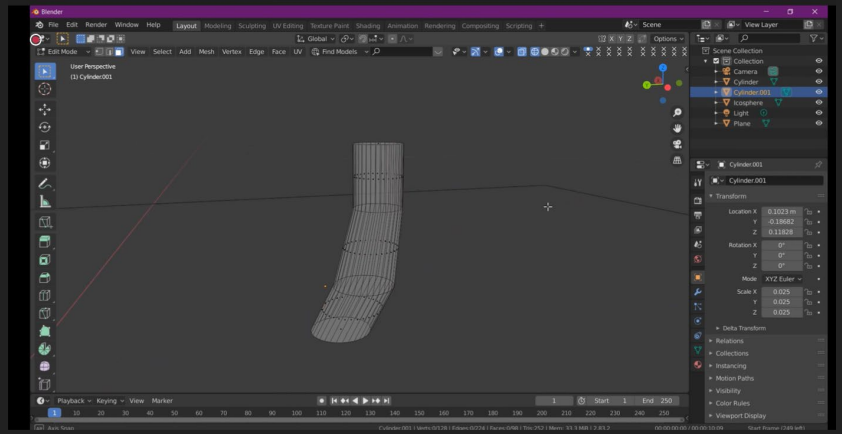
Use Case Scenarios

Link to the Miro Board

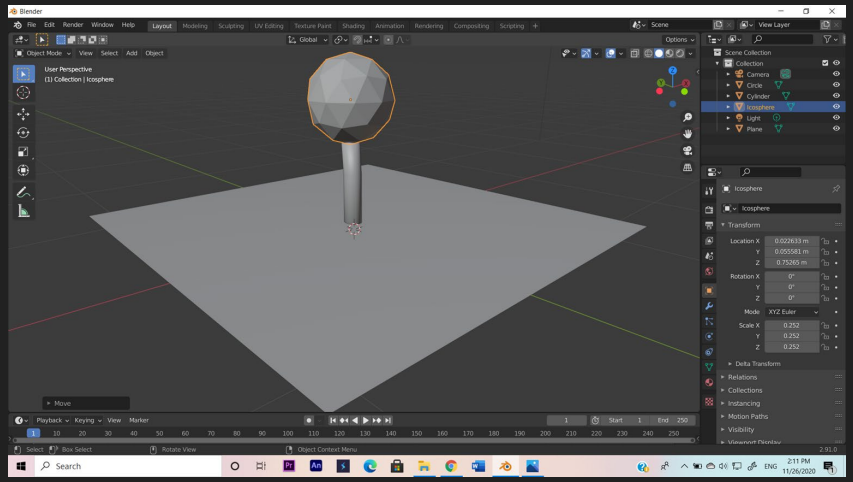
:https://miro.com/app/board/o9J_ldSJ38A=/

