Capítulo 33

Simulador de realidad virtual de máquinas de reacción en cadena

Diego García Orozco
Ramsses Alejandro Chavez Lopez
Jairo Alejandro Navarro Serrano
Jose Luis David Bonilla Carranza
Centro Universitario de Ciencias Exactas e Ingenierías, (CUCEI, UDG)
diego.garcia6534@alumnos.udg.mx
ramsses.chavez6225@alumnos.udg.mx
jairo.navarro@alumnos.udg.mx
jose.bcarranza@academicos.udg.mx

Resumen

En este artículo presentamos el desarrollo de un proyecto en realidad virtual cuyo objetivo es proporcionar un entorno didáctico y entendible para personas que son nuevas en el uso de estas tecnologías. Mediante la implementación de escenas interactivas se buscó un diseño que permitiera la enseñanza, ya sea de forma autodidacta o guiada, del uso de la realidad virtual. Durante el proceso de desarrollo se encontraron varios problemas, entre ellos la compatibilidad de elementos que permiten la funcionalidad coherente entre las distintas partes que conforman el proyecto. A pesar de haberse logrado la construcción de escenas, controles, gestión de datos del proyecto y la integración de una inteligencia artificial, se concluye que la tecnología de realidad virtual es aún inmadura y que requiere de mejores ajustes para alcanzar un mayor potencial y alcance.

I. Introducción

En este artículo, se abordará el desarrollo de un proyecto de realidad virtual con el propósito de demostrar la aplicación de estas herramientas para uso didáctico. El proyecto modular surgió de la curiosidad por explorar el entorno de realidad virtual (Virtual Reality, VR, por sus siglas en inglés) y utilizarlo como medio para replicar acciones del mundo real en un entorno virtual. Para lograr esto, se utilizaron las principales herramientas de desarrollo en el campo, como Unity, Visual Studio Code y un LAMP stack en Docker, junto con el lenguaje de programación C#.

El objetivo principal de este proyecto es mostrar que es posible recrear acciones del mundo real en un entorno virtual y está dirigido a todas aquellas personas interesadas en experimentar el mundo de la realidad virtual. Además, busca ilustrar, a través de ejemplos concretos, la complejidad y los desafíos que implica el desarrollo de un proyecto de este tipo.

Resolver este tipo de proyectos proporciona un conocimiento único en el campo de la realidad virtual, el cual puede ser de utilidad para cualquier persona interesada en acercarse al desarrollo de este tipo de proyectos. Además de los aspectos técnicos, también se abordarán temas como el

costo y el esfuerzo requeridos para llevar a cabo este tipo de proyectos, brindando información clave para aquellos que deseen emprender iniciativas similares.

En resumen, este artículo ofrece una visión detallada sobre el desarrollo de un proyecto de realidad virtual con fines educativos, utilizando herramientas de vanguardia y destacando los desafíos y beneficios asociados. Se espera que el contenido sea una guía valiosa para aquellos interesados en adentrarse en el mundo de la realidad virtual y deseen comprender mejor los aspectos técnicos, las implicaciones y el proceso de desarrollo de proyectos en este campo.

II. Trabajos relacionados

El desarrollo en Unity es el método más común para la programación de proyectos enfocados en videojuegos. Unity es un motor gráfico multiplataforma 2D y 3D, que transforma día a día la industria del videojuego, pertenece a Unity Technologies y sirve para la creación de videojuegos.

Dentro de este se pueden desarrollar juegos para varios dispositivos sin cambiar de plataforma. Aunque cabe mencionar que también se usa para agregar experiencias con realidad virtual, en el cine, la animación, y sus funciones favorecen sectores como el sector salud, automotriz, el de la construcción, entre otros [1] [2].

Muchos juegos populares tienen como base un desarrollo en entorno Unity; sin embargo, el desarrollo en VR es una nueva apuesta, pues este tipo de proyectos son relativamente recientes, lo cual lleva a propuestas que pueden llegar a ser volátiles y pueden presentar un riesgo en cuestiones de tiempo y esfuerzo si se enfoca a un modelo práctico [3].

Este proyecto se desarrolló sobre la plataforma de Oculus Quest, que son unas gafas de realidad virtual que ofrecen la tecnología necesaria para ejecutar software que se encuentre en entorno de realidad virtual. El manejo de datos es un área común de la programación, necesaria para dar seguimiento a datos o guardar información, así como dar acceso a la misma de forma segura, para este proyecto se optó la última, haciendo una validación de un usuario registrado a una base de datos. Para el manejo de

datos se usó un encriptado con SHA-256, que es un algoritmo de hash seguro de 256 bits2 se utiliza para

la seguridad criptográfica. Los algoritmos de hash criptográfico generan hashes irreversibles y únicos. Cuanto mayor sea la cantidad de hashes posibles, menor será la probabilidad de que dos valores creen el mismo hash [12].

Por otro lado, las inteligencias artificiales tienen diferentes ámbitos de uso, desde el análisis de datos e incluso cubrir puestos de trabajo como un call-center. En la búsqueda de una inteligencia artificial que se acoplara a la particularidad del proyecto propuesto, se encontró que IBM Watson era compatible con Unity [23] [24]. Watson es una plataforma de inteligencia artificial que permite incorporar herramientas de IA a un entorno de software, sin importar donde estén alojados, logrando a través de diversas herramientas usar estos datos para mejorar la experiencia de usuario, optimizar procesos y diseñar flujos de trabajo. Mediante la revisión de varios proyectos, como por ejemplo un juego de realidad virtual desarrollado por la empresa Ubisoft *Star Trek: Bridge Crew* se determinó que sí se podía integrar en nuestro proyecto. [24]

Para el control de versiones del proyecto se usó GitHub, que es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprado por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no solo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Para el desarrollo del proyecto se tomaron de inspiración tres softwares de realidad virtual ya existentes:

- *Gadgeteer*, figura 1a, es un juego de rompecabezas de física en realidad virtual desarrollado por el estudio Metanaut, en el que se construyen máquinas de reacción en cadena para resolver y armar rompecabezas con partes móviles y fijas. Se inspira de las llamadas máquinas de Rube Goldberg, las cuales son aparatos principalmente mecánicos cuyo propósito es completar una tarea a través de una pista, cuyo circuito una vez completado termina una tarea (no es

relevante para este tipo de máquinas si la tarea es importante o no). [4]

- *Sports Scramble VR*, figura 1b, es un videojuego desarrollado por el estudio Armature que consiste en integrar una mezcla de deportes que hacen los partidos de tenis, béisbol y bolos con interacción en realidad virtual. [5]
- *Star Trek: Bridge Crew*, figura 1c, es un videojuego de realidad virtual desarrollado por Ubisoft ambientado dentro del universo de Star Trek, el cual consiste en dirigir a la tripulación de la nave Enterprise a través de diferentes misiones. Lo destacable de este software es la integración real de una inteligencia artificial asistida por IBM Watson. [6]

(a) Imagen de presentación del juego Gadgeteer.

