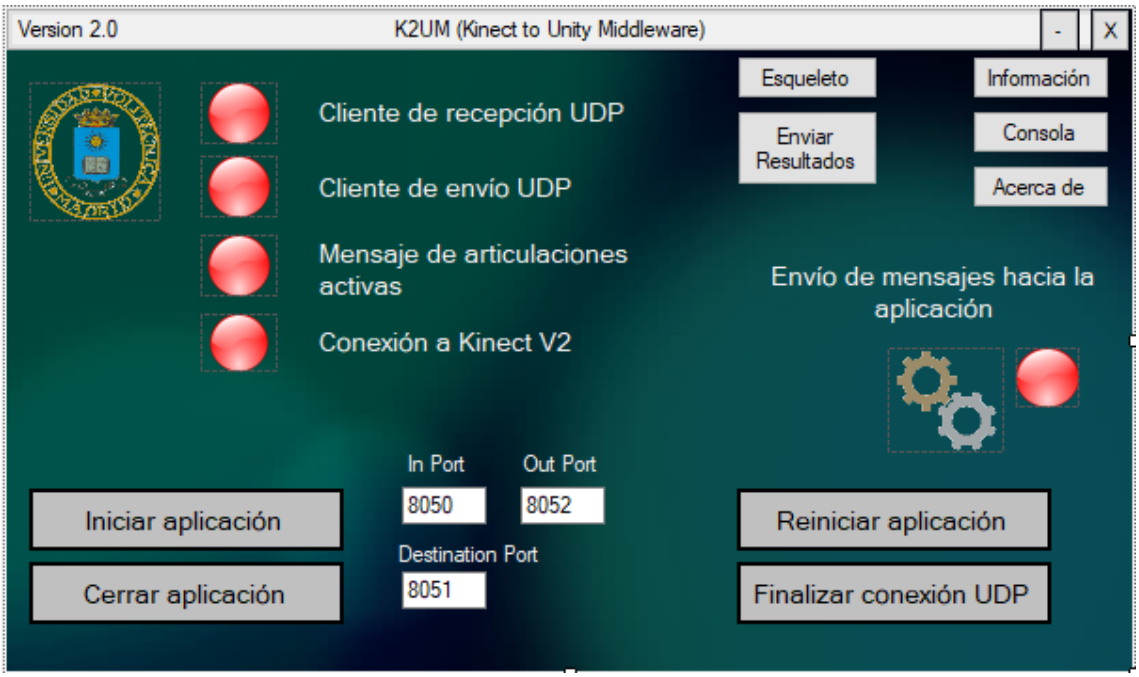
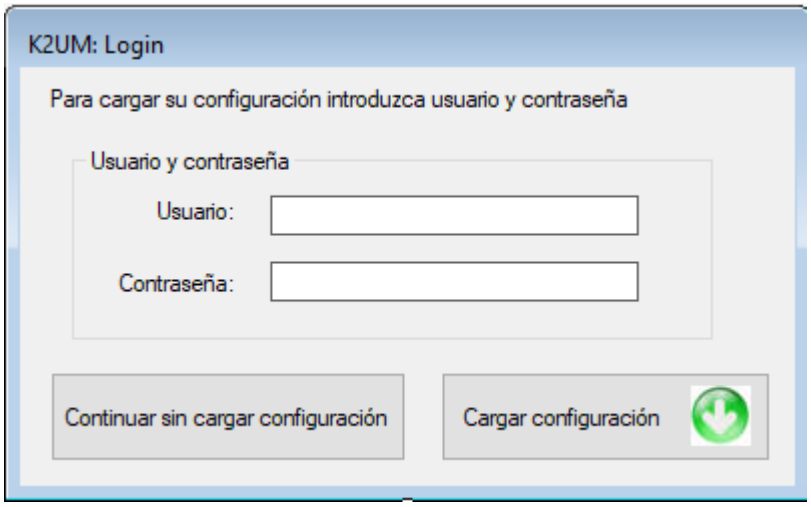
K2UM – Versión Abril 2019

Daniel Iglesias

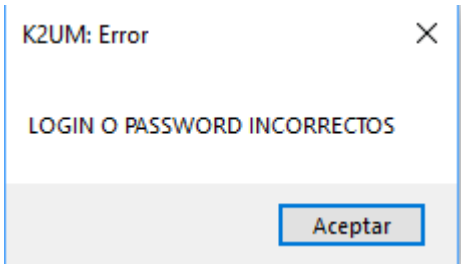


1. Inicio de la aplicación.

Al arrancar la aplicación se muestra la ventana que pide el usuario y contraseña.



Hay dos opciones: continuar sin descargar ningún fichero de configuración o introducir un nombre de usuario y contraseña. Si este usuario no está registrado en la web o su contraseña no es correcta saltará un aviso.

