CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL

VIGOr

Virtual reallty GlOves

Desarrollo de prototipo de un periférico para smartphone

SUSTENTANTES:

Omar Ildefonso Godinez Quiñones

David Uziel Guevara Hernández

DESARROLLO DE SOFTWARE

INFORMATICA Y COMPUTACION

ASESORES:

Asesor técnico Ing. Roa García de la Paz José Luis	Asesor metodológico Ing. Bravo López Juan Ramón

Tonalá, Jalisco. 26 DE 10 DE 2018

Contenido

Capítulo I

ANALISIS Y DEFINICIÓN DEL PROYECTO O PROTOTIPO

Antecedentes:

Google Cardboard:

Es una plataforma de realidad virtual desarrollada por Google para Smartphone. El Google cardboard puede ser armado por el usuario mientras este siga las instrucciones puestas por Google y tenga los materiales necesarios para construirlo.

Imagen 1. Capítulo 1

El visor cardboard se compone de una pieza de carbón cortada en una forma indicada por google con dos lentes de 45 mm focales colocados además de algunos magnetos.

El casco cardboard está pensado para acercar las tecnologías de realidad virtual a un público mayor, su precio es mucho más accesible que el precio de muchos otros cascos de realidad virtual.

El visor cuenta con varios kits de desarrollo creados por Google, estos kits permiten a otros programadores crear sus propias aplicaciones para el casco.

Data Glove:

El data glove fue creado por la compañía VPL Research la cual desarrollaba y vendía productos de realidad virtual. El guante fue sacado alrededor del año de 1987, este se conectaba a una computadora y sus sensores eran grupos de fibra optica que se usaban para rastrear los movimientos y orientación de la mano. Esos datos se transmitían a la computadora y permitía que se duplicaran los movimientos de las manos en un entorno virtual.

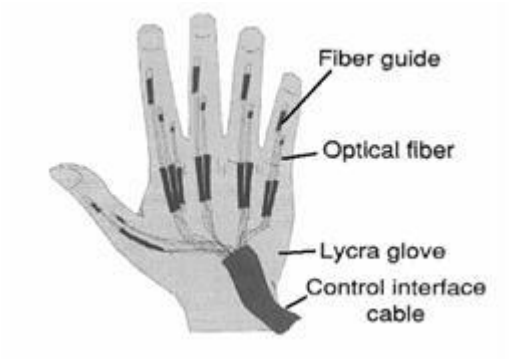

Imagen 2. Capítulo 1

Power Glove:

El Power Glove fue sacado al mercado en el año 1989 y desarrollado por ingenieros asociados a Nintendo. El power glove estaba hecho la tecnología del Data Glove pero con muchas modificaciones que le permitían funcionar con hardware de consumidor con un desempeño modesto además de reducir su precio considerablemente. El Data Glove detectaba varios tipos de rotación, el Power Glove solo podía detectar rotación en x, además de esto usaba sensores bañados en tinta conductora que reemplazaba la fibra óptica del Data Glove.

El Power Glove no fue bien recibido gracias a que su manufactura original era demasiado costosa, por lo que se crearon nuevas versiones con un diseño reducido y materiales más baratos, causando malfuncionamientos en muchas de las unidades.

Imagen 3. Capítulo 1.

Wii Mote:

El Wii mote tiene la capacidad de detectar movimiento en tres ejes gracias a su acelerómetro cuenta con un sensor óptico llamado PixArt el cual determina el lugar al que el Wii mote está apuntando. Esto permite que se use con precisión por 5 metros de distancia.

La barra de sensores es necesaria cuando el wiimote está controlando movimientos arriba-abajo o izquierda-derecha de un cursor en la pantalla del televisor para apuntar a opciones.

El control también usa un sensor infrarrojo para detectar posición, puede causar algunos problemas cuando otras fuentes de infrarrojos se encuentran alrededor, como bombillas incandescentes o velas. Esto puede ser fácilmente mitigado por el uso de luces fluorescentes alrededor de la Wii, ya que emiten poca o ninguna luz infrarroja.

Imagen 4. Capítulo 1.

Retroalimentación:

El Wii mote tiene un sistema de retroalimentación que informa al jugador de acciones que está realizando en el juego a través de audio y de vibraciones.

Myo Armband:

La Myo Armband es una pulsera control de movimiento que funciona a través de sensores EMG los cuales detectan la actividad eléctrica de los músculos para detectar cinco gestos hechos por tus manos, además de detectar orientación y rotación del ante brazo, la pulsera guarda los datos recibidos de los sensores sin acondicionar y acondicionados para que los desarrolladores los usen como quieran usarlos.

La Myo Armband está pensada más como una herramienta y no un control, aunque también es usada como un control.

La Myo Armband cuenta también con varios kits de desarrollo pensados para distintos sistemas operativos, lo cual facilita la creación de aplicaciones para la misma.

Imagen 5. Capítulo 1.

Planteamiento del problema:

La *realidad virtual* es una tecnología actual que tiene mucho potencial que no se aprovecha en su totalidad sobre todo en los dispositivos móviles.

Las tecnologías de interacción con la *realidad virtual* en los *dispositivos móviles* son muy básicas, ya que solo utilizan la inversión visual y los únicos movimientos que capta son los movimientos de la cabeza.

Debido a esto y a que los teléfonos inteligentes no tienen tanta potencia como las computadoras las apps que se encuentran disponibles en el mercado no brindan una *experiencia inversiva* como las que se pueden encontrar disponibles en la computadora, pero el equipo necesario para la realidad virtual en una computadora es bastante costoso, ya que el equipo básico (gafas y controles) esta alrededor de los 10,000$ pesos mexicanos y sin mencionar el precio del equipo de cómputo necesario.

En cambio, un teléfono es más accesible y las gafas más sencillas cuestan alrededor de 50$ pesos mexicanos. Y siendo tan accesible y teniendo tantas posibilidades no se ha trabajado en mejorar este campo de la realidad virtual.

Justificación:

Este proyecto está enfocado en crear un periférico para dispositivos Android cuya función será introducir los movimientos de la mano derecha a entornos de realidad virtual.

Analizando las aplicaciones de realidad virtual disponibles en la play store nos dimos cuenta de que la mayoría son aplicaciones que lo único que hacen es reproducir una escena animada o son reproductores de video en 360° y hay algunos juegos que debido a las limitantes de interacción, no son muy comodos de jugar o pueden ser aburridos.

En base a los cuestionarios que fueron realizados a diversos individuos podemos concluir que la gente anhela estar más conectados con los espacios virtuales y de un modo más inmersivo. También pudimos concluir que, aunque la *realidad virtual* se puede usar en campos más diversos que únicamente en los videojuegos, la gente no tiene estos conceptualizados. Se puede usar para aprender a través de experiencias y usarse como herramienta creativa.