(b) Imagen de presentación del juego Sports Scramble VR.

(c) Imagen de presentación del juego *Star Trek: Bridge Crew*

Figura 1: Ejemplos de tres videojuegos desarrollados en realidad virtual: *Gadgeteer* ilustra las mecánicas de los objetos en realidad virtual, *Sports Scramble VR* es ejemplo en el movimiento de controles y coordinación con el jugador y *Start Trek: Bridge Crew* es ejemplo en el uso de inteligencia artificial dentro de las dinámicas de un juego.

III. Desarrollo del proyecto

Para el desarrollo de este proyecto se requirió del siguiente equipo:

- Oculus Quest, figura 2:
    - Versión de Sistema 49845030259200400
    - Versión de Release 50.0.0.195.257.454885341
    - Almacenamiento 128 GB [7]
- Laptop Asus TUF FX506H
    - CPU Intel Core i5-11400H 2.70 GHz
    - 32 GB RAM, 1 TB SSD M.2.
    - GPU Nvidia GeForce RTX 3050Ti 4 GB
- Computadora ensamblada
    - CPU Intel Core i7-10700 2.90 GHz
    - 16 GB RAM, 500 GB SSD M.2.
    - GPU Nvidia GeForce RTX 3060Ti 8 GB

Y el siguiente software:

- Windows 10 Home Edition, Version 22H2, OS Build 19045.2913
- Unity 2019.4.26f1
    - Unity-sdk-core-1.2.3
- Docker versión 4.12.0
    - Xampp versión 8.2.4 sobre imagen con Debian 10
- IBM Watson Assistants 92
- Visual Studio Code version 1.78
- Blender 3.4

El proyecto modular parte de una escena básica en Unity para empezar a desarrollar, figura 4, se tuvo que investigar cómo crear los controles y que fueran compatibles con el VR. Unity proporciona nativamente algunos, pero solo eran disponibles con la versión de Unity 2021.3.9 [8]; debido a que IBM Watson solo se podía usar en la versión de Unity 2019.4.26 y de esta versión a previas, debido a que IBM dejó de dar soporte a versiones

posteriores; en la versión que se trabajó se tuvo que diseñar los controles desde cero que no fue una tarea sencilla, se diseñó manos en Blender y mediante programación en lenguaje C#, se logró tener unos controles funcionales, a partir de esta se fueron añadiendo las principales necesidades de un juego, siendo estas el movimiento libre sobre el mapa y controles de interacción usando para esto Visual Studio Code. [17]

Figura 2: Set de Oculus Quest, el visor de realidad virtual al centro y los controles a los extremos.

En el control de versiones se utilizó la herramienta GitHub, esta fue conectada por medio de la terminal incorporada en Visual Studio Code, permitiendo tener el proyecto en un repositorio y actualizar el mismo con los cambios que efectuaban durante el desarrollo del proyecto; también fue gracias a esta herramienta que una vez actualizada la versión de Unity y darnos cuenta de que esto causaba conflictos se pudo restaurar el proyecto a una versión anterior por medio del Git SHA de un commit, algo parecido a moverse a un branch de desarrollo diferente, permitiendo nuevamente la estabilidad entre los diferentes módulos realizados.

Al momento de integrar la interfaz de realidad virtual se encontró el primer problema, el cual era conseguir una computadora con los requerimientos necesarios para ejecutar el proyecto; los proyectos en VR necesitan de especificaciones que se encuentran en computadoras de altas prestaciones o denominación *gamer* [10]. Una vez solucionado este primer problema, se desarrollan objetos en Blender para generar assets virtuales con los cuales interactuar, figura 3, y se codifican las físicas del juego; así cubriendo las necesidades básicas del módulo.

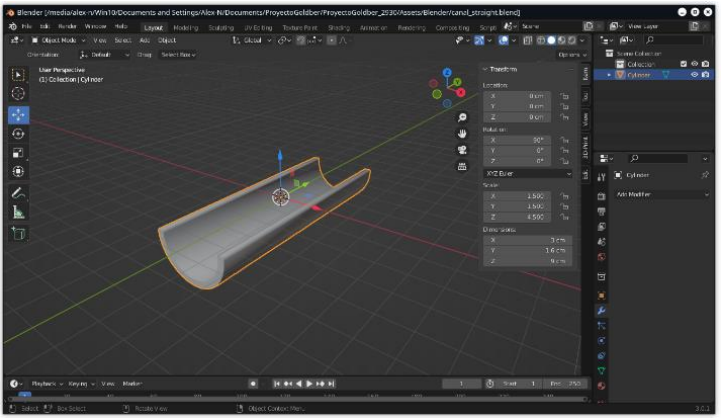


Figura 3: Ejemplo de creación de un asset usando Blender.

Para el módulo II se optó por usar una nueva escena dentro del juego con Unity con una interfaz de login y registro de usuarios básica que registra el nombre del usuario y la contraseña, también se asigna un ID para cada usuario que se registra la contraseña está encriptada mediante SHA-256.

Nuevamente usando Visual Studio Code se codificó un back-end, este para conectarlo a una base de datos remota alojada en un contenedor Docker; a partir de este punto se tuvieron algunas dificultades para poder hacer la conexión con la base de datos, pero mediante a varias revisiones se logra hacer la conexión con éxito.

El módulo III se intentaron diferentes modos para implementar la inteligencia artificial, primero se planteó utilizar MLAgents, que es un framework de IA integrado al entorno de desarrollo de Unity, pero tras diferentes pruebas y errores, no se logró integrar este framework al trabajo con realidad virtual.

La integración con IA se logró por medio de IBM Watson, el cual consiste una conexión por medio de API y árboles de decisión alojada en una base de datos de IBM, la funcionalidad base del chatbot es detectar palabras clave que permite responder al usuario por medio de voz a través de Text-to-Speech.