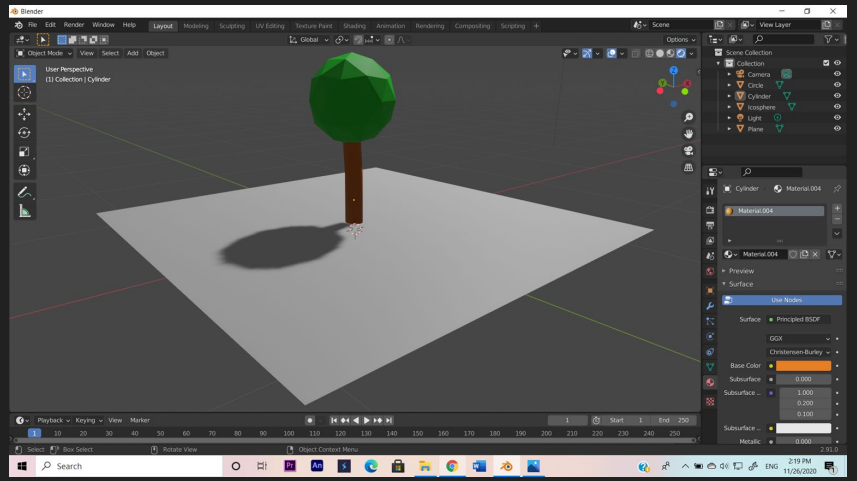




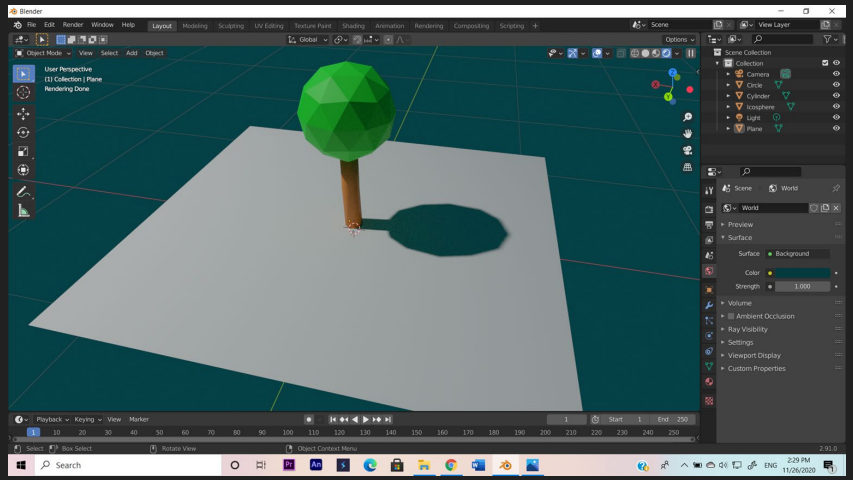


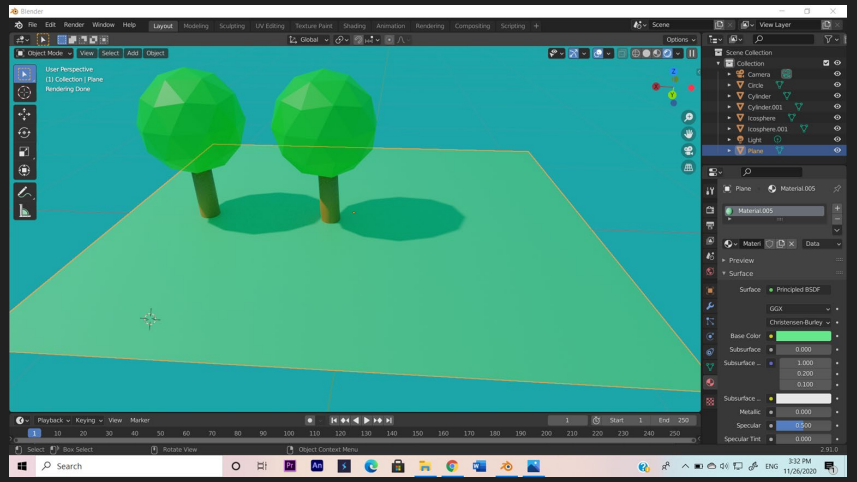
CREATING THE VARIOUS COMPONENTS FOR THE GAME INTERFACE ON BLENDER

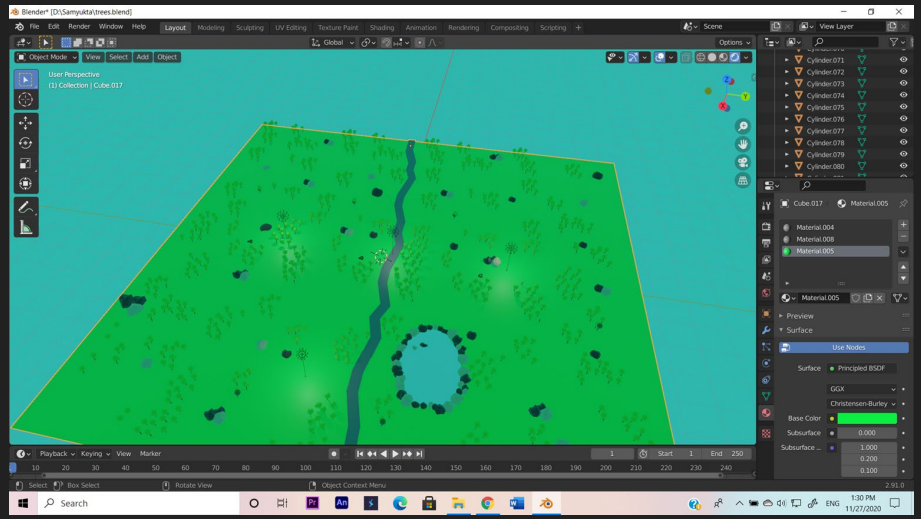


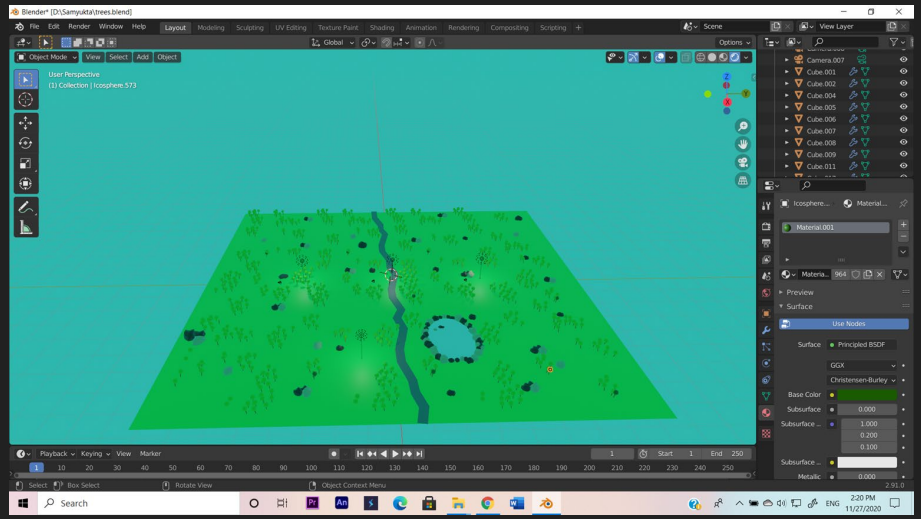


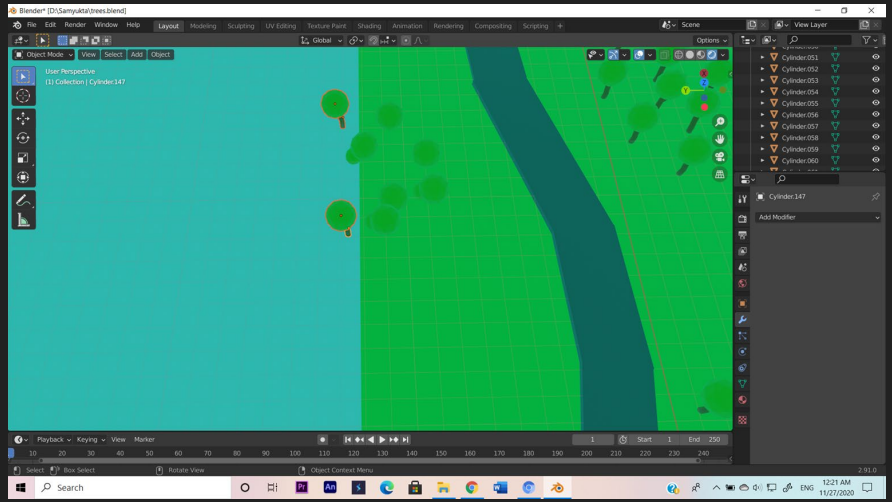
CREATING THE BACKGROUND

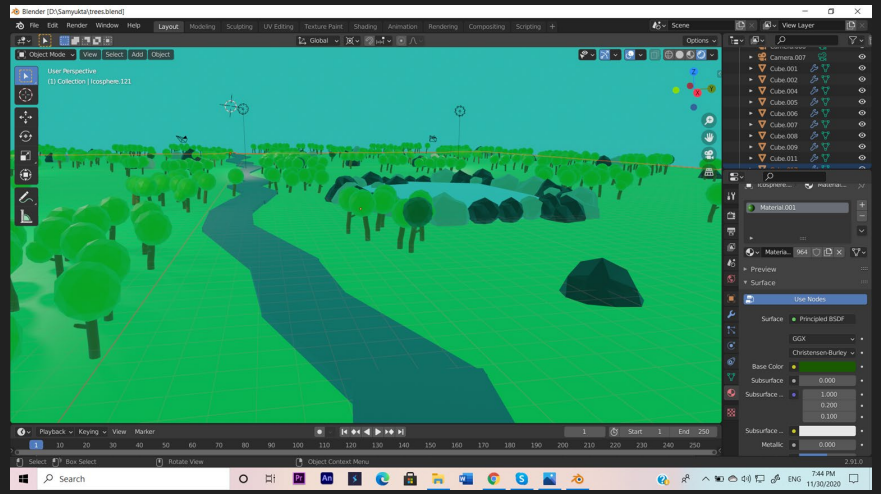


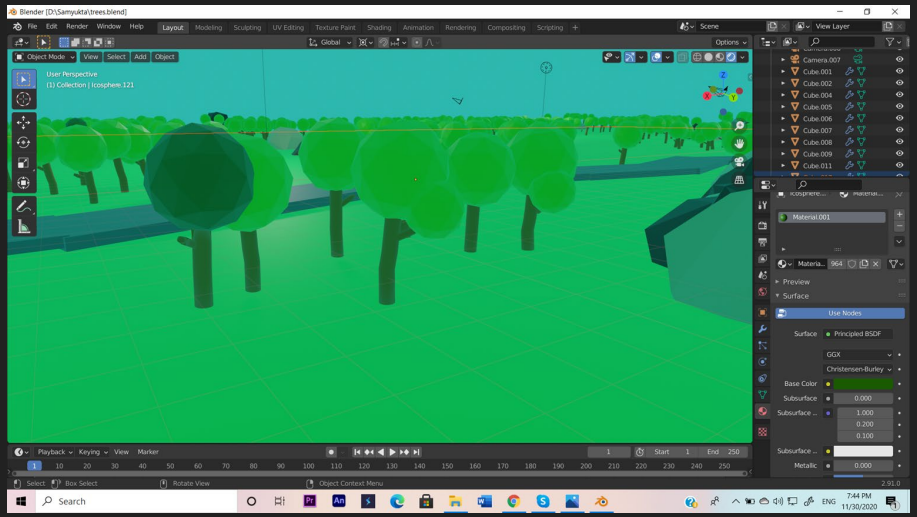


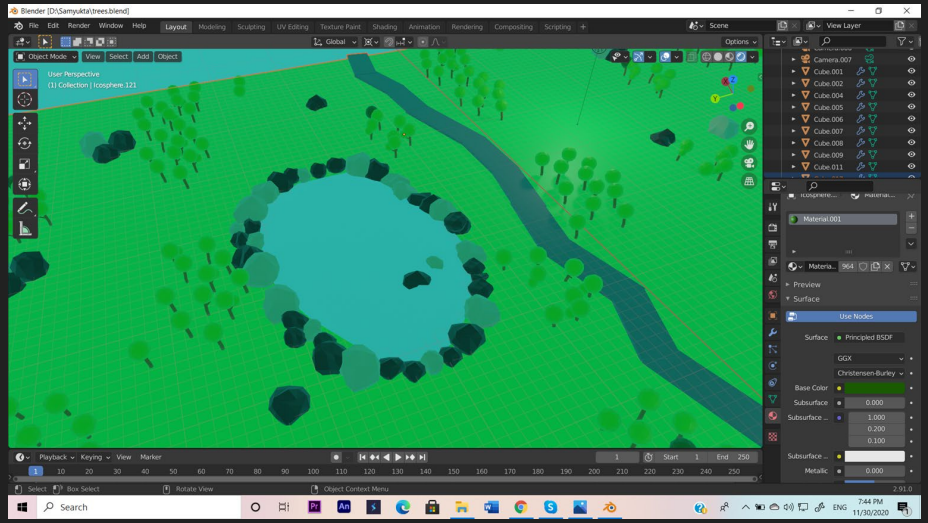


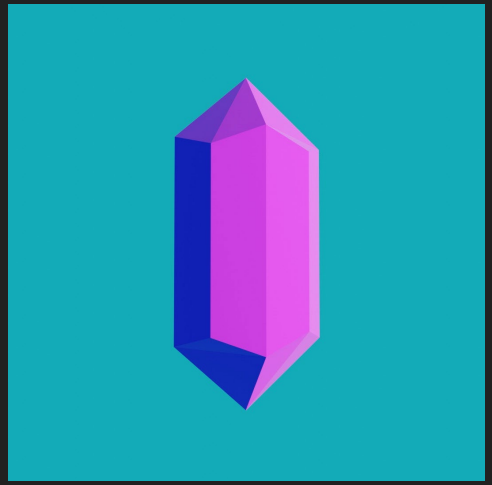


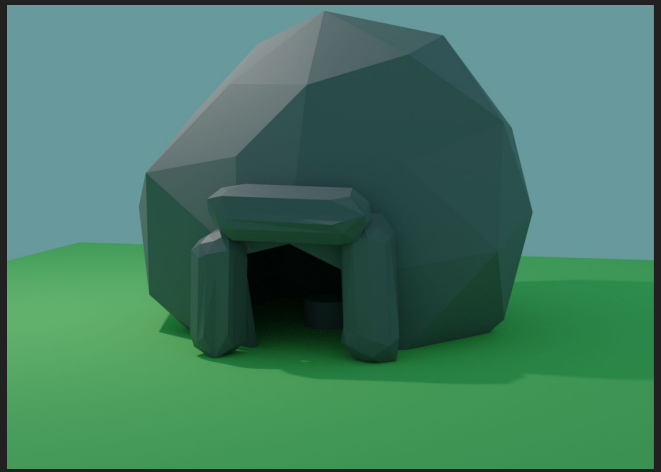


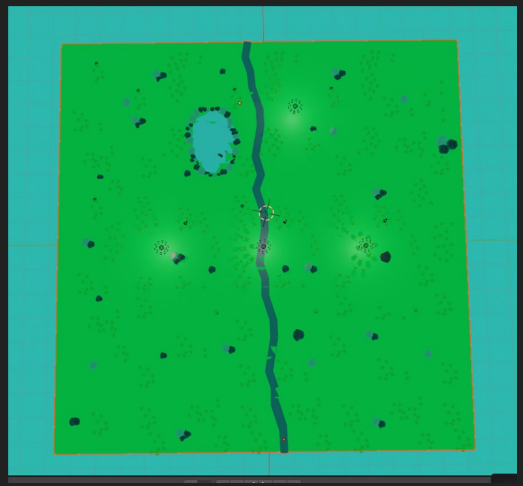




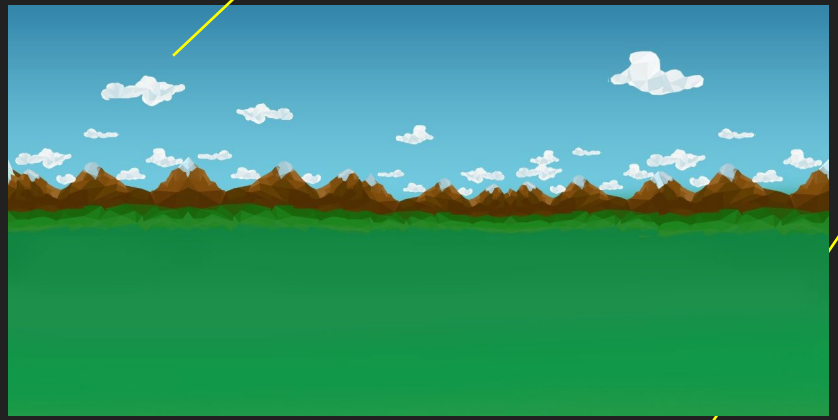




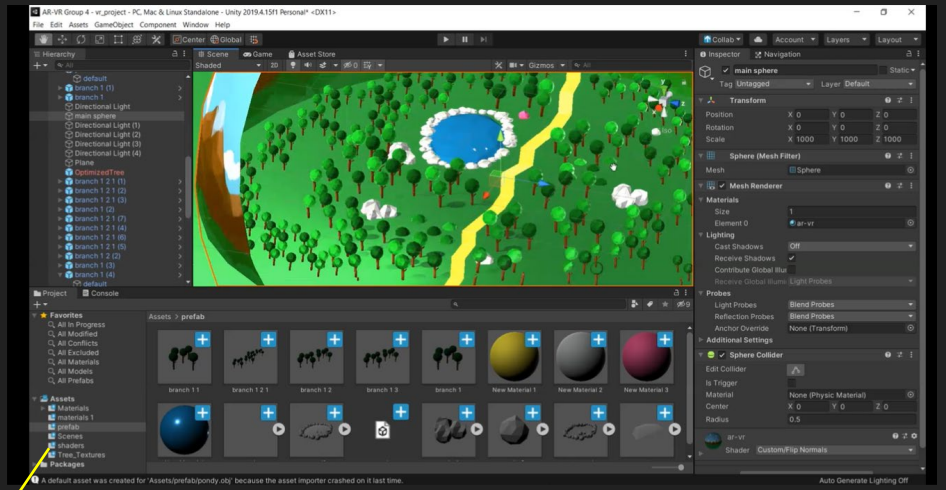




Drawing of the background done on photoshop which was then converted into a sphere in which the game was made



Compiling all the components of the background on to Unity



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

public class trigg : MonoBehaviour

    {public AudioSource level2mid;
    // Start is called before the first frame update
    void Start()

    {level2mid = GetComponent<AudioSource> ();


    }

    // Update is called once per frame
    void Update()
    {

    }
    void OnCollisionEnter (Collision collision) {
        if (collision.gameObject.tag == "target") {
