TECNOLOGÍA EDUCATIVA ISSN: 2519-9463

# VIDEOJUEGO DE PIRÁMIDE HOLOGRÁFICA CON REALIDAD AUMENTADA Y CONTROL DE AVATAR POR MANDO EMBEBIDO

HOLOGRAPHIC PYRAMID VIDEO WITH AUGMENTED REALITY AND CONTROL OF AVATAR BY EMBEDDED CONTROLLER

Ing. Manuel Alejandro García Romero

manuel.garcia@unillanos.edu.co

Dr. C. Javier Andrés Vargas

javier.andres.vargas@unillanos.edu.co

Universidad de los Llanos, Colombia

**Resumen**

El artículo presenta el desarrollo de un videojuego simple para pirámide holográfica de cuatro caras en la plataforma Unity, para esto se situaron cuatro cámaras con resolución de 640x640 pixeles, a la misma altura, distanciadas simétricamente entre sí y enfocando un mismo punto, con un mismo ángulo respecto a la horizontal en el espacio 3d de la plataforma, luego en la pantalla de 1080x1080 pixeles, se escalonó cada cámara, rotándolas y ubicándolas en la pantalla de juego hasta conseguir las vistas de proyección en la pirámide, después se emparentó las cámaras del avatar con una quinta cámara que es situada encima del punto enfocado por las otras cuatro con un ángulo de $90^{o}$ con respecto a la horizontal, esta cámara es de la API de Vuforia para realidad aumentada en Unity, configurada para los paquetes de marcadores necesarios en el juego. Se desarrollaron los personajes en Blender y se empaquetan en un archivos .fbx compatible con Unity, una vez creado el .fbx del avatar y el de los marcadores, se importa a la carpeta Assets del proyecto en Unity y de paso al espacio 3d, seguido a esto se le asocia un script que permite la recepción de datos por el puerto serial para el manejo de acciones y animaciones del mismo. El sistema de control consiste de un microcontrolador con un módulo bluetooth para el envío de la trama de datos al pc y un control embebido de dos ejes para los movimientos del avatar en el plano y un pulsador para saltar.

**Palabras clave**: Pirámide holográfica, realidad aumentada, videojuego, Unity, animaciones, microcontrolador.

**Abstract**

The paper presents the development of a simple video game for four-sided holographic pyramid on the Unity platform. For this were placed four cameras with a resolution of 640x640 pixels, at the same height, spaced symmetrically with each other and focusing on the same point, with the same angle with respect to the horizontal in the 3d space of the platform, then in the screen of 1080x1080 pixels. Each camera was staggered, rotating them and placing them on the game screen to get the projection views in the pyramid. The cameras of the avatar were then paired with a fifth chamber which is situated above the point focused by the other four at an angle of 90 degrees with respect to the horizontal; This camera is from the Vuforia API for augmented reality in Unity and configured for the necessary marker packs in the game. The characters were developed in Blender and packaged in a .fbx files compatible with Unity. Once created the .fbx of the avatar and the bookmarks, it is imported to the Assets folder of the project in Unity then, it is sub imported to 3d space. Followed step, it is associated a script that allows the reception of data by the serial port for the handling of actions and animations of the same. The control system consists of a microcontroller with a bluetooth module for sending the data frame to the pc and a joystick of two axes for the movements of the avatar in the plane and a button to jump.

**Keywords**: Holographic pyramid, augmented reality, video game, unity, animations, microcontroller.

## 1. Introducción

Esta idea nació de la nueva tendencia en holografía con pirámides de cristal, mostrándose como una industria creciente en el negocio del marketing y la publicidad, actualmente hay empresas que se dedican a la comercialización de pirámides holográficas para establecimientos comerciales o usos domésticos como es el caso de las empresas: Holoimagine y DreamocXL.

Estas pirámides no tienen una interacción directa y real entre el consumidor y el holograma, siendo este un factor primordial para ganar la atención del público en general, logrando mejorar las ventas de los productos allí expuestos.

Consideramos que es una excelente alternativa muy diferente al desarrollo convencional de videojuegos para consola, PC o dispositivos móviles, aunque en esencia continúa siendo un juego para computadores. Puede prestar una experiencia de juego donde se mire el personaje desde distintos ángulos, cubriendo una vista de 360 grados a su alrededor.

**Figura 1. Pirámide holográfica con realidad aumentada y mando embebido en funcionamiento.**

Los espacios tridimensionales permiten visualizar procesos y cosas que normalmente no podemos percibir, de hecho nos abre un portal a cualquier realidad, ya que en teoría, es posible crear todo lo que deseemos y animarlo de la forma que necesitemos, pudiendo manipular el tiempo y el espacio a nuestro parecer. Los videojuegos que jugamos desde niños son el mejor ejemplo de ello, crearon todos los mundos y personajes posibles, todo lo que alguien pudiera imaginar para un videojuego, la tecnología de simulación y animación de las plataformas 3d permitió y permitirá cada vez más detalles en los mundos de los juegos.

## 2. Materiales y métodos

En la Figura 2 se muestra el mapa de la metodología trazada.

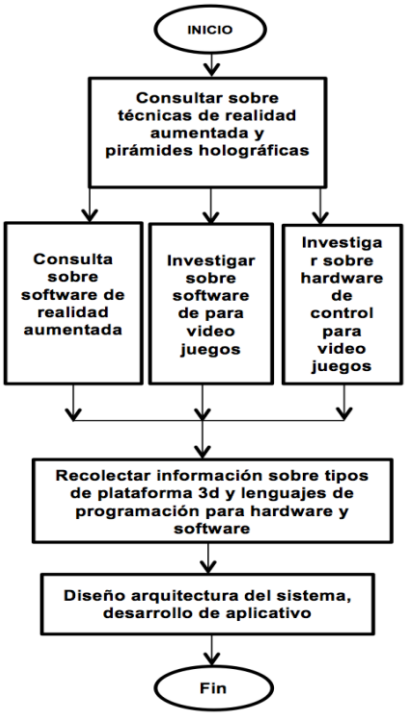



**Figura 2. Mapa de la metodología trazada**

A continuación se presentan las características de los software utilizados a partir de la consulta a varios manuales.