III-A. Módulo I Gestión de la tecnología de información

Se utilizaron tres escenarios con sus respectivos scripts en lenguaje de programación C#, fue a partir de aquí donde se generó código específico para manipular las físicas del juego, objetos y controles, además de las interfaces visuales en forma de manos humanas para estos mismos.

Figura 4: Vista inicial del programa en VR, tal y como se ver´ıa dentro de Unity.

Manejo de controles de movimiento: Los scripts de manejo de controles se programaron desde los elementos más básicos del framework de trabajo de Unity debido a que para la versión utilizada no hay presentes elementos predefinidos para los controles. Si bien fue un proceso ligeramente laborioso, se obtuvo un resultado satisfactorio, teniendo control de los objetos de manera intuitiva tanto con la toma de objetos de forma remota (control láser o raycast) y la forma manual (similar a la vida real). Este código es un script de movimiento continuo en Unity para aplicaciones de realidad virtual que permite al jugador moverse en el entorno virtual utilizando un controlador de movimiento de mano, como los controladores de Oculus Touch. Las variables de dicho código se listan a continuación [18]:

- Variables públicas:
    - `speed`: La velocidad de movimiento del jugador.
    - `inputSource`: El controlador de mano que se utilizará para el movimiento.
    - `gravity`: La fuerza de gravedad aplicada al jugador.

- `groundLayer`: La capa en la que se considerará que el jugador está en el suelo.
- `additionalHeight`: Altura adicional del jugador para ajustar la posición de la cápsula de colisión.

- Variables privadas:
  - `fallingSpeed`: La velocidad de caída del jugador.
  - `rig`: El componente XRRig que representa el cuerpo del jugador en el mundo virtual.
  - `inputAxis`: El vector de entrada del controlador de mano, que se utiliza para determinar la dirección de movimiento.

Estas variables son invocadas en la función `ContinuousMovement` como se ilustra en el siguiente código:

```
public class ContinuosMovement:MonoBehaviour
    {
    public float speed = 1;
    public XRNode inputSource; public float
    gravity = -9.81f; public LayerMask
    groundLayer;
      public float additionalHeight = 0.2f;
      private float fallingSpeed;
      private XRRig rig;
      private Vector2 inputAxis;
      private CharacterController character;
      // Start is called before the firstframe update
      }
```

La función Start obtiene los componentes `CharacterController` y `XRRig` para el objeto al que se adjunta el script:

```
void Start()
    {
    character =GetComponent<CharacterController>();
    rig = GetComponent<XRRig>();
    }
```

La función `Update` obtiene el dispositivo de entrada (controlador de mano) especificado por `inputSource`. Intenta obtener el valor del eje 2D primario del dispositivo, que se utiliza para el movimiento:

```
void Update()
    {
    InputDevice
device=InputDevices.GetDeviceAtXRNode(inputSource);
    device.TryGetFeatureValue(CommonUsages.primary2DAxi
s,out inputAxis);
    }
```

La función `FixedUpdate` llama a otra función llamada `CalspuleFollowHeadset` para ajustar la altura y posición de la cápsula de colisión según la posición de la cabeza del jugador en el mundo virtual. Calcula la dirección de movimiento en función de la orientación de la cabeza y el vector de entrada del controlador de mano. Mueve al personaje utilizando el componente `CharacterController`, multiplicando la dirección por la velocidad y el tiempo transcurrido desde el último fotograma fijo. Aplica la gravedad al personaje, ajustando su velocidad de caída. Si el personaje está en el suelo, la velocidad de caída se restablece a cero. Mueve al personaje hacia arriba según la velocidad de caída y el tiempo transcurrido desde el último fotograma fijo.

```
private void FixedUpdate()
{
    CapsuleFollowHeadset(); Quaternion
  headYaw =
  Quaternion.Euler(0,rig.cameraGameObject.tr
  ansform.eulerAngles.y,0);
    Vector3 direction = headYaw * new
  Vector3(inputAxis.x,0, inputAxis.y);
  character.Move(direction *Time.fixedDeltaTime *
  speed);
    //gravity
    bool isGrounded = CheckIfGrounded();
    if(isGrounded){fallingSpeed = 0;
    }
  else{
        fallingSpeed += gravity * Time.fixedDeltaTime;
    }
```

```
    character.Move(Vector3.up * fallingSpeed *
  Time.fixedDeltaTime);
}
```

La función `CapsuleFollowHeadset` ajusta la altura de la cápsula de colisión `character.height` para que coincida con la altura de la cabeza del jugador en el mundo virtual. Calcula el centro de la cápsula de colisión, `character.center`, en función de la posición de la cabeza del jugador en el espacio de la cápsula de colisión.

```
void CapsuleFollowHeadset()
{
    character.height = rig.cameraInRigSpaceHeight +
  additionalHeight; Vector3 capsuleCenter =
  transform.Inverse
  TransformPoint(rig.cameraGameObject.transform.posit
  ion); character.center = new
    Vector3(capsuleCenter.x,character.height/2 +
  character.skinWidth,capsuleCenter.z);
}
```

La función `CheckIfGrounded` realiza un raycast, que es una traza en el espacio para indicar o apuntar algún objeto y tener interacción remota con este, figura 5, hacia abajo desde el centro de la cápsula de colisión para comprobar si el personaje está en el suelo. Devuelve `true` si el raycast golpea un objeto en la capa especificada por groundLayer, indicando que el personaje está en el suelo.

```
bool CheckIfGrounded(){
    //tells us if ground Vector3
  rayStart = transform.TransformPoint
  (character.center);
    float rayLength = character.center.y + 0.01f;
    bool hasHit = Physics.SphereCast(rayStart,
  character.radius,Vector3.down, out RaycastHit
  hitInfo, rayLength, groundLayer);
    return hasHit;
}
```

En resumen, este conjunto de scripts permite al jugador moverse en el entorno virtual mediante un controlador de mano, con una cápsula de colisión que se ajusta a la altura de

la cabeza del jugador. También aplica la gravedad para que el personaje pueda caer y saltar en función de su interacción con el suelo.

