Memoria técnica

Práctica 2

Luis Antonio Ortega Andrés Guillermo Galindo Ortuño
Johanna Capote Robayna

Introducción

El objetivo final de la práctica es facilitar al usuario un entorno de realidad virtual donde pueda inspeccionar una serie de elementos de interés en la Alhambra y que no sería posible hacerlo debido a su fragilidad.

Para ello implementamos un programa en el lenguaje de programación `C#` encargado de comunicar un proyecto en `Unity` con el hardware especializado del `Kinect`.

La escena `Unity` nos permitirá interactuar con una serie de objetos presentes en la escena principal, que junto con la interfaz por gestos nos permitirá realizar diferentes transformaciones sobre la vista de los objetos como rotaciones o acercamientos.

Estructura del proyecto

Como el objetivo de la práctica es la interfaz por gestos que se ha implementado, haremos especial hincapié en aquellas partes enfocadas en esta tarea, únicamente mencionando brevemente la parte de correspondiente a Unity.

Podemos dividir este proyecto en las siguientes partes.

- `Scene`. Escena de `Unity` que aloja los objetos que vamos a inspeccionar.
- `BodySourceManager`.
- `BodySouceView`.
- `Gesture`. Clase cuya finalidad es gestionar los gestos realizados.
- `GestureSegments`. Cada gesto quedará determinado por una serie de "segmentos" que determinarán la transición del mismo.

Escena de Unity

La escena principal consta de 6 paredes que forman una cámara en cuyo centro se mostrará cada uno de los cinco objetos manipulables. Estos objetos se encuentran declarados en un array de `GameObject` llamado `Item`. Además de la cámara principal `MainCamera` también se declaran un objeto de tipo `ligth` que ilumina la cámara.

BodySourceManager

Este archivo es el encargado de realizar la conexión con el Kinect, recibir los datos del Kinect, almacenarlos para su posterior uso, y cerrar la conexión al cerrar la aplicación.

Por ello, al comenzar la escena se abre la conexión con el Kinect. Una vez abierta, cada vez que se actualice la escena, obtiene el último frame tomado por el sensor, y almacena el `Body` asociado en un atributo.

El script estará asociado a un objeto vacío de la escena, lo que permite que otras clases utilicen los datos obtenidos por este.

BodySourceView

Esta es la clase principal y es la encargada de, utilizando los datos del Kinect, y los gestos definidos en la clase `Gesture`, aplicar los cambios en la escena. Para conseguir los datos del Kinect, hay definido un atributo de tipo `GameObject` inicializado al objeto vacío que contendrá la instancia del `BodySourceManager`.

En primer lugar, se definen los siguientes atributos:

- Un array de `GameObject` que contiene cada uno de los objetos a manipular en la escena.
- Dos listas para almecenar la posición inicial de cada uno de los objetos anteriores.
- Un objeto `Gesture` asociado a cada gesto, correctamente inicializado.
- Un atributo de tipo `BodySourceManager`, que proporcionará los datos del sensor.

Ahora, al iniciar la escena, se asigna a cada gesto la función correspondiente que será la que indique la actualización a realizar en la escena al detectar el gesto. Vamos a explicar brevemente el contenido de cada uno de estas funciones.

Las funciones asignadas a los gestos *Next* y *Prev* permiten ciclar sobre todos los elementos. Estas ocultan el objeto actual, restablecen la posición del siguiente objeto a mostrar (en uno u otro sentido dependiendo del gesto), y lo hacen visible.

Las asignadas a cada gesto de rotación, que modifican la componente `rotation` del respectivo objeto rotándolo respecto del eje y la dirección adecuados.

Por último, las asignadas a los gestos de *ZoomIn* y *ZoomOut*, que acercan o alejan el objeto de la cámara en un rango predefinido.

Dicho esto, únicamente nos falta comentar la función `FixedUpdate()`, que es la encargada comunicar los datos obtenidos por el `BodySourceManager` con cada uno de los gestos definidos. Para ello, esta recoge los datos del atributo `BodySourceManager`, asignando al atributo respectivo la instancia almacenada en el objeto de unity con el método `GetComponent`. Ahora, almacena el `Body`, y llama al método update de cada uno de nuestros gestos pasándoles como argumento dicho objeto.

GestureSegment

Esta clase nos permite construir un segmento correspondiente a un gesto. Un objeto de esta clase debe tener implementado la función `update`, que será la encargada de indicar si dado un esqueleto detectado por el `Kinect` cumple una serie de restricciones.

Gesture

Esta clase es la encargada de construir cada uno de los gestos que utilizamos en la práctica además de gestionar y comunicar al resto de estructuras cuando un gesto está siendo ejecutado.

Un gesto está compuesto de un array de `GestureSegments` que corresponden a un segmento de gesto, es decir, un paso intermedio en el gesto.

Esta clase será la encargada de gestionar la transición entre cada uno de dichos segmentos que componen el gesto e indicar cuando han finalizado. Se pueden construir dos tipos de gestos, discretos y continuos.

Estos se diferencian en cuando dan respuesta al resto de componentes del proyecto, un gesto discreto devuelve una "señal" cuando el gesto ha sido completado en su totalidad y luego se reinicia. Sin embargo, un gesto contínuo no se reinicia y continúa enviando la respuesta mientras nos encontremos en su etapa final.

Los gestos que hemos implementado

- `Next` y `Previous`. Gestos discretos que nos permiten navegar entre los distintos objetos que presentamos.
- `ZoomIn` y `ZoomOut`. Gestos continuos que nos permiten alejar o acercar el objeto que estamos viendo.
- `RotateLeft`, `RotateRigth`, `RotateUp` y `RotateDown`. Gestos continuos que nos permiten rotar el objeto que estamos viendo en las distintas direcciones.

Cabe destacar que los gestos continuos se pueden fusionar, es decir, siempre que los movimientos sean compatibles, podemos ejecutar 2 de ellos a la vez, por ejemplo, acercar y rotar el objeto simultáneamente.

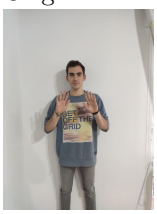
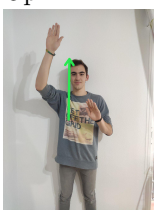
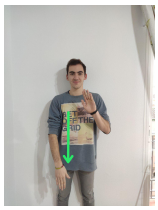
Para gestionar todo lo referente a los distintos gestos, el constructor de la clase nos obliga a pasarle un `string` indicando qué gesto queremos inicializar, con esto la propia clase se encarga de seleccionar los segmentos que le corresponden.

La clase implementa una función `update` que es llamada en cada frame de `Unity`, cuya finalidad es gestionar si se está cumpliendo el siguiente segmento del gesto. Para realizar el mismo tenemos una "ventana" de tiempo entre los diferentes segmentos, de forma que si la excedemos el gesto se resetea y sería necesario volver al segmento inicial del mismo.

Cada objeto de la clase tiene un miembro `EventHandler` que nos permite añadirle funciones con sus respectivos argumentos, de forma que la clase `BodyView` añade las funciones que afectan a la escena `Unity` al miembro del gesto correspondiente.

Veamos cuáles son estos gestos con los siguientes ejemplos que nuestros apuestos programadores han preparado.

Next	Prev
	  

Rotate				
Origin	Right	Left	Up	Down
				

Zoom In		Zoom Out