Aprende Unity. La programación con UnityScript es una guía paso a paso para aprender a hacer juegos Unity usando UnityScript. No necesita ningún conocimiento previo de programación o ninguna experiencia con otras herramientas de diseño como PhotoShop o Illustrator. A través de ejemplos prácticos de patrones de juego comunes, se aprende y aplican los fundamentos de la lógica y el diseño del juego. Muchos libros de programación de principiantes se refieren a la documentación que es demasiado técnico abstracto para un principiante de usar. Aprender Unity Programación con UnityScript le enseñará cómo leer y utilizar esos recursos para perfeccionar sus habilidades, y aumentar rápidamente su conocimiento en el desarrollo de juegos Unity. Aprenderá sobre animación, sonido, física, cómo manejar la interacción del usuario y mucho más. Janine Suvak ha ganado premios por su

desarrollo de juegos y está listo para mostrarle cómo comenzar su viaje como desarrollador de juegos. El motor de juego Unity3D es flexible, multiplataforma y es un gran lugar para comenzar su aventura de desarrollo de juegos, y UnityScript fue creado para ello.(Suvak, 2014)

Blender es una herramienta poderosa, estable, con un flujo de trabajo integral que permite entender su aprendizaje de la creación 3D con serenidad. Hoy en día, se considera que es uno de los paquetes 3D más completos del mercado y es gratuito y de código abierto. Es muy eficiente para muchos tipos de producciones, como películas animadas o en vivo en 3D, arquitectura, investigación o incluso creación de juegos con su motor de juego integrado y su uso del lenguaje Python.

A través de Blender, se podrán crear muchos tipos de proyectos usando un enfoque paso a paso. Se comienza por conocer las herramientas de modelado disponibles en Blender a medida que se crea un juguete robot 3D. Entonces, se descubrirán técnicas más avanzadas tales como escultura y re-topología creando un carácter extranjero. (Caudron, Nicq y Valenza, 2016)

Las aplicaciones de la electrónica, presentes actualmente en innumerables aspectos de la vida cotidiana, no serían posible sin los sensores, sin la capacidad que estos ofrecen de medir magnitudes físicas para su conocimiento o control. La utilización de sensores es indispensable en la automatización de industrias de procesos manufactureros, incluida la robótica y en la ingeniería experimental. (Pallás, 2003)

Unity Technologies ha dado un gran paso, audaz. No solo han entregado en algunas grandes promesas de un sistema de interfaz de usuario nuevo y mejorado para los proyectos de Unity, sino que también han hecho la fuente de la nueva interfaz de usuario totalmente de código abierto, dando a los desarrolladores de todos los días el acceso interno a la nueva interfaz de usuario. (Jackson, 2015)

Blender 3D Incredible Machines, le ayudará a desarrollar un conjunto completo de habilidades que cubre los aspectos claves del modelado mecánico. A través de este módulo, usted creará muchos tipos de proyectos, incluyendo una pistola, una nave espacial, un robot y un corredor. Al final de este módulo, habrá dominado un flujo de trabajo que podrá aplicar a sus propias creaciones. (Caudron, Nicq y Valenza, 2016)

Al investigar, descubrieron que algunos usuarios de Blender intentan aprender Blender tres veces y renunciar dos veces antes de que se sientan cómodos con la interfaz efectiva de este, si es inusual. Los editores de Packt y el autor decidieron que este era un problema que podría resolverse. La respuesta es explicar los fundamentos en profundidad, darle práctica para que sus manos puedan aprender Blender como lo hace su mente y luego construir sobre lo que ha aprendido. Esto no es solo un libro de referencia tema por tema. Es un libro para dar experiencia, este libro comenzará con una introducción a Blender y algunos antecedentes sobre los principios de animación, cómo se aplican a la animación por computadora, y cómo estos principios hacen que la animación sea mejor. (Fisher, 2014)

Hoy somos testigos del florecimiento de la realidad virtual (VR), una nueva y excitante tecnología que promete transformar de manera fundamental cómo interactuamos con nuestra información, con los amigos y con el mundo en general.

¿Qué es la realidad virtual del consumidor? Al usar una pantalla montada en la cabeza (como gafas), puede ver escenas 3D estereoscópicas. Usted puede mirar alrededor moviendo su cabeza y caminar alrededor usando controles manuales o sensores de movimiento. Usted puede participar en una experiencia totalmente inmersiva. Es como si estuvieras en otro mundo virtual. Este libro tiene un enfoque práctico, basado en proyectos para enseñarle los detalles del desarrollo de la realidad virtual con el motor de juego Unity 3D. Caminamos a través de una serie de proyectos prácticos, tutoriales paso a paso y discusiones en profundidad usando Unity 5 y otro software libre o de código abierto. Si bien la tecnología VR está avanzando rápidamente, trataremos de captar los principios y técnicas básicas que puede utilizar para hacer que sus juegos y aplicaciones VR sean inmersivos y cómodos. (Linowes, 2015)

## 3. Resultados y discusión

### 3.1 Desarrollo de software

El desarrollo del software fue en Unity, donde se utilizó el lenguaje de scripting C#, se hizo el script de recepción de datos procedentes del hardware donde se estableció la recepción de los ángulos roll, pitch y yaw.

Lo primero que establecimos fue el canal de comunicación físico, esto consistió en configurar el computador para recibir por el bluetooth y convertir este canal en un puerto virtual serial, en nuestro caso fue sencillo porque el sistema operativo de los I Mac trae esto por defecto, se reciben los tres ángulos en un string con un final de línea, esto nos permite hacer

la recepción de la cadena completa y hacer un Split utilizando como parámetro un espacio simple ‘ ‘ que nos almacenará los tres ángulos en variables separadas, aún como caracteres, se creó una línea que convertía estos caracteres a números flotantes y los almacena en un vector tripleta llamado rot, este vector lo tomamos y lo aplicamos como una transformación de rotación del objeto al que está asociado el script, finalmente limpiamos el buffer del serial para asegurarnos que llegue nueva.

**Figura 3. Prueba de protocolo de comunicación por bluetooth entre el mando embebido y el software 3d.**

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

public class serialtivac : MonoBehaviour {

     SerialPort stream = new SerialPort("COM8", 9600);
     float[] lastRot = {0,0,0};
     Vector3 rot;
     Vector3 offset;


     void Start () {
          stream.Open();
     }

     void Update () {
          string value = stream.ReadLine();
          string[] vec3 = value.Split(' ');
          if(vec3[0] != "" && vec3[1] != "" && vec3[2] != "")
          {
               rot = new Vector3(float.Parse(vec3[0]),
              float.Parse(vec3[1]),float.Parse(vec3[2]));
               transform.rotation = Quaternion.Slerp(transform.rotation,
              Quaternion.Euler(0,rot.x,rot.y), Time.deltaTime*3);
               stream.BaseStream.Flush();
          }
     }