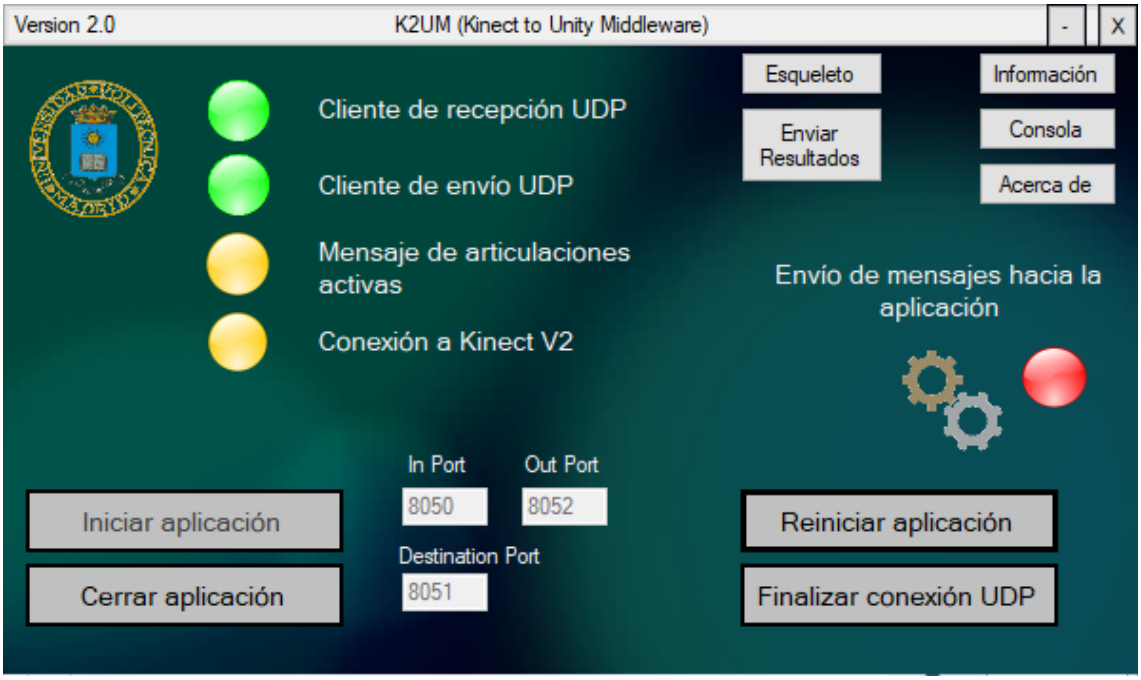


Esta parte del código se desarrolla en la clase **LoginForm.cs** la cual contiene las variables y métodos que se usan para la comunicación con la página web.

Al pulsar el botón *Cargar Configuración* se ejecuta el método **cargarConfiguración()** el cual envía un mensaje de solicitud de datos a la web mediante el protocolo HTTP. Esta parte de del código se ha copiado tal cual del middleware Chiro y se ha adaptado al actual.

La configuración que el middleware recoge de la web se guarda en un fichero de texto de nombre el login del usuario seguido de la extensión .txt. Este fichero se guarda en la carpeta donde esté instalado el programa en la subcarpeta **"\k2umfiles\users-settings**".

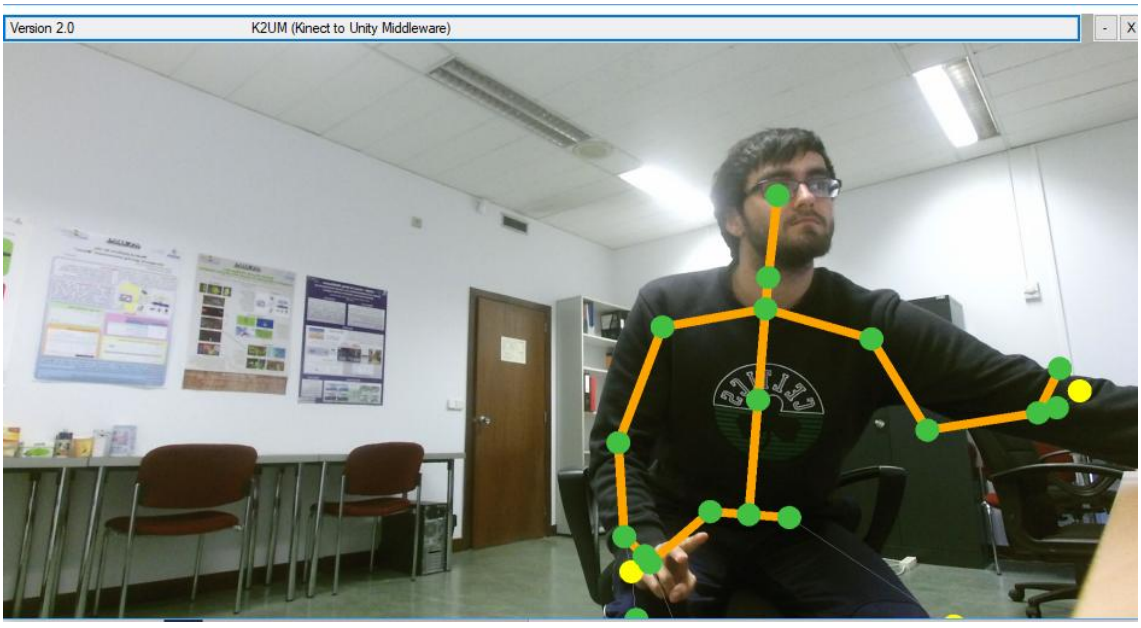
Al pasar el proceso de autenticación se abre la ventana principal gestionada por el formulario diseñado en **Form1.cs**. Al pulsar el botón *Iniciar aplicación* comienza la comunicación entre el middleware y la Kinect.



