2016/2017

SONRIE KINGDOM

David González Rodríguez

CITSEM

Índice

Introducción

Actualmente se utiliza la tecnología para casi cualquier ámbito de la vida cotidiana, desde la educación y la enseñanza, pasando por el ocio y el entretenimiento y terminando en la medicina. Este proyecto se basa en la utilización de la tecnología para conseguir ayudar a personas con problemas de movilidad reducida, centrándose en niños de entre cinco y doce años con parálisis cerebral. Debido a que los niños tienen grandes problemas de atención al a hora de realizar los ejercicios propuestos por los terapeutas, se decidió implementar estos de una forma llamativa y de fácil seguimiento mediante la utilización de los videojuegos, de aquí nace el proyecto SONRIE.

Primeramente, se creó un videojuego sencillo formado por cuatro pruebas, que se comentarán posteriormente, captadas mediante el sensor Kinect V1 de la consola Xbox 360. Este sensor capta diferentes puntos de la cara y envía el valor de cada uno de ellos al ordenador, desde ahí se establece si el ejercicio se ha realizado correctamente y si el usuario debe repetirlo o no. Estos ejercicios se basan en levantamiento de cejas, soplar, lanzar un beso y sonreír. Una explicación más detallada de los ejercicios junto con el código del primer programa utilizado para la realización de estos puede encontrarse en la tesis [1].

Poco tiempo después Microsoft lanzó al mercado su nuevo sensor Kinect V2 para la consola Xbox One, por lo que se decidió actualizar los proyectos creados para la primera versión. Debido a la poca información proporcionada para Kinect V2 y a la gran cantidad de problemas que tiene trabajar con una tecnología novedosa, se implementó un software para poder obtener los datos de Kinect, prepararlos a las necesidades previstas y enviarlos a cualquier programa o juego en los que se necesite. La mejora entre ambos sensores es muy significativa pues el último sensor permite obtener muchos más puntos que representan los músculos de la cara.

A parte de poder realizar el envío a distintos programas, se realizó un estudio de los nuevos puntos que se obtenían del sensor Kinect V2. Para ello se partió del programa creado por Jaime Jarrín, el cuál esta detallado en [2], con una pequeña modificación para mostrar los puntos en distintos colores en función del rango a estudiar, consiguiendo así obtener los más importantes para este proyecto.

Como el anterior videojuego estaba basado en Kinect V1, se decidió actualizar también este para conseguir un mayor nivel de detalle dentro del juego para aumentar la atención del usuario en los ejercicios y conseguir así una mejor realización de los mismos. Para ello, se utilizaron los programas Blender y Unity para el modelado y la lógica del videojuego respectivamente. En los próximos apartados se explicará cómo se desarrolló el juego y el Middleware, como está distribuida la información de del proyecto, como lanzar los programas y mejoras que aplicar en un futuro.

Desarrollo

Obtención de puntos

El primer objetivo de estas prácticas fue obtener los puntos importantes para los ejercicios recomendados por los terapeutas. Como ya se ha comentado, se partió del programa de Jaime Jarrín, explicado más profundamente en [2], con una pequeña variación del código para mostrar o colorear solo determinadas regiones de los puntos.

Los puntos no sirguen un orden lógico por lo que se tuvo que estudiar los puntos por regiones y posteriormente delimitar los más importantes a partir de estas.

En la tabla 1 puede verse las distintas regiones estudiadas de los puntos obtenidos por el sensor Kinect:

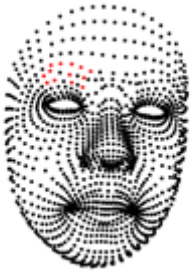
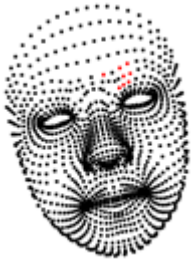
Tabla 1. Regiones de los puntos dados por Kinect.

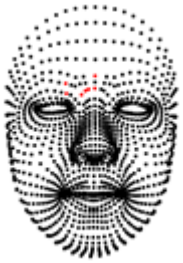
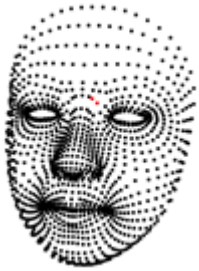
Puntos 0 - 199	
Puntos 200 - 399	
Puntos 400 - 599	
Puntos 600 - 799	
Puntos 800 - 999	

Puntos 1000 - 1199	
Puntos 1200 - 1346	

A partir de estas regiones se obtuvieron los puntos importantes para los ejercicios, los cuales se muestran en la tabla 2. Para simplificar los cambios necesarios en el programa a la hora de dibujar estos puntos, se añade la condición del "if" en la línea 188 del programa KinectFaceTestV2 de Jaime Jarrín.