     void OnGUI()
     {
          string newString = "Connected: " + transform.eulerAngles;
          GUI.Label(new Rect(10,10,300,100), newString);
          GUI.Label(new Rect(10,30,300,100), "\t" + rot);
     }
}
```

**Figura 4. Script para establecer comunicación y hacer la rotación del objeto asociado**

### 3.2 Creación de pirámide holográfica

La pirámide las hicimos con ángulos de inclinación con respecto a la horizontal mayor 45º y menor a 54º, con cristal cromado y normal, se encontró que lo óptimo es un ángulo de 45º con respecto a la horizontal en cristal normal. Las figuras 5 y 6 muestran las diferencias.

**Figura 5. Cristal cromado a 54º con video de ángel creado para hacer pruebas.**

**Figura 6. Cristal normal a 45º con Video de ángel creado para hacer pruebas.**

### 3.3 Escenografía

Lo primero fue situar cuatro cámaras con resolución de 640x640 pixeles, a la misma altura, distanciadas simétricamente entre sí, enfocando un mismo punto, con un mismo ángulo respecto a la horizontal del espacio 3d en plataforma, luego se seleccionó una pantalla de juego de 1080x1080 pixeles, las cuatro cámaras son escaladas a un noveno de la pantalla de juego, se ubican haciendo una cruz con espacio vacío en el centro igual a una cámara y ubicado en el centro de la pantalla de juego, la cámara derecha se sitúa frente a la izquierda y con sus partes superiores orientadas al espacio vacío, igualmente la frontal frente a la trasera y con una rotación en la cual la parte superior de la cámara quede orientada al centro vacío.

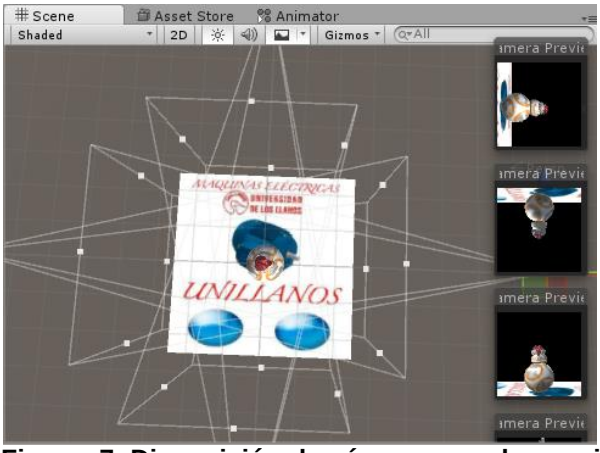



**Figura 7. Disposición de cámaras en el espacio 3d.**

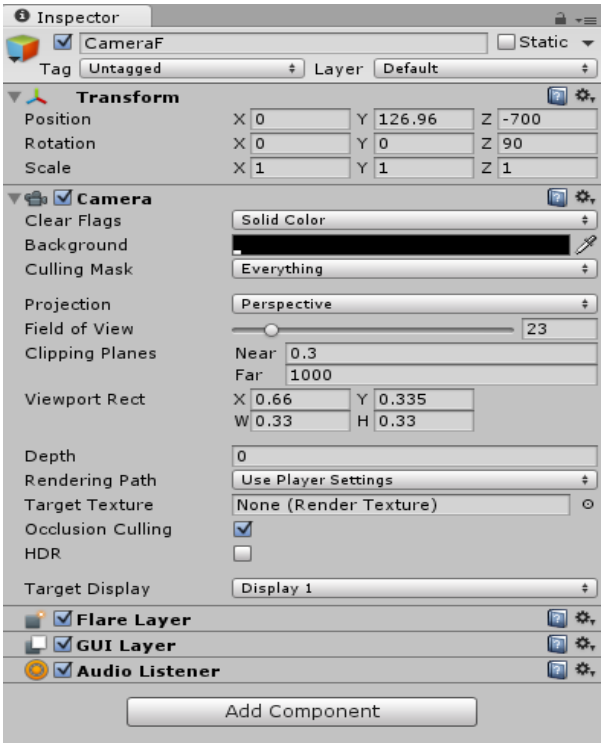



**Figura 8. Barra de herramientas para ubicación y escala de cada cámara**

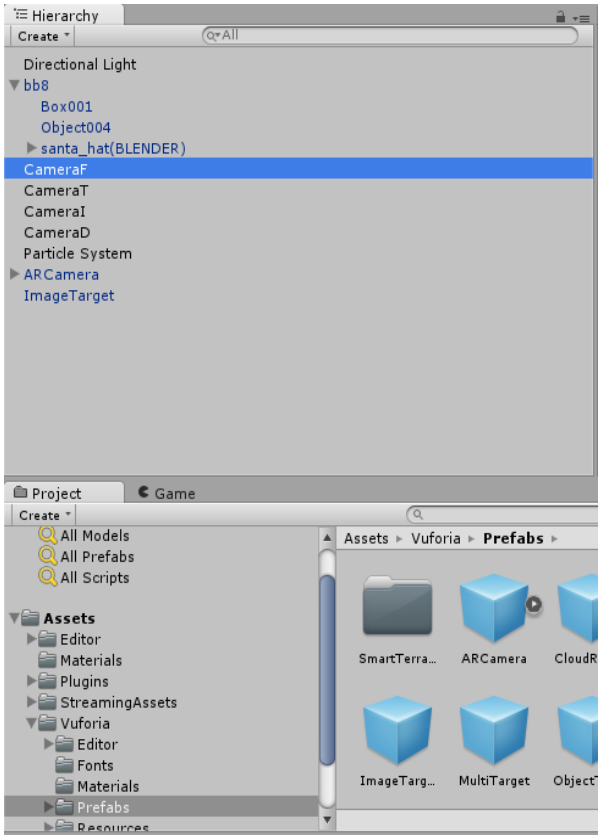



**Figura 9. Jerarquía de los objetos en la escena.**

Para proyectar sobre pirámide se utiliza un video proyector para lograr mayor área de visualización en las caras de la pirámide, luego se emparentan las cámaras a nuestro avatar.

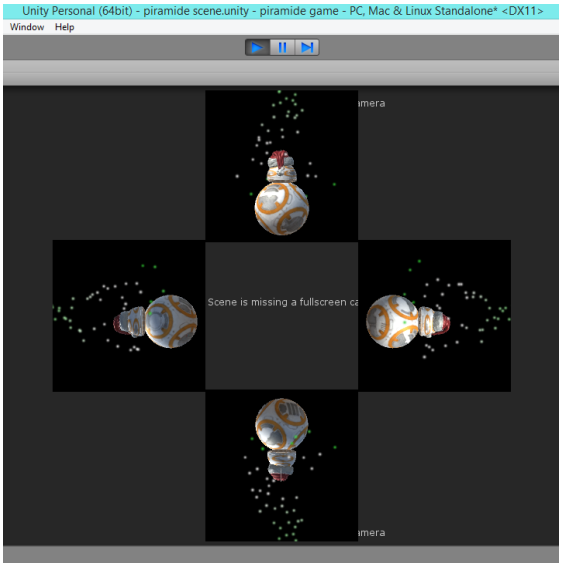



**Figura 10. Visualización de la interfaz del video juego funcionando con las cámaras dispuestas para proyectar en las cuatro caras de la pirámide**

Se colocó una quinta cámara que se ubicó justo encima del punto enfocado por las otras cuatro cámaras y con un ángulo de noventa grados con respecto a la horizontal, esta cámara debe ser de la API de Vuforia para realidad aumentada para Unity, se configura la cámara para los paquetes demarcadores necesarios en el juego, previamente configurados y descargados en la página de Vuforia.

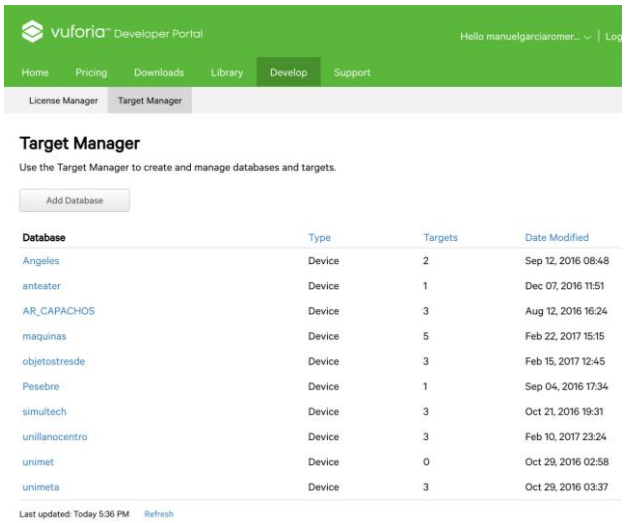



**Figura 11. Portal de Vuforia para desarrolladores.**

Se colocó en el espacio 3d y se configuró un módulo de la API Vuforia de la carpeta de prefab llamado Image Target donde se configura la imagen que activará la realidad aumentada. Este Image Target se ubicó espacialmente debajo de avatar; el avatar se coloca dentro de la Jerarquía del Image Target.

A la cámara de Vuforia se le dio una licencia que proporciona Vuforia desde su portal web para los desarrolladores, esta licencia no es válida para uso comercial. Además se configuró para que cargue la imagen creada anteriormente y para que cuando detecte el Image Target no muestre la imagen de la cámara real si no en cambio muestre el escenario del video juego, como se muestra en la Figura 12.

### 3.4 Modelamiento

Para el modelamiento de los personajes se utiliza una multiplataforma 3d de código abierto llamada Blender, que se puede descargar de forma gratuita desde su portal web http://www.blender.org.

Para crear el cuerpo del avatar se crearon sus partes por separado y luego se ensamblaron en un solo sólido, las partes deben ser creadas de acuerdo con las articulaciones necesarias para un movimiento natural en el momento de la animación. Primero se crea un nuevo proyecto con el motor BlenderRender, se elimina el cubo oprimiendo X y enter, se crea una esfera, se pasa al modo edición de malla pulsando la tecla tabulador, se selecciona todo el objeto oprimiendo la letra A, en el panel de herramientas situado en la parte inferior de la ventana 3d debe estar seleccionado el modo de caras, luego haciendo uso del mouse para el desplazamiento en el espacio 3d de Blender y las acciones de rotación, escalar, movimiento y del módulo de escultura de malla, se da forma a la parte, cuando está lista esa parte se vuelve al modo objeto pulsando de nuevo la tecla tabulador, se repite este proceso para el resto de partes del avatar.

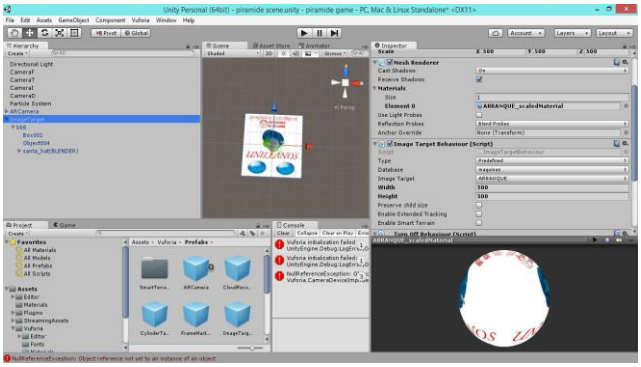


**Figura 12. Configuración cámara de realidad aumentada de Vuforia.**

Creadas las piezas se ubican en el espacio en 3d de forma que construyan el cuerpo completo del avatar, teniendo en cuenta que las partes del cuerpo se crearon como objetos independientes en el espacio y que se desea obtener un solo cuerpo, se selecciona la parte más grande y luego las otras partes con la tecla Shift oprimida y clic derecho, se suelta la tecla Shift para oprimir luego Control+J. Se suavizan las caras de la malla con la opción de suavizado en la paleta de herramientas que se despliega oprimiendo la tecla T con el mouse encima del volumen. Luego se exportó el objeto como un archivo .fbx

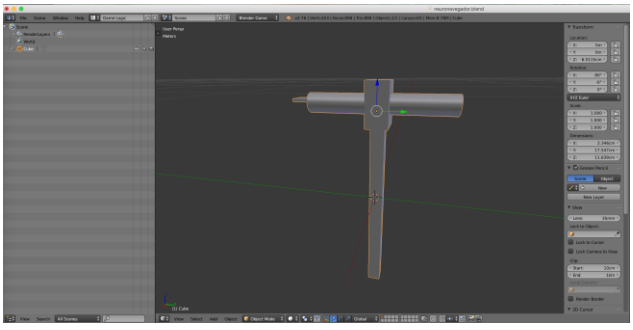


**Figura 13. Creación de primer avatar con la forma del mando embebido para probar el sistema de posicionamiento.**

### 3.5 Desarrollo de hardware

El desarrollo del hardware consistió en tomar los datos procedentes del giroscopio, acelerómetro y magnetómetro minimu-9 5v de Pololu, con un microcontrol adorarduino nano, lo primero fue descargar el repositorio en github MinIMU9AHRS que como su nombre lo indica es un mini imu que entrega 9 variables y crea un AHRS por sus siglas en inglés Attitude and Heading Reference Systems, en español: "Sistema de referencia de actitud y rumbo", que entrega un sistema de orientación de un cuerpo en el espacio en tres variables roll, pitch y yaw, las cuales son enviadas al ordenador por un protocolo de comunicación serial, el protocolo de comunicación físico se hizo mediante un módulo bluetooth HC-05. Se estableció que la forma de enviar los datos sería un string con finalización de línea "\n" donde los ángulos se organizaron para el envío así: primero Roll luego el Pitch y finalmente el Yaw, separados por un espacio simple.

MinIMU9AHRS | Compass | DCM | I2C | Output § | Vector | matrix

