Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Videojuego para la evaluación del movimiento

Video game for movement assessment

Sandro Jesús Socas Méndez

La Laguna, 14 de julio de 2023

D. **Jonay Tomás Toledo Carrillo**, con N.I.F. 78.698.554-Y profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Bibiana Fariña Jerónimo**, con N.I.F. 54.063.772-H profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Videojuego para la evaluación del movimiento”

ha sido realizada bajo su dirección por D. **Sandro Jesús Socas Méndez**,

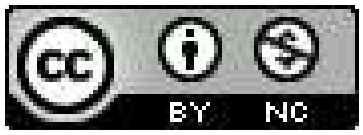
con N.I.F. 78.649.717-K.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 14 de julio de 2023

Agradecimientos

Quiero agradecer en primer lugar, a todos mis familiares y mi pareja por haberme apoyado y guiado durante todos estos años, por haber confiado tanto en mí y estar presentes en los buenos y malos momentos. También me gustaría agradecer a todos aquellas personas que he conocido a lo largo de mi paso por la universidad, que hicieron de mi estancia en la misma una de las mejores experiencias de mi vida y que se han convertido en amigos de confianza. Y, finalmente a mi tutor Jonay por confiar en mí y guiarme durante el desarrollo de este proyecto.

Licencia



Resumen

*Este proyecto tiene como objetivo principal la creación de un **videojuego** en **Unity** que permita evaluar el movimiento de **niños/as con movilidad reducida** mediante una serie de ejercicios físicos que serán presentados en pantalla.*

*El videojuego estará diseñado para que el niño/a pueda repetir los ejercicios físicos propuestos y un dispositivo **Microsoft Kinect** se encargará de almacenar la información del movimiento real del mismo. Además, el sistema proporcionará una realimentación en tiempo real, invitándolo a mejorar su movilidad según el cumplimiento de los objetivos propuestos.*

*La importancia de este proyecto radica en la necesidad de proporcionar a este colectivo herramientas lúdicas y motivadoras para mejorar su movilidad y calidad de vida. El desarrollo de un videojuego que permita la evaluación y mejora del movimiento en estos niños/as puede ser una alternativa eficaz y atractiva para el cumplimiento de los **objetivos terapéuticos**.*

Palabras clave: Videojuego, Unity, Evaluar el movimiento, Niños/as con movilidad reducida, Microsoft Kinect, Objetivos terapéuticos.

Abstract

*This project aims to create a **Unity video game** that allows the **evaluation of the movement** of **children with reduced mobility** through a series of physical exercises presented on the screen.*

*The video game will be designed so that the child can repeat the proposed physical exercises, and a **Microsoft Kinect** device will be responsible for storing the real movement information of the child. Additionally, the system will provide real-time feedback to the child, encouraging them to improve their mobility based on the achievement of the proposed objectives.*

*The importance of this project lies in the need to provide children with reduced mobility with playful and motivating tools to improve their mobility and quality of life. Developing a video game that allows the evaluation and improvement of movement in these children can be an effective and appealing alternative to achieve **therapeutic objectives**.*

Keywords: Unity, Video game, Evaluation of the movement, Children with reduced mobility, Microsoft Kinect, Therapeutic objectives.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Definición del problema

La **discapacidad motriz** [1], también conocida como movilidad reducida, es una condición que afecta al funcionamiento del sistema locomotor de una persona. Esta condición provoca una disminución parcial o total de la movilidad en uno o más miembros del cuerpo, lo cual dificulta la realización de actividades físicas.

Existen diversas causas que pueden originar movilidad reducida, entre ellas se encuentran las causas infecciosas, virales, reumáticas, neurológicas, musculares y los traumatismos. Estas causas pueden afectar al correcto desarrollo y funcionamiento de los músculos, huesos, articulaciones o incluso el sistema nervioso, resultando comprometida la salud de los individuos afectados.

En España, se calcula que existen aproximadamente más de 4,3 millones de personas con algún tipo de discapacidad, según los datos del **Instituto Nacional de Estadística (INE)** en la encuesta 'Discapacidad, Autonomía Personal y situaciones de Dependencia'[2] correspondientes al año 2020 y publicados en 2022. Según los datos recopilados en dicha encuesta, el grupo más representativo de discapacidad en personas mayores de 6 años, residentes en hogares, es el relacionado con la movilidad, como se puede observar en el gráfico adjunto.

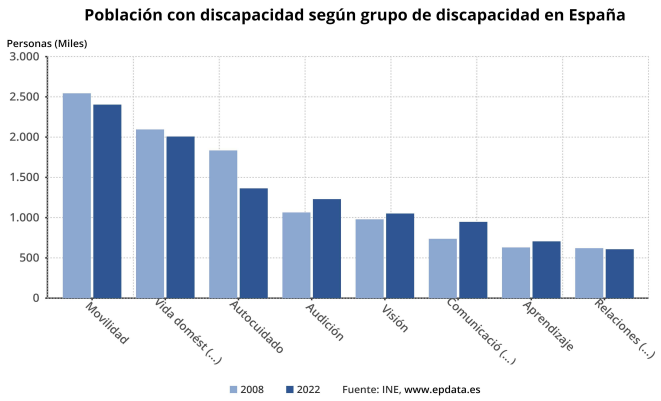


Figura 1.1: Grafico población con discapacidad en España

En el caso específico del grupo de población de 6 a 15 años, que es el colectivo al que se enfoca el desarrollo de este proyecto, la discapacidad motriz se posiciona como una de las cinco discapacidades más frecuentes, como se muestra en la gráfica adjunta.

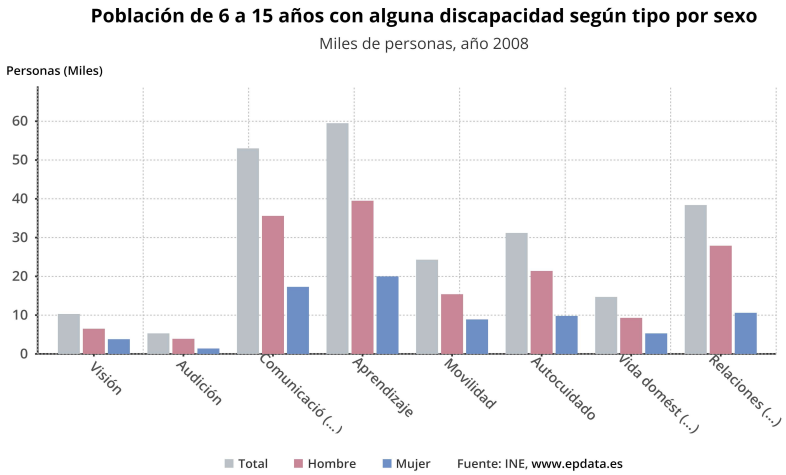


Figura 1.2: Gráfica población de 6 a 15 años con discapacidad según tipo

El problema de la movilidad reducida en niños/as representa un desafío significativo que afecta tanto a nivel individual como a nivel social. La disminución de la movilidad en estos niños/as dificulta su capacidad para realizar actividades motoras convencionales, lo que puede tener un impacto negativo en su calidad de vida y desarrollo personal.

Además, los datos recopilados demuestran que la discapacidad motriz es una de las principales formas de discapacidad en la población de 6 a 15 años. Esto resalta la importancia de abordar de manera efectiva este problema y brindar a estos niños/as las herramientas necesarias para mejorar su movilidad y fomentar su participación activa en la sociedad.

1.2 Justificación

La falta de acceso a intervenciones terapéuticas y recursos adaptados para promover la movilidad en niños/as con discapacidad motriz tiene un impacto significativo en su desarrollo físico, emocional y social. Es fundamental buscar alternativas innovadoras que puedan abordar esta problemática y brindar a estos niños oportunidades de mejora en su movilidad de manera efectiva y motivadora.

En este contexto, el desarrollo de un videojuego que permita evaluar y mejorar el movimiento de niños con movilidad reducida se presenta como una solución prometedora. Este enfoque ofrece a los niños una experiencia lúdica y atractiva, al mismo tiempo que los motiva a trabajar en mejorar su movilidad y les proporciona retroalimentación en tiempo real sobre su desempeño.

Diversos estudios, como "Health Benefits of Digital Videogames for the Aging Population: A Systematic Review" [3] y "Videogames and Health Improvement: A Literature Review of Randomized Controlled Trials" [4], han demostrado el efecto positivo del uso de videojuegos con fines terapéuticos en

el tratamiento de pacientes con diversas patologías. Sin embargo, son escasos los videojuegos desarrollados específicamente para evaluar y mejorar la situación de los niños que sufren de discapacidad motriz o tienen dificultades relacionadas directamente con la movilidad.