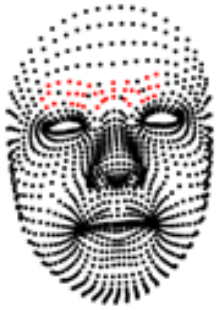
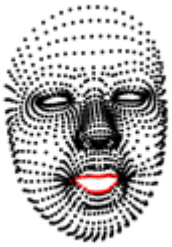
Tabla 2. Puntos importantes para los ejercicios y condición del if.

Puntos 200-201/208/210 213/217-219/222-224 345-347		`(index >= 200 && index < 202)||(index == 208) ||(index == 210) || (index == 213)|| ((index >= 217 && index < 220))|| ((index >= 222 && index < 225)) || (index >= 345 && index < 348)`
Puntos 1080-1085/ 1120-1154		`(index >= 1080 && index < 1086)/*Cejas*/ || (index >= 1120 && index < 1155)/*Boca*/`
Puntos 820/823-824/ 838-839/858		`(index == 820 || (index >= 823 && index <= 825) || (index >= 838 && index <= 840) || index == 858)`

Puntos 16/27/ 161-162/183/194-195		(index == 16 || index == 27 || (index == 161) || (index == 162) || index == 183 || index == 194 || index == 195)
Puntos 774/792		index==774 || index ==792

En la tabla 3 puede verse el conjunto de los puntos para cejas y boca respectivamente, juntos con la condición del "if" completa.

Tabla 3. Conjunto de puntos para cejas y boca.

Cejas		(index == 200 || index == 201 || index == 208 || index == 210 || index == 213 || (index >= 217 && index < 220) || (index >= 222 && index < 225) || (index >= 345 && index < 348)|| (index >= 1080 && index < 1086) || index == 820 || (index >= 823 && index <= 825) || (index >= 838 && index <= 840) || index == 858 || index == 16 || index == 27 || index == 161 || index == 162 || index == 183 || index == 194 || index == 195 || index == 774 || index == 792)
Boca		(index >= 1120 && index < 1155)

Middleware

Para el desarrollo del middleware se tuvo como base la memoria de las prácticas [3]. En este documento se explica cómo crear un Middleware para enviar la información de los puntos del sensor Kinect V2 hasta el programa Blender mediante un protocolo UDP. El proyecto actual utiliza una trasmisión mediante TCP debido a que Unity (al ser una licencia gratuita) tiene muchas librerías vetadas a la hora de usarlas en los Scripts. Este protocolo tiene mejores características que UDP pues asegura la transmisión de los paquetes además de enviarlos en orden.

Previo a la creación del Middleware se programó un software de prueba que simula el envío de la información del os 70 puntos a estudiar para confirmar que el envío era correcto, posteriormente se añadieron las líneas de código necesarias para conseguir que Kinect realice esta transmisión.

Se ha de comentar que el Middleware está dividido en dos partes, la parte programada en Visual Studio (correspondiente al programa de simulación y al programa de utilización de Kinect) y la parte realizada dentro de Unity en diferentes Scripts. Esta última parte es una simulación rápida de la recepción del programa en Unity que posteriormente debe ser introducida en el videojuego.

En las Figuras 1, 2 y 3 puede verse una representación de los programas tanto de la parte de Visual Studio como de Unity:

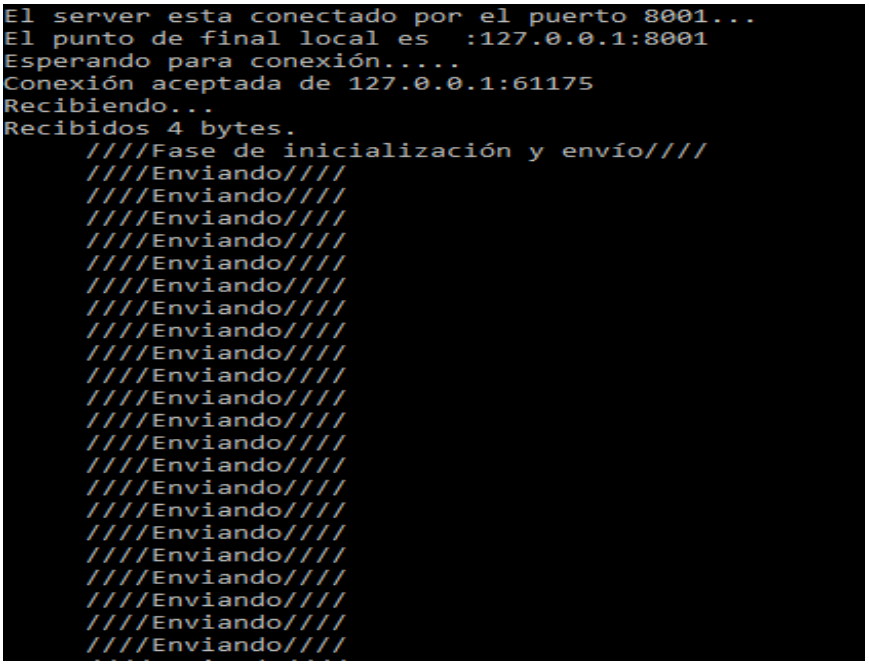


Figura 1. Representación del Middleware de simulación de envío de datos.

Figura 2. Representación del Middleware de envío de datos obtenidos por Kinect.

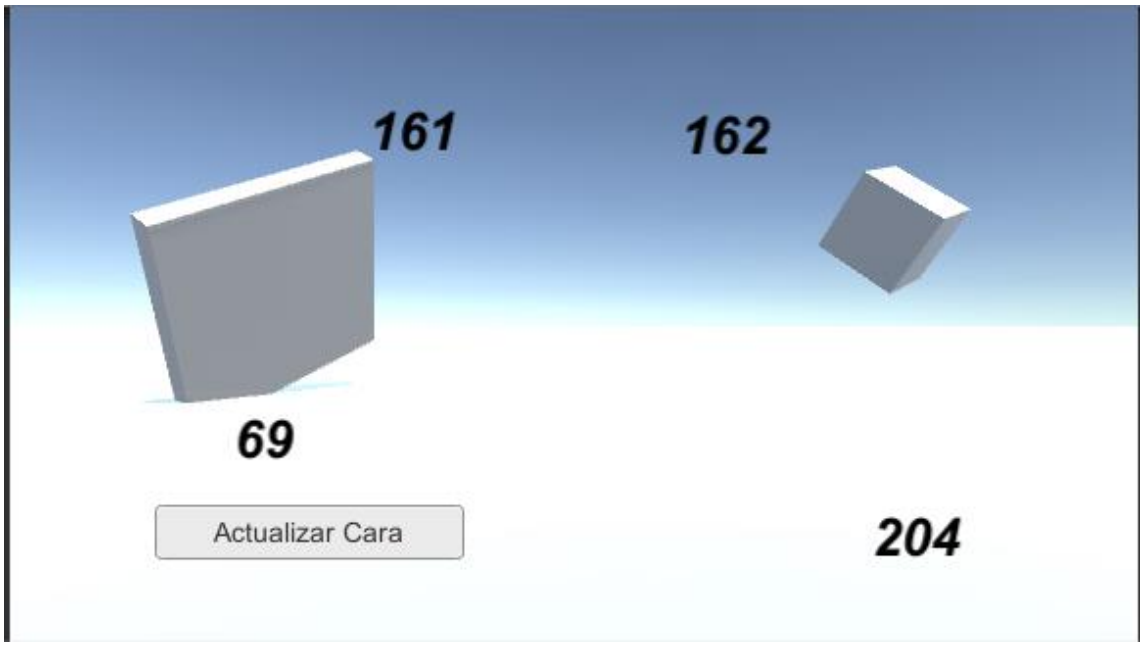


Figura 3. Representación del programa desarrollado en Unity para la obtención de datos enviados mediante TCP.

Estos programas junto con su utilización se explican más detalladamente en el apartado siguiente.

SONRIE Kingdom

El desarrollo del videojuego se divide en dos partes, la primera el modelado y animación mediante Blender de los objetos y personajes que aparecen y la unión de los modelos junto con la lógica del videojuego en Unity.

El diseño de todos los objetos ha sido realizado mediante el programa Blender, también se han utilizado algunos modelos de la asignatura síntesis y animación de imágenes SAI.

En las figuras 4 y 5 pueden verse algunos de ellos.

