K2UA

Kinect to Unity Asset

Memoria técnica

Registro software

UNIVERSIDAD POLITÉCNICA DE MADRID

Autores: César Luaces Vela, Martina Eckert

Fecha: Mayo de 2018

Contenido

1. Descripción del programa 2
1.1. Objetivos 2
1.2. Realización 2
2. El lenguaje de programación 3
3. El entorno operativo 3
3.1. El receptor del esqueleto “SkeletonReceiver” 3
3.2. El controlador: KinectReceiver 4
3.3. El amplificador de movimiento: AmplifyManager 6
4. Listado de ficheros 6
5. El diagrama de flujo 6

1. Descripción del programa

1.1. Objetivos

El objetivo principal del programa a registrar “K2UA – *Kinect to Unity Asset*”, es su uso como complemento de Unity 3D, para realizar el procesado de los datos recibidos vía mensaje UDP, de manera que, estos sean utilizados como sistema de control para un personaje implementado como un modelo 3D. En concreto, los datos recibidos son datos de movimiento de 25 articulaciones de un esqueleto humano, capturado por la cámara Kinect y enviado via UDP por el middleware “K2UM – *Kinect to Unity Middleware*”.

La finalidad del conjunto “Middleware & Videojuego” es el uso para fines terapéuticos, en concreto la realización de videojuegos para realizar ejercicio físico, especialmente para personas con diversidad funcional.

1.2. Realización

Se ha generado el *asset* que contiene los scripts que permiten la recepción de datos enviados desde Kinect V2 y el posterior procesado de estos para su utilización como control del videojuego. Para su identificación se ha exportado con el nombre de **KinectAsset.unitypackage**.

Para el acceso a este *asset*, una vez creada una sesión de trabajo en Unity 3D, este debe ser importado como *Custom Package* a la misma. Una vez hecho, en la carpeta `Assets` de la ventana `Project` aparece una nueva carpeta llamada `KinectAsset` con los *prefabs* y *scripts* asociados a este ítem (Figura 1).

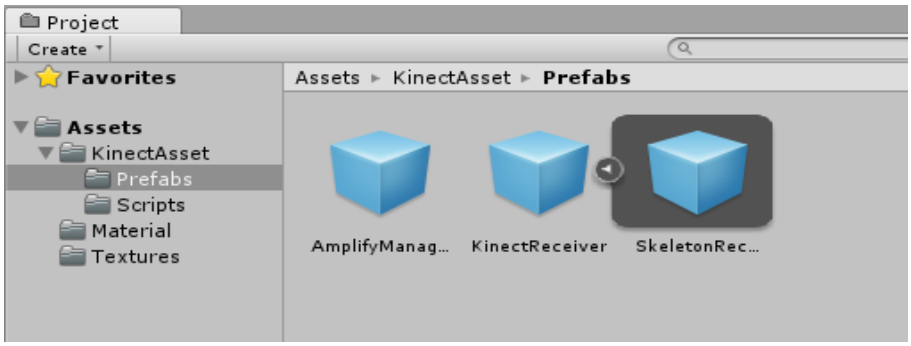

Figura 1. Objetos importados con el asset "KinectAsset.unitypackage"

Unity 3D llama *prefab* a un objeto que se almacena con todas su características y *scripts* asociados, de manera que actúe como plantilla a partir de la cual se pueden crear nuevas instancias al objeto en la escena. Así, para el uso del *asset* que contiene todas las herramientas diseñadas, solamente es necesario arrastrar cada uno de los *prefabs* generados a la escena de trabajo actual.

Tal y como se muestra en la figura 1, este *asset* está compuesto por una carpeta con dos *prefabs* y otra con una serie de *scripts* que estos *prefabs* tienen asociados. El primero de estos *prefabs* se llama *KinectReceiver* y contiene, por un lado, toda la lógica para la recepción y tratamiento de los datos enviados por el *middleware*, y por otro, el esqueleto receptor al que se aplican estos datos. El esqueleto receptor se muestra como otro *prefab* vinculado a *KinectReceiver* dado que su uso siempre se realiza de forma conjunta. El otro *prefab* que compone este *asset*, llamado *AmplifyManager*, contiene toda la lógica necesaria para realizar la amplificación de los movimientos recibidos procedentes del *middleware*. Se facilita como *prefab* independiente de manera que, si no es desea amplificar el movimiento realizado por el usuario captado, no esté presente en la escena.