Por lo tanto, este proyecto se centra en la creación de un videojuego en Unity que permita evaluar el movimiento de niños con movilidad reducida mediante la repetición de una serie de ejercicios físicos presentados en pantalla. Estos movimientos serán capturados por un dispositivo Microsoft Kinect y analizados para proporcionar una retroalimentación en tiempo real al niño, invitándolo a mejorar su movilidad de acuerdo con los objetivos propuestos.

Este proyecto busca llenar un vacío en el campo de los videojuegos terapéuticos dirigidos a niños con discapacidad motriz, con el objetivo de mejorar su calidad de vida y promover su inclusión activa en la sociedad. Al proporcionar una herramienta lúdica y efectiva para la evaluación y mejora de la movilidad, esperamos contribuir significativamente al bienestar y desarrollo de estos niños/as.

1.3 Objetivos

El objetivo de este proyecto se centra en la utilización de tecnologías innovadoras y en la integración de conocimientos de diferentes disciplinas para lograr la creación de un videojuego útil para la evaluación y mejora de la movilidad en niños con movilidad reducida.

Para lograr este hito, se han establecido una serie de tareas a realizar:

- Realizar una revisión bibliográfica para identificar los estudios más relevantes y actuales sobre el uso de videojuegos y tecnologías innovadoras en la evaluación y mejora del movimiento en niños con movilidad reducida.
- Adquirir el conocimiento necesario para trabajar con el software Unity [5], una herramienta fundamental en el desarrollo de videojuegos.
- Desarrollar software en Unity de generación de movimientos.
- Realizar pruebas con el dispositivo Microsoft Kinect [6] para verificar su compatibilidad y funcionalidad con el software desarrollado en Unity.
- Integrar el dispositivo Microsoft Kinect en el software desarrollado en Unity para la captura del movimiento real del niño/a.
- Medir los resultados obtenidos en la evaluación del movimiento de los niños con discapacidad física utilizando el software desarrollado en Unity y el dispositivo Microsoft Kinect.

Capítulo 2
Estado del arte

Para llevar a cabo un correcto desarrollo del proyecto es muy importante conocer en profundidad las tecnologías que se van a utilizar. Es por ello que esta sección muestra un desglose de las tecnologías empleadas en el desarrollo del proyecto, así como los resultados obtenidos en el periodo de aprendizaje.

2.1 Tecnologías a utilizar

En la actualidad, contamos con diversas opciones altamente poderosas para el desarrollo de videojuegos. Para este proyecto en particular, se plantea utilizar Unity como motor base para el videojuego, y el Microsoft Kinect v1 como dispositivo capturador de movimientos. En cuanto a la funcionalidad, se llevará a cabo mediante scripts desarrollados en el lenguaje de programación C# [7], que serán los encargados de controlar y gestionar las interacciones del videojuego, garantizando una experiencia fluida y personalizada para los usuarios.

2.1.1 Unity

Es una plataforma de desarrollo de videojuegos ampliamente utilizada que ofrece herramientas y recursos para la creación de experiencias interactivas en diversas plataformas, como ordenadores, consolas, dispositivos móviles y realidad virtual. Se destaca por su enfoque en la facilidad de uso y la versatilidad, lo que la convierte en una opción popular tanto para desarrolladores principiantes como para expertos.

Figura 2.1: Logo Unity

Algunas de las características clave de Unity son las siguientes:

- Unity cuenta con un motor de juego potente y flexible que permite crear gráficos de alta calidad, efectos visuales, física realista y sonido

envolvente. Esto proporciona una base sólida para el desarrollo de juegos y experiencias interactivas inmersivas.

- Unity ofrece un editor visual intuitivo que permite a los desarrolladores crear y manipular fácilmente objetos, escenas, animaciones y efectos especiales. Esto facilita el proceso de diseño y prototipado, sin necesidad de tener conocimientos avanzados de programación.
- Unity admite múltiples lenguajes de programación, siendo el más utilizado C#. Proporcionando a los desarrolladores la flexibilidad de elegir el lenguaje con el que se sientan más cómodos.
- Unity cuenta con una amplia biblioteca de assets, que incluye modelos 3D, texturas, sonidos y scripts predefinidos. Estos assets pueden ser utilizados y compartidos en proyectos, acelerando el proceso de desarrollo.

Unity está disponible a través del sitio web oficial y ofrece soporte en varios sistemas operativos, incluyendo Windows, macOS y Linux. A su vez, Unity ofrece diferentes planes para satisfacer las necesidades de los desarrolladores:

- Unity Personal: es una opción ideal para desarrolladores independientes, estudiantes y pequeños estudios. Brinda acceso a la mayoría de las características y funcionalidades básicas de Unity. Los juegos y aplicaciones desarrollados con Unity Personal pueden ser publicados en múltiples plataformas sin tener que pagar regalías.
- Unity Plus: es una alternativa de suscripción mensual. Está dirigido a desarrolladores que desean llevar sus proyectos al siguiente nivel. Unity Plus ofrece beneficios como soporte prioritario, opciones de personalización de la pantalla de inicio y más.
- Unity Pro: es una elección para estudios de desarrollo y empresas con proyectos más complejos. Esta versión es la mas completa y avanzada de Unity. Proporciona todas las características y funcionalidades disponibles, incluyendo herramientas y servicios adicionales para el desarrollo en equipos de trabajo más grandes.
- Unity Enterprise: está dirigido a empresas y organizaciones con necesidades específicas en el desarrollo de juegos y experiencias interactivas. Brinda una solución personalizada y escalable para abordar los proyectos más complejos y demandantes. Este plan ofrece una variedad de beneficios adicionales, como soporte especializado, herramientas de colaboración avanzadas y opciones de licencia flexibles.

LEGEND Included Additional costs Not included	PERSONAL Start creating with the free version of Unity Free Get started	PLUS More functionality and resources to power your projects from €369 /yr Choose plan	PRO Complete solution for professionals to create and operate from €1,877 /yr Choose plan	ENTERPRISE Manage and optimize complex projects for teams of any size Contact us
Create				
Unity real-time development platform	✓	✓	✓	✓
Unity Visual Scripting	✓	✓	✓	✓
Unity Version Control (3 users and 5GB storage)	✓	✓	✓	✓
Splash screen customization		✓	✓	✓
Deploy to game consoles			✓	✓
Unity Mars authoring tools for AR/MR			✓	✓
Havok Physics for Unity			✓	✓
Extended LTS support for 3 years				✓
Build Server license capacity			⊕	✓
Source Code Access				✓
Source Code Adapt				⊕

Figura 2.2: Comparativa entre planes que Unity ofrece (1)

	PERSONAL	PLUS	PRO	ENTERPRISE
Crash and error reporting				
Cloud Diagnostics (Personal)	✓	✓	✓	✓
Cloud Diagnostics (Plus, Pro, Enterprise)		✓	✓	✓
Integration with collaboration tools	pick 1	✓	✓	✓
Monetize				
Unity Ads	✓	✓	✓	✓
In-App Purchase plug-in	✓	✓	✓	✓
Support and learning				
Priority queue for Customer Service			✓	✓
Partner Advisor			✓	1-19 seats
Technical support			⊕	✓
On-demand training			⊕	20+ seats
Starter Success			⊕	1-19 seats
Partner Relations Manager (PRM)			⊕	20+ seats
Bug Fixing and LTS Backporting			⊕	100+ seats

Figura 2.3: Comparativa entre planes que Unity ofrece (2)

2.1.2 Microsoft Kinect v1

El dispositivo Microsoft Kinect V1, comúnmente conocido como Kinect for Xbox 360, es un sensor de movimiento desarrollado por Microsoft. Su fecha de lanzamiento se remonta a noviembre de 2010, saliendo originalmente como complemento para la consola Xbox 360. Dispone de una combinación de cámaras RGB, sensores de profundidad y micrófonos para capturar y rastrear los movimientos del cuerpo humano sin la necesidad de controladores. Esto le permite realizar una interacción más natural y basada en gestos con las aplicaciones compatibles.

Figura 2.4: Dispositivo Mirosoft Kinect

Kinect V1 surge como consecuencia directa del proyecto "Project Natal" [8] de Microsoft, que buscaba crear una experiencia de juego mucho más inmersiva y sin necesidad de mandos. Gracias a su amplio abanico de características en relación al control por voz y detección de movimiento, el Kinect V1 revolucionó la forma en que los jugadores interactúan con sus consolas Xbox 360.