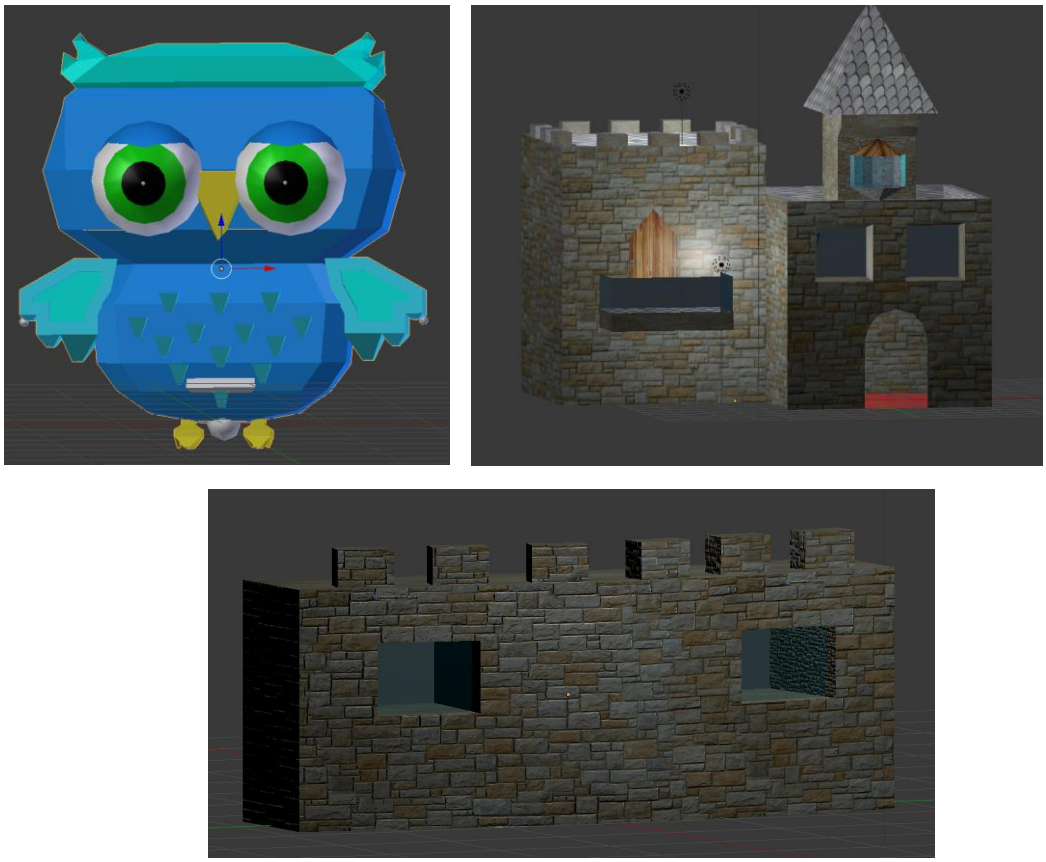


Figura 4. Diseño del personaje búho, la ciudadela principal y la muralla exterior.

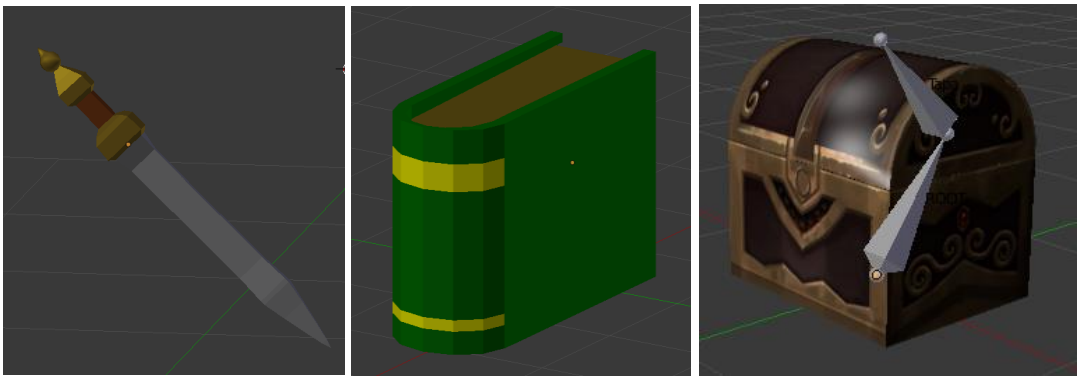


Figura 5. Diseño de objetos varios utilizados en el proyecto.