La realidad virtual tiene mucho futuro, pero en el campo más accesible para el usuario promedio es donde está menos desarrollado. Los controles de movimiento específicamente desarrollados para RV en dispositivos móviles son algo que prácticamente no existe en estos momentos.

Es por eso que queremos desarrollar un mando para captura de movimiento de la mano derecha que se use en dispositivos móviles. Con este periférico se podrían hacer aplicaciones de realidad virtual mas completa.

Objetivo General:

- Crear un periférico en forma de guante que sirva para interactuar con un entorno virtual

Objetivos específicos:

- Darle al usuario la capacidad de interactuar con sus manos en un entorno virtual
- Programar un entorno virtual que explore la capacidad del guante.
- Lograr captar los movimientos en 3 ejes de desplazamiento y en 3 ejes de rotación para que la movilidad en el entorno virtual sea una simulación realista.

Metodología:

Diseño:

1. Desarrollo de la idea: Buscaremos un problema al que le podamos resolver o brindar una mejora.
2. Planteamiento del problema: Buscaremos los puntos malos o débiles del problema.
3. Desarrollo de la justificación: Desarrollaremos el porqué de la solución o mejoras que aplicaremos para resolver el problema.
4. Investigación de los antecedentes: Buscaremos productos y empresas que hicieron aportes en la solución del problema.
5. Redacción de antecedentes: Redactaremos un resumen de los principales antecedentes de los que nos basaremos en crear nuestro proyecto.
6. Redacción de planteamiento del problema: Redactaremos una descripción del problema que queremos resolver.

7. Redacción de justificación: Redactaremos el porque queremos y porque se debe resolver el problema.
8. Desarrollo y redacción de objetivos general y específicos: Redactaremos lo que queremos lograr con nuestro proyecto.
9. Planeación de metodología: Desarrollaremos el método por el cual desarrollaremos nuestra idea.
10. Definición de conceptos: Definiremos los conceptos mas importantes del desarrollo de nuestro proyecto.
11. Desarrollo del marco de referencia: Haremos un listado de todas las referencias que utilizamos en la elaboración del proyecto.
12. Esquematización de diagramas: Deseñaremos esquemas representativos de nuestro proyecto.

Desarrollo:

1A. Creación del receptor de datos por bluetooth: Diseñaremos un software receptor de señales bluetooth para la app.

1B. Creación del emisor de datos por bluetooth: Diseñaremos un software que tomara los datos del sensor y los enviara por bluetooth.

2. Prueba y corrección de bluetooth: Realizaremos pruebas de comunicación y aplicaremos correcciones de ser necesario.

3. Hacer la parte estructural del guante: Crearemos un guante de dos piezas en el que se montara el sensor y el circuito emisor.

4. Colocar el sensor en la estructura del guante: Le implementaremos la parte electrónica al guante.

5. Realizar pruebas y correcciones de calibración: Probaremos que los datos están calibrados y listos para que la app los procese.

6. Crear el sistema de calibración y tratamiento de datos: Según el resultado de las pruebas diseñaremos un software que calibre el sensor y que procese la información para que sea la apropiada para la app.

7. Implementar físicas 3D en demo: Comenzaremos con el desarrollo de la aplicación, implementado las físicas y objetos base.

8. Crear jugabilidad y entorno: Crearemos un juego con el que se demuestre el funcionamiento del guante en un entorno de realidad virtual.

9. Implementar pulido gráfico: Trabajaremos en que el entorno visual sea agradable y ligero para el procesamiento

Definiciones Generales del Prototipo:

Realidad virtual: es aquella que es generada por computadoras o sistemas informáticos, que proyectan un escenario en que el usuario tiene la sensación de estar, pudiendo interactuar en ese nuevo mundo y los objetos que allí se encuentren a menor o mayor grado, de acuerdo con los equipos que tenga a su alcance.

Imagen 6. Capítulo 1.

Captura de movimiento: La captura de movimiento es el proceso de grabar los movimientos de un actor y recrearlos en modelos de personajes digitales.

Capítulo II

MARCO CONCEPTUAL DEL PROYECTO O PROTOTIPO

Aspectos Generales:

Se utilizará un giroscopio y un acelerómetro. El acelerómetro será usado para calcular la velocidad a la que se mueve el dispositivo en los tres ejes de movimiento en un espacio 3D, esto es, X, Y, Z. El giroscopio se utilizará para calcular el desplazamiento rotativo del dispositivo en Cuaternions Solo se usarán estos por motivos de precio y tamaño, ya que uno de nuestros objetivos es hacer llegar esta tecnología al mayor publico posible.

La demo técnica será realizada en Unity, la razón de esto es que Unity es un entorno con herramientas para desarrollar aplicaciones 3D en RV ya construidas, encima de esto, cuenta con soporte para plataformas Android, lo cual será lo más conveniente por cuestiones de tiempo y aparte de que es el entorno más utilizado para la creación de aplicaciones de RV en dispositivos Android.

La comunicación será a través de bluetooth siendo el guante el esclavo. Al utilizar esta comunicación evitaríamos el uso de cables y confiable.

La estructura del guante será ajustable a través de velcro para que se pueda ajustar a la mano del usuario y tendrá los dedos descubiertos, en la parte posterior de la mano.

Utilizaremos la tarjeta Arduino como el controlador del guante ya que tiene las características que necesitamos, su tamaño reducido nos servirá a posicionarlo de una forma más cómoda, su bajo costo y también porque la programación del Arduino es bastante sencilla.

Teoría Fundamental:

Acelerómetro y giroscopio:

El acelerómetro empleado en este prototipo es un acelerómetro semiconductor, funciona con un electrodo con la suficiente masa para moverse cuando mueves el acelerómetro, este se encuentra rodeado de placas metálicas que juntas trabajan como capacitores, cuando el electrodo central se mueve, cambia la capacitancia del acelerómetro. Del mismo modo se mide la rotación con el giroscopio, esta vez, sin embargo, se mide el movimiento angular en lugar del lineal.

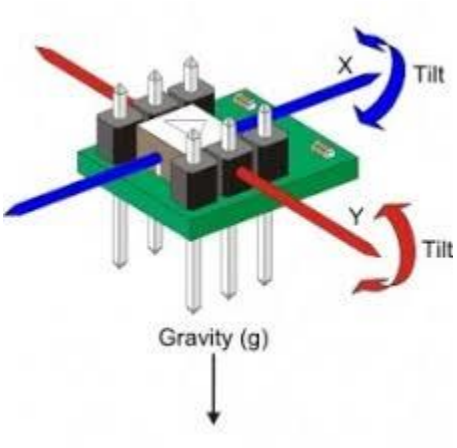

Imagen 7. Capítulo 2.

