K2UM

Kinect to Unity Middleware

Memoria técnica

Registro software

UNIVERSIDAD POLITÉCNICA DE MADRID

Autores: César Luaces Vela, Martina Eckert

Fecha: Mayo de 2018

Contenido

1. Descripción del programa

1.1. Objetivos

El objetivo principal del programa a registrar "K2UM – *Kinect to Unity Middleware*", es haciendo la función de un intermediario entre un hardware (la cámara de captura de movimiento Microsoft Kinect V2) y un software (un videojuego realizado en Unity 3D). Transmite la información captada por la cámara al videojuego, donde es procesada y aplicada a un modelo 3D tipo personaje virtual. Para este fin, el middleware trabaja conjuntamente con un complemento para Unity 3D (K2UA – *Kinect to Unity Asset*) que permite la interpretación de los datos recibidos y su posterior utilización como controlador de los movimientos del modelo 3D.

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

El middleware y el hardware Kinect V2 realizan una comunicación vía USB (*Universal Serial Bus*) mediante la cual el middleware almacena, de forma temporal, los datos asociados al movimiento de un individuo. Tras esto, dichos datos son procesados y empaquetados para su posterior envío al videojuego (realizado en Unity 3D) mediante un socket de transmisión interno. La Fig. 1 visualiza el funcionamiento de conjunto y las tareas realizadas en cada parte.

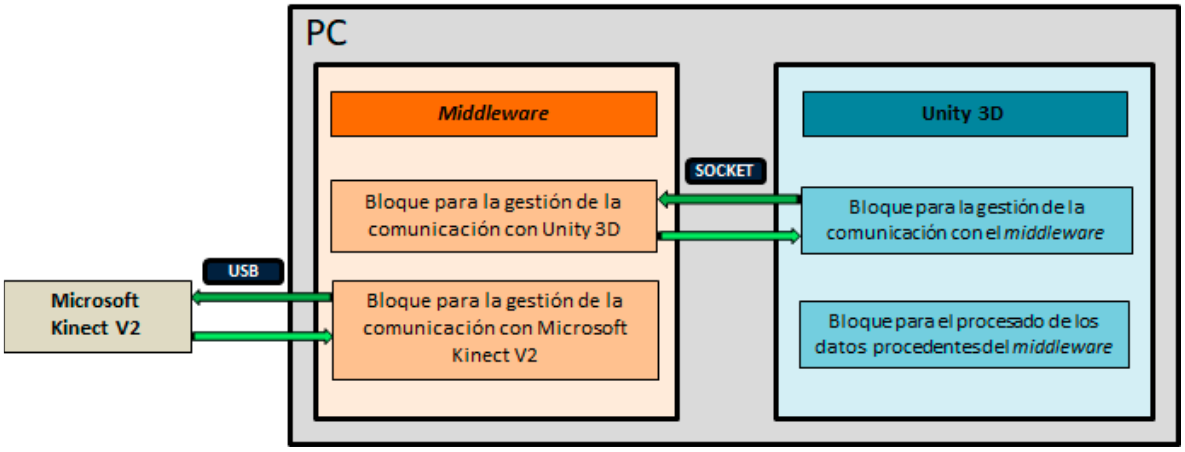

Figura 1. Diagrama de bloques general del sistema

Una vez iniciada la transmisión, el hardware realiza una detección de los usuarios que se encuentran frente a la cámara. Es posible procesar el movimiento de hasta dos de los seis usuarios que Kinect V2 puede detectar, pero desde el videojuego de Unity 3D solo almacena y trata la información relacionada con uno de ellos.