El dispositivo cuenta con un sensor de infrarrojos y una cámara RGB integrada para medir la distancia y la forma de los objetos y personas en la escena, permitiendo un seguimiento preciso de los movimientos. A su vez, permite capturar imágenes en color y proporciona una visión clara del escenario. Posee reconocimiento y control de voz. Además, puede rastrear y reconocer el esqueleto de una persona, identificando las articulaciones y los movimientos corporales con precisión. También, dispone de un amplio campo de visión, lo que le permite rastrear movimientos de varias personas al mismo tiempo en una misma habitación.

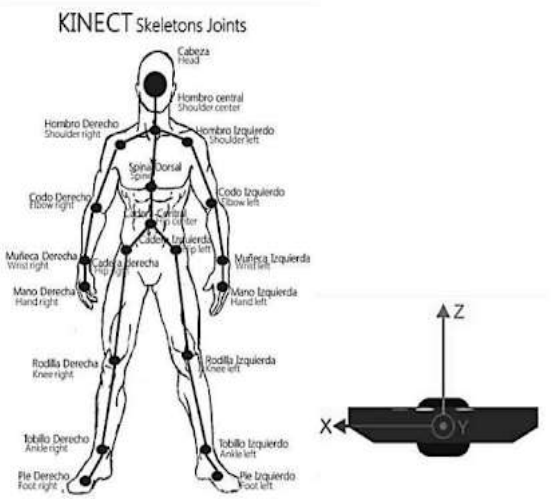


Figura 2.5: Esquema reconocimiento de esqueleto

El Kinect V1 ha sido ampliamente utilizado, no solo en videojuegos, sino también en otros campos y disciplinas, como pueden ser: la medicina, investigación, arte interactivo y proyectos de realidad aumentada.

2.1.3 C#

Es un lenguaje de programación diseñado por Microsoft, que tiene sus raíces en la familia de lenguajes C. Está orientado a objetos y posee seguridad de tipos. Se trata de un lenguaje natural en el que crear y usar componentes de software. Además, permite a los desarrolladores crear diversos tipos de aplicaciones seguras y sólidas que se ejecutan en .NET.

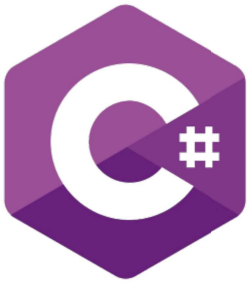


Figura 2.6: Logo lenguaje C#

Es un lenguaje que se mantiene actualizado, puesto que desde su origen, ha agregado características para admitir nuevas cargas de trabajo y prácticas de diseño de software emergentes. Además, resalta el control de versiones para garantizar que los programas y las bibliotecas puedan evolucionar con el tiempo de manera compatible.

2.2 Aprendizaje de Unity

Una vez finalizado el estudio de las tecnologías que se utilizaron en el desarrollo del proyecto, llegó el momento de poner en práctica todos los conocimientos adquiridos necesarios para trabajar con el software Unity. Para poner a prueba los dichos conocimientos y con el objetivo de facilitar el posterior desarrollo del prototipo final, se planteó la creación de un pequeño proyecto 3D que integrase movimiento a un avatar en un entorno sencillo.

2.2.1 Instalación y lectura de documentación

En primer lugar se ha realizado la preparación del entorno de trabajo. Para ello se ha descargado e instalado la aplicación Unity Hub [9]. A través de esta, se pueden administrar los proyectos, instalar distintas versiones del editor, activar la licencia e instalar componentes complementarios para el desarrollo de los proyectos en Unity.

A continuación, se ha instalado la versión 2021.3.24f1 LTS del editor de Unity, que será la utilizada a lo largo de todo el proceso de desarrollo del proyecto.

Una vez instalados todos los componentes que forman parte del entorno, se ha realizado una lectura en profundidad de la documentación [10] correspondiente a la versión del editor instalada. A partir de esta lectura, se ha logrado comprender una serie de conceptos básicos que son fundamentales en la realización de cualquier proyecto de Unity 3D. Entre los conceptos aprendidos se incluyen los siguientes:

- GameObjects: son objetos fundamentales en Unity que representan personajes, accesorios, escenarios y más. Cada objeto en el juego es un GameObject y existen en entornos 3D llamados escenas.

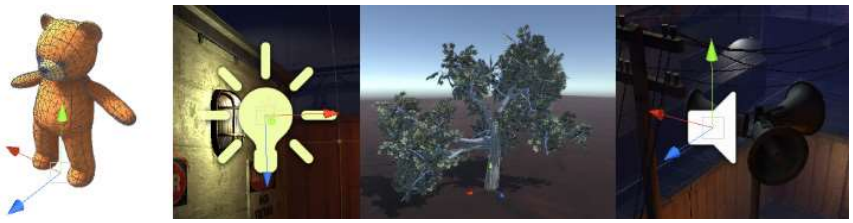


Figura 2.7: Ejemplos de GameObjects

- Transform: es un componeente que determina la posición, rotación y escala de cada GameObject en la escena. Todos los GameObjects tienen un componente Transform.

Figura 2.8: Componente Transform

- Mesh Filter: este componente define la forma de un GameObject en 3D, lo que le permite tener una apariencia visual específica.
- Cámaras: son GameObjects especialmente configurados que capturan y muestran el mundo del juego al jugador, permitiendo su visualización desde diferentes perspectivas.
- Rigidbody: permiten que los GameObjects interactúen con el sistema de física en Unity, incluyendo la gravedad y las colisiones entre objetos.

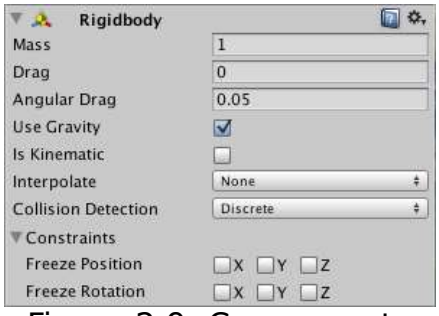


Figura 2.9: Componente Rigidbody

- Colliders: define la forma de un GameObject en 3D con el propósito de detectar y gestionar colisiones físicas entre objetos en la escena.

Asimismo se han adquirido conocimientos básicos sobre el funcionamiento de los 3D Assets y la incorporación de scripts de movimiento a modelos 3D.

2.2.2 Creación del proyecto de prueba

Utilizando los conocimientos mencionados anteriormente, se ha creado el proyecto de prueba. En primer lugar, se ha descargado el modelo 3D y una serie de animaciones desde la plataforma Mixamo [11]. Adobe Mixamo es una solución de animación en 3D que proporciona a los usuarios una colección de personajes 3D de alta calidad totalmente texturizados y animaciones de personajes de cuerpo completo capturadas por actores de movimiento profesionales las cuales pueden usarse de forma gratuita.

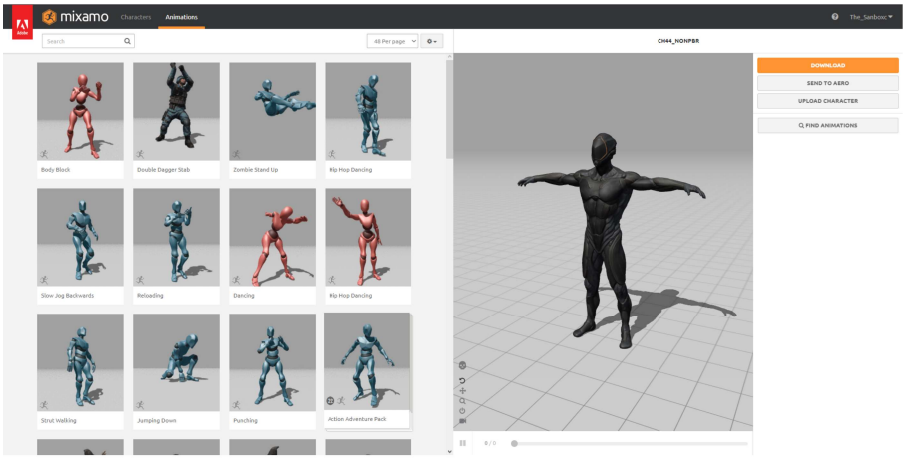


Figura 2.10: Plataforma Adobe Mixamo

En segundo lugar, se ha importado el modelo 3D y las animaciones al proyecto. Se ha utilizado el formato FBX, para asegurar que el modelo y las animaciones se importan correctamente. A continuación, se ha creado un script en C# que se encarga de controlar el desplazamiento del personaje en función de las interacciones del jugador. Este script utiliza funciones para permitir el movimiento suave y natural del personaje en respuesta a las acciones del jugador.