Manejo de controles de jugador: Para los controles de jugador se construyó una representación de manos en Unity que permitiera mostrar un modelo de mano o un controlador 3D en función de los dispositivos de entrada de VR detectados

[19]. El modelo se compone de las siguientes variables:

- Variables públicas:
    - `showController`: Un booleano que determina si se muestra un controlador 3D o un modelo de mano.
    - `controllerCharacteristics`: Las características de los dispositivos de entrada que se utilizarán para determinar los controladores disponibles.
    - `controllerPrefabs`: Una lista de objetos prefabricados (modelos) que representan los diferentes controladores de VR.
    - `handModelPrefab`: El objeto prefabricado (modelo) que representa una mano.
- Variables privadas:
    - `targetDevice`: El dispositivo de entrada seleccionado para la representación de la mano.
    - `spawnedController`: El objeto instanciado que representa el controlador 3D.
    - `spawnedHandModel`: El objeto instanciado que representa el modelo de mano.
    - `handAnimator`: El componente Animator asociado al modelo de mano.

Para iniciar los dispositivos de entrada se invoca a la función `Start` que a su vez anida a la función `TryInitialize`, que obtiene los dispositivos de entrada disponibles con las

características especificadas. Si se encuentran dispositivos, se selecciona el primero de la lista. Busca el modelo de control correspondiente al nombre del dispositivo y le hace una instancia correspondiente. Si no se

encuentra el modelo de control, se muestra un mensaje de error y se llama al primer elemento de la lista de modelos de control. También se hace una instancia del modelo de mano y se obtiene el componente Animator asociado a este.

```
    void Start()
    {
       TryInitialize();
    }
    void TryInitialize()
    {
         List<InputDevice> devices = new
       List<InputDevice>(); InputDevices.
       GetDevicesWithCharacteristics(controllerCharacteri
       stics,devices);
        //InputDevices.GetDevices(devices)
        foreach (var item in devices{
            Debug.Log(item.name+item.characteristics);
        }
  if(devices.Count > 0){ targetDevice = devices[0];
     GameObject prefab
     =controllerPrefabs.Find(controller =>
     controller.name == targetDevice.name);
           if(prefab){
               spawnedController = Instantiate(refab,
             transform);
           }
           else{Debug.LogError("Modelo de control no
         encontrado"); spawnedController =
         Instantiate(controllerPrefabs[0],transform);
           }
           // spawnedController.SetActive(true);
         spawnedHandModel =
         Instantiate(handModelPrefab,transform);
         spawnedHandModel.SetActive(false);handAnimator=
         spawnedHandModel.GetComponent<Animator>();
       }
    }
```

La función `updateHandAnimation` actualiza las animaciones de la mano en función de los valores de entrada del dispositivo, obtiene el valor del gatillo y lo asigna al parámetro `Trigger` del `Animator` y

además el valor del agarre y lo asigna al parámetro `Grip` del `Animator` [20].

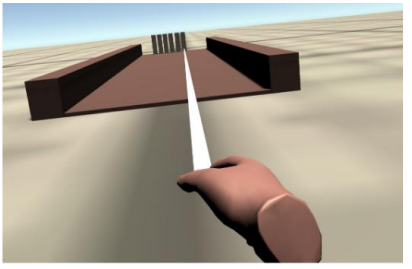
```
void updateHandAnimation () {
        if(targetDevice.TryGetFeatureValue(CommonUsag
     es.trigger, out float triggerValue))
    {
            handAnimator.SetFloat("Trigger",triggerVa
         lue);
    }
  else{
        handAnimator.SetFloat("Trigger", 0);
    }
    if(targetDevice.TryGetFeatureValue(CommonUsages
  .grip, out float gripValue))
    {
        handAnimator.SetFloat(Grip", gripValue);
    }
  else{
        handAnimator.SetFloat("Grip", 0);
    }
}
```

La función `Update` comprueba si el dispositivo de entrada seleccionado sigue siendo válido, si no es válido, se intenta inicializar nuevamente. Si el dispositivo es válido, se muestra el controlador 3D o el modelo de mano según el valor de `showController`: si es verdadero, se desactiva el modelo de mano y se activa el controlador 3D; si es falso, se activa el modelo de mano y se desactiva el controlador 3D. Luego, se actualizan las animaciones de la mano.

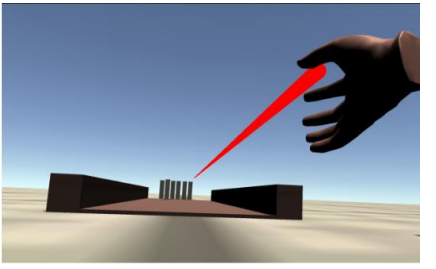
```
void Update()
{
    if(!targetDevice.isValid)
    {
        TryInitialize();
    }
  else
    {
        if(showController)
      {
```

```
            spawnedHandModel.SetActive(false);
          spawnedController.SetActive(true);
          }
      else {
         spawnedHandModel.
          ‹ SetActive(true); spawnedController.
             ‹     SetActive(false);
         updateHandAnimation();
         }
      }
}
```

Este conjunto de scripts permite mostrar una representación visual de las manos del usuario en una aplicación de realidad virtual. Puede mostrar un modelo de mano o un controlador 3D según la preferencia del usuario. Además, anima la mano en función de las entradas del dispositivo de VR, como el gatillo y el agarre.



(a) Mano y raycast, en un ángulo donde se apunta a un objeto.



(b) Mano y raycast, en un ángulo donde no se apunta a un objeto.

Figura 5: Ejemplos de cómo se visualiza la mano de control y raycast, el cuál cambia de color de acuerdo con se tenga (a) apuntado a un objeto o (b) no se esté apuntando a nada.

III-B. Módulo II Sistemas robustos, paralelos y distribuidos

Para el manejo de cuentas de usuario, se implementó una escena previa al acceso al juego que permitía a los usuarios realizar acciones básicas como el registro y la lectura de usuarios, figura 6. Con el objetivo de lograr esta funcionalidad, se utilizó una solución basada en contenedores de Docker. Docker es una plataforma que permite empaquetar y distribuir aplicaciones junto con sus dependencias en contenedores, lo que facilita la portabilidad y la escalabilidad del sistema.