        level2mid.Play ();
            Destroy (collision.gameObject);
        }
    }
}
```

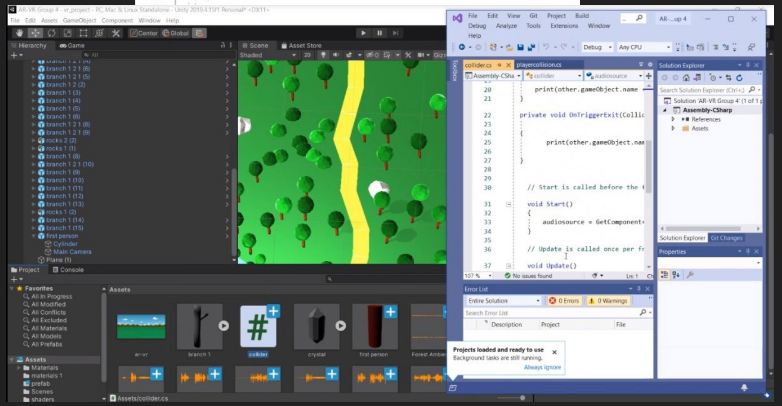
lider.cs

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

public class collider : MonoBehaviour
{
  Audiosource audiosource;
 private void OnTriggerEnter(Collider other)

 {
     print(other.gameObject.name + "has entered the cube");


     audiosource.Play();

 }
 private void OnTriggerStay(Collider other)
 {

     print(other.gameObject.name + "is still the cube");
 }
 private void OnTriggerExit(Collider other)

 {
       print(other.gameObject.name + "has exit the cube");


 }


  // Start is called before the first frame update
  void Start()
  {
      audiosource = GetComponent<Audiosource>();
  }

  // Update is called once per frame
  void Update()
  {
```

ovement 1.cs

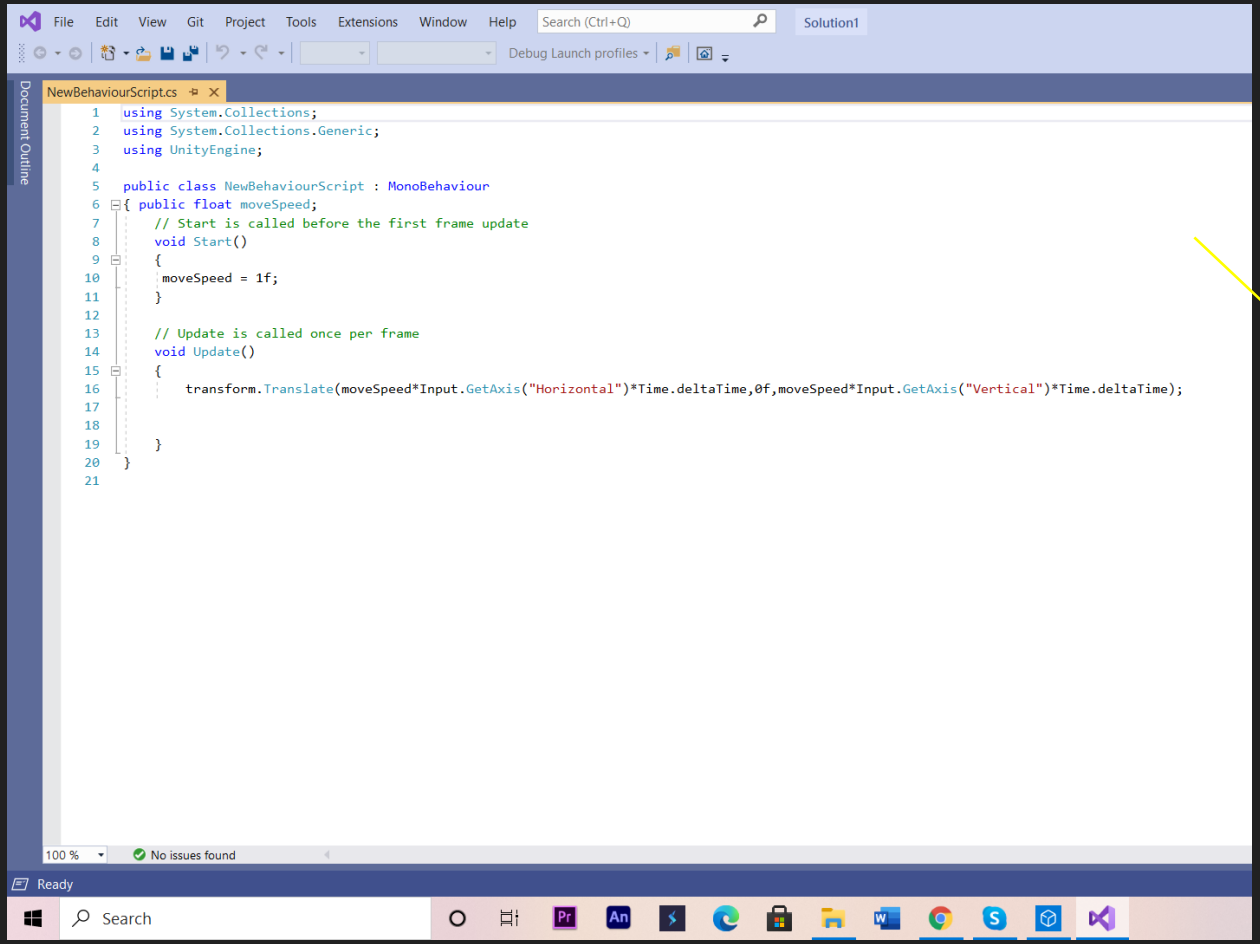
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class movement1 : MonoBehaviour
{
    public Transform [] target;
    public float speed;

    private int current;



    void Update()
    { if(transform.position != target[current].position)
    {
        Vector3 pos = Vector3.MoveTowards(transform.position,target[current].position,speed * Time.deltatime)
        GetComponent<Rigidbody>().MovePosition.(pos);