Listado 2.1: Método que controla el movimiento del personaje

```
void Update {
        x = Input.GetAxis("Horizontal");
        y = Input.GetAxis("Vertical");
        transform.Rotate(0, x * Time.deltaTime *
        velocidad_rotacion, 0);
        transform.Translate(0, 0, y * Time.deltaTime *
        velocidad_movimiento);
        animacion.SetFloat("VelX", x);
        animacion.SetFloat("VelY", y);
}
```

Para que sea posible la visualización de las animaciones al controlar el personaje, es necesaria la creación de un componente Animator. Una vez creado, se han configurado algunos parámetros como las condiciones de activación de cada animación y se han asignado dichas animaciones al

componente creado. Además se ha hecho uso de las variables y los estados del componente Animator para controlar la reproducción de las animaciones según la posición del personaje respecto al plano del juego.

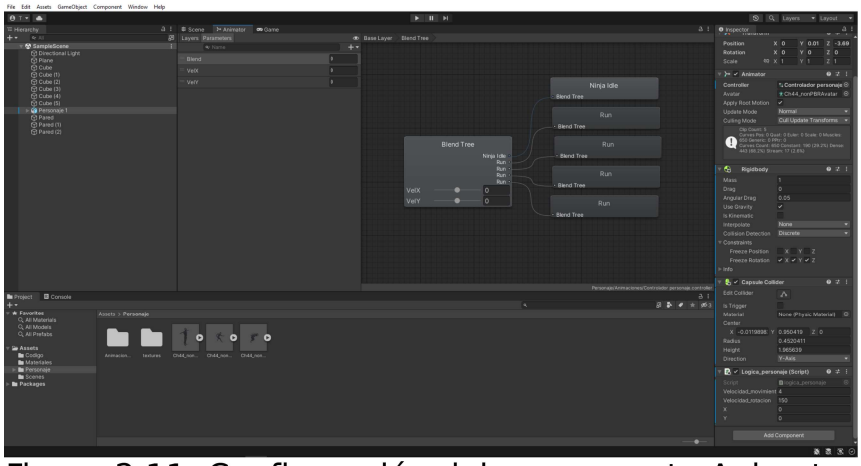


Figura 2.11: Configuración del componente Animator

Por último, se ha asignado el componente Animator y el script al objeto que contiene el modelo 3D del personaje permitiendo así que pudiera moverse de manera realista y fluida en el entorno del juego, ejecutando las animaciones correspondientes según su posición y las interacciones del propio jugador.

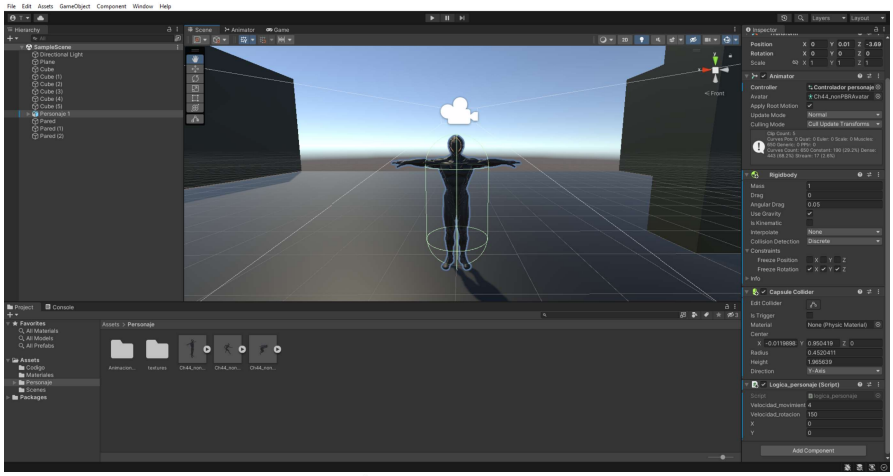


Figura 2.12: Asignación de los componentes al avatar

Capítulo 3
Desarrollo del videojuego

En este capítulo veremos todas las tareas llevadas a cabo para desarrollar el videojuego.

Cabe destacar que el nombre del videojuego es **Just Repeat**, una clara referencia al popular juego de baile “Just Dance”, el cuál hace énfasis en el principal objetivo del mismo, la simple repetición de una serie de movimientos.

3.1 Conexión del dispositivo Kinect con Windows

Al igual que en el desarrollo del proyecto de prueba mencionado en el capítulo anterior, para el correcto desarrollo del videojuego es necesario preparar primero el entorno de trabajo. Todo el entorno Unity ha sido desplegado exitosamente, por lo que el siguiente paso es preparar la conexión del dispositivo Kinect para que pueda ser usado como dispositivo captador de movimientos desde nuestro ordenador.

Antes de proceder a la integración del Kinect v1 con Unity, se deben cumplir varios requisitos previos. Es necesario disponer del dispositivo Kinect y contar con su adaptador USB para poder conectarlo al ordenador.

Figura 3.1: Adaptador
USB para Kinect v1

Una vez cumplidos los requisitos mencionados, se ha llevado a cabo la conexión del Kinect v1 con el ordenador. Para ello, se ha descargado e instalado la herramienta Kinect for Windows SDK v1.8 [12]. Dicha herramienta fue creada por Microsoft y lanzada en junio de 2013, con el objetivo principal de proporcionar a los desarrolladores una plataforma y un conjunto de

herramientas para crear aplicaciones utilizando el sensor Kinect en entornos de desarrollo de Windows. A su vez, permite acceder a los datos capturados por el sensor de Kinect y usarlos de la forma que le plazca al usuario.

Para verificar que se ha establecido la conexión de manera exitosa entre Kinect y el ordenador, se ha accedido al administrador de dispositivos y se ha comprobado que haya sido detectado y aparezca como un equipo conectado en funcionamiento.

Figura 3.2: Pantalla del administrador de dispositivos

Concluida esta etapa, se procede a la integración de Kinect con Unity, donde se aprovecharán las características del dispositivo para completar el desarrollo del videojuego.

3.2 Integración de Kinect con Unity

Una vez establecida la conexión de forma exitosa, se procede a la creación del proyecto. Lo primero que ha sido realizado es la creación de un nuevo proyecto en Unity, sobre el cual se alojará posteriormente la integración con Kinect.

3.2.1 Importación del asset Kinect with MS-SDK

Para la realización de dicha integración se ha importado como base en el directorio raíz del proyecto un asset personalizado desde la **Asset Store** de Unity llamado Kinect with MS-SDK [13]. Dicho asset, desarrollado y publicado por RF Solutions, es un conjunto de ejemplos de Kinect v1 que utiliza varios scripts principales, agrupados en una carpeta. Utiliza el SDK/Runtime de Kinect proporcionado por Microsoft para mostrar cómo utilizar avatares controlados por Kinect, detectar gestos y otras funcionalidades relacionadas con Kinect en tus propios proyectos de Unity. Esto permite ahorrar tiempo en el desarrollo del videojuego puesto que se ha tomado como base scripts ya creados de uso libre que permiten manipular los datos recogidos por el dispositivo Kinect.

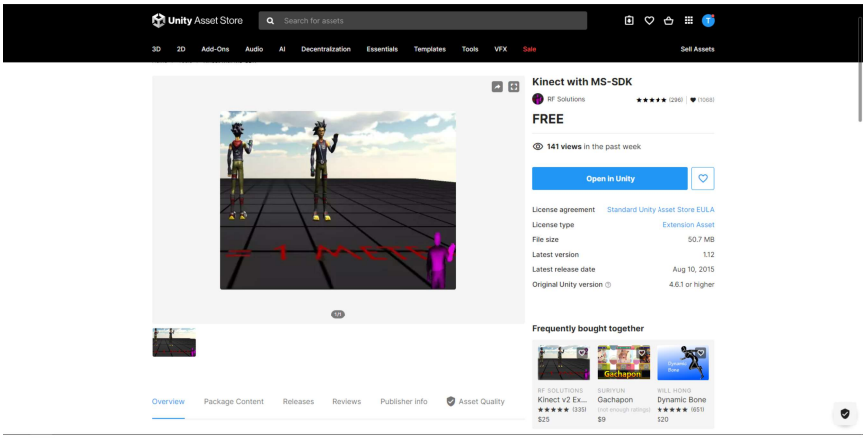


Figura 3.3: Descarga del asset MS-SDK

Debido a que el asset importado no se actualiza desde 2015, después de su importación en el proyecto surgieron una serie de errores por consola con código CS0618 y CS1061. Dichos errores se deben a referencias a métodos y librerías obsoletos presentes en los scripts para el manejo de Kinect. El motivo principal de estos errores se debe a que este asset esta pensado para funcionar integrado en Unity 5.3, y como ya se ha comentado en el apartado 2.2.1. en este proyecto se ha utilizado la versión 2021.3.24f1 LTS.