La parte de desarrollo de Unity es también muy rápida de explicar, pues solo se tuvo que generar un terreno, importar los objetos creados en Blender y colocarlos en él. Además, se creó una capsula para simular al personaje (el usuario nunca podrá verse a sí mismo) en la que se emparentó una cámara y el búho que acompaña a nuestro personaje durante todo el viaje.

Las figuras 6, 7 y 8 muestran la ciudadela creada a partir de los objetos importados de Blender y distintas vistas del videojuego.

Figura 6. Vista superior de la ciudadela.

Figura 7. Vista interior de la ciudadela.

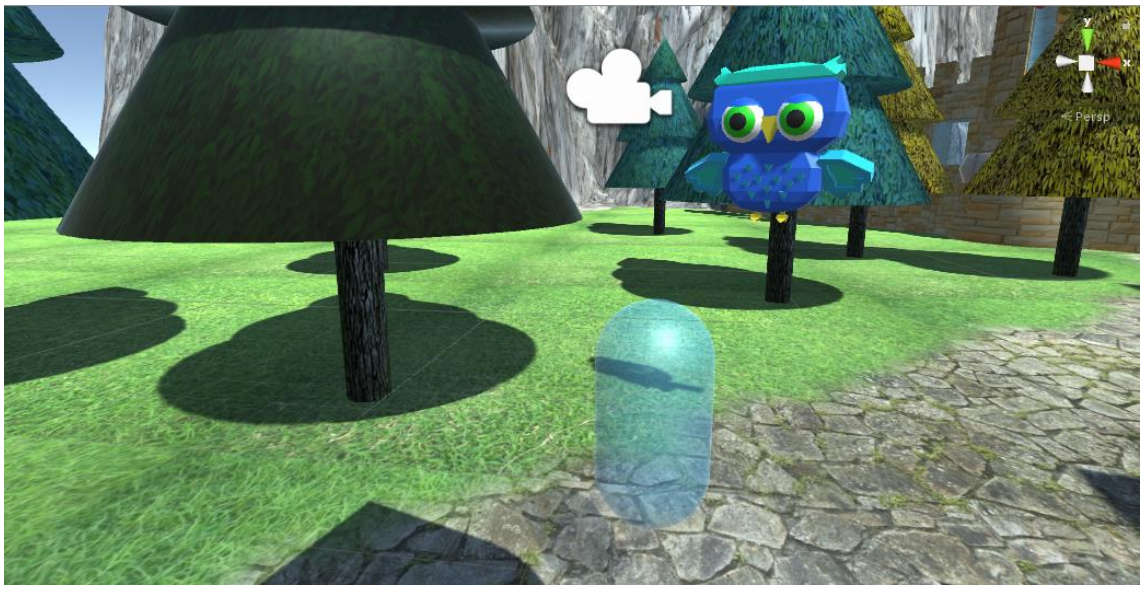


Figura 8. Cápsula del personaje con la cámara.

Figura 9. Vista principal del juego.

Distribución

En este apartado se explicará cómo está distribuido el proyecto en función de los archivos y carpetas. EL proyecto completo puede encontrarse en drive en este enlace: https://drive.google.com/open?id=0B9GBysG-_5EGdFJOMm9ZcDF6Szg

La carpeta principal es SONRIE_Kingdom la cual está dividida en tres subcarpetas, Documentación, Middleware y SONRIE_Kingdom.

Middleware

Esta carpeta está divida a su vez en dos apartados, los programas relacionados con el envío de datos aleatorios y obtenidos por Kinect y los archivos relacionados con el programa de recepción de Unity.

La carpeta Kinect contiene:

Carpeta Jaime: programa utilizado para obtener los puntos y que aparezcan coloreados (en este software se deben aplicar las condiciones del "if" anteriormente comentados para que los puntos aparezcan coloreados).

Carpeta MiddlewareTCP_Vicente: programa que genera los datos aleatorios que simulan la información de los puntos obtenidos por Kinect y los envía mediante TCP a Unity.

Carpeta MiddlewareTCPK: programa que obtiene los valores reales de los puntos a partir de Kinect y los envía mediante TCP a Unity.

Carpeta puntos: Fichero Word que contiene la explicación de las tablas 1 y 2.

Fichero MiddlewareTCP_1: fichero de código C# del programa de Jaime Jarrín. (Hubo problemas a la hora de abrirlo y se decidió dejar ahí para evitar errores).

La carpeta Unity contiene:

Carpeta Script (Unity--Kinect): programa de Unity que recibe el valor de los datos de los puntos (ya sean aleatorios o los verdaderos valores de los puntos).