    } else current = (current + 1) % target.Length;

    }
}
```

ITERATIONS FOR THE CODE TO DETECT OBJECT COLLISION AND TRIGGER SOUND

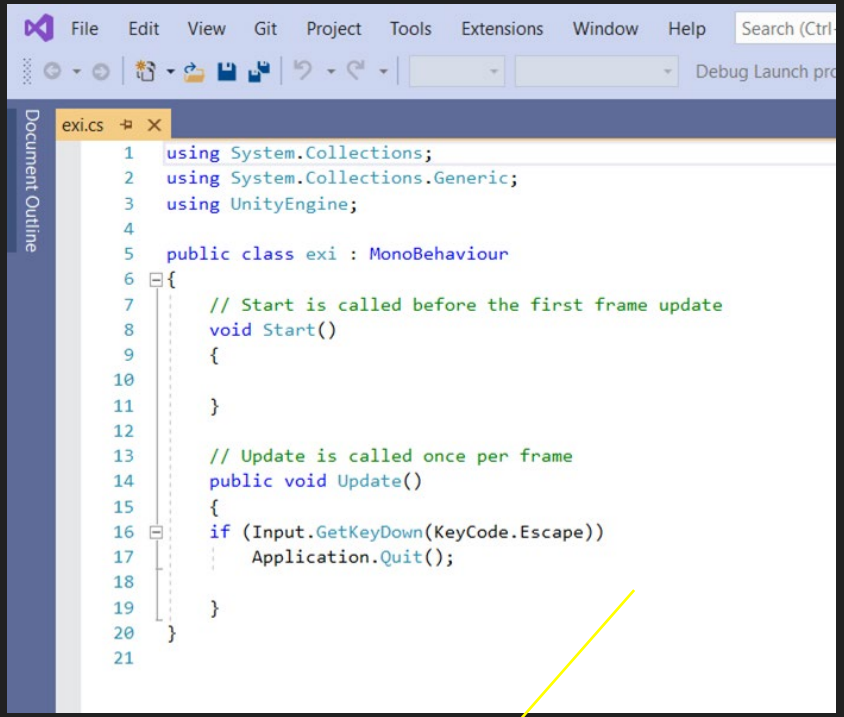


embly-CSharp playercollision

```
using UnityEngine;

Unity Script | 0 references
public class playercollision : MonoBehaviour
{
    public AudioSource playSound;
    Unity Message | 0 references
    void OnCollisionEnter(Collision collisionInfo)
    {
        if (collisionInfo.collider.tag == "colliding")
        {
            playSound.Play();
            Debug.Log(collisionInfo.collider.name);

        }
    }
}
```



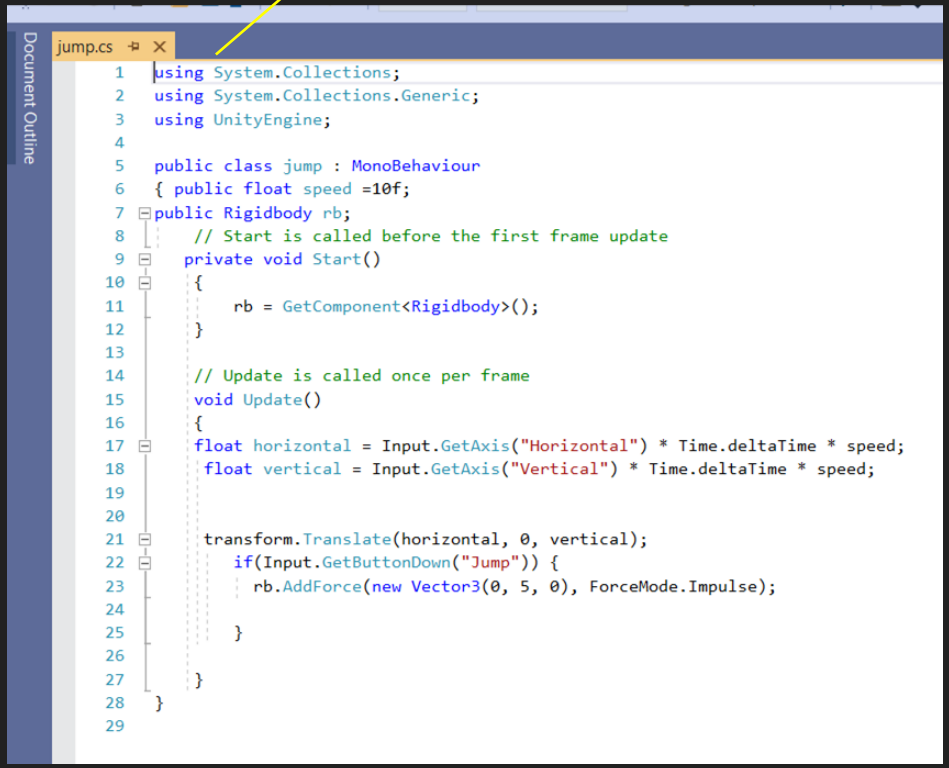
Code to move through the game scene using the arrow keys




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

public class exi : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    public void Update()
    {
    if (Input.GetKeyDown(KeyCode.Escape))
        Application.Quit();

    }
}
```

Code to exit the level once completed using the ESC key

Code to make an object jump using the spacebar




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

public class jump : MonoBehaviour
{ public float speed =10f;
public Rigidbody rb;
    // Start is called before the first frame update
   private void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    void Update()
    {
    float horizontal = Input.GetAxis("Horizontal") * Time.deltaTime * speed;
     float vertical = Input.GetAxis("Vertical") * Time.deltaTime * speed;


     transform.Translate(horizontal, 0, vertical);
        if(Input.GetButtonDown("Jump")) {
          rb.AddForce(new Vector3(0, 5, 0), ForceMode.Impulse);

        }

    }
}
```

Assembly-CSharp | mouse_look

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

Unity Script | 0 references
public class mouse_look : MonoBehaviour
{
    public float mouseSensitivity = 100f;

    public Transform playerBody;

    float xRotation = 0f;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X")* mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y")* mouseSensitivity * Time.deltaTime;

        xRotation -= mouseY;
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);

        transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
        playerBody.Rotate(Vector3.up * mouseX);
    }
}
```

CODE FOR THE CAMERA ANGLE OF THE PERSON SO HE/SHE IS ABLE TO LOOK AROUND IN ANY DIRECTION

CODE SO THE PROJECTION OF THE SPHERE WILL BE ON THE INSIDE

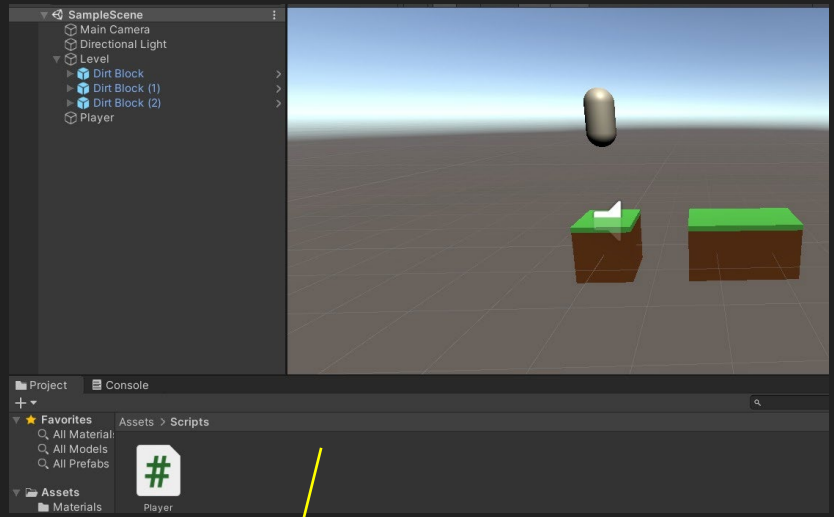
collid.cs | playercollision.cs | NewBehaviourScript.cs | player_moveme

```
Shader "Custom/Flip Normals" {
    Properties {
        _MainTex ("Base (RGB)", 2D) = "white" {}
    }
    SubShader {

        Tags { "RenderType" = "Opaque" }

        Cull Off

        CGPROGRAM

        #pragma surface surf Lambert vertex:vert
        sampler2D _MainTex;

        struct Input {
            float2 uv_MainTex;
            float4 color : COLOR;
        };

        void vert(inout appdata_full v) {
            v.normal.xyz = v.normal * -1;
        }

        void surf (Input IN, inout SurfaceOutput o) {
            fixed3 result = tex2D(_MainTex, IN.uv_MainTex);
            o.Albedo = result.rgb;
            o.Alpha = 1;
        }

        ENDCG

    }

    Fallback "Diffuse"
}
```