Kinect V2 es capaz de detectar un total de 25 articulaciones, y generar un elemento tipo body, que contiene toda la información asociada al movimiento captado, con una tasa de envío de 30 por segundo, es decir, un paquete cada 33 ms aproximadamente. Además, ya que Kinect V2 proporciona una gran cantidad de datos acerca de la imagen captada, el elemento tipo body generado también contiene mucha más información de interés además de la relacionada con las posiciones y rotaciones de cada articulación. El estado de detección del usuario, es decir, si existe algún error en la detección del esqueleto por parte de Kinect V2, la altura a la que se encuentra el plano del suelo o el estado abierto o cerrado de las manos, son otros datos de interés asociados a este elemento que son tenidos en cuenta a lo largo de la ejecución de la aplicación.

2. El lenguaje de programación

Para el desarrollo de aplicaciones compatibles con el hardware de captura de movimiento Kinect V2, Microsoft proporciona el conjunto de desarrollo oficial (SDK – *Software Development Kit*), versión 2.0. En él, la mayor parte de los recursos que se encuentran son librerías para ser utilizadas en el lenguaje C#, de manera que este es el lenguaje en el que se implementa el middleware. El desarrollo se ha llevado a cabo mediante el entorno Microsoft Visual Studio 2015.

- Microsoft “Kinect for Windows SDK 2.0”
  https://www.microsoft.com/en-us/download/details.aspx?id=44561

3. El entorno operativo

Una vez el middleware ha obtenido los datos asociados al movimiento que Kinect V2 envía, transmite estos datos almacenados al software donde van a ser procesados, en este caso un videojuego programado en Unity 3D. Su función principal es trasladar en tiempo real los movimientos detectados por Kinect V2 a un modelo 3D, haciendo que este se mueva de manera idéntica al usuario captado. Para la realización de esta tarea, el sistema de transmisión debe elegirse

teniendo en cuenta, por un lado, que la velocidad de transmisión de Kinect V2 es de un paquete cada 33 ms aproximadamente, y que, por otro, la información de cada nuevo paquete no está relacionada con el anterior, es decir, que si uno de ellos se pierde o si su información almacenada es errónea, Unity 3D puede completar el movimiento con el siguiente paquete correcto recibido. Así, en el sistema de transmisión a utilizar debe primar la velocidad de transmisión y no el control de la correcta recepción de cada paquete transmitido. Teniendo en cuenta esto, el protocolo de transmisión que mejor se adapta a estas necesidades es el UDP (*User Datagram Protocol*), de manera que, es el que utiliza el middleware para llevar a cabo esta tarea.

En el sistema se manejan dos diferentes tipos de mensajes, que son mensajes de datos y mensajes de control.

- Los **mensajes de datos** contienen en el campo "Tipo" la cadena de texto **DD** y en el campo "Datos" la información asociada al movimiento. También contienen el identificador que Kinect V2 asocia al usuario detectado. De esta manera, en caso de que hubiese más de un usuario detectado por Kinect V2, Unity 3D puede distinguir entre la información recibida de ambos.
- Los **mensajes de control** se identifican mediante la cadena de texto **CC** en el campo "Tipo" y contienen información asociada al funcionamiento del middleware y Unity 3D. Pueden ser enviados tanto desde el middleware a Unity 3D como desde Unity 3D al middleware y engloban tanto mensajes únicamente de control como mensajes de error asociados a la recepción de otros mensajes de control previos o problemas en la conexión con Kinect V2.

En la Fig. 2 se muestra el esquema de transmisión esperado en una comunicación en la que los datos solicitados al middleware desde la aplicación son los asociados tanto a la posición como a la rotación captadas por Kinect V2.