```
int ro,pi,ya;
void printdata(void)
{

      #if PRINT_EULER == 1
      ro = ToDeg(roll);
      pi = ToDeg(pitch);
      ya = ToDeg(yaw);
      snprintf(report, sizeof(report),
      "%d,%d,%d",ToDeg(roll),ToDeg(pitch),ToDeg(yaw));
      miserial.println(report);
      Serial.print(report);
      #endif
      #if PRINT_ANALOGS==1
      #endif
      #if PRINT_DCM == 1
      #endif

}

long convert_to_dec(float x)
{
  return x*10000000;
}
```

**Figura 14. Script para microcontrolador, función de envío printdata(void)**

Una vez desarrollado el sistema de comunicación con el motor 3d, y establecidos los parámetros de dicha comunicación se procede a aplicar este sistema al desarrollo de un videojuego que aprovechara al máximo este sistema de comunicación, junto con el sistema de visualización. Lo primero que se pensó fue en mejorar fue el control remoto, haciéndolo inalámbrico por conexión bluetooth al ordenador, además de esto se cambia el giroscopio por una palanca resistiva de dos ejes que servirán para el desplazamiento del avatar en el terreno, también se añaden dos pulsadores que servirán para salto y defensa, este sistema consta del mismo principio mencionado anteriormente que se basa en la comunicación serial de una cadena de caracteres separados por coma concatenados en un orden especifico. Este nuevo control es contenido en una carcasa desarrollada a medida en la multiplataforma 3d Blender como se muestra en las figuras 16, 17, 18 y 19. Donde se puede apreciar las partes de la carcasa y su posterior impresión en ABS que es un plástico muy común en electrodomésticos y juguetes.