Materiales:

Ingreso		Egreso	
Omar	$3,800	Chip GY-521	$85
David	$400	Arduino uno	$160
		Material para guante	$150
		Circuito de potencia	$100
		Teléfono (Android)	$3500
		Gafas de RV (para celular)	$72.25
Total de ingreso	$4,200	Total de egreso	$4,067.25

Capítulo III

DISEÑO/DESARROLLO DEL PROYECTO O PROTOTIPO

Diseño:

El guante será negro y contará con lazos de velcro para ser ajustable, en dorso de la mano estará el sensor mpu5060 cuyos cables saldrán por la parte de inferior para llegar al Arduino. El Arduino estará en una segunda pieza que se ubicará en antebrazo junto con la fuente de poder y el módulo de bluetooth.

Escenario:

Para el escenario optamos por hacerlo parecido a un juego de carnaval, utilizamos la menor cantidad de objetos posibles para que no se requiera de tanto procesamiento.

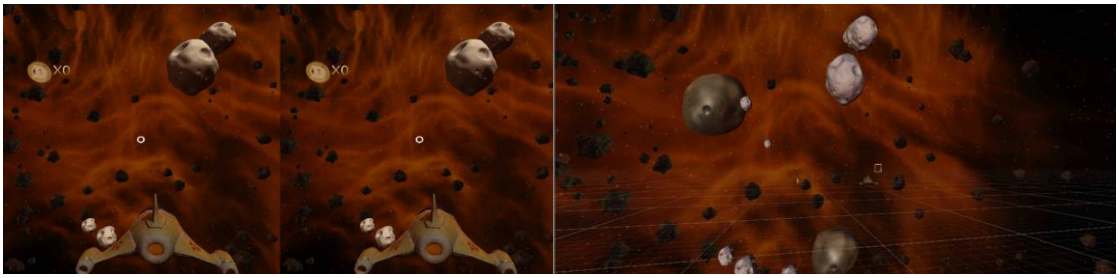

Imagen 8 y 9. Capítulo 3.

En las siguientes imágenes podemos ver lo que ve el usuario con las gafas y como se ve sin las gafas.

Imagen 10 y 11. Capítulo 3.

Código de Arduino

```
#include "I2Cdev.h"
#include<SoftwareSerial.h>
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
MPU6050 mpu;
SoftwareSerial BT(10,11);
//MPU6050 mpu(0x69);
float ACCELERATION_RC_CONSTANT = 30;
#define LED_PIN 13
bool blinkState = false;
bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];
Quaternion q;
VectorInt16 aa;
VectorInt16 aaReal;
VectorInt16 aaWorld;
VectorFloat gravity;
float outputDataX = 0;
float prevOutputDataX = 0;
float outputDataY = 0;
float prevOutputDataY = 0;
float outputDataZ = 0;
float prevOutputDataZ = 0;
```

```
float unfilteredPreviousX = 0;
float unfilteredPreviousY = 0;
float unfilteredPreviousZ = 0;
float startTime = 0;
float currentTime = 0;
float currentTimeSeconds = 0;
float seconds = 0;
float a = 0;
String data = "";
volatile bool mpuInterrupt = false;
void dmpDataReady() {
    mpuInterrupt = true;
}
void setup() {
    Wire.begin();
    TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    BT.begin(9600);
    Serial.begin(9600);
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();
    calibrateThatShit();
    if (devStatus == 0) {
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);
        Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
```

```
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;
        packetSize = mpu.dmpGetFIFOPacketSize();
    }
    pinMode(LED_PIN, OUTPUT);
    mpuInterrupt = false;
}
void loop() {
    mpuIntStatus = mpu.getIntStatus();
    fifoCount = mpu.getFIFOCount();
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        mpu.resetFIFO();
        currentTime = 6176;
    } else if (mpuIntStatus & 0x02) {
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
        mpu.getFIFOBytes(fifoBuffer, packetSize);
        fifoCount -= packetSize;
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetAccel(&aa, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
        data = (String)"%"+q.w+"#"+q.x+"#"+q.y+"#"+q.z+"~";
        currentTime =3140;
        BT.print(data);
    }
}
void calibrateThatShit(){
```

```
  mpu.setXGyroOffset(95);
  mpu.setYGyroOffset(-50);
  mpu.setZGyroOffset(-10);
  mpu.setXAccelOffset(-2702);
  mpu.setYAccelOffset(1820);
  mpu.setZAccelOffset(1050);
 }
 void getRealWorldAcc(){
  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetAccel(&aa, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
  mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
 }
```

A continuación, están los códigos de los scripts que se ejecutan en el menú.

Código del Menu (MainMenu)

```
using System.Text.RegularExpressions;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class MainMenu : MonoBehaviour
{
    public TextMeshProUGUI OutText = null; //Objeto con el cual se mostrarán los mensajes
    public TextMeshProUGUI UserNameShow = null;

    public InputField UserName = null; //Cuadros de texto de donde se obtendrán la información del registro
    public InputField Email = null;
    public InputField Password = null;
    public InputField reEnterPassword = null;

    public InputField PasswordLog = null; //Cuadros de texto de donde se obtendrán la información del login
    public InputField UserNameLog = null;

    public float rotateSpeed; //Variable para la velocidad con la que girara el fondo

    private NetworkManager networkManager = null;
```

```
    string expression = "\\w+([-+.']\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*";//Verificador de la estructura del email

    private void Awake()//Al iniciar la aplicación busca el objeto NetworkManager
    {
        networkManager = GameObject.FindObjectOfType<NetworkManager>();
    }

    void Update()
    {
        RenderSettings.skybox.SetFloat("_Rotation", Time.time * rotateSpeed); //Cada frame el fondo gira segun el valor de rotateSpeed
        UserNameShow.text = Data.userName;//Muestra el nombre de usuario
    }

    public void submitRegister()//Acción del botón de registro
    {
        if (UserName.text == "" || Email.text == "" || Password.text == "" || reEnterPassword.text == "")//Verifica que no haya recuadros vacíos
        {
            OutText.text = "Llene los campos faltantes";//Muestra mensaje
        }
        else
        {
            if (Regex.IsMatch(Email.text, expression))//Verifica que la estructura se cumpla
            {
                if (Regex.Replace(Email.text, expression, string.Empty).Length == 0)
                {
                    if (Password.text == reEnterPassword.text)//Verifica que la contraseña se haya escrito correctamente
                    {
                        int length = (Password.text).Length;//Cuenta el tamaño del string
                        if (length >= 6)//Si es mayor o igual a 6
                        {
                            OutText.text = "Procesando...";//Muestra mensaje
                            networkManager.Registrar(UserName.text, Email.text, Password.text);//Pasa los datos del registro a NetworkManager
                        }
                        else
                        {
                            OutText.text = "La contraseña debe tener almenos 6 caracteres";//Muestra mensaje
                        }
                    }
                    else
                    {
                        OutText.text = "Las contraseñas no son idénticas";//Muestra mensaje
                    }
                }
                else
                {
                    OutText.text = "La direccion de email no es valida";//Muestra mensaje
                }
            }
            else
            {
                OutText.text = "La direccion de email no es valida";//Muestra mensaje
            }
        }
    }

    public void submitLogin()//Acción del botón de login
    {
```

```
        if (UserNameLog.text == "" || PasswordLog.text == "")//Verifica que no haya recuadros vacíos
        {
            OutText.text = "Llene los campos faltantes";//Muestra mensaje
        }
        else
        {
            OutText.text = "Procesando...";//Muestra mensaje
            networkManager.IniciaarSecion(UserNameLog.text, PasswordLog.text);//Pasa los datos del login a
NetworkManager
            PasswordLog.text = "";//Vacía el campo Password
        }
    }

    public void puntajes()//Acción del botón de puntajes
    {
        networkManager.Score();//Llama al networkManager
    }

    public void CloseSession()//Cerrar sesión
    {
        Data.userName = "";//Se borra la variable de sesión
    }

    public void PlayGame()//Acción del botón Empezar
    {
        //Cambia de escena para dar inicio al juego
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    public void QuitGame()//Acción del botón Cerrar
    {
        //Cierra la aplicación
        Application.Quit();
    }
    public void puntajes2()//Acción del botón de puntajes
    {
        networkManager.Score2();//Llama al networkManager
    }
}
```