SONRIE Kingdom

Esta carpeta esta subdividida en dos apartados más, los modelos y diseños relacionados con Blender y el videojuego principal en el que se importaron todos los objetos y se creó el entorno.

La carpeta Blender contiene:

Carpeta Estructura: modelos de los edificios y objetos utilizados para el diseño de la estructura de la ciudadela.

Carpeta FBX: Subdividida en varios apartados. Contiene el modelo de los objetos de Blender exportados a archivos .fbx. (Evita errores a la hora de importar texturas y animaciones.)

Carpeta Imágenes_Fondo: Contiene las imágenes de fondo utilizadas para generar los objetos.

Carpeta Objetos: contiene todos los modelos de los distintos objetos utilizados en la ambientación del videojuego (árboles, decoración…)

Carpeta Personajes: contiene los modelos de los personajes creados (El búho definitivo utilizado en el videojuego es Buho_Extended).

Carpeta Texturas: carpeta que contiene todas las texturas utilizadas en los objetos utilizados.

Fichero Blender to fbx: archivo .doc que explica la mejor forma de exportar los modelos de Blender a archivos fbx para importar posteriormente a Unity.

La carpeta Unity contiene:

Carpeta Sonrie_Kingdom: Videojuego final basado en Unity. En él se encuentran todos los modelos importados a Unity como .fbx. Este programa tiene creado el terreno del escenario y colocados todos los modelos. Además, tiene preparado la cápsula que representa al usuario, con su cámara y el personaje búho que le seguirá durante toda la historia.

Funcionamiento

En este apartado se comentará como se han de lanzar los programas para que funcione el envío entre Visual Studio y Unity. También se comentará la idea principal de juego además de explicar la estructuración de los objetos dentro de este.

Middleware

Para lanzar correctamente los programas de envío, se debe de realizar siempre de la misma manera. Será diferente en caso del Middleware de número aleatorios y en el caso del Middleware con datos de Kinect.

Middleware de envío aleatorio.

Paso 1: abrir el programa de Visual Studio Middleware TCP (alojado en la carpeta MiddlewareTCP_Vicente).

Paso 2: abrir el programa de Unity de recepción (alojado en Script (Unity--Kinect)→Pureba_Middleware→Assets→Scenes→Server).

Paso 3: lanzar el programa Middleware TCP. Esperar hasta que aparezca la ventana de comandos y escriba la siguiente línea "esperando para conexión….."

Paso 4: lanzar el programa de recepción de Unity clicando en el botón "Play".

Paso 5: cada vez que se quiera obtener el valor de los puntos de la cara (valores aleatorios en este caso) se debe clicar el botón actualizar cara.

En la figura 10 se muestra una representación del programa.

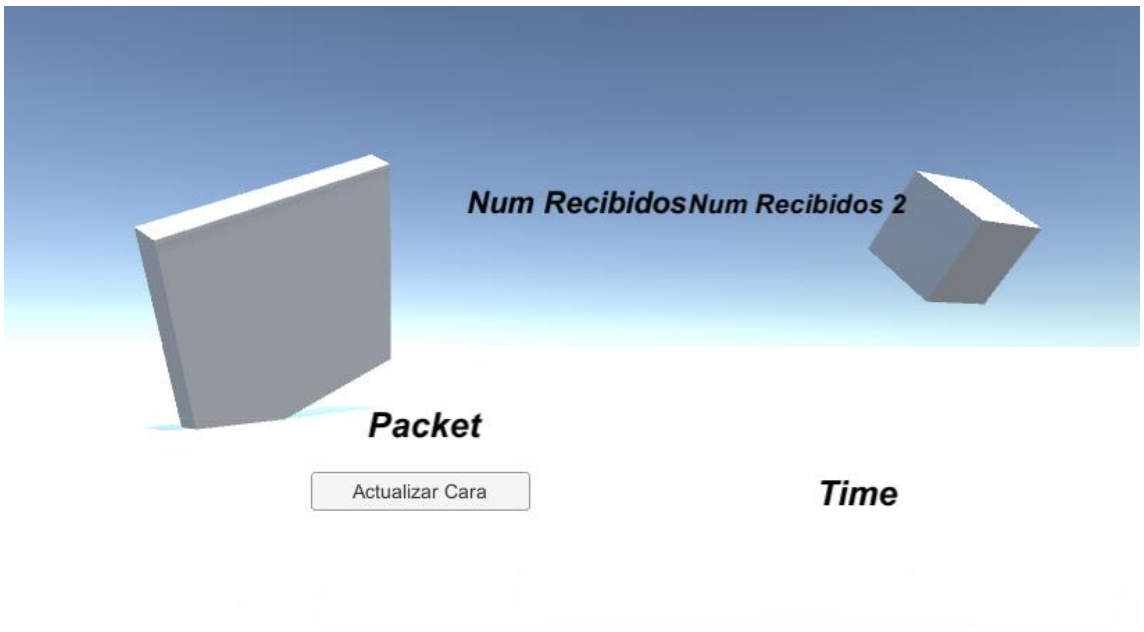


Figura 10. Representación del programa de recepción en Unity.