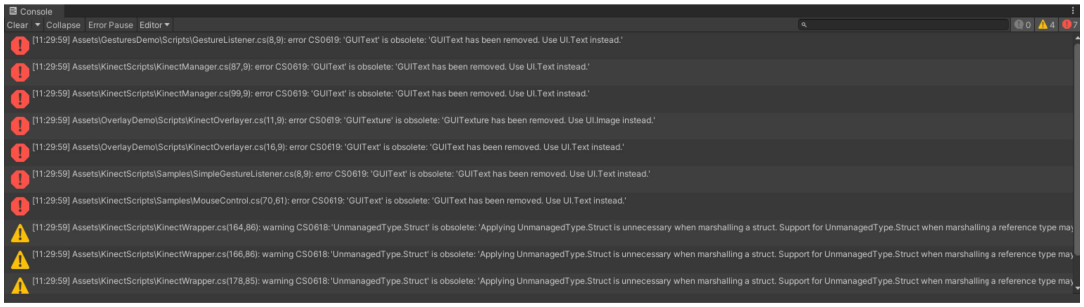


Figura 3.4: Errores de incompatibilidad entre versiones

Dichos errores se han solucionado modificando directamente el código fuente de los scripts, cambiando las importaciones de las librerías obsoletas y llamadas a métodos inexistentes en la versión utilizada.

Se ha modificado el método obsoleto Application.LoadLevel(int) utilizando el método SceneManager.LoadScene en su lugar. Se han corregido todas las importaciones de la librería GUItext desfasada a la actual Text. Una vez realizados los cambios mencionados, los scripts están listos para ser probados y para agregar métodos personalizados posteriormente.

3.2.2 Creación y gestion de escenas

El enfoque principal en el desarrollo del videojuego ha sido la funcionalidad, otorgando menor importancia al apartado gráfico. Esto se debe a que la relevancia del juego radica en su capacidad para detectar y evaluar los movimientos realizados por el niño/a, más que en su apariencia visual. Es por este mismo motivo que el prototipo se ha diseñado para poseer la menor cantidad de escenas posibles, haciendo que luzca una estetica minimalista y simple.

Existen dos tipos de escenas:

- Menú principal: es la pantalla principal del juego, que incluye un menú interactivo sencillo con varias opciones, como Jugar, Opciones y Salir.
- Escena de Juego: es la pantalla en la que se muestra al avatar y en la que se realiza la detección de los gestos por medio del dispositivo Kinect.

Para diseñar los elementos presentes en el menú, se han creado objetos UI dentro del área Canvas de Unity. Se ha utilizado un objeto UI de tipo Image para el fondo, uno de tipo Text para el título y objetos de tipo Button para las opciones seleccionables en el menú.

Además, se han agregado dos objetos vacíos para separar el menú principal del menú que se despliega al hacer clic en Opciones. Después de crear todos

los elementos y configurar su apariencia visual, se ha desarrollado un script que proporciona la funcionalidad al menú.

Se ha creado una clase llamada MainMenu, que contiene dos funciones:

- PlayGame: este método permite al jugador pasar a la pantalla de juego cuando se selecciona la opción "Jugar".

Listado 3.1: Código función PlayGame

```
public void PlayGame() {
        SceneManager.LoadScene
        (SceneManager.GetActiveScene().buildIndex + 1);
}
```

- QuitGame: este método permite al jugador salir del juego al presionar el botón "Salir".

Listado 3.2: Código función QuitGame

```
public void QuitGame() {
        Debug.Log("Ha salido sin problema del juego");
        Application.Quit();
}
```

Estas funciones proporcionan la interactividad necesaria para que el menú principal del juego sea funcional y permita al jugador navegar entre las diferentes opciones.

Figura 3.5: Pantalla principal del videojuego

Figura 3.6: Menu de opciones

Para el caso del menu opciones es mucho más sencillo. Se ha establacido su visibilidad por defecto a false, y en el objeto de tipo UI button “Opciones”, en la propiedad On Click() se han establecido dos condiciones. La primera es activar la visivilidad del objeto correspondiente al menu de opciones y la segunda es desactivar la visivilidad del menu principal. Para el caso del menu de opciones se ha seguido el mismo procedimiento aplicado sobre el botón “Volver”

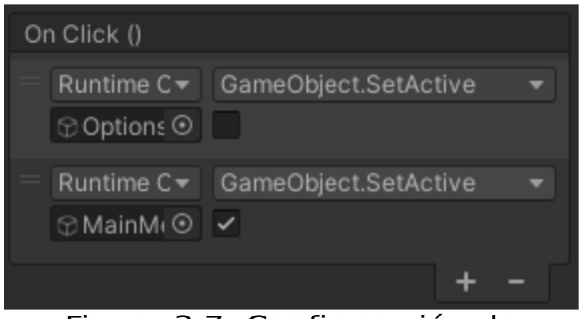


Figura 3.7: Configuración de propiedad On Click()

En cuanto a la pantalla de juego, se ha reciclado una escena que viene por defecto al momento de importar el asset. Dicha escena contiene un modelo 3D importado de un avatar de Unity sobre el cuál se ha asignado el script AvatarController. Dicho script se encarga de controlar y gestionar el comportamiento y la interacción del avatar en el juego. Es decir, utiliza los datos del seguimiento de esqueletos proporcionados por el Kinect para actualizar la posición y la rotación del modelo del avatar de acuerdo con los movimientos del jugador.

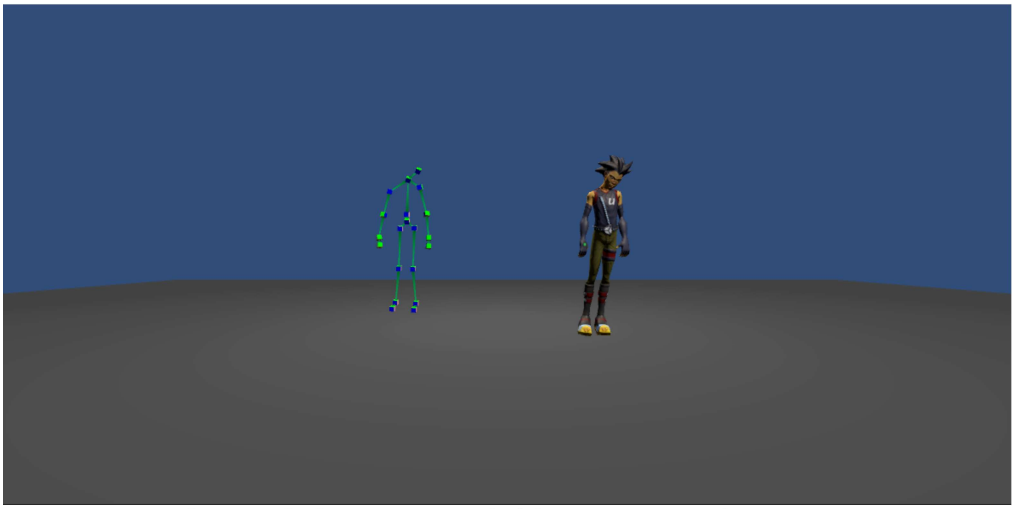


Figura 3.8: Escena del juego con el Avatar

Además en esta escena existe un componente denominado KinectManager asociado al objeto MainCamera. Es una parte clave del sistema, puesto que es el responsable de administrar la comunicación y el control entre el sensor Kinect y la aplicación Unity. Entre sus funciones principales destacan las siguientes:

- Se encarga de inicializar y establecer la conexión con el sensor Kinect.
- Detecta y realiza un seguimiento de los cuerpos de los usuarios presentes en el campo de visión. Incluyendo la detección de articulaciones y movimientos de los personajes.
- Puede identificar cuando un usuario realiza uno de estos gestos y notificar a la aplicación Unity para que realice acciones específicas en respuesta.
- Recopila y proporciona acceso a los datos capturados por el sensor Kinect, como las coordenadas 3D de las articulaciones de los usuarios

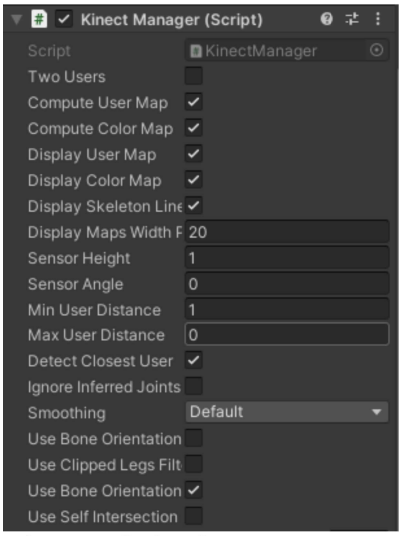


Figura 3.9: Componente KinectManager

Por último, cabe destacar la existencia del objeto Cubeman, el cual es utilizado para mostrar el seguimiento y la detección de los usuarios en tiempo real. Suele estar compuesto por diversos cubos que representan diferentes partes del cuerpo. Dichos cubos, se actualizan para reflejar la postura y los movimientos de los usuarios detectados por el sensor Kinect.