Código de las conexiones con la base de datos (NetworkManager)

```
using System.Collections;
using TMPro;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;

public class NetworkManager : MonoBehaviour
{
    public TextMeshProUGUI OutText = null; //Objeto con el cual se mostrarán los mensajes
    public GameObject logpanel = null;
    public GameObject logtext = null;
    public GameObject OutConection = null;
    public Image icon = null;
    public GameObject logbutton = null;
    public GameObject closebutton = null;
    public GameObject userNameShow = null;
    public GameObject data = null;
    private HighScore score = null;

    private void Awake()//Al iniciar la aplicación busca el objeto HighScore
```

```
    {
        score = GameObject.FindObjectOfType<HighScore>();
        if (GameObject.FindObjectOfType<Data>() == null)//Si no existe Data lo crea
        {
            Instantiate(data);
        }
    }

    void Update()//Revisa la conexión a internet en cada frame
    {
        if (Application.internetReachability != NetworkReachability.NotReachable && Data.userName == "")//si ha conexión a internet
        {
            icon.sprite = Resources.Load<Sprite>("Images/vacio");//No muestra el icono de "fuera de línea"
            logbutton.SetActive(true);//Activa el botón de login
            closebutton.SetActive(false);//Desactiva el botón de cerrar sesión
            OutConection.SetActive(false);//Desactiva el mensaje de fuera de línea
            userNameShow.SetActive(false);//Desactiva el nombre del usuario
        }
        else if (Application.internetReachability != NetworkReachability.NotReachable && Data.userName != "")//si ha conexión a internet
        {
            icon.sprite = Resources.Load<Sprite>("Images/vacio");//No muestra el icono de "fuera de línea"
            logbutton.SetActive(false);//Desactiva el botón de login
            closebutton.SetActive(true);//Activa el botón de cerrar sesión
            userNameShow.SetActive(true);//Activa el nombre del usuario
            OutConection.SetActive(false);//Desactiva el mensaje de fuera de línea
        }
        else//En caso opuesto
        {
            icon.sprite = Resources.Load<Sprite>("Images/icono1");//Muestra el icono de "fuera de línea"
            logbutton.SetActive(false);//Desactiva el botón de login
            closebutton.SetActive(false);//Desactiva el botón de cerrar sesión
            userNameShow.SetActive(false);//Desactiva el nombre del usuario
            OutConection.SetActive(true);//Activa el mensaje de fuera de línea
        }
    }

    public void IniciaarSecion(string user, string pass)
    {
        StartCoroutine(Login(user, pass));//Inicia el IEnumerator del Login
    }

    public void Registrar(string user, string email, string pass)
    {
        StartCoroutine(Register(user, email, pass));//Inicia el IEnumerator del Registro
    }

    public void Score()
    {
        StartCoroutine(Puntuaciones());//Inicia el IEnumerator del Score
    }
    public void Score2()
    {
        StartCoroutine(Coins());//Inicia el IEnumerator del Score
    }

    IEnumerator Login(string user, string pass)//Hilo que manda los datos del login al servidor y espera respuesta
    {
        var uri = "http://omargodinez.com/DAR2/login.php?user=" + user + "&pass=" + pass;//Crea el acceso al servidor y manda los datos
```

```
        using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))//Realiza la conexión
        {
            yield return webRequest.SendWebRequest();//Solicita y espera la página deseada

            if (webRequest.isNetworkError)//Si ocurrió un error
            {
                OutText.text = "No se logró conectar con el servidor";//Muestra mensaje
            }
            else
            {
                if(webRequest.downloadHandler.text == "nop")//En caso de respuesta negativa
                {
                    OutText.text = "Los datos no concuerdan";//Muestra mensaje
                }else if(webRequest.downloadHandler.text == "sirve")//En caso de respuesta positiva
                {
                    OutText.text = "Éxito";//Muestra mensaje
                    Data.userName = user;//Guarda el nombre del usuario con sesión abierta
                    OutText.text = "";//Borra el mensaje
                    //GameObject child = logpanel.transform.GetChild(3).gameObject;
                    //child.SetActive(false);
                    logpanel.SetActive(false);//Oculta la ventana de login
                    logtext.SetActive(false);//Oculta el OutText
                }
                else//En cualquier otro caso
                {
                    OutText.text = "Ocurrió un problema";//Muestra mensaje
                }
            }
        }
    }

    IEnumerator Register(string user, string email, string pass)//Hilo que manda los datos del registro al
servidor y espera respuesta
    {
        var uri = "http://omargodinez.com/DAR2/createuser.php?user=" + user + "&email=" + email +
"&pass=" + pass;//Dirección del servidor al cual se accederá
        using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))//Realiza la conexión
        {
            yield return webRequest.SendWebRequest();//Solicita y espera la página deseada

            if (webRequest.isNetworkError)//Si ocurrió un error
            {
                OutText.text = "No se logró conectar con el servidor";//Muestra mensaje
            }
            else
            {
                if (webRequest.downloadHandler.text == "usuario")//Si el usuario ya existe
                {
                    OutText.text = "Ya existe el usuario";//Muestra mensaje
                }else if (webRequest.downloadHandler.text == "email")//Si el email ya está registrado
                {
                    OutText.text = "Ya está registrado el email";//Muestra mensaje
                }else if (webRequest.downloadHandler.text == "registrado")//En caso de respuesta positiva
                {
                    OutText.text = "Usuario creado con éxito";//Muestra mensaje
                }else
                {
                    OutText.text = "Ocurrió un problema";//Muestra mensaje
                }
            }
        }
    }
```

```
    IEnumerator Puntuaciones()//Hilo que pide los puntajes al servidor y espera respuesta
    {
        var uri = "http://omargodinez.com/DAR2/score.php";//Dirección del servidor al cual se accederá
        using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))//Realiza la conexión
        {
            yield return webRequest.SendWebRequest();//Solicita y espera la página deseada

            if (webRequest.isNetworkError)//Si ocurrió un error
            {
                score.colocarP("");//Pasa un dato vacío a HighScore para que este muestre error de conexión
            }
            else
            {
                score.colocarP(webRequest.downloadHandler.text);//Pasa los datos del servidor a HighScore
            }
        }
    }
    IEnumerator Coins()//Hilo que pide los puntajes al servidor y espera respuesta
    {
        var uri = "http://omargodinez.com/DAR2/score2.php";//Dirección del servidor al cual se accederá
        using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))//Realiza la conexión
        {
            yield return webRequest.SendWebRequest();//Solicita y espera la página deseada

            if (webRequest.isNetworkError)//Si ocurrió un error
            {
                score.colocarP("");//Pasa un dato vacío a HighScore para que este muestre error de conexión
            }
            else
            {
                score.colocarP(webRequest.downloadHandler.text);//Pasa los datos del servidor a HighScore
            }
        }
    }
}
```