En la figura 10 aparecen diferentes elementos:

Num Recibidos: Muestra el valor de un paquete recibido (el número del paquete a mostrar depende del código elegido en el Script Boton_Cara)

Num Recibidos2: Muestra el valor de un paquete recibido (el número del paquete a mostrar depende del código elegido en el Script Boton_Cara)

Packet: Muestra el valor del último paquete recibido (Debe ser siempre 69 si no se modifica el código).

Time: Muestra el valor de la variable time. Esta variable aumenta en 1 cada frame.

Las piezas que aparecen en la figura 10 se utilizan para corroborar que el programa no se ha bloqueado. La pieza de la izquierda debe girar constantemente y la de la derecha solo cuando se pulse el botón "Actualizar Cara".

Middleware de envío de datos obtenidos por Kinect.

Paso 1: abrir el programa de Visual Studio KinectFaceTestv2 (alojado en la carpeta MiddlewareTCPK).

Paso 2: abrir el programa de Unity de recepción (alojado en Script (Unity--Kinect)→Pureba_Middleware→Assets→Scenes→Server).

Paso 3: lanzar el programa KinectFaceTestv2. Esperar hasta que aparezca la ventana blanca en la que se mostrará la representación de la cara.

Paso 4: lanzar el programa de recepción de Unity clicando en el botón "Play".

Paso 5: El programa de Unity no funcionará correctamente hasta que se haya detectado una cara por Kinect. Por ello, se debe seleccionar la ventana en blanco y conseguir que Kinect detecte nuestra cara.

Paso 6: cada vez que se quiera obtener el valor de los puntos de la cara (valores reales de Kinect en este caso) se debe clicar el botón actualizar cara.

El programa de recepción de ambos es el mismo, por lo que la explicación de la figura 10 sirve para ambos casos.

SONRIE Kingdom

La idea principal del juego es la siguiente:

El usuario aparece en la entrada de la ciudadela. En este instante se acerca el búho y le cuenta la historia del castillo (una historia por la cuál debe realizar las pruebas para llegar hasta el príncipe/princesa y salvar el reino. Por ejemplo, el castillo está encantado, todo el mundo es de piedra y debe romper el hechizo). El usuario no puede mover al personaje, este se moverá solo. Una vez contada la historia, el personaje empieza andar de forma autónoma mientras el búho le sigue, sin salir de la pantalla, hasta el primer punto de las pruebas. En este momento el búho solicita al jugador que debe levantar las cejas para abrir las puertas del castillo, el usuario tiene 3 intentos para conseguirlo, si no consigue superar la prueba se realiza automáticamente. Se halla superado la prueba satisfactoriamente o no, aparecen estrellas y/o humo felicitando al jugador por haber resuelto la prueba (recomendable utilizar el sistema de partículas de Unity). Una vez abiertas las puertas, continuamos dentro del edificio de la ciudadela. Al entrar, se ve una mesa a la izquierda con una vela. Ahora el búho pide al usuario que sople para apagar la vela antes de que se provoque algún desastre (Segunda prueba). En este caso sucede lo mismo, se tienen 3 intentos y se felicita al usuario de una forma creativa se halla cumplido correctamente o no la prueba. Se continua hacia dentro de la ciudadela y se llega a la cama donde el príncipe/princesa esta tumbado y dormido. En este momento, el búho solicita al jugador que le tire un beso para poder despertarlo y conseguir así romper el hechizo (Tercera prueba). Ocurre lo mismo que en los casos anteriores, se tienen 3 intentos y en cualquier caso se felicita al jugador de una forma llamativa. Por último, una vez se ha despertado el príncipe/princesa se pide devolverle una sonrisa a este personaje y realizar así la cuarta prueba. También sucede lo mismo que en los casos anteriores, 3 intentos y siempre con una felicitación llamativa al final del ejercicio.