A dicho objeto, se encuentra asociado el script CubemanController, el cual cuyo nombre indica, es el controlador encargado de actualizar la posición y rotación de los cubos que representan las articulaciones del personaje. En cuanto al código fuente del script, este comienza declarando una serie de variables públicas que controlan el movimiento vertical y si se desea reflejar movimientos. Posteriormente, se crean las variables GameObject que representan las articulaciones del avatar, como la cabeza, los hombros, los codos, las muñecas, las manos, las rodillas, los tobillos y los pies. Además, se declara una variable tipo LineRenderer llamada SkeletonLine, que será la encargada de dibujar las lineas que representan los huesos en el espacio.

El script posee dos métodos principales:

- Método Start(): encargado de inicializar los conjuntos de huesos y líneas, almacenando la posición y rotación iniciales del personaje. A su vez, asigna la variable SkeletonLine a cada hueso para dibujar la línea entre ellos.

```
void Start ()
{
    //store bones in a list for easier access
    bones = new GameObject[] {
        Hip_Center, Spine, Shoulder_Center, Head,  // 0 - 3
        Shoulder_Left, Elbow_Left, Wrist_Left, Hand_Left,  // 4 - 7
        Shoulder_Right, Elbow_Right, Wrist_Right, Hand_Right,  // 8
        Hip_Left, Knee_Left, Ankle_Left, Foot_Left,  // 12 - 15
        Hip_Right, Knee_Right, Ankle_Right, Foot_Right  // 16 - 19
    };

    parIdxs = new int[] {
        0, 0, 1, 2,
        2, 4, 5, 6,
        2, 8, 9, 10,
        0, 12, 13, 14,
        0, 16, 17, 18
    };

    // array holding the skeleton lines
    lines = new LineRenderer[bones.Length];

    if(SkeletonLine)
    {
        for(int i = 0; i < lines.Length; i++)
        {
            lines[i] = Instantiate(SkeletonLine) as LineRenderer;
            lines[i].transform.parent = transform;
        }
    }

    initialPosition = transform.position;
    initialRotation = transform.rotation;
    //transform.rotation = Quaternion.identity;
}
```

Figura 3.10: Método Start() de CubemanController

- Método Update(): obtiene una instancia de KinectManager y verifica si se detecta un jugador. En caso de no detectar jugador, se restauran la posición y rotación inicial del avatar y se desactivan los objetos que representan las articulaciones. En caso contrario, se obtiene la posición del jugador en el espacio y se actualiza la del personaje en base a esta información.

```
void Update ()
{
    KinectManager manager = KinectManager.Instance;

    // get 1st player
    uint playerID = manager != null ? manager.GetPlayer1ID() : 0;

    if(playerID <= 0)
    {
        // reset the pointman position and rotation
        if(transform.position != initialPosition)
        {
            transform.position = initialPosition;
        }

        if(transform.rotation != initialRotation)
        {
            transform.rotation = initialRotation;
        }

        for(int i = 0; i < bones.Length; i++)
        {
            bones[i].gameObject.SetActive(true);

            bones[i].transform.localPosition = Vector3.zero;
            bones[i].transform.localRotation = Quaternion.identity;

            if(SkeletonLine)
            {
                lines[i].gameObject.SetActive(false);
            }
        }

        return;
    }
```

Figura 3.11: Método Update() de CubemanController

Por último, se actualiza la posición y rotación de cada articulación en base a los datos capturados por Kinect. Los huesos se activan o desactivan según si el sensor detecta el seguimiento de las articulaciones correspondientes.

3.3 Creación de gestos personalizados y almacenamiento de movimientos

En este apartado vamos a ver los pasos que se han seguido para completar la creación de un gesto personalizado asi como la adición al proyecto de la detección y reconocimiento del mismo.

Para agregar un gesto personalizado se ha modificado el script que venía importado por defecto en el proyecto, ubicado en la ruta Assets/KinectScripts/KinectGestures.cs. En primer lugar, dentro del enum Gestures, se ha asignado el nombre del gesto que se desea crear. A continuación, para implementar la detección del gesto se ha agregado un case al switch() ubicado en la función CheckForGesture(), puesto que cada case procesa la detección de cada gesto definido en el enum Gestures.

Cada gesto de ejemplo cuenta con su propio su propio switch() interno para verificar y cambiar el estado actual del gesto. Cada gesto es como una máquina de estados con estados numéricos. Su estado actual, junto con otros datos, se almacena en una estructura interna de tipo GestureData. Esta estructura de datos se crea para cada gesto que necesita ser detectado en la escena.

Listado 3.3: Estructura de los gestos

```
public struct GestureData {

public uint userId;
```

```
        public Gestures gesture;
        public int state;
        public float timestamp;
        public int joint;
        public Vector3 jointPos;
        public Vector3 screenPos;
        public float tagFloat;
        public Vector3 tagVector;
        public Vector3 tagVector2;
        public float progress;
        public bool complete;
        public bool cancelled;
        public List<Gestures> checkForGestures;
        public float startTrackingAtTime
    }
```

El estado inicial de cada gesto es siempre 0. En ese estado, el código necesita detectar si el gesto está comenzando o no. Para hacer esto, se verifica y almacena la posición de una articulación. Si la posición de la articulación es correcta para iniciar el gesto se incrementa el estado. En el siguiente estado se verifica si dicha articulación ha alcanzado la posición requerida dentro de un intervalo de tiempo.

```
// check for RaiseLeftHand
case Gestures.RaiseLeftHand:
    switch(gestureData.state)
    {
        case 0:  // gesture detection
            if(jointsTracked[leftHandIndex] && jointsTracked[leftShoulderIndex] &&
                    (jointsPos[leftHandIndex].y - jointsPos[leftShoulderIndex].y) > 0.1f)
            {
                SetGestureJoint(ref gestureData, timestamp, leftHandIndex, jointsPos[leftHandIndex]);
            }
            break;

        case 1:  // gesture complete
            bool isInPose = jointsTracked[leftHandIndex] && jointsTracked[leftShoulderIndex] &&
                (jointsPos[leftHandIndex].y - jointsPos[leftShoulderIndex].y) > 0.1f;

            Vector3 jointPos = jointsPos[gestureData.joint];
            CheckPoseComplete(ref gestureData, timestamp, jointPos, isInPose, KinectWrapper.Constants.PoseCompleteDuration);
            break;
    }
    break;
```

Figura 3.12: Código ejemplo de gesto

Si la articulación alcanza la posición que tiene marcada como objetivo dentro del intervalo de tiempo, el gesto se considera completado. En caso contrario, se marca como cancelado. Posteriormente el estado del gesto puede ser reseteado a 0 para reiniciar el proceso de detección del gesto.

En resumen, la forma de proceder ha sido comprender cómo funcionan gestos relativamente simples de los ejemplos que ya venían creados en el asset, para posteriormente modificar sú código ajustándolo a las necesidades

requeridas.

Ya se ha comentado como crear gestos personalizados, sin embargo, aún no se ha mencionado como dotar de detección y reconocimiento de gestos a la escena del juego. Como se ha mencionado en el anterior apartado, en la escena del juego existe un componente asociado a la MainCamera denominado KinectManager. Dentro de este componente existe una propiedad denominada “Gestos del jugador”,la cual te permite seleccionar el numero de gestos a detectar y asignar que gestos del enum Gestures se van a reconocer.

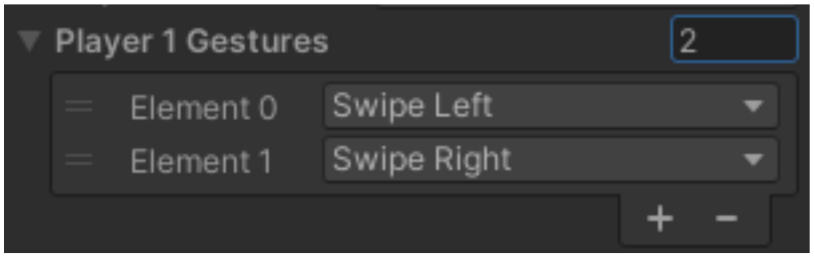


Figura 3.13: Propiedad gestos del jugador

Una vez han sido asignados los movimientos que se van a reconocer es hora de poner a prueba el proyecto y comprobar que todo está en orden.