2. El lenguaje de programación

Para modificar las características en tiempo de ejecución o definir las relaciones entre los objetos que componen una escena, Unity 3D permite la ejecución de scripts codificados en distintos lenguajes de programación. Esta ejecución se realiza a través de *MonoDevelop*, una implementación de código abierto de *.NET Framework*, principalmente diseñada para C#. Así, este es el lenguaje utilizado para el desarrollo del *asset* de recepción de Kinect V2.

3. El entorno operativo

3.1. El receptor del esqueleto "SkeletonReceiver"

El *middleware* envía la información asociada al movimiento captado en forma de mensajes con el nombre de la articulación y la rotación o posición espacial que esta tiene. Para aplicar dicho movimiento en un modelo (avatar), *KinectAsset* utiliza un esqueleto receptor intermedio llamado *SkeletonReceiver*.

En la figura 2 se muestra un objeto *SkeletonReceiver* posicionado en una escena de un proyecto en desarrollo de Unity 3D.

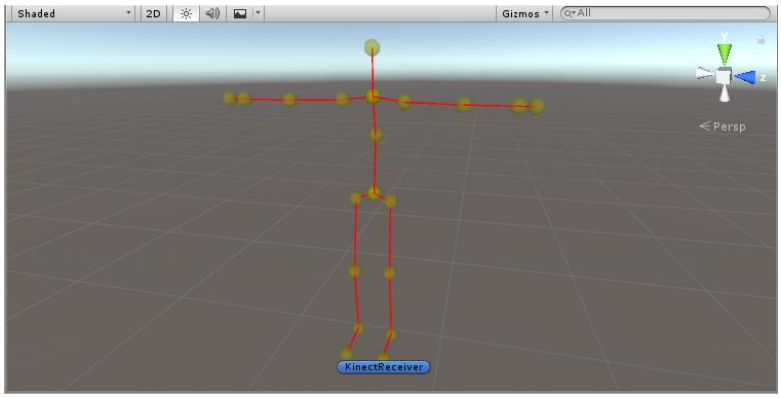

Figura 2. Esqueleto receptor asociado al objeto SkeletonReceiver

Una vez llevada a cabo la tarea de reproducir los datos de movimiento captados por Kinect V2, y transmitidos a través del *middleware,* el siguiente paso es conseguir que estos movimientos se apliquen en el esqueleto receptor de Unity3D y mediante este a un modelo 3D importado.

3.2. El controlador: KinectReceiver

Para realizar esta tarea se tiene en cuenta que, todo modelo 3D que se genere con previsión de ser animado, es decir que se le aplique cierto movimiento de forma independiente para cada articulación, imprescindiblemente tienen que tener algún tipo de esqueleto asociado a él. De esta manera, a través del objeto *KinectReceiver*, es posible seleccionar cada una de las articulaciones que posee a dicho esqueleto y emparentarlas con las articulaciones del esqueleto receptor de Kinect V2.

Así, solo es necesario posicionar el esqueleto receptor de Kinect dentro del modelo 3D importado y desplazar sus articulaciones hasta la posición que ocupan en dicho modelo. Una vez iniciada la aplicación, el sistema de emparentamiento hace que, con cada movimiento del esqueleto receptor, se realice el mismo movimiento en la articulación del esqueleto del modelo 3D a la que está emparentada, y dado que, ambas ocupan la misma posición espacial, se realiza una transferencia óptima del movimiento del esqueleto receptor al modelo 3D importado.

Mediante esta técnica, es posible utilizar el esqueleto receptor implementado, en cualquier tipo de modelo 3D, independientemente de su esqueleto. Se puede desplazar cada una de las articulaciones que posee el esqueleto receptor, y es posible su usarlo en conjunto con cualquier modelo 3D, sin importar el tipo de fisionomía utilizada en su esqueleto interno p.ej. un personaje extraterrestre con articulaciones desproporcionadas como muestra la figura 3.