Código del Volumen del menú (VolumeValueChange)

```
using UnityEngine;

public class VolumeValueChange : MonoBehaviour
{

    // referencia al componente de sonido
    private AudioSource audioSrc;

    // variable modificadora del volumen
    private float musicVolume = 1f;

    void Start()
    {

        // asignación del componente de audio
        audioSrc = GetComponent<AudioSource>();
    }

    void Update()
    {

        // aplicación de la opcion de volumen en el componente de sonido
```

```
        audioSrc.volume = musicVolume;
    }

    // Toma el valor de la slider y lo pasa a la variable del volumen
    public void SetVolume(float vol)
    {
        musicVolume = vol;
    }
}
```

Código de la table de puntuaciones (HighScore)

```
using TMPro;
using UnityEngine;

public class HighScore : MonoBehaviour
{

    public TextMeshProUGUI Name1 = null;//Bloques de texto de donde se pondrá la información de los
puntajes
    public TextMeshProUGUI Point1 = null;
    public TextMeshProUGUI Name2 = null;
    public TextMeshProUGUI Point2 = null;
    public TextMeshProUGUI Name3 = null;
    public TextMeshProUGUI Point3 = null;
    public TextMeshProUGUI Name4 = null;
    public TextMeshProUGUI Point4 = null;
    public TextMeshProUGUI Name5 = null;
    public TextMeshProUGUI Point5 = null;
    public TextMeshProUGUI Error = null;

    public void colocarP(string datos)//Acción de recibir y mostrar las mejores puntuaciones
    {
        if(datos == "" || datos == "Fallo la conexión")//Si recibe un parámetro vacío o un mensaje de fallo
notifica el error
        {
            Error.text = "No se logró conectar con el servidor";//Muestra mensaje
        }
        else//Si no hay error
        {
            string[] valores;//Arreglo donde se guardaran los datos de los puntajes ya separados
            valores = datos.Split(' ');//Separa la cadena recibida en cada espacio vacío
            Name1.text = valores[0];//Muestra el nombre
            Point1.text = valores[1];//Muestra el puntaje
            Name2.text = valores[2];
            Point2.text = valores[3];
            Name3.text = valores[4];
            Point3.text = valores[5];
            Name4.text = valores[6];
            Point4.text = valores[7];
            Name5.text = valores[8];
            Point5.text = valores[9];
        }
    }
}
```

Código de almacenamiento de variables (Data)

```
using UnityEngine;

public class Data : MonoBehaviour
{
    public static string userName = "";//Variable que guarda el nombre del usuario

    public void Awake()//Al iniciar la aplicación se declara como no destructible
    {
        DontDestroyOnLoad(gameObject);//El objeto no se destruirá al cambiar de escena
    }
}
```

Códigos de scripts que se ejecutan en el juego de la nave:

Código del botón de inicio y final del juego(startB)

```
using UnityEngine;
using UnityEngine.SceneManagement;


namespace RayCaster.Utils
{
    public class startB : MonoBehaviour
    {
        public InteractivoVR RayCast;//Captador de raycast
        public GameObject Objeto, TObj;//El objeto que contiene el script
        private float time = 0;//Contador
        public bool start = false;//Si el botón es de inicio
        public Material m1;//materiales para cambiar de color
        public Material m2;
        public Material m3;
        public Material non;
        public GameObject spawn;

        void Update()//Se actualiza cada frame
        {
            if (RayCast.EstasMirando == true)//Si el raycaster detecta la retícula
            {
                time += Time.deltaTime;//Se le suma 1 a time cada segundo
                if (time < 1)//Si el tiempo es menor a 1
                {
                    TObj.gameObject.GetComponent<Renderer>().material = m1;//Cambia el material
                }
                time += Time.deltaTime;//Se le suma 1 a time cada segundo
                if (time > 1 && time < 2)//Si el tiempo es mayor a 1 y menor que 2
                {
                    TObj.gameObject.GetComponent<Renderer>().material = m2;//Cambia el material
                }
                time += Time.deltaTime;//Se le suma 1 a time cada segundo
                if (time > 2 && time < 3)//Si el tiempo es mayor a 2 y menor que 3
                {
                    TObj.gameObject.GetComponent<Renderer>().material = m3;//Cambia el material
                }
                if (time > 3)//Si time es mayor a 3
                {
                    if (start)//Si es un botón de inicio
```

```
                {
                    spawn.SetActive(true);
                    Ship.speed = -10.0f;
                    Objeto.SetActive(false);//Se deshabilita el objeto para evitar errores al destruir
                    Initiation.ban = true;//Inicia Initiation
                    Destroy(Objeto); //Se destruye el objeto
                }
                else
                {
                    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);//Cambia de
escena al Return
                }
            }
        }
        else if (RayCast.EstasMirando == false)//Si el raycaster no detecta la retícula
        {
            TObj.gameObject.GetComponent<Renderer>().material = non;//Cambia el material al estado
original
            time = 0;//El valor de time es 0;
        }
    }
  }
}
```