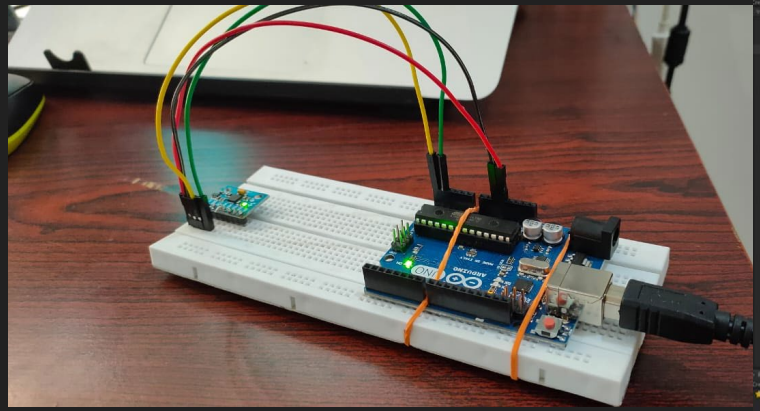
No issues found

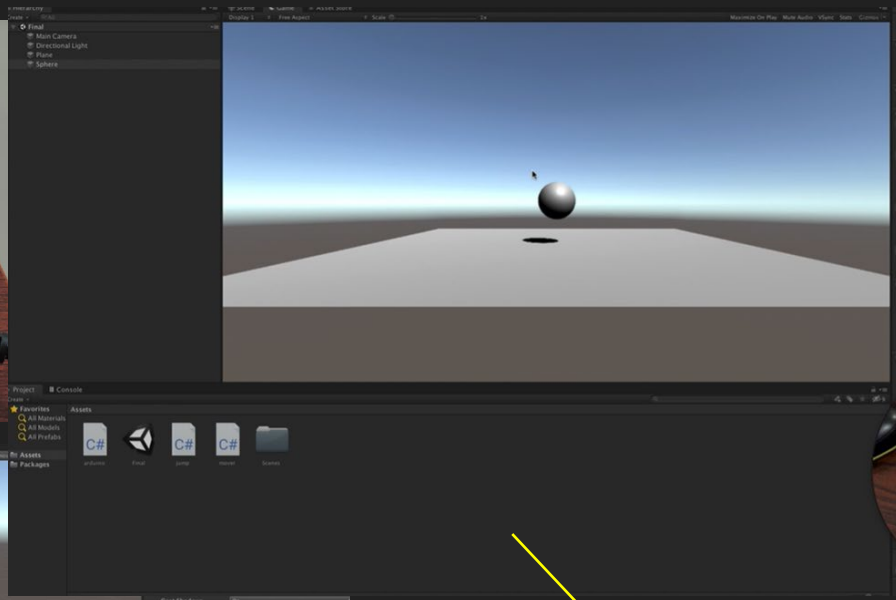
Assembly-CSharp | Player

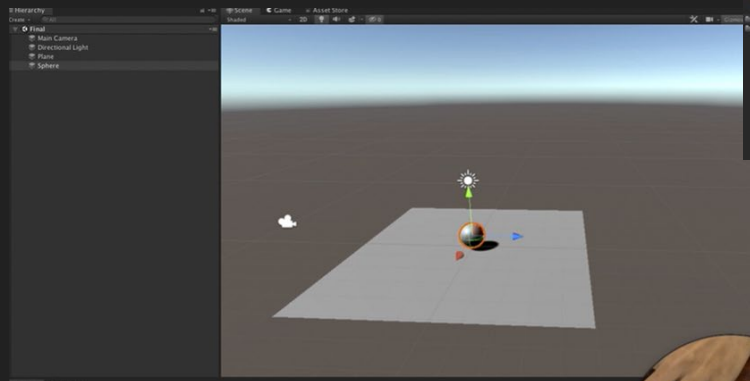
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
   Unity Script | 0 references
5  public class Player : MonoBehaviour
6  {
7      private bool jumpKeyWasPressed;
8      private float horizontalInput;
9      private Rigidbody rigidBodyComponent;
10
11     // Start is called before the first frame update
       Unity Message | 0 references
12     void Start()
13     {
14         rigidBodyComponent = GetComponent<Rigidbody>();
15     }
16
17     // Update is called once per fra me
       Unity Message | 0 references
18     void Update()
19     {
20         // Check if Space key is pressed down
21         if (Input.GetKeyDown(KeyCode.Space) == true)
22         {
23             jumpKeyWasPressed = true;
24
25             // Debug.Log("Space Key was pressed Down");
26         }
27
```

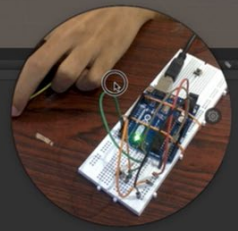


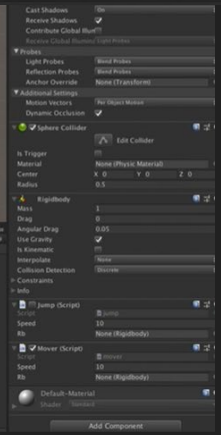
Code to make the capsule move and jump up and down











Scenes from level 1

CODE ITERATIONS FROM LEVEL 1

ITERATION 1 - USING ARDUINO TO CONTROL THE POSITION OF A CUBE IN UNITY

Arduino_Test_-_Serial

```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
int a =1;
int b=2;

Serial.println(a);
delay(1000);

Serial.println(b);
delay(1000);
}
```

```
using UnityEngine;
using System.Collections;
using System.IO.Ports;

public class My1stScript : MonoBehaviour {
    private float AmountToMove;
    SerialPort serial = new SerialPort("COM9",9600);

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (!serial.IsOpen)
            serial.Open ();

        int data = int.Parse (serial.ReadLine());
        AmountToMove = 2;
        MoveObject (data);

    }
    void MoveObject(int direction){

        if (direction == 1) {
            transform.Translate (Vector3.left * AmountToMove, Space.World);
        }


        if (direction == 2) {
            transform.Translate (Vector3.right * AmountToMove, Space.World);
        }



    }

}
```

CODE ITERATIONS FROM LEVEL 1

ITERATION 2 - USING UNITY TO CONTROL THE BUILT-IN LED ON THE ARDUINO BOARD

```
int data;

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(13,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available()){
  data = Serial.read();
  if(data == 'A'){
    digitalWrite(13,HIGH);
  }
  else {
    digitalWrite(13,LOW);
  }

}
}
```

Assets > C# led.cs

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

public class led : MonoBehaviour
{
    public SerialPort serial = new SerialPort ("/dev/cu.usbmodem14101",9600);
    private bool lightState = false;
    public void OnMouseDown(){
        if(serial.IsOpen== false){
            serial.Open();
        }

        if(lightState == false){
            serial.Write("A");
            lightState = true;
        }else{
            serial.Write("a");
            lightState = false;
        }
    }
}
```

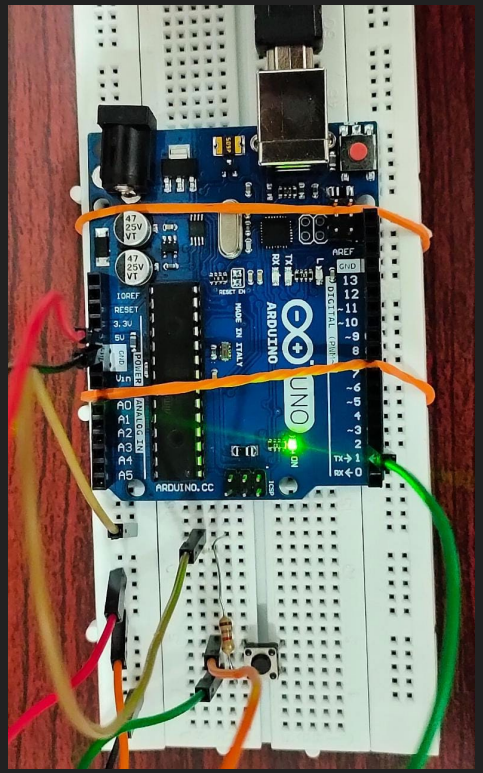
CODE ITERATIONS FROM LEVEL 1

ITERATION 3 - USING BUTTON IN ARDUINO TO MAKE A SPHERE JUMP IN UNITY

UnityButton

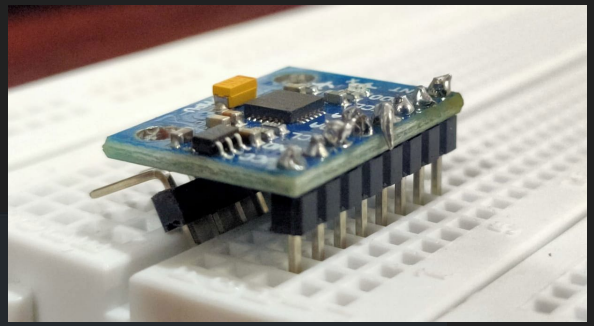
```
int switchPin1 =2;
const int ledPin =  13;
void setup() {
  Serial.begin(9600);
  pinMode(switchPin1, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  if(digitalRead(switchPin1) == HIGH){
    Serial.write(1);
    Serial.flush();
    digitalWrite(ledPin, HIGH);
    delay(20);
  }
  else{
    digitalWrite(ledPin, LOW);
  }
}
```