```
  Update_Matrix[0][1]=-G_Dt*Gyro_Vector[2];//-z
  Update_Matrix[0][2]=G_Dt*Gyro_Vector[1];//y
  Update_Matrix[1][0]=G_Dt*Gyro_Vector[2];//z
  Update_Matrix[1][1]=0;
  Update_Matrix[1][2]=-G_Dt*Gyro_Vector[0];
  Update_Matrix[2][0]=-G_Dt*Gyro_Vector[1];
  Update_Matrix[2][1]=G_Dt*Gyro_Vector[0];
  Update_Matrix[2][2]=0;
 #endif

  Matrix_Multiply(DCM_Matrix,Update_Matrix,Temporary_Matrix); //a*b=c

  for(int x=0; x<3; x++) //Matrix Addition (update)
  {
    for(int y=0; y<3; y++)
    {
      DCM_Matrix[x][y]+=Temporary_Matrix[x][y];
    }
  }
}

void Euler_angles(void)
{
  pitch = -asin(DCM_Matrix[2][0]);
  roll = atan2(DCM_Matrix[2][1],DCM_Matrix[2][2]);
  yaw = atan2(DCM_Matrix[1][0],DCM_Matrix[0][0]);
}
```

**Figura 15. Script para hallar el pitch, roll y yaw de una matriz de estados.**

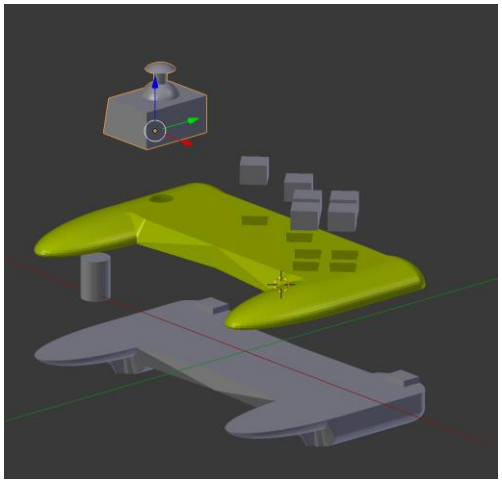


**Figura 16. Mando explosionado, cada agujero fue hecho a medida y probado con piezas hechas en 3d con las medidas de los dispositivos reales que se usaran posteriormente en el ensamblaje.**

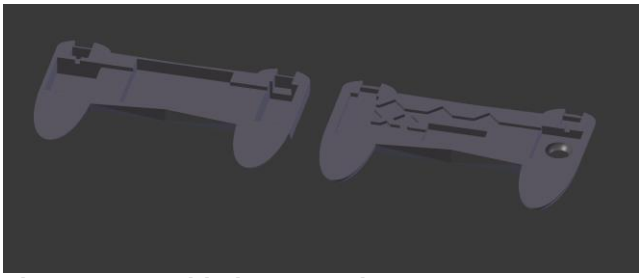


**Figura 17. Cavidades para elementos y rutas para cableado.**

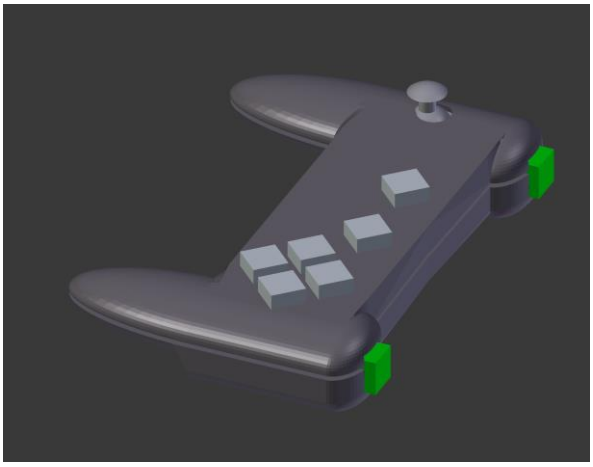


**Figura 18. Ensamblaje virtual de piezas.**

**Figura 19. Mando impreso con dispositivos reales.**

Una vez creado el mando que puede ser usado para el desarrollo de un videojuego tipo consola o de computadora, se consideró crear un videojuego en tercera persona desestimando el uso de la realidad aumentada, y en el cual se consideraron las siguientes etapas para su posterior desarrollo:

- Game-Art.
- Creación del Terreno 3D.
- Creación de personajes 3D.
- Creación de animaciones.
- Los personajes (Zombies).
- El jugador (Player).
- Configuración de las animaciones para cada personaje.
- *Scripts actions*.

Siendo el *Game art* la parte del desarrollo de un videojuego donde se plantean a groso modo las ideas que deben quedar plasmadas en los aspectos más importantes del videojuego como los las características de los personajes, de los escenarios, los roles de los actores, la interfaz grafica, el tipo de modelamiento entre otras muchas, se plantean los siguientes parámetros puntuales para el juego:

Idea Principal: Juego de supervivencia de un niño en un apocalipsis zombie, donde el personaje principal "Alejo" un niño adolecente delgado y temeroso que se encuentra perdido en un bosque con poca visibilidad infestado de zombies violentos, zombies que en su vida humana pudieron ser hombre y mejeres de la vida civil como un campesino o una siple mujer o tal vez un militar. Estos zombies perciben a proximidad a sus victimas perdidas en el bosque como es el caso de Alejo, quien se encuentra en busca de su hermana mayor perdida en dicho bosque.

Para la creación del terreno se tratan de seguir los parámetros especificado en el *game art*, para lo cual se crea un terreno con la herramienta *terrain* nativa de Unity donde se permite la creación rápida de árboles y la creación de accidente geográficos de una forma fácil y sencilla, como se muestra en las figuras 20 y 21.

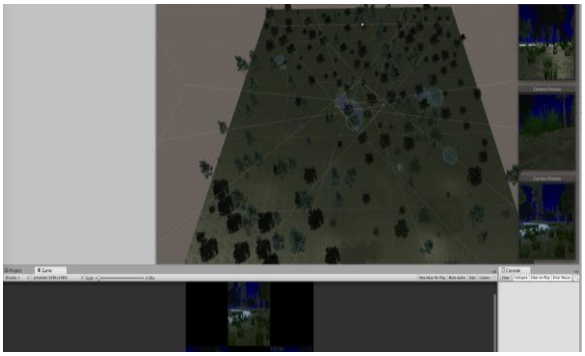


**Figura 20. Vista superior del terreno donde se denota su extensión.**

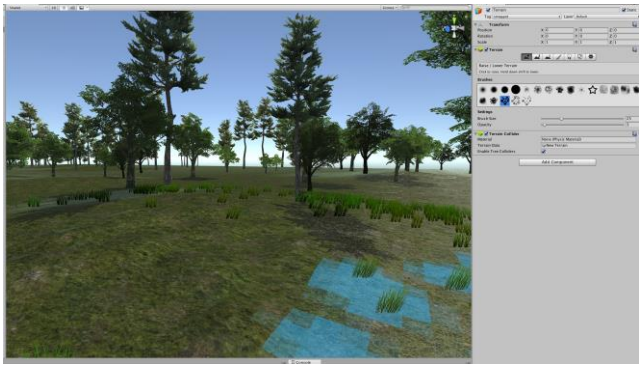
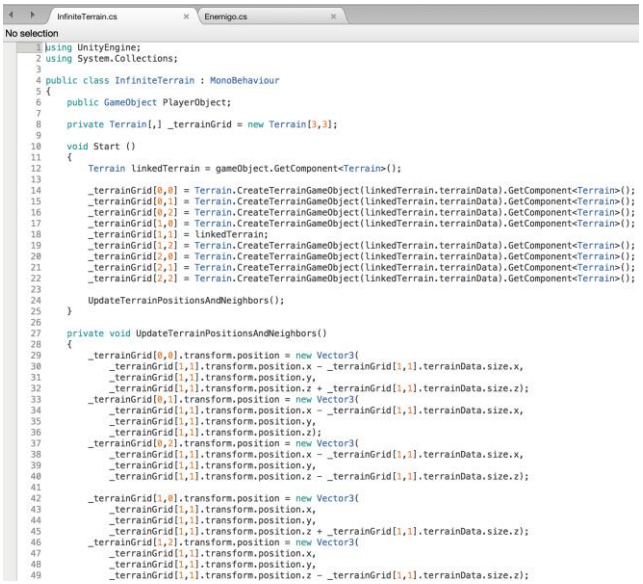


**Figura 21. Vista de cámara en posición horizontal.**

Como se puede apreciar en la figura 20 su extensión es limitada, lo que ocasionaría que el personaje al llegar al extremo del terreno pueda caer y esto seria un error en el desarrollo de un video juego, ya que un personaje no debe caer ni salir en ningún momento del escenario que lo confina, es por eso que se hace uso de un script de regeneración de terreno que utiliza la posición del personaje principal para regenerar el terreno a medida que él avanza en una dirección, es script repite el terreno originalmente creado con las mismas características incluidos los personajes secundarios a él asociado.