Código del puntaje (Points)

```
using TMPro;
using UnityEngine;

public class Points : MonoBehaviour
{
    public TextMeshProUGUI ScoreBar = null;//Text que mostrara el puntaje
    private int num = 0;//Valor inicial
    public static int points;//Variable donde se almacenarán los puntos
    public bool coins = false;

    private void Awake()
    {
        points = num;//Restaura valor en 0 al iniciar
    }
    public void OnGUI()
    {
        if (!coins)//Si son monedas
        {
            ScoreBar.text = "" + points;//Muestra los puntos
        }
        else
        {
            ScoreBar.text = "X" + points;//Muestra los puntos
        }
```

```
    }
}
```

Código de los Eventos (Effect)

```
sing System.Collections;
using TMPro;
using UnityEngine;
using UnityEngine.Networking;

public class Effect : MonoBehaviour
{
    public GameObject effect;
    private int score;
    public TextMeshProUGUI pointsText;//Text donde mostrara el puntaje final
    public TextMeshProUGUI  OutText;//Text donde mostrara mensaje de la base de datos
    public GameObject panel;//Pantalla de game over

    public void GameOver()//Proceso que se ejecuta cuando Timer llega a 0
    {
        panel.SetActive(true);//Aparece la pantalla de game over
        score = Points.points;//Obtiene el puntaje
        pointsText.text = "" + score + " Points";//muestra puntaje
        StartCoroutine(NewScore(score));//Inicia el IEnumerator del NewScore
    }

    public void Create(Vector3 position)//Obtiene la posicion de la diana destruida
    {
        Instantiate(effect, position, Quaternion.identity);//Crea una animación en la posición de la diana
        StartCoroutine(NewCoins());
    }

    IEnumerator NewScore(int score)//Hilo que manda los datos del login al servidor y espera respuesta
    {
        var user = Data.userName;
        var uri = "http://omargodinez.com/DAR2/newScore.php?user=" + user + "&points=" + score;//Crea el
acceso al servidor y manda los datos
        using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))//Realiza la conexión
        {
            yield return webRequest.SendWebRequest();//Solicita y espera la página deseada

            if (webRequest.isNetworkError)//Si ocurrió un error
            {
                OutText.text = "No se logró conectar";//Muestra mensaje
            }
            else
            {
                if (webRequest.downloadHandler.text == "registrado")//Si se registró el puntaje
                {
                    OutText.text = "Top 5";//Muestra mensaje
                }
            }
        }
    }

    IEnumerator NewCoins()//Hilo que manda los datos del login al servidor y espera respuesta
    {
        score = Points.points;//Obtiene el puntaje
```

```
        pointsText.text = "X  " + score;//muestra puntaje
        var user = Data.userName;
        var uri = "http://omargodinez.com/DAR2/newCoins.php?user=" + user + "&points=" + score;//Crea el
acceso al servidor y manda los datos
        using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))//Realiza la conexión
        {
            yield return webRequest.SendWebRequest();//Solicita y espera la página deseada

            if (webRequest.isNetworkError)//Si ocurrió un error
            {
                OutText.text = "No se logró conectar";//Muestra mensaje
            }
            else
            {
                if (webRequest.downloadHandler.text == "registrado")//Si se registró el puntaje
                {
                    OutText.text = "Top 5";//Muestra mensaje
                }
            }
        }
    }
}
```

Código de los asteroides (Asteroid)

```
using UnityEngine;

public class Asteroid : MonoBehaviour
{
    public Ship ship;
    public Transform player;
    public Destroid effect;
    public float speedR = 5.0f;
    public float speed = 0.0f;
    private float dist = 0.0f;
    public bool IsXL, IsXXX = false, IsXX = false, IsX = false, IsI = false, fallow = false;
    public GameObject I, X, XX, XXX, This;
    private Vector3 target;

    private void Awake()//Al iniciar la aplicación busca el objeto Ship y
    {
        if (GameObject.FindObjectOfType<Ship>() != null)
        {
            player = GameObject.FindObjectOfType<Ship>().GetComponent<Transform>();
        }
        ship = GameObject.FindObjectOfType<Ship>();
        effect = GameObject.FindObjectOfType<Destroid>();
    }
    private void Start()
    {
        if (IsI || IsX)
        {
            Destroy(This, 25);
        }
        if (Random.Range(0.0f, 100.0f) > 50.0f)
        {
            speedR *= -1;
        }
        if (Random.Range(0.0f, 100.0f) > 70.0f || IsXL)
        {
```

```
            fallow = true;
        }
    }

    // Update is called once per frame
    void Update()
    {
        if (GameObject.FindObjectOfType<Ship>() == null)
        {
            destroid();
        }
        else
        {
            dist = Vector3.Distance(player.transform.position, transform.position);
            target = player.transform.position;
        }
        if ((IsI || IsXL) && fallow)
        {
            if(dist > 10)
            {
                transform.Rotate(Vector3.up * speedR * Time.deltaTime);
                transform.position = Vector3.MoveTowards(transform.position, target, speed);
            }
            else
            {
                transform.Rotate(Vector3.up * speedR * Time.deltaTime);
                GetComponent<Rigidbody>().AddRelativeForce(Vector3.forward * speed, ForceMode.Impulse);
            }
        }
        else
        {
            transform.Rotate(Vector3.up * speedR * Time.deltaTime);
            GetComponent<Rigidbody>().AddRelativeForce(Vector3.forward * speed, ForceMode.Impulse);
        }
    }

    private void OnTriggerEnter(Collider other)//En caso de colisión
    {
        if (other.gameObject.name == "shipA")//Si el objeto colisionador fue Dart
        {
            ship.GameOver();//Llama la función de destrucción
        }
        else if (other.gameObject.name == "AsteroidXL(Clone)")
        {
            destroid();
        }
        else if (other.gameObject.name == "AsteroidXXX(Clone)" && ((IsI || IsX) || (IsXX || IsXXX)))
        {
            destroid();
        }
        else if (other.gameObject.name == "AsteroidXX(Clone)" && (IsI || IsX))
        {
            destroid();
        }
        else if (other.gameObject.name == "AsteroidX(Clone)" && (IsI))
        {
            destroid();
        }
    }

    private void destroid()
    {
```

```
        //Dependiendo del tamaño sera los meteoritos que apareceran por la colicion
        Vector3 vec = new Vector3(transform.position.x, transform.position.y, transform.position.z);//Guarda la posición
        effect.DestroidX(vec);
        if (IsX)
        {
            Quaternion rot = Quaternion.Euler(new Vector3(Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f)));//Se asignan los valores de rotación
            Quaternion rot2 = Quaternion.Euler(new Vector3(Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f)));//Se asignan los valores de rotación
            Instantiate(I, vec, rot);//Instancia el objeto
            Instantiate(I, vec, rot2);//Instancia el objeto
        }
        else if (IsXX)
        {
            Quaternion rot = Quaternion.Euler(new Vector3(Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f)));//Se asignan los valores de rotación
            Quaternion rot2 = Quaternion.Euler(new Vector3(Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f)));//Se asignan los valores de rotación
            Instantiate(X, vec, rot);//Instancia el objeto
            Instantiate(X, vec, rot2);//Instancia el objeto
        }
        else if (IsXXX)
        {
            Quaternion rot = Quaternion.Euler(new Vector3(Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f)));//Se asignan los valores de rotación
            Instantiate(XX, vec, rot);//Instancia el objeto
        }
        else if (IsXL)
        {
            Quaternion rot = Quaternion.Euler(new Vector3(Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f), Random.Range(0.0f, 360.0f)));//Se asignan los valores de rotación
            Instantiate(XXX, vec, rot);//Instancia el objeto
        }
        Destroy(this.gameObject);
    }
}
```