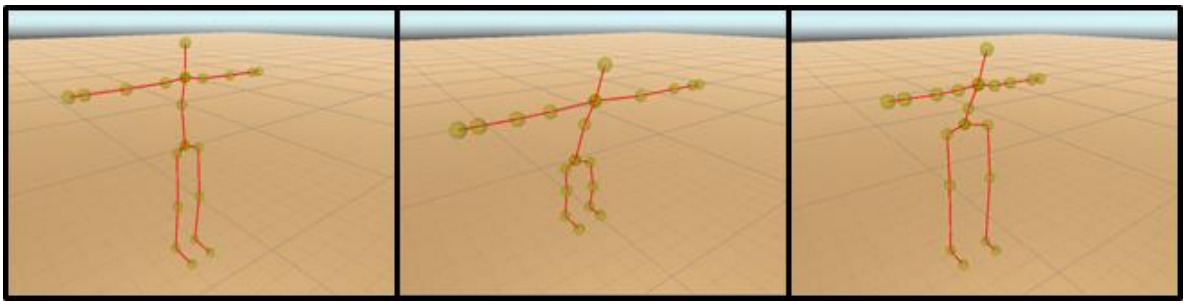

Figura 3. Muestra de las posible posiciones que pueden ser adoptadas por el esqueleto receptor

Para asociar cada una de sus articulaciones a sus equivalentes en el esqueleto receptor, Unity 3D permite la creación de variables públicas durante el desarrollo de un *script*, a las que se puede asociar objetos de la escena arrastrándolos a una casilla creada para ellas en el menú *Inspector* asociado al objeto. Dado que la ejecución de este *asset* depende de una gran cantidad de *scripts*, la tarea de localizar todas las variables que deben de ser asignadas en cada uno de estos *scripts* sería muy compleja para alguien que no conociera su funcionamiento. Así, para simplificar este proceso, se ha utilizado una herramienta de Unity 3D que, mediante código, permite al usuario modificar el *Inspector* asociado al objeto y crear uno propio que se ajuste a sus necesidades de uso.

La figura 4 muestra el *Inspector* inicial que se muestra al seleccionar el objeto *KinectReceiver* en la escena.

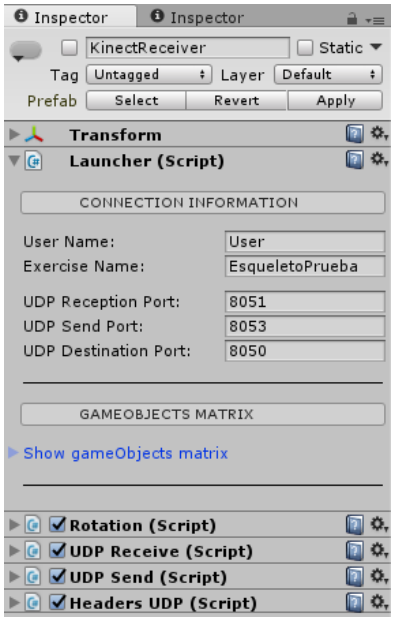

Figura 4. Ventana de inspector asociada a un objecto KinectReceiver

Tal y como se comentaba anteriormente, el *Inspector* de este objeto muestra todos los scripts que tiene asociados. El primero de todos ellos, llamado *Launcher*, actúa como controlador al contener todas las variables públicas a las que el resto de *scripts* acceden y que deben ser asignadas por el usuario de forma previa a la ejecución. También tiene asociado un *script* que modifica su aspecto y comportamiento para simplificar el proceso de configuración y utilización del *asset*.

Visualmente, el *Inspector* asociado al objeto *KinectReceiver*, está dividido en dos zonas. La primera de ellas, situada en la parte superior, permite al usuario introducir la información asociada a la comunicación con el *middleware*. En ella se definen, tanto los campos con el nombre del usuario y del ejercicio, como los puertos asociados a los *sockets* por los que se realiza la comunicación UDP. El nombre del usuario y del ejercicio se envían al *middleware* en forma mensajes de control de clase **nombre de usuario (UN)** y **resultados de ejecución (FR)** respectivamente,

La siguiente zona del *Inspector*, situada bajo la anterior, es donde el usuario puede asociar cada una de las articulaciones del modelo 3D importado a Unity 3D con las de su equivalente del esqueleto receptor.

3.3. El amplificador de movimiento: AmplifyManager

El último *prefab* contenido dentro del asset *KinectAsset* es el *AmplifyManager*. Este objeto tiene asociado una serie de *scripts* que permiten amplificar la rotación llevada a cabo por cualquiera de las articulaciones del esqueleto receptor y aplicar esta rotación amplificada a la articulación equivalente del modelo 3D. Dado que la amplificación directa del ángulo de rotación de una articulación se torna demasiado compleja debido a las dependencias de articulaciones de menor nivel, se opta por el uso de una técnica llamada cinemática inversa.