En el contexto de este proyecto, se creó un contenedor Docker con el sistema operativo Debian 10. Este sistema operativo proporciona una base estable y confiable para el funcionamiento del servidor. Además, se instaló Xampp dentro del contenedor Docker, lo que permitió configurar un LAMP Stack (Linux, Apache, MySQL y PHP) para el manejo de la base de datos y el servidor web de PHPMyAdmin. El LAMP Stack proporciona una infraestructura sólida para el desarrollo de aplicaciones web, ya que combina el sistema operativo Linux como base, el servidor web Apache para la gestión de las peticiones HTTP, el sistema de gestión de bases de datos MySQL para almacenar y administrar la información de los usuarios, y el lenguaje de programación PHP para la interacción entre la aplicación y la base de datos. Mediante esta configuración, se pudo establecer un entorno de servidor completo dentro del contenedor Docker, lo que permitió gestionar las cuentas de usuario de manera eficiente. Los usuarios podían registrar nuevas cuentas y acceder a sus perfiles existentes a través de la interfaz proporcionada por el servidor web en el contenedor.

Para lograr la comunicación entre el programa de realidad virtual y el servidor, se implementaron scripts en C# que interactúan con otro conjunto de scripts en PHP. Esta elección se basó en el hecho de que PHP es un lenguaje comúnmente utilizado en el desarrollo de bases de datos y aplicaciones de backend en entornos de nube.

La configuración de esta comunicación se realizó dentro del servidor Docker, que actúa como un entorno aislado y seguro para ejecutar la aplicación. Dentro del servidor Docker, se configuró un sistema de base de datos utilizando phpMyAdmin, una herramienta popular para la administración de bases de datos MySQL mediante una interfaz web [13].

La configuración de esta comunicación se realizó dentro del servidor Docker, que actúa como un entorno aislado y seguro para ejecutar la aplicación. Dentro del servidor Docker, se configuró un sistema de base de datos utilizando phpMyAdmin, figura 7, una herramienta popular para la administración de bases de datos MySQL mediante una interfaz web. El código que se requirió para el desarrollo de la conexión con base de datos desde Unity es un reto considerable, pues Unity como tal es simplemente una interfaz interactiva que compila los scripts del juego. Las líneas de

código que se presentan a continuación definen cómo se abordó el problema.

(a) Pantalla de inicio de sesión.

(b) Pantalla de registro de usuario.

Figura 6: Las pantallas de inicio de sesión (a) y de registro de usuario (b)son el par de pantallas que el usuario ve por primera vez al ejecutar el programa.

La primera sección del código de acceso al juego es la clase, la cual es una base generada por Unity con un nombre dado al generar un script, en este caso Login. Dentro de la misma línea encontramos `MonoBehaviour` [21], la cual es una clase nativa de Unity que contiene un conjunto de variables, objetos y funciones internas [14].

```
public class Login : MonoBehaviour
{
public InputField nameField; public InputField
passwordField; public Button submitButton;
public void CallLogin(){
      StartCoroutine(LoginPlayer());
}
```

Debajo de esto tenemos 3 variables, que se definen de la siguiente manera: dos `Input field nameField` y `passwordField`, y un `Button.submitButton`, los primeros 2 intuitivamente los usaremos para la captura de datos de usuario y el botón para la confirmación de estos datos. Estas variables tienen la propiedad `public` con la finalidad de depurar de errores estos campos abiertamente desde la interfaz Unity. Finalmente, para este bloque de código se tiene la llamada de función, `CallLogin()`, la cual inicializa una función interna de Unity de tipo `Coroutine(startCoroutine)` [9] que llama a la función principal de inicio de sesión (`LoginPlayer()`). En esta sección se utiliza un

`IEnumerator`, el cual solamente se puede utilizar por parte de las `Coroutine`; dentro del bloque de código tenemos una variable basada en `WWWForm`, este se utiliza para mandar información a un servidor web - el cual en nuestro caso sería nuestra base de datos.

```
IEnumerator LoginPlayer(){ WWWForm form = new
WWWForm();
form.AddField("name",nameField.text);
form.AddField("password",passwordField.text);
```

Esta variable form podría considerarse un arreglo de información, en la cual empuja los campos "name" y "password" referenciando los `InputField` del bloque anterior, especificando que únicamente necesitamos la propiedad de texto como indica la rutina `.text`.

La siguiente sección de código sencillamente se encarga de comunicarse con la web con el form que previamente llenamos con los datos de usuario [16].

```
WWW www = new
WWW("http://localhost:41062/sqlconnect/login.php",
form);
yield return www;
```

Finalmente, tenemos la sección de validación de datos; verificamos que el texto de nuestro sitio no esté vacío indicando que sí existe una conexión a base de datos web; una vez validado, se utiliza el administrador de escenas del motor de Unity para movernos a una escena del juego, sí la validación falla no se permite el acceso al juego y dentro de la interfaz de depuración de Unity recibimos un mensaje.

```
if(www.text[0]== '0'){
UnityEngine.SceneManagement.SceneManager.LoadSc
ene(4);
     }
else{
    Debug.Log("User login failed.Error #"+ www.text );
     }
```

Todos estos archivos son accesibles al público dentro del repositorio del juego, anexado en la sección de referencias número [27]; el código que acabamos de describir anteriormente se puede encontrar bajo la ruta `\Assets\Scenes\Login.cs`.

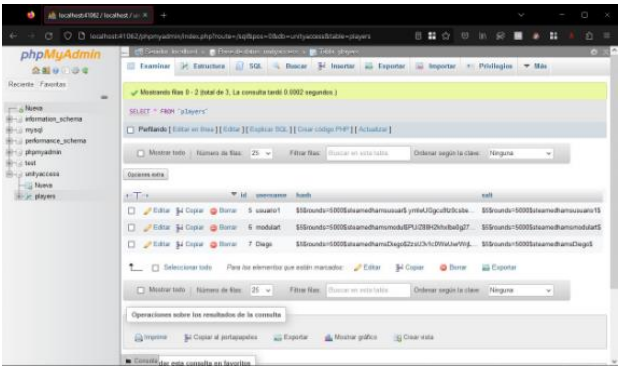


Figura 7: Vista principal del administrador de base de datos PHPMyAdmin, el cuál es ejecutado en el navegador web conectándose a la IP del contenedor de Docker que alberga el sistema de registro.

Para finalizar el detalle sobre los registros en base de datos, se incluye el uso de Salt es una medida de ciberseguridad que mejora la protección de las contraseñas almacenadas en la base de datos. El proceso de encriptación mediante el uso de Salt garantiza que incluso si la base de datos se ve comprometida, las contraseñas no serán fácilmente descifrables.

Esta forma resultó ser más práctica para mostrar un sistema distribuido, puesto que permite ahorrar en montar una red física y aprovecha el potencial de Docker en materia de práctica para sistemas distribuidos.