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

public class mover : MonoBehaviour
{   public float speed =10f;
    public Rigidbody rb;
    SerialPort serial = new SerialPort ("/dev/cu.usbmodem14101",9600);

    // Start is called before the first frame update
    void Start()
    {
        serial.Open();
        serial.ReadTimeout = 1;
        rb = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    void Update()
    { float horizontal = Input.GetAxis("Horizontal") * Time.deltaTime * speed;
     float vertical = Input.GetAxis("Vertical") * Time.deltaTime * speed;


        if(serial.IsOpen){
           try
           {
               // move(serial.ReadByte());
               int direction = serial.ReadByte();
               if(direction == 1){
               transform.Translate(horizontal, 0, vertical);
               if(Input.GetButtonDown("Jump")) {
               rb.AddForce(new Vector3(0, 5, 0), ForceMode.Impulse);}}
           }
           catch (System.Exception)
           {

           }
       }
```



CODE ITERATIONS FROM LEVEL 1

ITERATION 4 - USING ACCELEROMETER TO CONTROL A CUBE IN UNITY (IN PROGRESS)



```ino
#include<Wire.h>
const int MPU=0x68;  // I2C address of the MPU-6050
String dataString;
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
void setup(){
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  Serial.begin(115200);
}
void loop(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,14,true);  // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

  dataString = AcX;
  dataString += ";";
  dataString += AcY;
  dataString += ";";
  dataString += AcZ;
  dataString += ";";
  dataString += GyX;
  dataString += ";";
  dataString += GyY;
  dataString += ";";
  dataString += GyZ;
  dataString += " ";
  Serial.println(dataString);
  Serial.flush();
  delay(200);
}
```

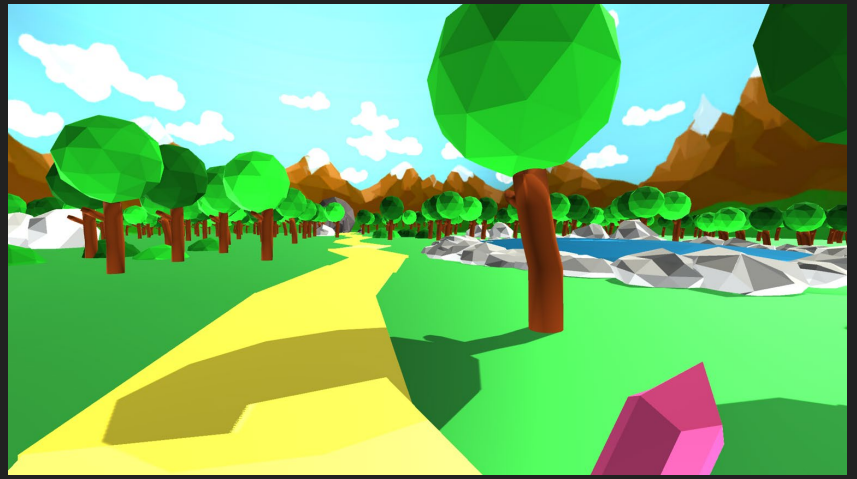
Users > malayvasa > Unity > Arduino Test - Cube > Assets > C# mpuTest.cs

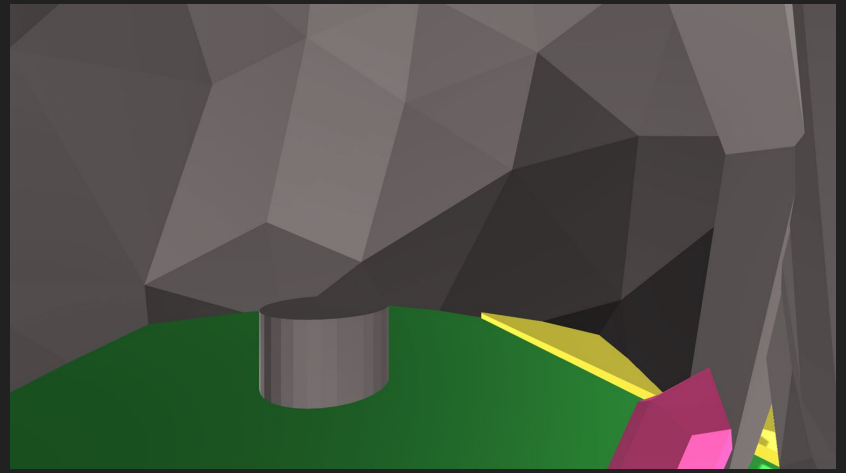
```cs
using UnityEngine;
using System.Collections;
using System;
using System.IO.Ports;


public class mpuTest : MonoBehaviour
{   SerialPort stream;
    float acc_normalizer_factor = 0.00025f;
    float gyro_normalizer_factor = 1.0f / 32768.0f;   // 32768 is max value captured during test on imu

    float curr_angle_x = 0;
    float curr_angle_y = 0;
    float curr_angle_z = 0;

    float curr_offset_x = 0;
    float curr_offset_y = 0;
    float curr_offset_z = 0;

    // Increase the speed/influence rotation
    public float factor = 7;
    // SELECT YOUR COM PORT AND BAUDRATE
    string port = "/dev/cu.usbmodem14101";
    int baudrate = 115200;
    int readTimeout = 6000;

    void Start()
    {
        // open port. Be shure in unity edit > project settings > player is NET2.0 and not NET2.0Subset
        stream = new SerialPort(port, baudrate);

        try
        {
            stream.ReadTimeout = readTimeout;
        }
        catch (System.IO.IOException ioe)
        {
            Debug.Log("IOException: " + ioe.Message);
        }
```

ITERATION 4 - MORE OF THE CODE

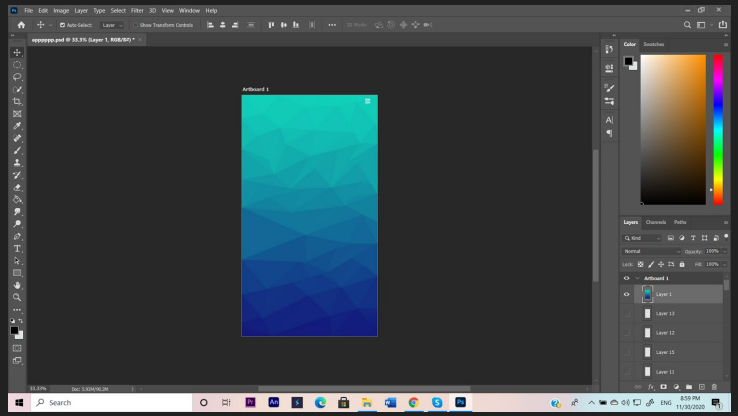
```
        stream.Open();
    }

    void Update()
    {
        string dataString = "null received";

        if (stream.IsOpen)
        {
            try
            {
                dataString = stream.ReadLine();
                Debug.Log("RCV_ : " + dataString);
            }
            catch (System.IO.IOException ioe)
            {
                Debug.Log("IOException: " + ioe.Message);
            }

        }
        else
            dataString = "NOT OPEN";
        Debug.Log("RCV_ : " + dataString);

        if (!dataString.Equals("NOT OPEN"))
        {
            // recived string is  like  "accx;accy;accz;gyrox;gyroy;gyroz"
            char splitChar = ';';
            string[] dataRaw = dataString.Split(splitChar);