4. Listado de ficheros

KinectUnityAsset

5. El diagrama de flujo

Los bloques mostrados en el diagrama de flujo (Fig. 6) se dividen en tres categorías que se distinguen mediante el tipo de flecha que tenga asociada:

Sucesión de tareas realizadas por cada uno de los hilos secundarios generados tanto por el *middleware* como por Unity 3D. Las tareas están situadas de forma secuencial, de manera, que pueda apreciarse el orden en el que se realizan y la dependencia que existe entre ellas.

Mensajes que las aplicaciones intercambian durante su conexión, mostrando su procedencia y el direccionamiento interno que se les aplica.

Los distintos flujos de datos creados por las aplicaciones durante su ejecución, especificando, en cada caso, el tipo de canal de comunicación utilizado.

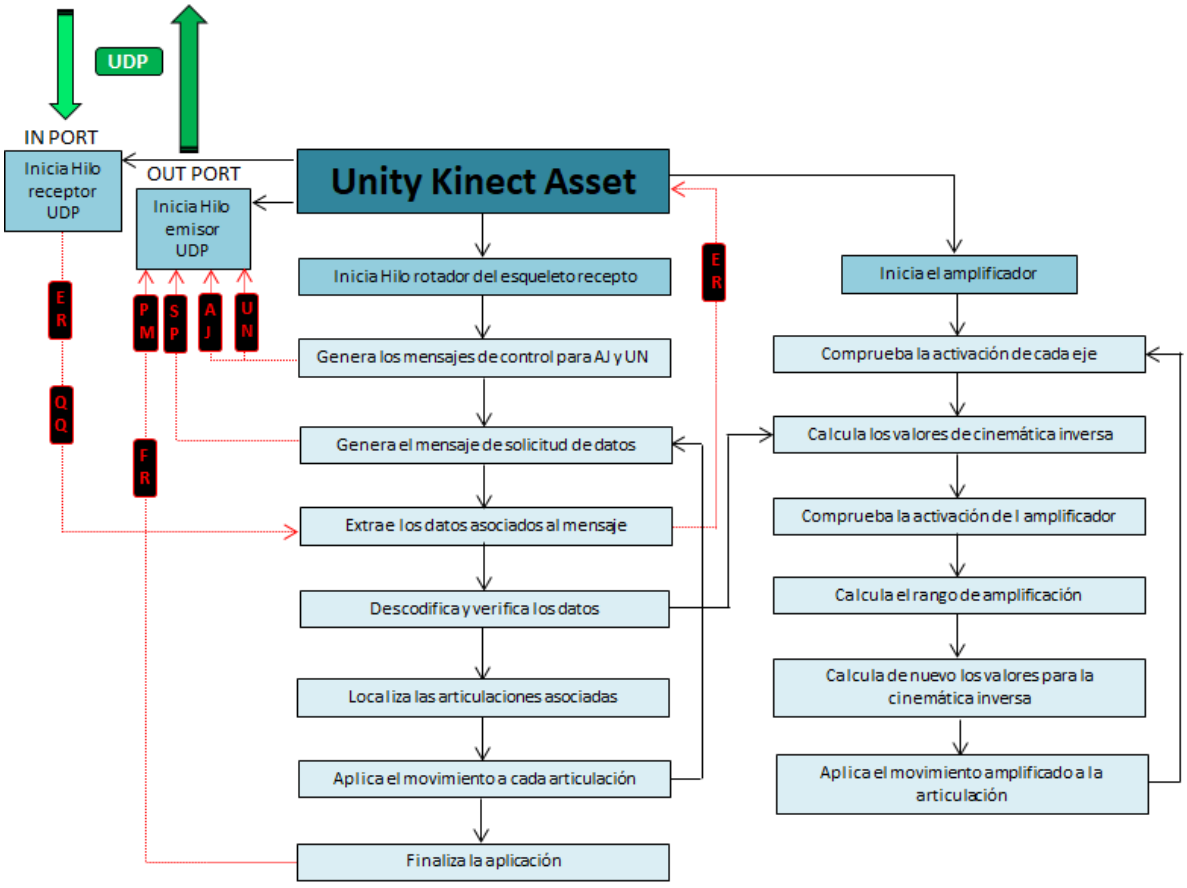

Figura 2. Diagrama de bloques final asociado al middleware K2UM.