Código de la cámara (Camer4)

```
using UnityEngine;

public class Camer4 : MonoBehaviour
{
    public GameObject ship;       //Public variable to store a reference to the player game object
    private Vector3 offset, last;         //Private variable to store the offset distance between the player and camera
    double z = 0.000f, x = 0.000f;

    // Use this for initialization
    void Start()
    {
        //Calculate and store the offset value by getting the distance between the player's position and camera's position.
        offset = new Vector3(0, 4, -13);
    }

    void LateUpdate()
    {
        if (GameObject.FindObjectOfType<Ship>() != null)
```

```
        {
            //Según el Angulo de la nave se posicionará la cámara atrás de la nave
            float Ay = ship.transform.transform.eulerAngles.y;
            if (Ay <= 180)
            {
                z = 26 - (Ay * 0.144);
                if (Ay > 0 && Ay < 90)
                {
                    x = Ay * 0.144;
                }
                else
                {
                    x = (Ay - 90) * 0.144;
                    x = 13 - x;
                }
            }
            else
            {
                z = (Ay - 180) * 0.144;
                if (Ay > 180 && Ay < 270)
                {
                    x = ((Ay - 180) * 0.144) * -1;
                }
                else
                {
                    x = ((Ay - 180) * 0.144) - 26;
                }
            }
            float variableZ = (float)z;
            float variableX = (float)x;
            last = new Vector3(variableX, 0, variableZ);
            transform.position = ship.transform.position + offset + last;
        }
    }
}
```

Código de las monedas (DogCoin)

```
using UnityEngine;

public class DogCoin : MonoBehaviour
{
    public Points point;//Objeto Points
    private float speed = 30.0f;

    private void OnTriggerEnter(Collider other)//En caso de colisión
    {
        if (other.gameObject.name == "shipA")//Si el objeto colisionador fue la nave
        {
            Points.points += 1;//Suma los puntos
            Destroy(this.gameObject);//Se destruye
        }
    }

    void Update()
    {
        transform.Rotate(Vector3.up * speed * Time.deltaTime);//Rotación de la moneda
```

```
    }
}
```

Código de destrucción de meteoros (Destroid)

```
using UnityEngine;

public class Destroid : MonoBehaviour
{
    public GameObject effectXL;//Tipos de meteoros
    public GameObject effectXXX;
    public GameObject effectXX;
    public GameObject effectX;

    public void DestroidXL(Vector3 position)//Obtiene la posición
    {
        Instantiate(effectXL, position, Quaternion.identity);//Crea una animación en la posición del objeto
    }
    public void DestroidXXX(Vector3 position)//Obtiene la posición
    {
        Instantiate(effectXXX, position, Quaternion.identity);//Crea una animación en la posición del objeto
    }
    public void DestroidXX(Vector3 position)//Obtiene la posición
    {
        Instantiate(effectXX, position, Quaternion.identity);//Crea una animación en la posición del objeto
    }
    public void DestroidX(Vector3 position)//Obtiene la posición
    {
        Instantiate(effectX, position, Quaternion.identity);//Crea una animación en la posición del objeto    }
}
```

Código de la nave (Ship)

```
using UnityEngine;

public class Ship : MonoBehaviour
{

    public GameObject VrCamera, Over;
    public Effect effect;
    public static float speed = 0.0f;
    float num = 0.0f;
    Vector3 vec, cam;

    private void Awake()
    {
        speed = num;//Restaura valor en 0 al iniciar
        Over.transform.Translate(new Vector3(0,0,0));
        Over.SetActive(false);
    }

    void Update()//Cada frame se mueve hacia adelante
    {
        Vector3 direccion = new Vector3(transform.forward.x, 0, transform.forward.z).normalized * speed *
Time.deltaTime;
```

```
        Quaternion rotacion = Quaternion.Euler(new Vector3(0, transform.root.eulerAngles.y, 0));
        transform.Translate(rotacion * direccion);
    }

    public void GameOver()//Al chocar
    {
        Vector3 vec = new Vector3(transform.position.x, transform.position.y, transform.position.z);//Guarda la
posición
        cam = VrCamera.transform.position;
        Vector3 rec = new Vector3(0, -1, 0);
        Vector3 last = cam + rec;
        Over.transform.Translate(last);//Posiciona la ventana de game over
        Over.SetActive(true);
        effect.Create(vec);//efecto de destrucción
        Destroy(this.gameObject);//Destruye el objeto
    }
}
```

Código de transacción al menu (ReturnMenu)

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ReturnMenu : MonoBehaviour
{
    private float num = 5;//Valor inicial
    public static float time;//Variable para la cuenta regresiva

    private void Awake()//Al iniciar
    {
        time = num;//Se le coloca el valor inicial a Time
    }

    void Update()//Se actualiza en cada frame
    {
        time -= Time.deltaTime;//Se le resta 1 a time cada segundo
        if (time < 0)//Si time es menor a 0
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 2);//Cambia de escena
al Menú
        }
    }
}
```

Códigos de PHP en el servidor, que realizan las interacciones con la Base de datos

Código de establecimiento de conexión con la BD (conection)

```
<?php

        $servidor = 'localhost';
```

```
        $usuario = 'dova';
        $pass = 'Danielcot#1234';
        $db = 'game';
        $conexion = new mysqli($servidor, $usuario, $pass, $db);
?>
```

Código de creación de usuarios (createuser)

```
<?php
        include "conection.php";
        $user = $_GET['user'];
        $email =$_GET['email'];
        $pass = hash("sha256", $_GET['pass']);
        if(!$conexion){
                echo "Fallo la conexion";
        }else{//Revisa si el nombre de usuario existe
                $sql = "SELECT userName FROM user WHERE userName = '$user'";
                $resultado = mysqli_query($conexion, $sql);
                if(mysqli_num_rows($resultado) > 0)
                {
                        echo "usuario";
                        exit();
                }else{//Si no existe el usuario revisa si el correo esta registrado
                        $sql = "SELECT email FROM user WHERE email = '$email'";
                        $resultado = mysqli_query($conexion, $sql);
                        if(mysqli_num_rows($resultado) > 0)
                        {
                                echo "email";
                                exit();
                        }else{//Si no encuentra ninguna coincidencia crea el usuario
```