```
using UnityEngine;
using System.Collections;

public class InfiniteTerrain : MonoBehaviour
{
    public GameObject PlayerObject;

    private Terrain[,] _terrainGrid = new Terrain[3,3];

    void Start ()
    {
        Terrain linkedTerrain = gameObject.GetComponent<Terrain>();

        _terrainGrid[0,0] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();
        _terrainGrid[0,1] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();
        _terrainGrid[0,2] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();
        _terrainGrid[1,0] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();
        _terrainGrid[1,1] = linkedTerrain;
        _terrainGrid[1,2] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();
        _terrainGrid[2,0] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();
        _terrainGrid[2,1] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();
        _terrainGrid[2,2] = Terrain.CreateTerrainGameObject(linkedTerrain.terrainData).GetComponent<Terrain>();

        UpdateTerrainPositionsAndNeighbors();
    }

    private void UpdateTerrainPositionsAndNeighbors()
    {
        _terrainGrid[0,0].transform.position = new Vector3(
            _terrainGrid[1,1].transform.position.x - _terrainGrid[1,1].terrainData.size.x,
            _terrainGrid[1,1].transform.position.y,
            _terrainGrid[1,1].transform.position.z + _terrainGrid[1,1].terrainData.size.z);
        _terrainGrid[0,1].transform.position = new Vector3(
            _terrainGrid[1,1].transform.position.x - _terrainGrid[1,1].terrainData.size.x,
            _terrainGrid[1,1].transform.position.y,
            _terrainGrid[1,1].transform.position.z);
        _terrainGrid[0,2].transform.position = new Vector3(
            _terrainGrid[1,1].transform.position.x - _terrainGrid[1,1].terrainData.size.x,
            _terrainGrid[1,1].transform.position.y,
            _terrainGrid[1,1].transform.position.z - _terrainGrid[1,1].terrainData.size.z);