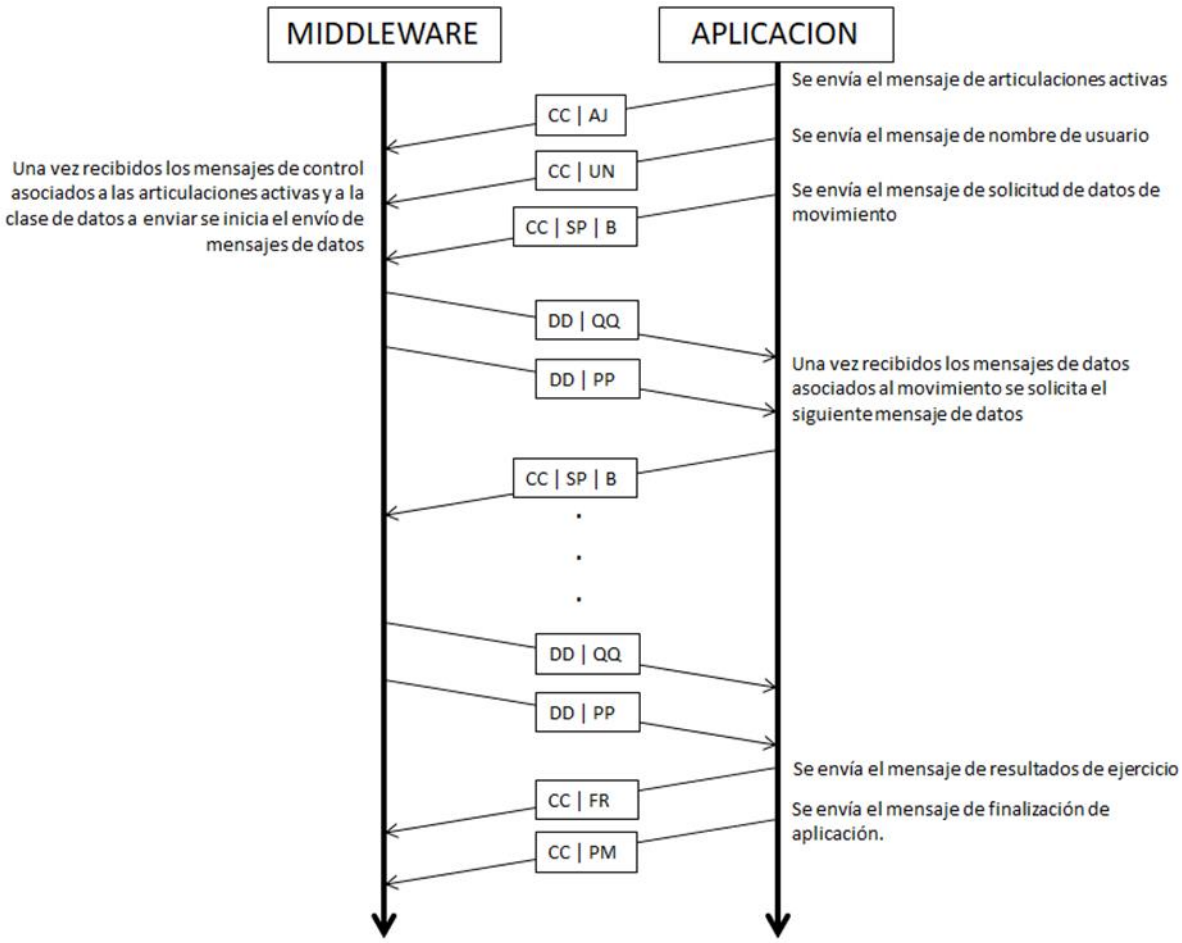

Figura 2. Esquema de transmisión en una comunicación entre el middleware y la aplicación

Para su utilización como aplicación independiente, este *middleware* se implementa como una aplicación de escritorio directamente ejecutable por el usuario. A su vez está compuesta por varias ventanas, cada una de ellas con una funcionalidad distinta. Desde ellas es posible iniciar, pausar o finalizar el *middleware*, tanto como acceder a toda la información de control del funcionamiento del mismo.

En Fig. 3 se muestra la ventana principal con las siguientes funciones:

1. Botones de control principal para inicio, cierre y reinicio. También permiten finalizar únicamente la conexión UDP, liberando los sockets asociados a ella, con la posibilidad de hacer cambios en los puertos configurados sin reiniciar el middleware.
2. Puertos asociados a la comunicación UDP: Los puertos de entrada, salida y destino que utiliza el middleware durante la creación de los sockets UDP. Pueden ser modificados únicamente cuando la comunicación UDP no ha sido todavía iniciada o ha sido finalizada previamente. Para facilitar su uso, tanto el middleware como Unity 3D poseen inicialmente una serie puertos predefinidos de manera que este valor no es necesario que sea introducido.