Durante todo el recorrido del videojuego, el personaje del búho debe mantener la atención de jugador, dando más información sobre la historia o contando alguna anécdota graciosa entre prueba y prueba.

La estructuración de los objetos dentro de Unity es sencilla y puede verse en la figura 11.

Figura 11. Distribución de los objetos dentro del videojuego en Unity.

Como se aprecia en la figura 11, existen una distribución muy sencilla de los objetos.

Luz: Representa la luz del videojuego.

Buho_Extended_Animated: Modelo del búho importado en Unity.

Jugador: Corresponde con la cápsula que representa al jugador y la cámara del videojuego.

Terreno: Representa el terreno del juego.

Estructura: Contiene todos los objetos importados de Blender que forman la estructura exterior de la ciudadela (Murallas, portón, banderas, torres…)

Ciudadela: Corresponde con los edificios dentro de la muralla y la fuente central. Esta subdivido a su vez en dos, el edificio de la ciudadela (donde transcurre la historia principal) y el edificio de los establos.

> Ciudadela: Contiene los objetos que forman el edificio ciudadela. El apartado decoración corresponde con todos los objetos dentro del edificio.
>
> Establos: Contiene todos los objetos que forman el edificio establos.

Mejoras

El proyecto esta inacabado, faltando juntar el videojuego con el Middleware en el mismo proyecto de Unity. Es una tarea sencilla pues solo hay que adaptar el programa de recepción de Unity al videojuego.

Esta creada la pantalla inicial del juego, pero no se ha aplicado lógica a los botones para que redirija a la opción seleccionada (conectar Kinect, empezar a jugar, opciones…)

No esta aplicada la lógica del videojuego. En este caso solo se debe conseguir que el jugador ande automáticamente hasta los lugares donde realizará las pruebas y pedir que las realice.

Se han de grabar los audios para el sonido ambiente y la voz del búho que cuenta la historia.

No se ha diseñado al personaje príncipe/princesa del final del videojuego.

Kinect no devuelve valores sencillos de tratar, por lo que se deberá generar un algoritmo para posicionar correctamente los puntos y saber si se ha realizado correctamente el ejercicio o no.

La ciudadela no tiene puerta en el videojuego, está separada a parte debido al esqueleto que permite que se habrá. Se debe importar y colocar correctamente.

Como mejoras para el proyecto, se aconseja añadir más objetos al ambiente de la ciudadela para conseguir mantener la atención del usuario mientras se anda de una prueba a otra. También se aconseja amoldar la ciudadela a la historia elegida (en caso de haber caído en un hechizo que haga a todos de piedra, añadir gente de color gris en el patio de la ciudadela...). Además, el juego no cuenta con muchas animaciones que ayuden al jugador a estar pendiente del juego, por lo que se aconseja añadir más efectos visuales como por ejemplo mariposas volando, un caballo en el establo etc.

Por último, aunque la fuente tiene textura de agua, no se ha podido aplicar una “Normal Image” para conseguir la sensación de movimiento. También estaría bien añadir algún tipo de caída de agua a los chorros de la misma.

En el apartado de bibliografía se han añadido diferentes canales de YouTube que se han utilizado para aprender a realizar diferentes efectos o propiedades dentro de Unity y Blender.

Bibliografía

[1] Sampedro, E. (2015). *Sistema de Apoyo Terapéutico con Kinect para niños con parálisis cerebral infantil*. Universidad Politécnica de Madrid.

[2] Jarrín, J. (2016). *Memoria de prácticas externas*. Universidad Politécnica de Madrid.

[3] Sánchez-Rico, D. (2016). *Reconocimiento de movimientos faciales mediante Kinect V2*. Universidad Politécnica de Madrid.

[4] Rendón, E. (2013). *Enrique RA*. [online] YouTube. Available at: https://www.youtube.com/channel/UCLesiY5dN259bglUwLFvvuA

[5] Institute, U. (2014). *Unity Institute*. [online] YouTube. Available at: https://www.youtube.com/channel/UCxDZ3d_9hF55liPZkzpBgMg

[6] Videojuegos, H. (2005). *Hagamos Videojuegos*. [online] YouTube. Available at: https://www.youtube.com/user/juande

[7] YouTube. (2008). *nodician*. [online] Available at: https://www.youtube.com/channel/UC393VScZGe721UVQf3u1NZw

[8] FLA, N. (2017). *Nando FLA*. [online] YouTube. Available at: https://www.youtube.com/channel/UCTrdPtgSvosLY6D1r193QOA