        _terrainGrid[1,0].transform.position = new Vector3(
            _terrainGrid[1,1].transform.position.x,
            _terrainGrid[1,1].transform.position.y,
            _terrainGrid[1,1].transform.position.z + _terrainGrid[1,1].terrainData.size.z);
        _terrainGrid[1,2].transform.position = new Vector3(
            _terrainGrid[1,1].transform.position.x,
            _terrainGrid[1,1].transform.position.y,
            _terrainGrid[1,1].transform.position.z - _terrainGrid[1,1].terrainData.size.z);
```

**Figura 22. Script utilizado para la regeneración de terreno.**

Ya creado el terreno se procede con la creación de los personajes, para este caso se hace uso de un programa de Adobe llamado Fuse, que se encuentra en su versión Beta, y cuya descarga y uso es gratuito, este programa permite la creación de personajes de una forma variada, detallada y rápida.

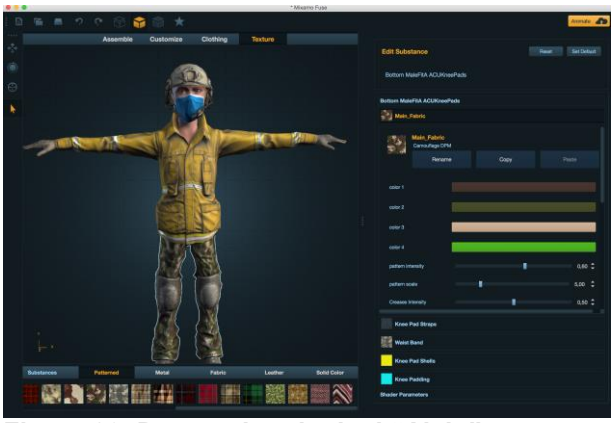


**Figura 23. Personaje principal "Alejo"**

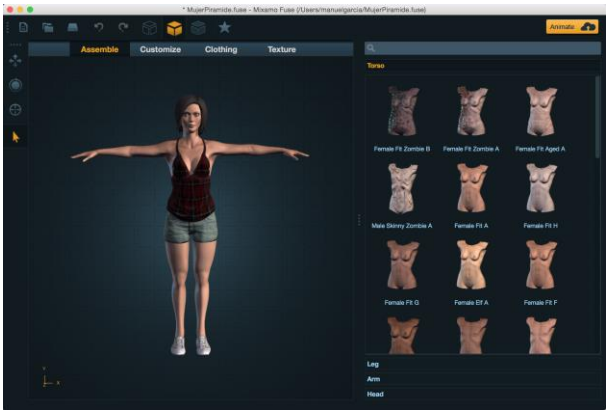



**Figura 24. Hermana de Alejo**

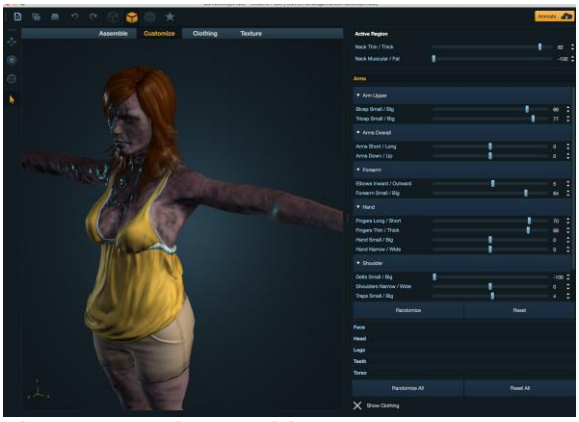



**Figura 25. Mujer Zombie No 1.**

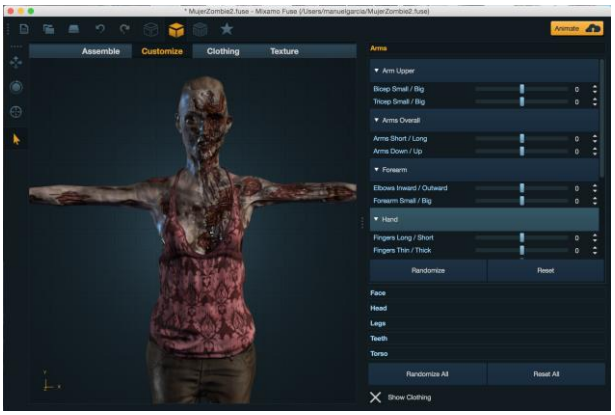



**Figura 26. Mujer Zombie No 2.**

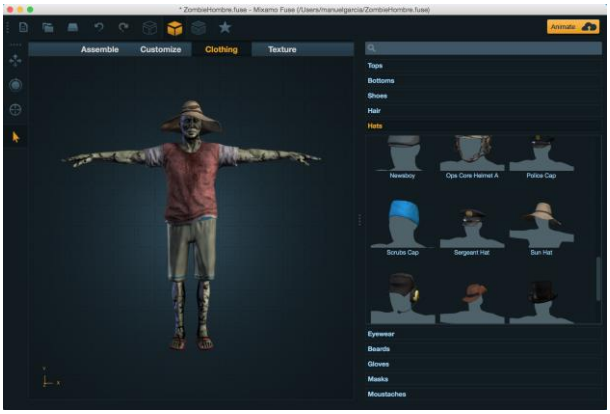



**Figura 27. Hombre de campo Zombie N. 1.**

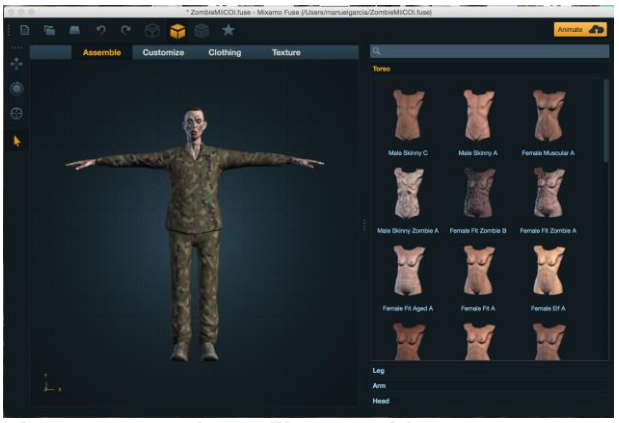



**Figura 28. Hombre militar Zombie.**

Fuse de Adobe viene complementado con una plataforma web para la animación de los personajes creados con Fuse llamada Mixamo con la URL: **https://www.mixamo.com/.** La cual cuenta con un paquete de animaciones prefabricadas adaptables a los modelos exportados por la herramienta de modelado, se pueden empaquetar varias animaciones para un solo personaje, esta plataforma permite además la asociación de los huesos que permiten el movimiento en los motores 3d, finalmente los exporta en archivos ejecutables por las diferentes plataformas para creación de videojuegos como lo son Unity, Blender y Unreal. El proceso es rápido y sencillo pues solo se deben escoger las animaciones y asociárselas al modelo para luego poderlo descargar en el tipo de archivo necesitado en el desarrollo. Las imágenes del uso de la plataforma se pueden ver en las figuras 29, 30, 31, y 32.

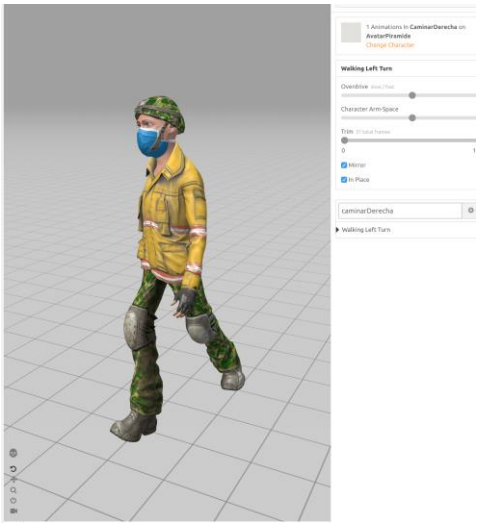


**Figura 29. Animando a Alejo.**

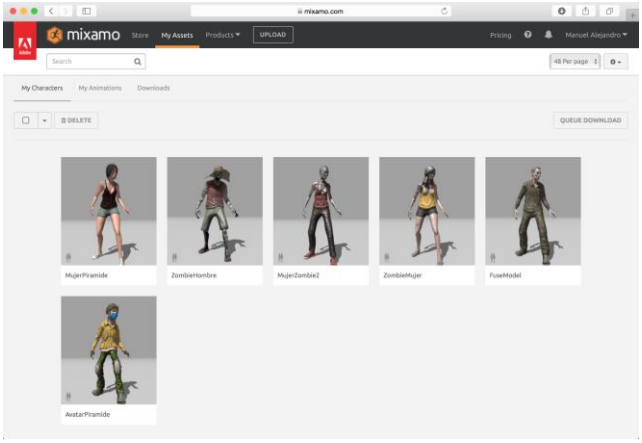


**Figura 30. Personajes del videojuego en la plataforma Mixamo.**

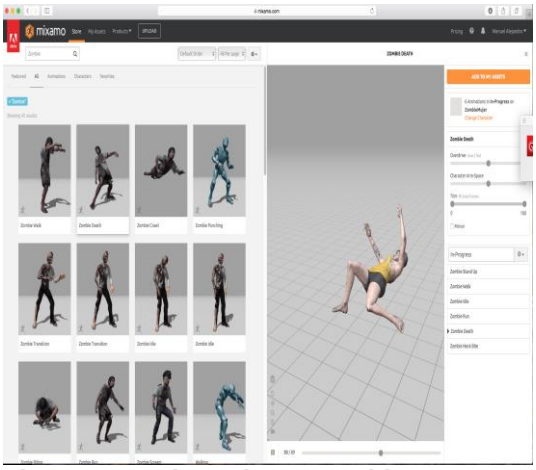


**Figura 31. Animando un zombie.**

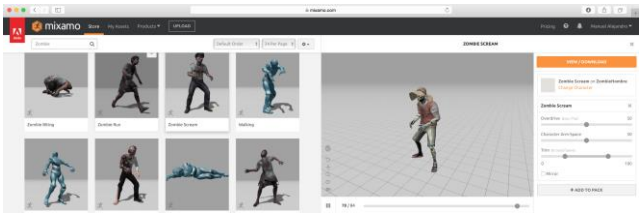


**Figura 32. Zombie y animaciones.**

Los personajes después de descargados son importados a Unity para hacer el árbol de animaciones que serán activadas a través de los scripts. Como se muestra en las figuras 33 y 34.

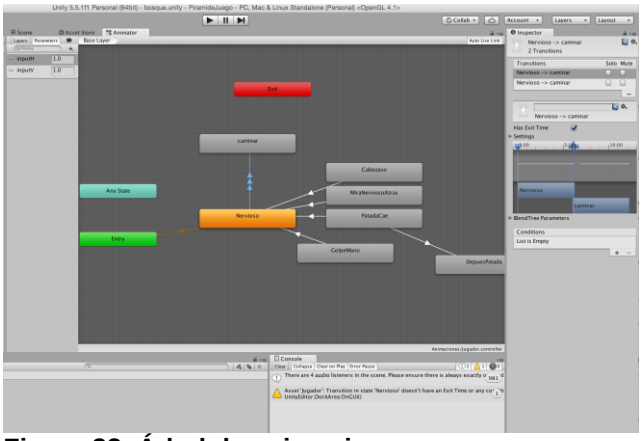