3. Indicadores de conexión UDP: Indican el estado de conexión a los clientes de envío y recepción UDP.
4. Indicador de mensaje de articulaciones activas: Indica si se ha recibido un mensaje comunicando las articulaciones activas en este envío.
5. Indicador de conexión a Kinect V2: Indica el estado de la conexión entre el middleware y el hardware Kinect V2.
6. Indicador de estado de envío: Indica el estado del envío de mensajes de tipo datos entre el middleware y Unity 3D.
7. Botones de cambio de pestaña: Permiten abrir las ventanas “Información” y “Consola”.

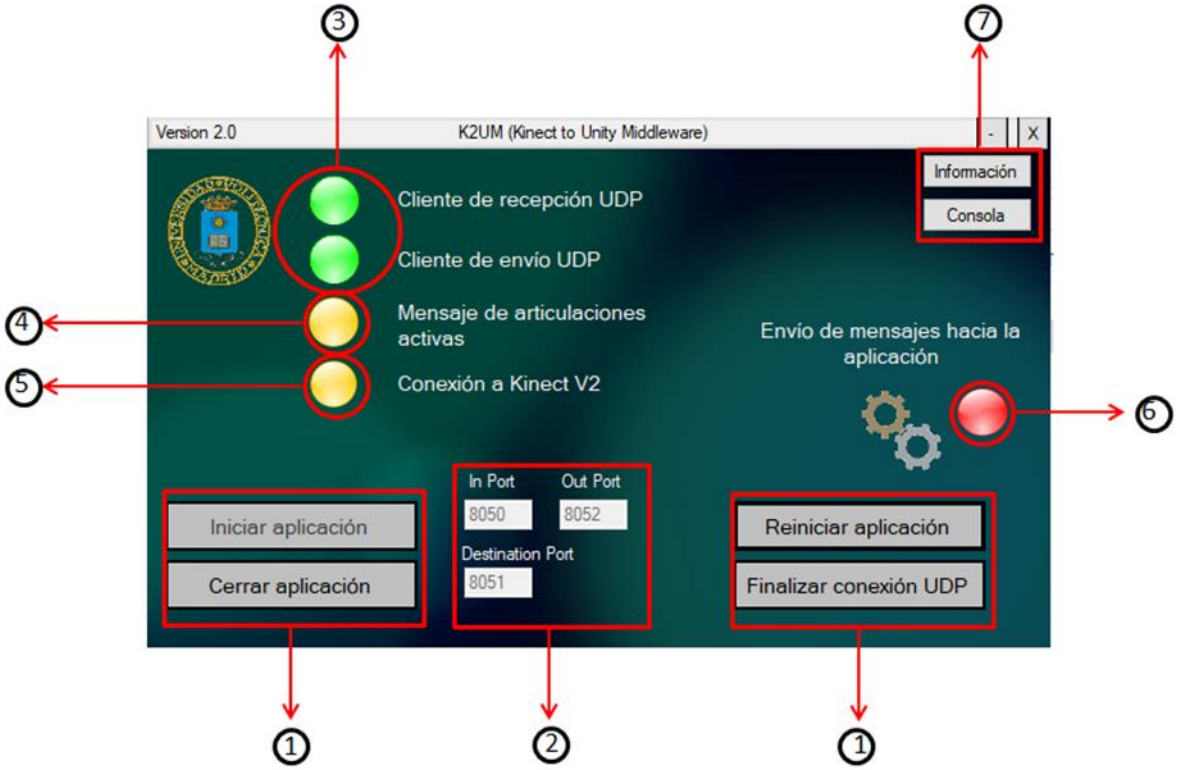

Figura 3. Ventana de controles del interface gráfico del middleware

La ventana “Información” mostrada en Fig. 4 contiene información detallada relacionada con el envío y la recepción de datos vía UDP. Así, permite conocer desde el número de mensajes transmitidos en ambas direcciones hasta el número de errores producidos en este proceso. Desde ella, también es posible, acceder a información de control, como el contenido del campo “Datos” en los últimos mensajes enviados y recibidos, el número de articulaciones activas enviadas en esta transmisión, los datos relacionados con el último usuario conectado o los resultados obtenidos durante la ejecución de la última aplicación conectada.