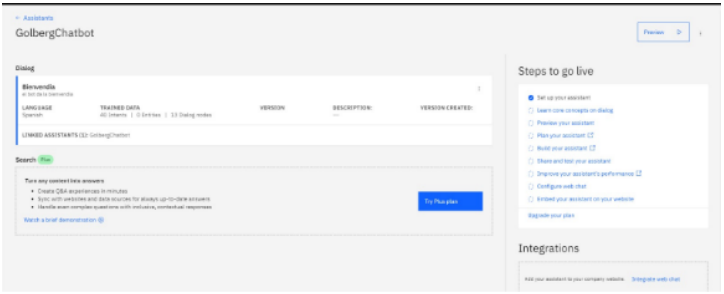
III-C. Módulo III Justificación de Cómputo Flexible (soft-computing)

Para el tercer módulo se desarrolló el Chatbot de GoldbergChatBot. En este chatbot se utilizó tecnología de IBM Watson, figura 8a, el cual es un elemento interactivo a base de reconocimiento de voz con inteligencia artificial utilizando una base de conocimientos o árbol de decisiones alojado en servidores de IBM, por los cuales se analiza una cadena de caracteres obtenida de la voz por parte del usuario; al finalizar el análisis de la voz dentro del motor de IBM Watson se ejecuta el árbol de decisiones, una vez

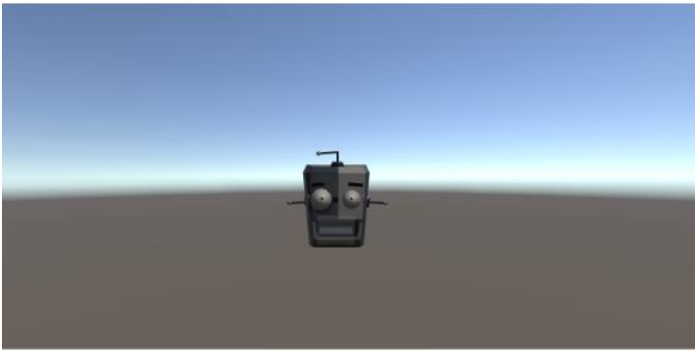
hecho esto, el árbol de decisiones devuelve una respuesta, la cual es escuchada por medio de text-to-speech para responder a las preguntas del usuario [26].

IBM Watson funciona a partir de la construcción de árboles de decisión, figura 8b. Los árboles de decisión son una técnica de aprendizaje automático utilizada en IBM Watson para la toma de decisiones basada en una serie de reglas y condiciones. Estos árboles se utilizan para representar y visualizar las opciones y resultados posibles en forma de un árbol jerárquico. Estos árboles de decisión se crean mediante algoritmos de aprendizaje automático que analizan un conjunto de datos de entrenamiento y generan un modelo predictivo. Este modelo se construye utilizando características o atributos del conjunto de datos, y establece reglas y condiciones que permiten clasificar o predecir nuevas instancias [23].

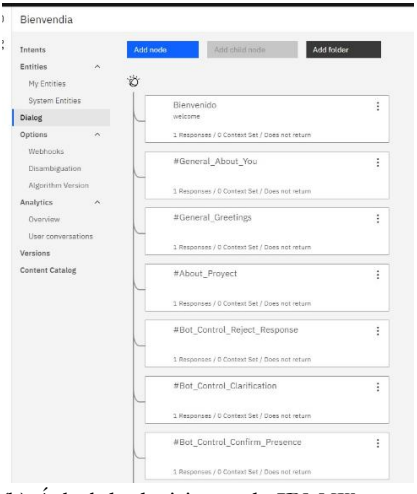
Dentro de este funcionamiento, se implica dividir el conjunto de datos inicial en función de las características que mejor discriminan las distintas clases o categorías de interés. Esta división se basa en la selección del atributo más informativo en cada paso, utilizando medidas como la ganancia de información o la impureza de Gini. Una vez que se ha construido el árbol de decisión, este se puede utilizar para realizar predicciones o clasificar nuevas instancias según las reglas y condiciones establecidas. La instancia se sigue por el árbol, evaluando las características en cada nodo y siguiendo las ramas correspondientes hasta llegar a una hoja, donde se toma una decisión final. Los árboles de decisión se utilizan en diversas aplicaciones que requieran una toma de decisiones. Estos modelos permiten automatizar procesos y ayudar a los usuarios a tomar decisiones informadas basadas en datos y patrones identificados en el conjunto de entrenamiento.



(a) Página principal de IBM Watson.



(b) Escena de IA con un asset que le representa.



(b) Árbol de decisiones de IBM Watson.

Figura 8: El trabajo con la IA involucro la inclusión del motor de IBM Watson (a) basándose en árboles de decisión (c) que toma diferentes opciones basándose en el diálogo con el usuario. Un asset con forma de cabeza de robot (b) se encarga de representar a la IA que interactúa con el usuario.

IV. Resultados obtenidos

Durante la realización del proyecto con realidad virtual, se buscaba crear una experiencia inmersiva para los usuarios. Sin embargo, es importante destacar que los resultados obtenidos fueron contrastantes, con logros significativos en ciertas áreas, pero también dificultades importantes en otras.

En primer lugar, el proyecto logró desarrollar escenas primordiales con detalle suficiente como para que los usuarios aprendieron a dirigirse con los controles para realidad virtual y se encontrarán inmersos en entornos que les permitieron experimentar lo suficiente. La calidad visual y la interactividad lograron crear una sensación envolvente, lo que contribuyó a la creación de una experiencia emocionante y cautivadora.

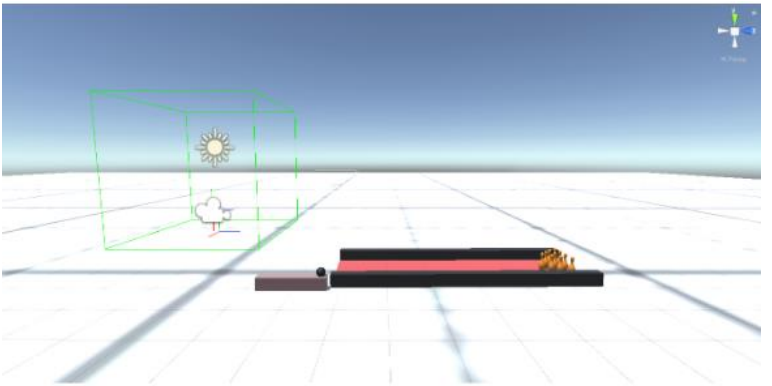
Además, se logró proporcionar una visión prometedora de las posibilidades y limitaciones de la realidad virtual en la creación de experiencias inmersivas. Los resultados positivos en cuanto a la calidad visual y la creación de escenas lo suficientemente detalladas y que puedan comunicar por sí mismas su propósito demuestran el potencial de esta tecnología para sumergir a los usuarios en mundos virtuales convincentes.