            // normalized accelerometer values
            float ax = Int32.Parse(dataRaw[0]) * acc_normalizer_factor;
            Debug.Log(ax);
            float ay = Int32.Parse(dataRaw[1]) * acc_normalizer_factor;
            Debug.Log(ay);
            float az = Int32.Parse(dataRaw[2]) * acc_normalizer_factor;
            Debug.Log(az);
```

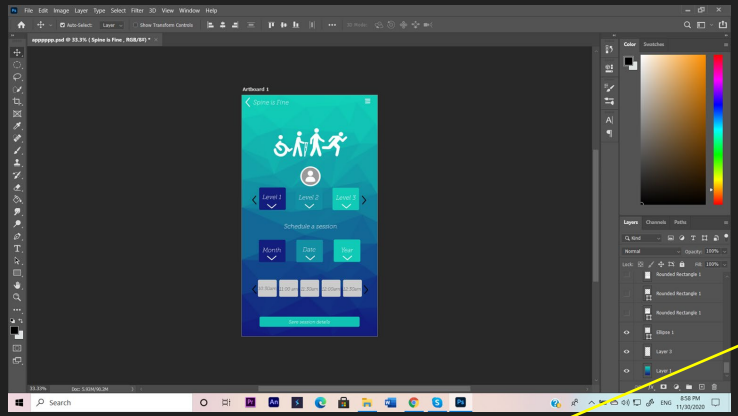
```
            // normalized gyrocope values
            float gx = Int32.Parse(dataRaw[3]) * gyro_normalizer_factor;
            Debug.Log(gx);
            float gy = Int32.Parse(dataRaw[4]) * gyro_normalizer_factor;
            Debug.Log(gy);
            float gz = Int32.Parse(dataRaw[5]) * gyro_normalizer_factor;
            Debug.Log(gz);

            // prevent
            if (Mathf.Abs(ax) - 1 < 0) ax = 0;
            if (Mathf.Abs(ay) - 1 < 0) ay = 0;
            if (Mathf.Abs(az) - 1 < 0) az = 0;


            curr_offset_x += ax;
            curr_offset_y += ay;
            curr_offset_z += 0; // The IMU module have value of z axis of 16600 caused by gravity


            // prevent little noise effect
            if (Mathf.Abs(gx) < 0.025f) gx = 0f;
            if (Mathf.Abs(gy) < 0.025f) gy = 0f;
            if (Mathf.Abs(gz) < 0.025f) gz = 0f;

            curr_angle_x += gx;
            curr_angle_y += gy;
            curr_angle_z += gz;