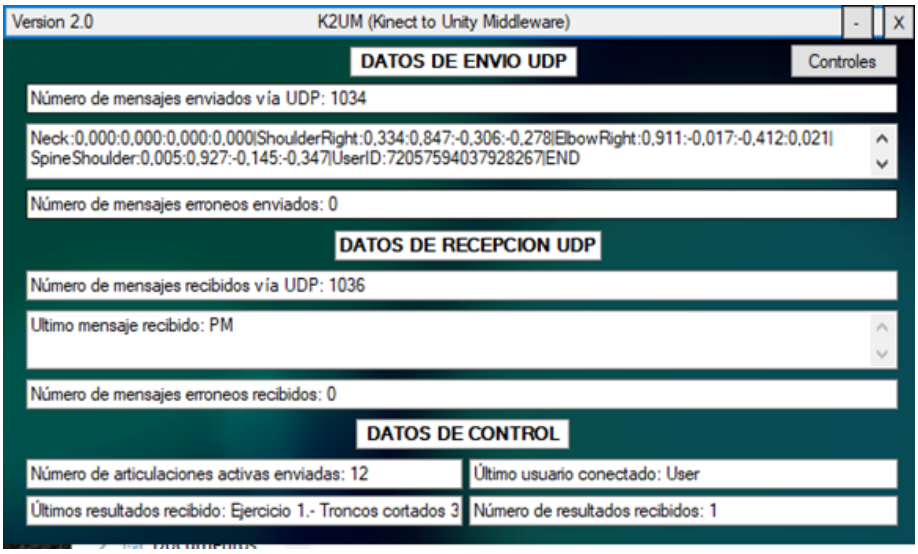

Figura 4. Ventana de información del interface gráfico del middleware

La ventana "Consola" mostrada en Fig. 5 informa acerca de los procesos internos que realiza el middleware durante su ejecución. Cada vez que uno de los hilos lanzados realiza un proceso considerado de interés para el usuario, se añade un mensaje al texto mostrado en esta ventana, indicando que tipo de proceso se ha realizado y el momento en el que se ha llevado a cabo.

Toda la información mostrada en esta ventana se almacena en un archivo temporal llamado **tempOut.info** que, dado que actualmente no es necesario almacenar ningún dato del proceso de transmisión realizado, es eliminado cuando se finaliza la ejecución del middleware.