Sin embargo, también se encontraron dificultades importantes en ciertas áreas del proyecto. Por ejemplo, la optimización del rendimiento fue un desafío, ya que se requería un equilibrio entre la calidad visual y el rendimiento fluido en diferentes dispositivos. Además, la creación de interacciones complejas y realistas en el entorno virtual resultó ser un proceso complejo y laborioso.

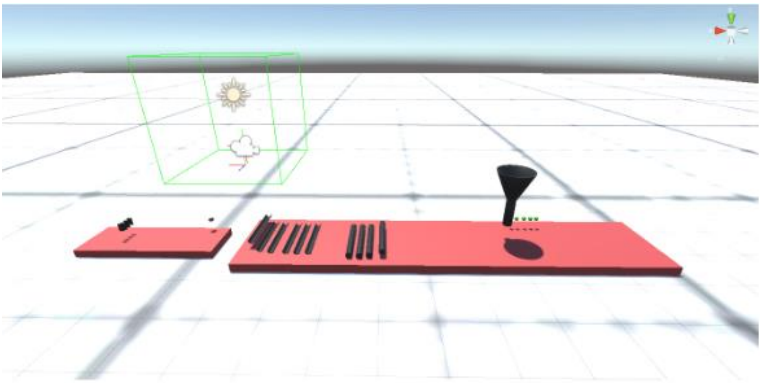
A pesar de estas dificultades, se lograron avances significativos y se sentaron las bases para futuros desarrollos en el campo de la realidad virtual. Es importante destacar que este proyecto sirvió como punto de partida y aprendizaje para comprender mejor los desafíos y oportunidades que surgen al crear experiencias inmersivas con tecnología VR.

El proyecto de realidad virtual logró ofrecer una experiencia inmersiva y emocionante para los usuarios, con escenas lo suficientemente entendibles y una calidad visual aceptable, figuras 9 y 8c. Si bien se enfrentaron dificultades en ciertas áreas, los resultados obtenidos demuestran el potencial de la realidad virtual para crear mundos virtuales convincentes.

Estos hallazgos proporcionan una base sólida para futuros trabajos en el campo de la realidad virtual, donde se podrán abordar los desafíos identificados y seguir mejorando las experiencias inmersivas para los usuarios.



(a) Escena de Boliche.



(b) Escena de máquina de Goldberg.

Figura 9: Escenas finalizadas del proyecto: escena de boliche, (a), en disposición tradicional de juego de boliche, en la cual el usuario puede tomar una bola y tirar de esta sobre una pista para tirar un grupo de pinos. Escena de máquina de Goldberg, (b), se presenta un conjunto de objetos interactivos, tanto móviles como fijos, entre canicas, canaletas y un embudo, el usuario puede tomar cualquiera de estos elementos y ordenarlos a placer.

Asimismo, la implementación de un sistema de registro y acceso con cuenta fue un logro importante, pues generar este tipo de funcionalidades en juegos es ciertamente desafiante, resaltando el hecho de que en esta ocasión se utilizó un entorno VR, el cual se realiza de manera poco común

en este ámbito, lo cual abre las posibilidades de este tipo de programas. Por otro lado, uno de los aspectos más desafiantes fue la implementación de un teclado virtual dentro del entorno de realidad virtual. La versión de desarrollo utilizada presentó dificultades significativas en esta área, lo que afectó la usabilidad y la comodidad de los usuarios.

La interacción con el teclado virtual, aunque siendo poca para el usuario dentro del entorno virtual, resultó poco intuitiva y debía de hacerse de forma asistida desde el exterior (usando el teclado de la computadora), lo que limitó la capacidad de los usuarios para realizar tareas que requerían el uso de un teclado.

Los obstáculos encontrados en la implementación del teclado virtual destacan la importancia de contar con versiones de desarrollo estables y adecuadas para garantizar la funcionalidad y la usabilidad de las interacciones en la realidad virtual. Estos desafíos señalan la necesidad de una mayor investigación y desarrollo en el ámbito de las interfaces de usuario en la realidad virtual, con el fin de superar las limitaciones actuales y mejorar la experiencia general del usuario.

En resumen, se muestran dos resultados contrastantes en el proyecto. Por un lado, se lograron implementaciones significativas en la creación de escenas realistas y envolventes, lo que brindó a los usuarios una experiencia emocionante. Sin embargo, la implementación de aspectos como el teclado virtual se vio obstaculizada por la versión de desarrollo utilizada, lo que plantea desafíos importantes en términos de usabilidad y funcionalidad. Estos resultados destacan la necesidad de seguir explorando y mejorando la tecnología de realidad virtual para maximizar su potencial en la creación de experiencias inmersivas y satisfactorias para los usuarios.

V. Conclusiones y trabajo a futuro

Durante el desarrollo de este proyecto, se lograron crear tres escenas de realidad virtual, incluyendo también las de sesión o inicio de sesión, las cuales incorporaron un proceso de verificación utilizando tecnologías de back-end como PHP y bases de datos para gestionar los datos básicos de registro de usuarios y autenticación.

Uno de los principales desafíos encontrados durante el desarrollo de este proyecto fue asegurar la compatibilidad entre los diferentes componentes de software requeridos, especialmente entre la capa de desarrollo para realidad virtual (Unity) y el componente de Inteligencia Artificial utilizado.

Estos componentes están en constante evolución, lo que a veces ocasiona rupturas o limitaciones en el desarrollo de proyectos que combinan estas tres características. No obstante, se logró superar estos obstáculos y se concluyeron satisfactoriamente las escenas y la funcionalidad del programa, demostrando así un ejemplo práctico del uso de la realidad virtual para el aprendizaje interactivo.

Es importante destacar que la tecnología de realidad virtual aún se encuentra en una etapa temprana de desarrollo y presenta ciertas limitaciones. Aunque se han logrado avances significativos, se puede considerar que la realidad virtual todavía carece de adaptabilidad y flexibilidad suficiente para convertirse en un producto viable de uso diario en diversos ámbitos.