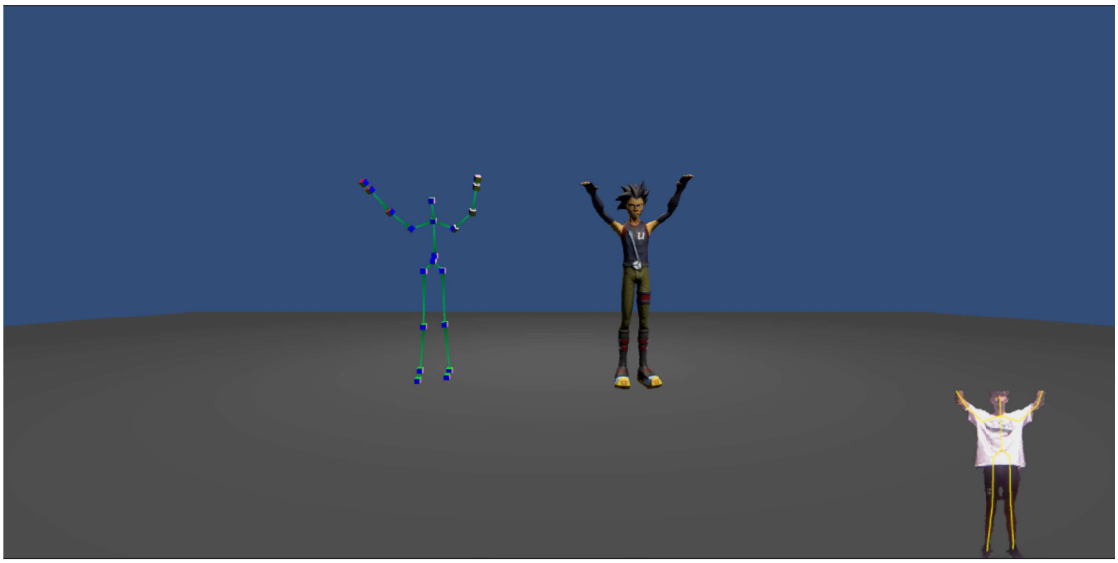


Figura 3.14: Pruebas de gestos con Kinect (1)

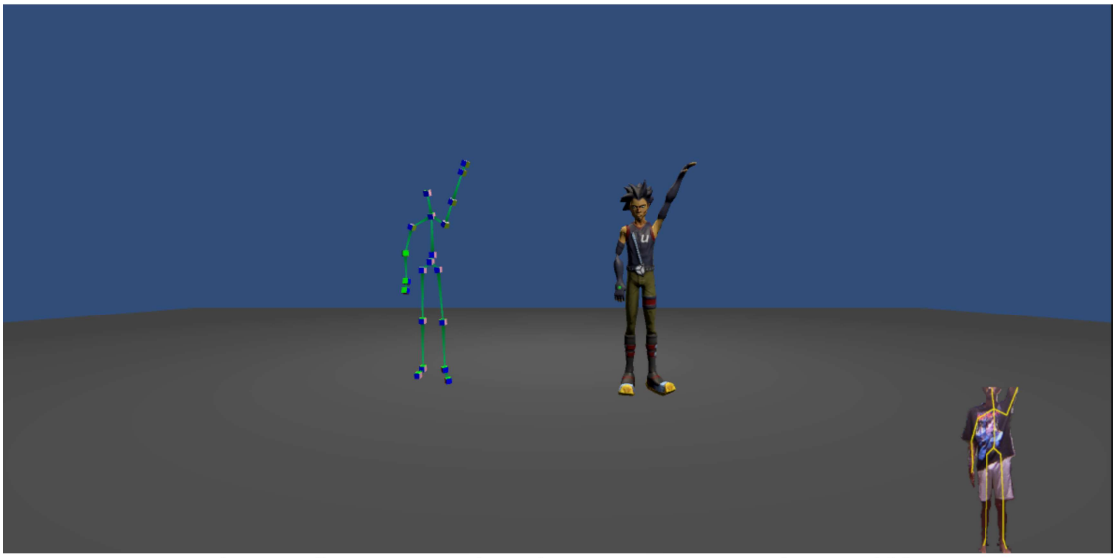


Figura 3.15: Prueba de gestos con Kinect (2)

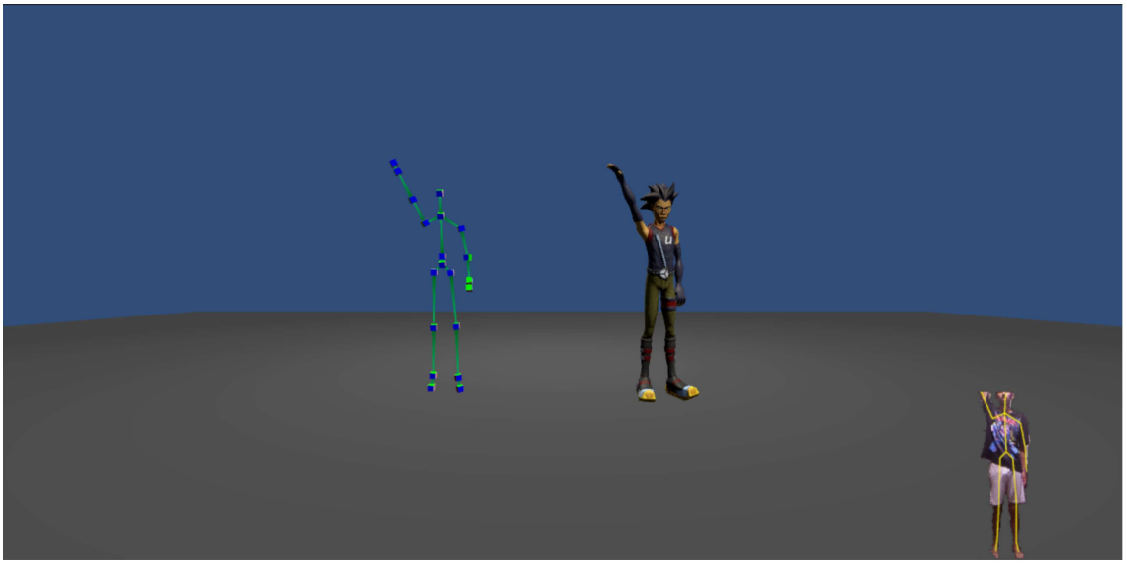


Figura 3.16: Prueba de gestos con Kinect (3)

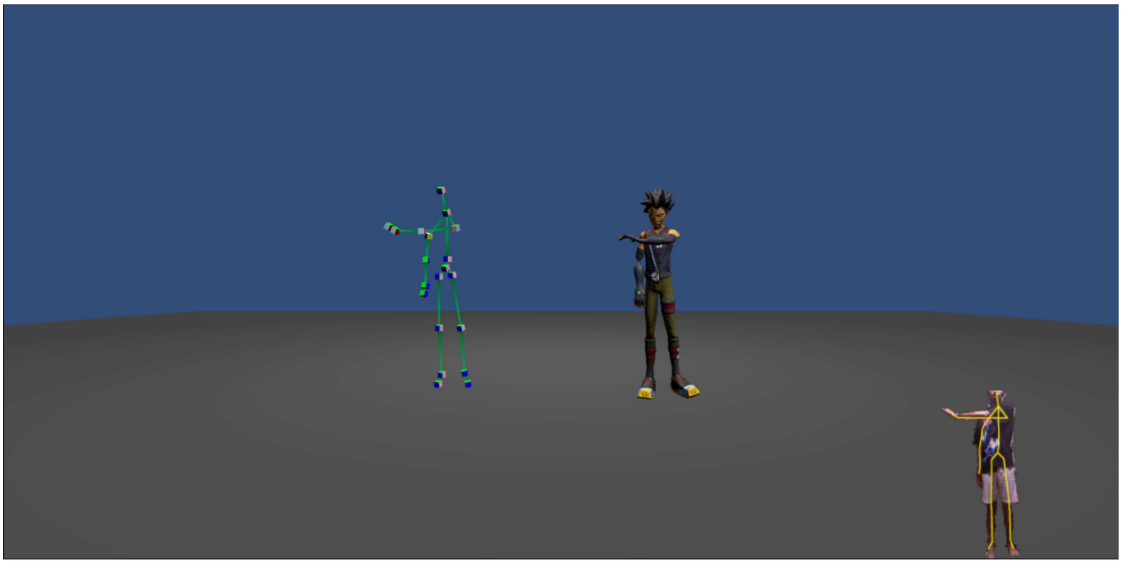


Figura 3.17: Prueba de gestos con Kinect (4)