2. Ventana Kinect Skeleton

Una vez iniciada la conexión se pueden usar el resto de funciones del middleware como la información de los mensajes enviados o la ventana de depuración para el seguimiento de la aplicación.

Al pulsar el botón *Esqueleto* se abre la ventana independiente que muestra la imagen de la cámara de Kinect y dibuja si detecta algún cuerpo el esqueleto del mismo.



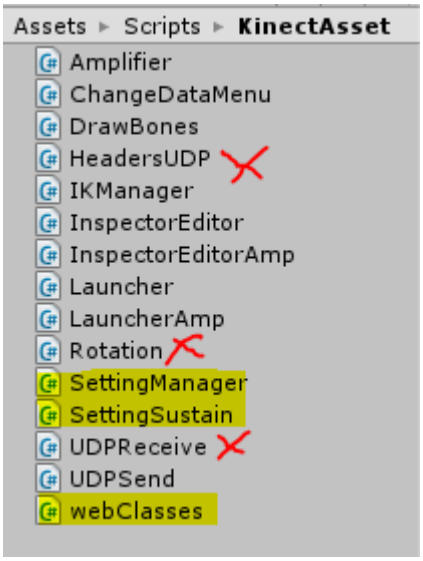
El seguimiento del código utilizado comienza en la clase **Form1.cs.** Al pulsar el botón *Esqueleto* se crea un hilo secundario llamado **skeletonThread** el cual

activa la ventana de la cámara. El resto del proceso se realiza en la clase **Form2.cs**.

En ella se toman los datos que manda la Kinect (30 frames por segundo) y se diferencian si son datos de imagen (**ColorFrame)** o datos de los esqueletos (**BodyFrame**). Ambos datos se combinan para formar la imagen resultante. Kinect puede captar hasta 6 esqueletos distintos al mismo tiempo.

3. Kinect Asset

Una vez iniciada la conexión con Kinect se puede establecer conexión con Unity a través del código incluido en el asset. Al iniciar una escena que contenga el objeto kinectReceiver del asset de Kinect, que puede ser por ejemplo el menú principal, se inicializan todas las variables para la conexión UDP con el middleware y se envía la solicitud de datos de configuración al mismo. Este proceso involucra los scripts **udpSend.cs y settingManager.cs**.



En amarillo, scripts nuevos añadidos. En rojo scripts existentes modificados.

Para que este proceso sea posible se han implementado dos nuevos tipos de mensajes de control, mensaje de solicitud de configuración (*Setting Request* – SR) y mensaje de configuración enviada desde el middleware (*Setting Sent* - SS).

Desde la clase **SettingManager.cs** se solicita el archivo de configuración tras cargar la escena. El middleware recibe la solicitud de configuración (SR), la cual contiene información del juego cuyos datos solicita y tras esto deserializa la información correspondiente al juego solicitado, contenida en el fichero de configuración descargado. La información del juego se serializa de nuevo en formato JSON y se envía a Unity (SS) donde la clase **UDPReceive.cs** se encarga de recibir el mensaje, comprobar qué tipo de mensaje es y colocarlo en la cola de mensajes recibidos. La clase **Rotation.cs** desencola el mensaje y extrae la cadena de texto en formato JSON. Seguidamente la clase **SettingManager.cs**

deserializa la información del JSON obteniendo la configuración de todos los ejercicios del juego y los guarda en un diccionario de ejercicios. Este diccionario está definido en la clase **SettingSustain.cs** que se asignará como componente a un gameObject "*settingObject"* que no se destruirá al cambiar de escena y permitirá recoger la configuración para cada ejercicio.

También se ha desarrollado un ejemplo de script para extraer la configuración y al terminar el ejercicio mandar los resultados, como ayuda para los desarrolladores de los juegos. Este script se llama **exerciseManager.cs.**

Cuando el usuario decida puede enviar los resultados de los ejercicios obtenidos hasta el momento usando el botón "Enviar Resultados" del middleware. Los resultados que se van llegando se guardan en un fichero de texto de nombre el login del usuario seguido de la extensión .txt y almacenado en el directorio del middleware dentro de **"\k2umfiles\users-settings**".

El componente **SettingManager.cs** debe estar presente en el objeto *kinectReceiver* en la escena en la que se decida solicitar el archivo de configuración, por ejemplo el menú del juego.
Todas las clases que estructuran los datos de configuración y resultados que se envían a la web se guardan en el fichero **webClasses.cs**.