A pesar de las limitaciones actuales, este proyecto ha proporcionado una perspectiva valiosa sobre el potencial y las posibilidades de la realidad virtual en el ámbito educativo. La capacidad de recrear entornos y experiencias inmersivas ofrece una nueva forma de aprendizaje y exploración interactiva. Si bien aún hay desafíos por superar, este proyecto demuestra el valor de incorporar la realidad virtual en el ámbito didáctico y cómo puede mejorar la forma en que las personas aprenden e interactúan con la información.

En conclusión, el desarrollo de este proyecto de realidad virtual ha demostrado la aplicación de la tecnología VR en el ámbito educativo, permitiendo a los usuarios aprender e interactuar de manera inmersiva. Aunque se enfrentaron desafíos técnicos, se logró completar con éxito la creación de las escenas y la funcionalidad del programa. Si bien la realidad virtual aún está en desarrollo y presenta desafíos en términos de compatibilidad y madurez tecnológica, este proyecto resalta el potencial de la realidad virtual como herramienta educativa y sugiere un camino prometedor para futuras investigaciones y mejoras en el campo de la realidad virtual.

Reconocimientos

Para el desarrollo de este proyecto nos gustaría agradecer a las siguientes personas:

- Juan Alberto García Orozco Apoyo en pruebas con el chatbot.
- Isaac García Orozco, Juan Jose García Hernandez, y Maria Julia Leticia Orozco Galvan con el apoyo en pruebas de los minijuegos.
- Leonardo Daniel Santiago por las pruebas y correcciones en el código de la presentación de este artículo.

Referencias

[1] Medium. 2022. How is AI Used in Sports. [online] Available at: https://medium.com/@mygreatlearning/how-is-ai-used-i9n-sports-cdfbdd97ad82.

[2] Deepmind.com. 2022. Advancing sports analytics through AI research. [online] Available at: https://www.deepmind.com/blog/advancing-sports-analytics-through-ai-research.

[3] Elmqaddem, N., 2019. Augmented Reality and Virtual Reality in Education. Myth or Reality?. International Journal of Emerging Technologies in Learning (iJET), [online] Available at: https://online-journals.org/index.php/i-jet/article/view/9289.

[4] Gadgeteer. 2022. [Online] Available at: https://gadgeteergame.com/

[5] Armature, Sports Scramble VR. 2022. [online] Availabe at: https://armature.com/games/sports-scramble/

[6] Ubisoft, Star Trek Bridge Crew. 2022. [online] Available at: https://www.ubisoft.com/en-gb/game/star-trek/bridge-crew

[7] Support.oculus.com. 2021. Explore Oculus Quest Features. [online] Available at: https://support.oculus.com/articles/in-vr-experiences/oculus-features/index-oculus-features.

[8] Technologies, U., 2021. Unity - Manual: Unity User Manual 2020.3 (LTS). [online] Docs.unity3d.com. Available at: https://docs.unity3d.com/Manual/index.html.

[9] Docs.unity3d.com. 2022. Coroutines. [online] Availabe at:

https://docs.unity3d.com/es/2018.4/Manual/Coroutines.html.
[10] Docs.unity3d.com. 2022. VR Overview. [online] Available at: https://docs.unity3d.com/540/Documentation/Manual/VROverview.html
[11] Docs.unity3d.com. 2022. WWWForm. [online] Available at: https://docs.unity3d.com/ScriptReference/WWWForm.html.
[12] Board To Bits Games. Unity & MySQL Databases, Part 1: Setting Up. (20 de abril de 2018). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=SKbY-0zt2VE.
[13] Board To Bits Games. Unity & MySQL Databases, Part 2: Registering Users. (27 de abril de 2018). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=4W90-mh70JY.
[14] Board To Bits Games. Unity & MySQL Databases, Part 3: Parsing Server Data. (4 de mayo de 2018). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=AjBtDvERlyg.
[15] Board To Bits Games. Unity & MySQL Databases, Part 4: User Login. (11 de mayo de 2018). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=NVdjlXgbiMM.
[16] Board To Bits Games. Unity & MySQL Databases, Part 5: Saving User Data. (18 de mayo de 2018). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=0QeRgd40q M.
[17] Valem. Introduction to VR in Unity - PART 1 : VR SETUP. (8 de abril de 2020). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=gGYtahQjmWQ.
[18] Valem. Introduction to VR in Unity - PART 2 : INPUT and HAND PRESENCE. (15 de abril de 2020). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=VdT0zMcggTQ.
[19] Valem. Introduction to VR in Unity - PART 3 : TELEPORTATION. (24 de abril de 2020). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=fZXKGJYri1Y.
[20] Valem. Introduction to VR in Unity - PART 4 : CONTINUOUS MOVEMENT. (19 de mayo de 2020). Accedido el 17 de mayo de 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=5NRTT8Tbmoc.

[21] GameDevTraum. 2022. ¿Qué es MnoBehaviour en Unity? [online] Available at: https://gamedevtraum.com/es/desarrollo-de-videojuegos-y-aplicaciones-con-unity/serie-manejo-general-del-motor-unity/que-es-monobehaviour-en-unity-funcionamiento/.

[22] IBM. 2022. Decision Trees [online] Available at: https://www.ibm.com/topics/decision-rees.

[23] ZDNET, IBM Watson: What are companies using it for? 2015. [online] Availabe at: https://www.zdnet.com/article/ibm-watson-what-are-companies-using-it-for/

[24] GitHub, Unity-SDK, 2022. [online] Available at: https://github.com/watson-developer-cloud/unity-sdk

[25] IBM, IBM and Ubisoft® Partner to Bring Voice Command with Watson to Virtual Reality in Star TrekTM: Bridge Crew. 2017. [Online] Available at: https://uk.newsroom.ibm.com/2017-05-11-IBM-and-Ubisoft-R-Partner-to-Bring-Voice-Command-with-Watson-to-Virtual-Reality-in-Star-Trek-TM-Bridge-Crew

[26] Scott Hwang, Setting Up a 3D Chatbot with Unity / IBM Watson / Oculus Lipsync. 2020. [online] Available at: https://www.youtube.com/watch?v=qz3Ac9Xi29g

[27] hrrhcksthl et al. Proyecto Goldberg. 2023. [online] Available at: https://github.com/hrrhcksthl/ProyectoGoldberg