            transform.rotation = Quaternion.Euler(curr_angle_x * factor, -curr_angle_z * factor, curr_angle_y * factor);
        }
    }
}
```

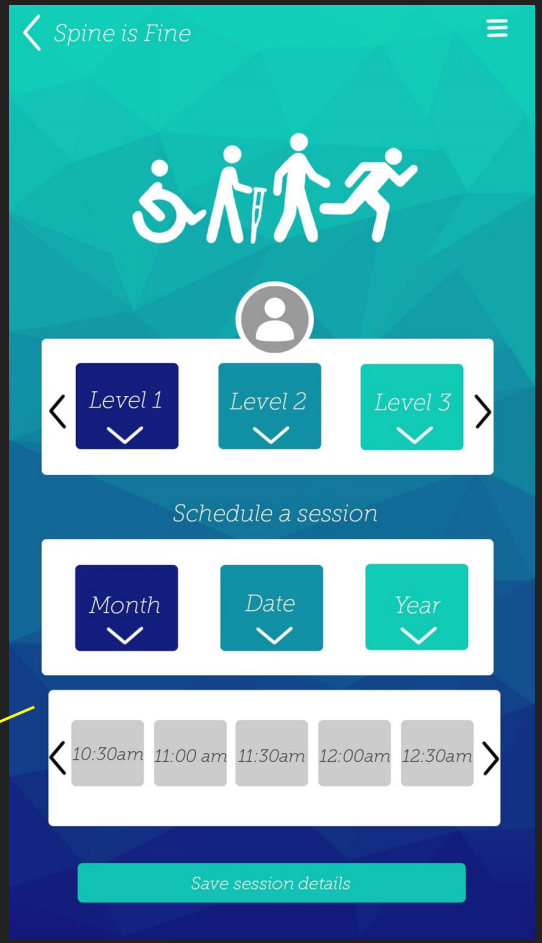


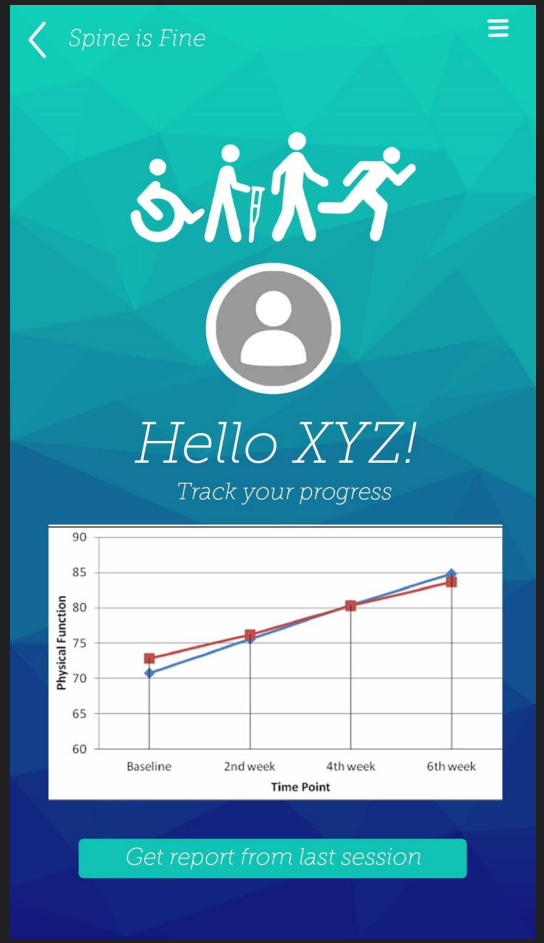


SCENES FROM LEVEL 2









MOCK APP INTERFACE

Category	Component	Score
Sitting balance	Sitting unsupported	3
Standing balance	Standing unsupported	2
	Standing with eyes closed	3
	Standing with feet together	3
	Standing on one foot	2
	Turning to look behind	3
	Retrieving object from floor	2
	Tandem standing	3
	Reaching forward with an outstretched arm	4
Dynamic balance	Sitting to standing	2
	Standing to sitting	3
	Transfer	3
	Turning 360 degrees	2
	Stool stepping	2
Total		37

Mock report that each patient will receive through the app to track progress as well as to send to a professional to assess. The report is based on the Berg Balance Test evaluation.

Progress Videos

References

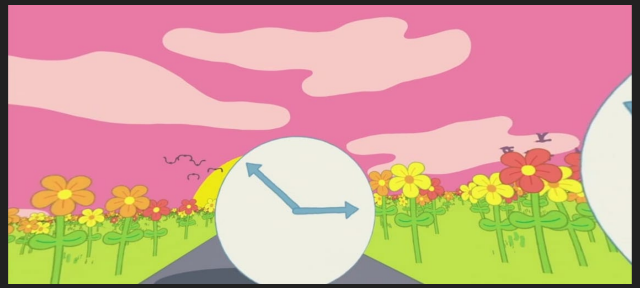
- https://youtu.be/0NCTAuP3BgU
- https://youtu.be/pwZpJzpE2lQ
- https://youtu.be/7nxKAtxGSn8
- https://youtu.be/6OT43pvUyfY
- https://youtu.be/1ZIxl-br0po
- https://www.youtube.com/watch?v=xvLMD2qWaKk
- https://www.youtube.com/watch?v=aN11LnlF89I&t=30s
- https://www.youtube.com/watch?v=VomZe-_WWsE&t=452s
- https://www.youtube.com/watch?v=fKWTpi70a_E&t=187s
- https://www.youtube.com/watch?v=WTGcs10Sj34
- https://www.youtube.com/watch?v=kSfgmh_pk1Y
- https://www.youtube.com/watch?v=sXQI_0ILEW4
- https://www.physio-pedia.com/Berg_Balance_Scale#:~:text=for%20item%20%23%2012.-,Interpretation,at%20greater%20risk%20of%20falling.
- https://www.inyourhometherapy.com/our-blog/types-physiotherapy-aware/
- https://www.healtheuropa.eu/vr-technology-and-3d-capture-could-allow-physiotherapy-at-home/99490/
- https://www.youtube.com/watch?v=necffi60Fs4
- https://www.youtube.com/watch?v=xq1eiI99dVQ
- https://youtu.be/qGAsgIJ-c38
- https://www.youtube.com/watch?v=7nxKAtxGSn8&t=301s
- https://www.youtube.com/watch?v=gAB64vfbrhI&t=266s
- https://www.youtube.com/watch?v=qGAsgIJ-c38&t=182s
- https://www.youtube.com/watch?v=KVhSCck_5yI&t=8s
- https://www.youtube.com/watch?v=dbAdLs-NlzI&t=25s
- https://www.youtube.com/watch?v=-SNVjj6wjlU
- https://www.youtube.com/watch?v=_QajrabyTJc&t=884s
- https://www.youtube.com/watch?v=eaqwszsH6Jg
- https://www.youtube.com/watch?v=9SYt4MDSAro
- https://forum.unity.com/threads/problem-cant-send-simple-x-y-and-z-variables-from-arduino-uno-to-unity.251387/
- https://stackoverflow.com/questions/57539387/how-can-i-use-mpu6050-and-arduino-uno-to-control-cube-so-the-cube-will-move-with
- https://www.arduino.cc/en/uploads/Tutorial/button.png

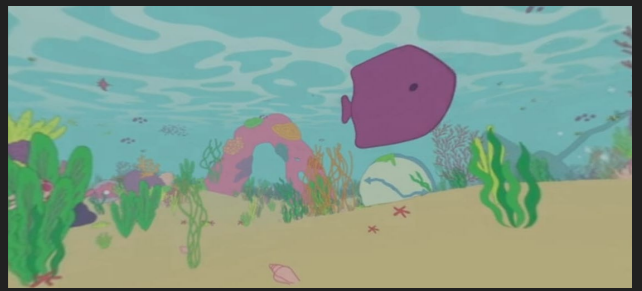
Primary Research

Conversation with **Dr. Mugdha Desai**, Sports Physiotherapist, Glasgow

Inspiration

- https://neurorehabilitation.m-iti.org/lab/rehabnet-2/
- https://www.franciscanhealth.org/health-care-services/neurological-physical-therapy-255
- https://www.with.in/watch/dear-lizzy?autoplay=true
- Minecraft
- Wizard Of Oz (Yellow Brick Road)
- Indiana Jones: Raiders of the Lost Ark





Moodboard - Aesthetics

Technologies learnt and used during this project

1. Blender to render the 3D objects
2. Unity to design and code the various elements of the game
3. Coding in Unity using C#
4. Photoshop to draw the background of the game interface
5. Coding Arduino Uno
6. Using Phone sensors such as Gyroscope
7. Editing Sound in Adobe Audition
8. Using External sensors (Arduino Kit)
9. Soldering and Calibrating Sensors

Individual Contributions of team members

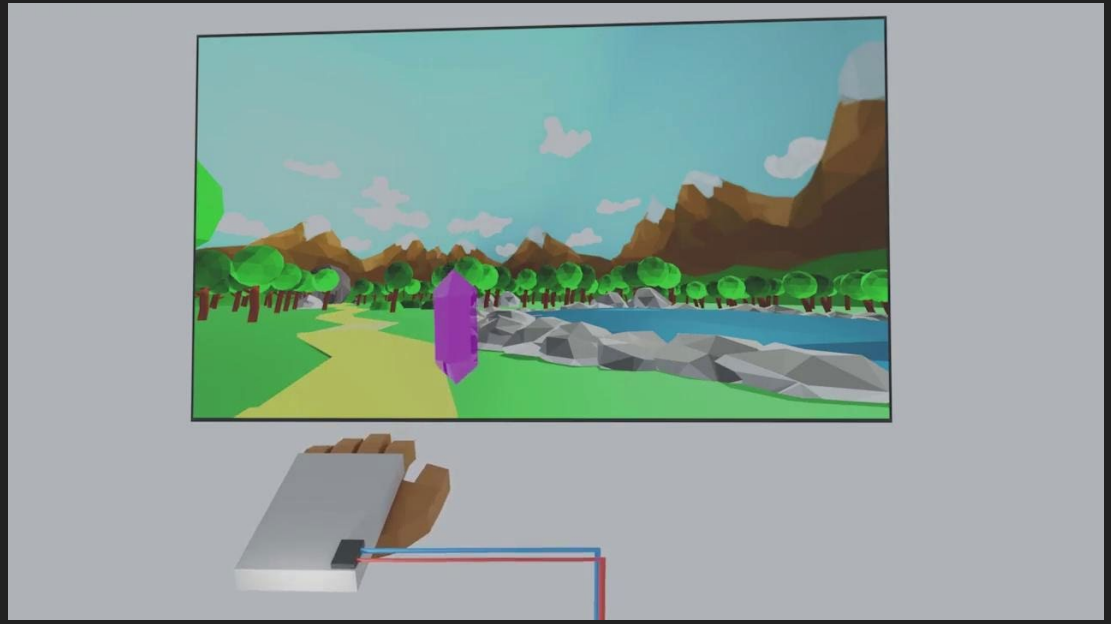
Aditya: Rendered a Concept Video animation for an iteration of Level 1, Rock Temple and the Crystal using Blender. Introduced teammates to Blender UI. Edited the sounds and came up with the lowpoly adventure theme for the Gameplay. Coded iterations for movement in Unity. Contacted a Physiotherapist for Primary Research.Troubleshooting.

Ishika: Painting and rendering spherical background, wrote the code for warping the image, attaching the camera to the first person controller and trigger detection with audio iteration. Putting the environment and the elements of the game together on Unity, rendering and exporting the game on Unity.Troubleshooting.

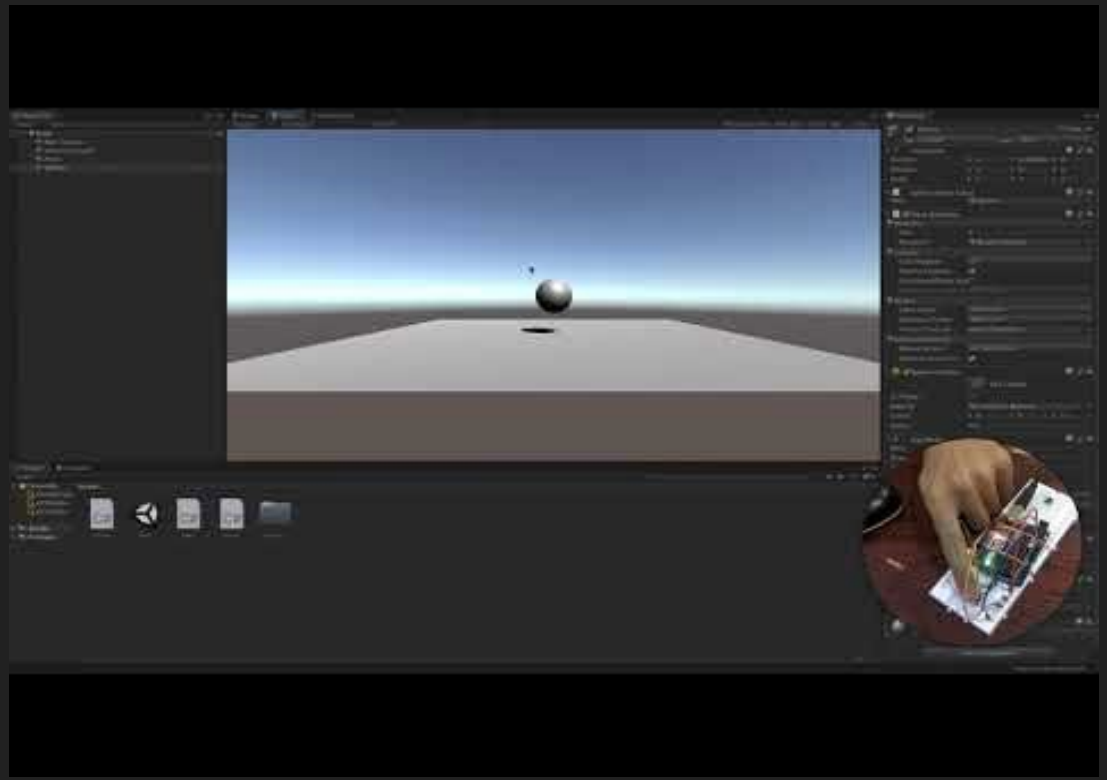
Malay: Worked on Interfacing Unity with Arduino, this would allow the digital aspect and physical aspect of the project to interact. This involved understanding how to use the Serial Port to establish a two way communication between Unity and Arduino. Soldered and Calibrated sensors. Integrated this code, with that of Level 1. I used the Arduino Uno & MPU-6050.

Samyukta: Initial ideation and research for the project + use cases. Rendered the 3D elements of the game on Blender. Wrote the code for - the movement of the character (level 2), collision detection+audio (level 2), exiting the game and coding the object to jump. Details of the app and report. Voice over for the 2 levels. Troubleshooting.

Concept Video (Level I)



Screen Recording of Arduino + Unity (Level I)



Screen Recording of Working Prototype (Level II)

Link to the Screen recordings

Level 1:

- https://drive.google.com/file/d/1fXzAjSx9E4lpTCJrFHT2p0z9wk3RbZ-z/view?usp=sharing
- https://youtu.be/njyyBnBHjEs

Level 2:

- https://drive.google.com/file/d/1MZGVXRsxv_0u_0Ld8TBIEf2Y16I5HOjc/view?usp=sharing

Process Video:

- https://drive.google.com/file/d/1PlW7dbV_Px-o3BvpylWjAwb1giWJsO0k/view?usp=sharing

Drive Link to Working prototype of Game (Level II)

https://drive.google.com/file/d/1rN-EoONtWccAKY1PEWR03R2MCmqjXqCO/view?usp=sharing

Link to the zipped game file:
https://drive.google.com/file/d/1w6kArXJbw3ThcymUpwVFz5lDN1cllfd/view?usp=sharing