**Figura 33. Árbol de animaciones.**

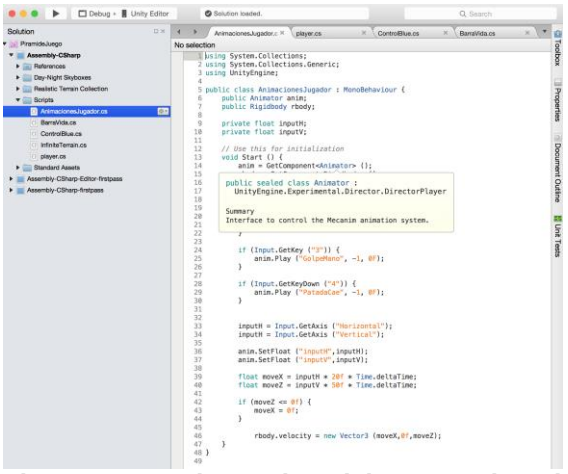


**Figura 34. *Script action* del personaje principal donde se activan las animaciones del árbol según la información recibida por el mando.**

En el siguiente enlace se puede encontrar un video en youtube del video juego funcionando con su respectivo control inalámbrico:
**https://goo.gl/OD0vS3**

## 4. Conclusiones

El vidrio polarizado o cromado no refracta la suficiente luz para ver claramente el escenario ni el personaje, mientras el vidrio normal presenta un mejor índice de refracción.

Un proyector o Video Beam presenta una mejor proyección en el cristal que un televisor led pantalla plana.

Vuforia es una herramienta muy completa para la creación de contenido con realidad aumentada, aunque sus costos de licencias para producciones comerciales son muy costosos en comparación con otras herramientas de código abierto que permiten la creación del mismo contenido de forma gratuita como ARtoolKit.

Una vez establecido el protocolo de comunicación entre el periférico, la ubicación de las cámaras, y la realidad aumentada, se deja un sistema flexible para creación de nuevos contenidos con nuevos requerimientos ya sea para el hardware o el software.

## 5. Referencias bibliográficas

Gordon Fisher (2014). *BLENDER 3D Basics Beginner's Guide* (2a ed). Packt Publishing Ltd. Ltd. 53 Livery Street Birmingham B3 2PB UK, ISBN: 978-1-78398-490-9.

Jonathan Linowes (2015). *Unity-Virtual-Reality-Projects*. Packt Publishing Ltd. 53 Livery Street Birmingham B3 2PB UK, ISBN:978-1-78398-855-6.

Janine Suvak (2014). *Learn Unity3D. Programming with UnityScript: Unity´s JavaScript for Beginners* (1ª ed.) ISBN-13 (pbk): 978-1-4302-6586-3, ISBN (electronic): 978-1-143026587-0, Springer Science+Business Media New York, fax (201) 348-4505.

Pallás, R. (2003). *Sensores y acondicionadores de señal*. Barcelona: Marcombo Boixareu Editores.

Romain Caudron, Pierre-Armand, Nicq y Enrico Valenza (2016). Blender 3D: Designing Objects Table of Contents (1ª ed.), Packt Publishing Ltd. 53 Livery Street Birmingham B3 2PB UK, ISBN: 978-1-78712-719-7.

Ramón Pallás Areny (2005). Sensores y acondicionadores de Señal (4ª ed.), MARCOMBO S.A. Barcelona, (+34) 95 458 3425 (Sevilla), ISBN: 84-267-1344-0.

Romain Caudron, Pierre-Armand Nicq, Enrico Valenza (2016). *Blender 3D: Characters, Machines , and Scenes for Artists Table of Contents* (1ª ed.), Packt Publishing Ltd. 53 Livery Street Birmingham B3 2PB UK, ISBN: 978-1-78355-361-7.

Simon Jackson, (2015). Unity 3D UI Essentials (1ª ed.), Packt Publishing Ltd. 53 Livery Street Birmingham B3 2PB UK, ISBN: 978-1-78355-361-7.

## 6. Reconocimientos

Los autores desean agradecer a la Universidad de los Llanos.

**Fecha de recepción**: 22 de abril de 2017

**Fecha de aceptación**: 15 de mayo de 2017