```
                                        $sql = "INSERT INTO user SET userName = '$user', email = '$email',
password = '$pass'";

                                        $resultado = mysqli_query($conexion, $sql);

                                        echo "registrado";

                                        exit();

                                }

                        }

                }
?>
```

Código de inicio de secion (login)

```
<?php
        include "conection.php";
        $user = $_GET['user'];
        $pass = hash("sha256", $_GET['pass']);
        if(!$conexion){
                echo "fallo";
        }else{//Busca al usuario y verifica que la contraseña coincida
                $sql = "SELECT * FROM user WHERE userName = '$user' AND password = '$pass'";
                $resultado = mysqli_query($conexion, $sql);
                if (mysqli_num_rows($resultado)>0) {
                        echo "sirve";
                }else{
                        echo "nop";
                }
        }
?>
```

Código que agrega los puntajes a la BD (newScore)

```
<?php
	$user = $_GET['user'];
	$points =$_GET['points'];
	$ban = false;
	if ($user != "" && $points > 0) {
		$resultado = array();
		$dbh = new PDO('mysql:host=localhost;dbname=game', "dova", "Danielcot#1234");
		if (!$dbh) {
			echo "Fallo la conexion";
		}else{//Obtiene y compara los mejores puntajes con el nuevo
			$dbh->query("SET NAMES 'utf8'");
			$sql="select * from puntajes order by points DESC limit 5";
			foreach ($dbh->query($sql) as $res)
			{
				$resultado[]=$res;
			}
			for ($i=0; $i < count($resultado); $i++) {
				if ($points > $resultado[$i]["points"]) {
					$ban = true;
				}
			}
			if ($ban) {//Si es mayor a un puntaje, se agrega
				$sql="INSERT INTO puntajes SET userName = '$user', points =
'$points'";
				$dbh->query($sql);
				echo "registrado";
			}else{
				echo "nop";
			}
		}
```

```
        }
?>
```

Código que muestra los mejores puntajes (score)

```
<?php
        $resultado = array();
        $dbh = new PDO('mysql:host=localhost;dbname=game', "dova", "Danielcot#1234");
        if (!$dbh) {
                echo "Fallo la conexion";
        }else{//Obtiene los 5 mejores puntajes de la BD
                $dbh->query("SET NAMES 'utf8'");
                $sql="select * from puntajes order by points DESC limit 5";
                foreach ($dbh->query($sql) as $res)
                {
                        $resultado[]=$res;
                }//Imprime los nombres y puntages
                echo $resultado[0]["userName"]." ".$resultado[0]["points"]." ".$resultado[1]["userName"]."
".$resultado[1]["points"]." ".$resultado[2]["userName"]." ".$resultado[2]["points"]."
".$resultado[3]["userName"]." ".$resultado[3]["points"]." ".$resultado[4]["userName"]."
".$resultado[4]["points"];
        }
        $conn = mysqli_connect("localhost", "dova", "Danielcot#1234", "game");
        if (!$conn) {
           die("Connection failed: " . mysqli_connect_error());
        }//Si hay muchos puntajes guardados, borra varios de los innecesarios
        $sql="select count(points) from puntajes";
        $resultado = $conn->query($sql);
        $valor = $resultado->fetch_assoc();
        if($valor["count(points)"] >= 10){
                $cont = $valor["count(points)"] - 6;
```

```
                $sql="DELETE FROM puntajes order by points ASC limit ".$cont."";
                $conn->query($sql);
        }
        mysqli_close($conn);
?>
```

Capítulo IV

ANALISIS DE RESULTADOS Y CONCLUCIONES

Resultados obtenidos:

El elaborar el guante nos dimos cuenta de que el método por el cual lo queríamos hacer es el más complicado y menos adecuado, esto se debe a que los IMU no son nada adecuados para el rastreo de movimiento, debido a que estos generan mucho Drift, lo que los hace poco precisos y difíciles de controlar.

Aun así, se logró crear una aplicación donde se interactúa utilizando el guante, generando una experiencia inversiva y entretenida, un punto malo que surgió fue, que se requiere de un teléfono más potente de lo esperado.

Referencias

Corporation, N. C. (2019). *C-SPAN.* Obtenido de https://www.c-span.org/organization/?112710/VPL-Research

El universal. (18 de Abril de 2019). *eluniversal.com*. Obtenido de https://www.eluniversal.com.mx/techbit/viajar-sin-salir-de-casa

Electronic Wings. (s.f.). *Electronic Wings*. Obtenido de https://www.electronicwings.com/pic/mpu6050-gyroscope-accelerometer-temperature-interface-with-pic18f4550

Expansión. (7 de Noviembre de 2018). *expansion.mx*. Obtenido de https://expansion.mx/tendencias/2018/11/07/la-realidad-virtual-puede-revolucionar-la-educacion

Fernández, G. C. (Junio de 2017). *repositoriounican.* Obtenido de https://repositorio.unican.es/xmlui/bitstream/handle/10902/12264/Cobo%20Fernandez%20Guillermo.pdf?sequence=1

G., J. B. (s.f.). *Monografias.com* . Obtenido de https://www.monografias.com/trabajos4/realvirtual/realvirtual.shtml

Gironi, D. (4 de Febrero de 2013). *davidegironi*. Obtenido de http://davidegironi.blogspot.com/2013/02/avr-atmega-mpu6050-gyroscope-and.html#.W-T_F_ZFzb3

Google. (2 de Abril de 2019). *Google VR*. Obtenido de https://developers.google.com/vr/discover/

Lanier, R. J. (2017). *Virtual Reality Society*. Obtenido de https://www.vrs.org.uk/virtual-reality-profiles/vpl-research.html

Metz, R. (20 de Junio de 2017). *Technology review*. Obtenido de https://www.technologyreview.es/s/7962/en-busca-del-elemento-que-le-falta-la-realidad-virtual-otras-personas

realidadvirtual.com. (30 de Octubre de 2005). Obtenido de www.realidadvirtual.com/que-es-la-realidad-virtual.htm

Thalmic Labs. (2016). *Myo blog*. Obtenido de https://developerblog.myo.com/

Unity. (s.f.). *Unity documentation*. Obtenido de https://docs.unity3d.com/es/current/Manual/UnityManual.html

Páginas de los modelos 3D:

https://www.turbosquid.com/

https://free3d.com/es/

https://www.freepik.es/

Música utilizada:

“Colossus & Guardian” del videojuego Digimon world next order

“Gestalt Refresh!” del videojuego Digimon world next order