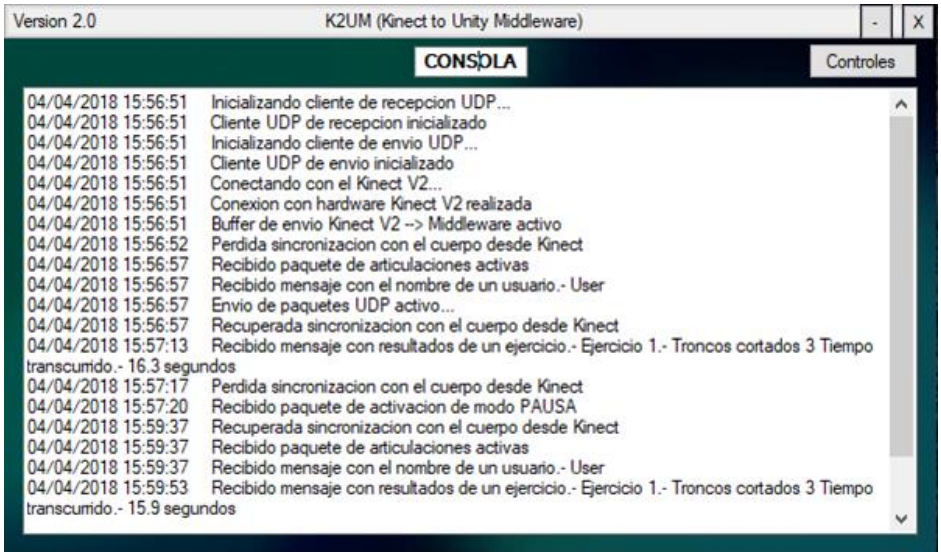

Figura 5. Ventana de consola del interface gráfico del middleware

4. Listado de ficheros

Middleware K2UM Fuentes.zip

Middleware K2UM Ejecutable.zip

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.

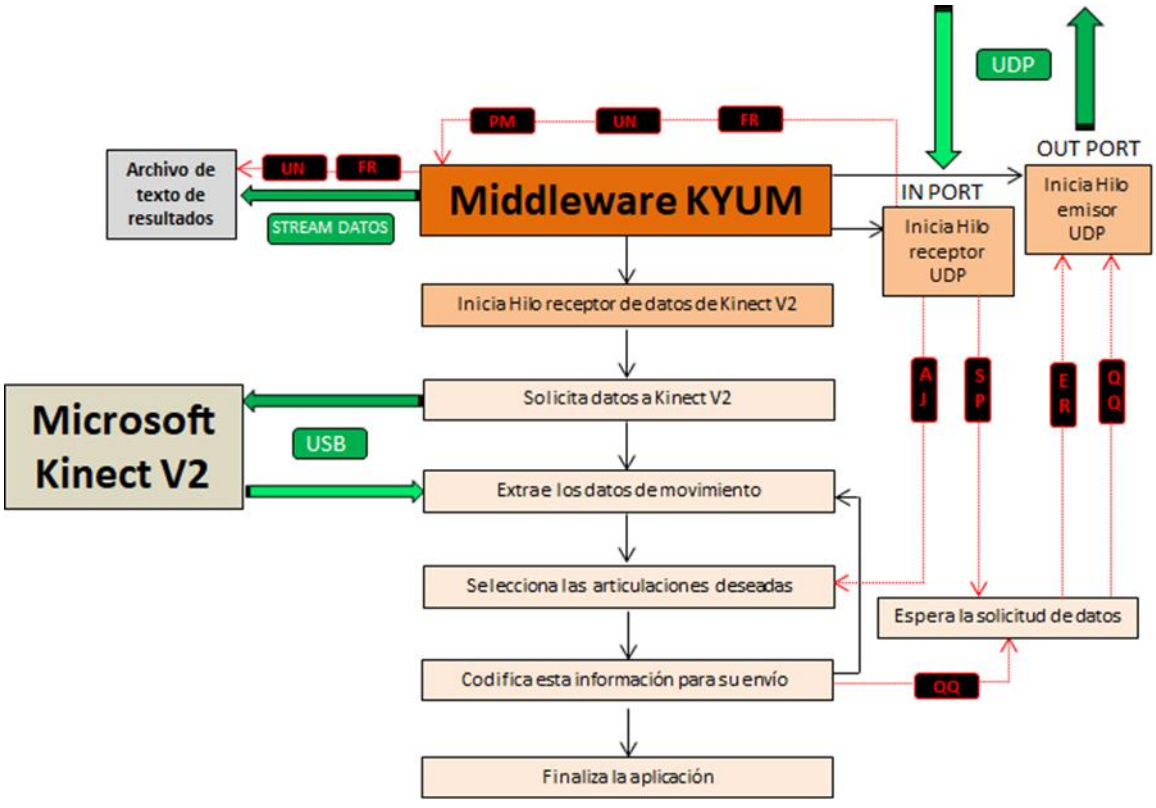

Figura 6. Diagrama de bloques final asociado al middleware K2UM.