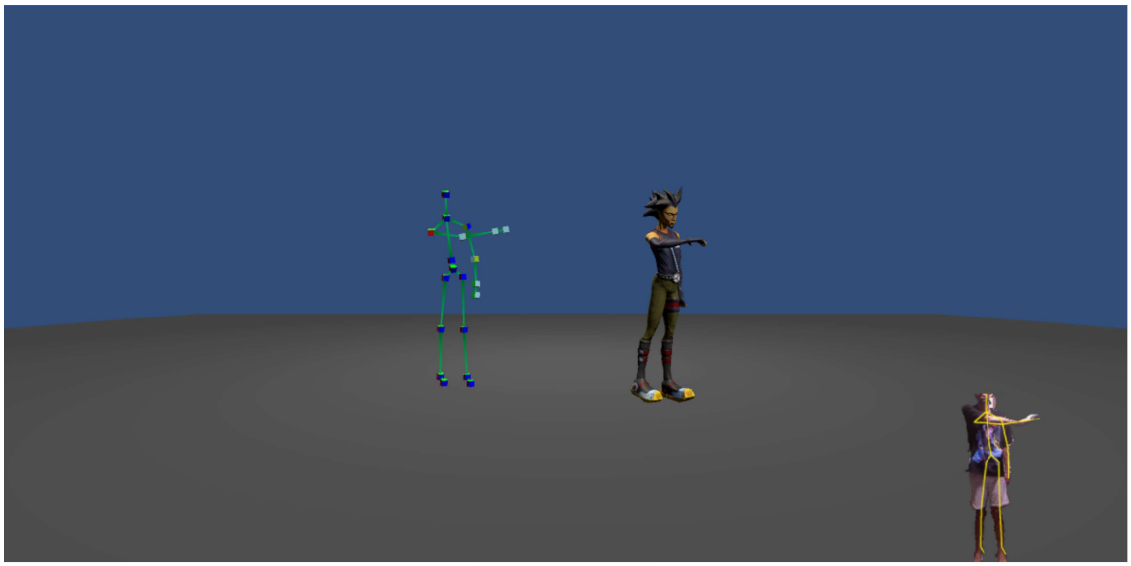


Figura 3.18: Prueba de gestos con Kinect (5)

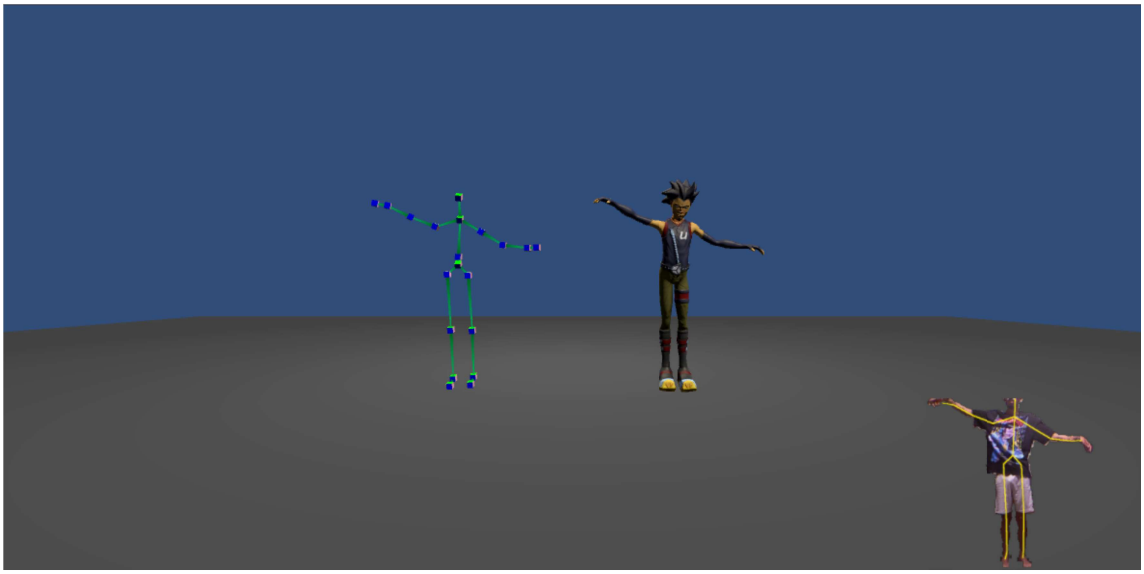


Figura 3.19: Prueba de gestos con Kinect (6)

Capítulo 4
Conclusiones y líneas futuras

A pesar de que el videojuego se ha enfocado en la funcionalidad y ha sacrificado la estética visual, se ha logrado cumplir su objetivo principal. Sin embargo, como mejoras futuras, recomiendo la contratación de un diseñador gráfico para mejorar el apartado visual del juego y dedicar tiempo al desarrollo de la retroalimentación en tiempo real.

Personalmente, este proyecto me ha permitido adquirir conocimientos en el desarrollo de aplicaciones utilizando Kinect y Unity. Además, he tenido la oportunidad de aprender y utilizar el lenguaje de programación C# en la implementación de scripts en Unity. Estas habilidades y conocimientos adquiridos resultarán valiosos en futuros proyectos y en mi desarrollo profesional en el ámbito de la programación.

Capítulo 5
Summary and Conclusions

Despite the video game's focus on functionality and the sacrifice of visual aesthetics, its main objective has been successfully achieved. However, for future improvements, I recommend hiring a graphic designer to enhance the game's visual aspect and dedicating time to the development of real-time feedback.

Personally, this project has allowed me to acquire knowledge in application development using Kinect and Unity. Additionally, I have had the opportunity to learn and utilize the C# programming language for script implementation in Unity. These acquired skills and knowledge will be valuable for future projects and my professional development in the programming field.

Capítulo 6
Presupuesto

Se ha llevado a cabo un presupuesto sobre el esfuerzo realizado durante el desarrollo de este TFG. La duración del mismo se encuentra alrededor de unas 300 horas, según lo establecido por el plan de estudios publicado el 21 de marzo de 2011, por la Universidad de La Laguna.

Tareas	**Duración**	**Coste**
Estudio de la situación actual del mercado	30h	15€/h
Estudio de las tecnologías a utilizar en el desarrollo del videojuego	80h	15€/h
Desarrollo del videojuego	140h	20€/h
Documentación	50h	15€/h
Total	**300h**	**5.200€**

Tabla 6.1: Presupuesto en base a horas invertidas

Materiales	**Características**	**Cantidad**	**Coste**
Ordenador	Intel Core i5-9400F 8 GB DDR4 SSD 240 GB NVIDIA GeForce GTX 1650	1	1.058€
Microsoft Kinect	Versión 1 USB 2.0 Profundidad máxima: 4m	1	110€

Adaptador USB Kinect	Alimentación de 100V-240V	1	3,50€
Total		**3**	**1.171,5€**

Tabla 6.2: Presupuesto de los materiales

Bibliografía

[1] “Todo lo que necesitas saber sobre discapacidad motriz” Último acceso: 2023-04-08. [Online]. Available: https://www.incluyeme.com/todo-lo-que-necesitas-saber-sobre-discapacidad-motriz/

[2] “Población con discapacidad en España, en gráficos” Último acceso: 2023-04-10. [Online]. Available: https://www.epdata.es/datos/poblacion-discapacidad-espana-graficos/631

[3] “Health Benefits of Digital Videogames for the Aging Population: A Systematic Review” Último acceso: 2023-04-10. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32589482/

[4] “Videogames and Health Improvement: A Literature Review of Randomized Controlled Trials” Último acceso: 2022-04-11. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26191999/

[5] “Unity” Último acceso: 2023-04-27. [Online]. Available: https://unity.com/es

[6] “Microsoft Kinect” Último acceso: 2023-05-02. [Online]. Available: https://zaguan.unizar.es/record/12845/files/TAZ-PFC-2013-649_ANE.pdf

[7] “Paseo por el lenguaje C#” Último acceso: 2023-04-29. [Online]. Available: https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

[8] “Project Natal,” Último acceso: 2023-05-04. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2010/01/Kudo_Fitzgibbons_Kinect_for_Xbox360_071210_FacSummit.pdf

[9] “Unity Hub” Último acceso: 2023-04-27. [Online]. Available: https://unity.com/unity-hub

[10] “Unity User Manual 2021.3 (LTS)” Último acceso: 2023-04-25. [Online]. Available: https://docs.unity3d.com/2021.3/Documentation/Manual/UnityManual.html

[11] “Adobe Mixamo” Último acceso: 2023-05-10. [Online]. Available: https://www.mixamo.com/#/

[12] “Microsoft Kinect SDK v1.8” Último acceso: 2023-06-02. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=40278

[13] “Kinect with MS-SDK” Último acceso: 2023-06-20. [Online]. Available: https://assetstore.unity.com/packages/tools/kinect-with-ms-sdk-7747