E.T.S. de Ingeniería de Sistemas Informáticos

Universidad Politécnica de Madrid

Memoria de Práctica Externa

<**Empresa**>

<Jiliang Qiu>
<Sistemas de información>
<4º>
<03/07/24>

Introducción.

La práctica desarrollada se ha basado en la programación de un programa escrito en CSharp para Kinect con el objetivo de mandar datos a Unity. Kinect es un dispositivo que permite a los usuarios interactuar y controlar la consola sin necesidad de contacto físico, mando o similares.
El programa desarrollado e implementado inicialmente por César, y más tarde optimizada por Philip con gran esfuerzo, trata de capturar los "Joints" (articulaciones). Estos son puntos colocados en cierta posición del cuerpo para que la cámara del dispositivo Kinect sea capaz de capturar la información y saber en qué posición del espacio está colocada la articulación permitiendo al personaje de Unity simular el movimiento capturado. Debido a que los datos capturados por la cámara no son enviados directamente, es necesario previamente transformarlos en formato json mediante la "serialización". Este método consiste en transformar los objetos a un formato transferible, en este caso string json.

El primer objetivo de la práctica ha sido entender el código creado por César y cambiado posteriormente por Philip. A continuación, ha sido modificado para poder mandar datos desde el reloj bangle.j, trabajo realizado en el anterior semestre, a unity mediante el programa desarrollado.

ÍNDICE DE CONTENIDOS

1. ACTIVIDADES Y COMPETENCIAS DESARROLLADAS.

El objetivo principal del proyecto es enviar los datos desde el programa, escrito en Csharp, a Unity. Antes de empezar, se ha analizado el funcionamiento del programa para posteriormente hacer un diagrama de flujo del mismo. Esto se llevó a cabo con dos objetivos en mente. En primer lugar, para ser capaz de entender el código sobre la cual se está trabajando. En segundo lugar, permitir a las personas que vayan a trabajar en este proyecto en el futuro implementar y personalizar el código sin dificultades.

Una vez entendido y realizado el diagrama de flujo, se empezó a manipular el código para cumplir el objetivo, es decir, enviar datos recolectades desde Kinect a Unity.

El primer intento fue introducir “corazón” como Joint. Sin embargo, dio error, ya que al Joint “heart” (como se llama mi objeto) le faltaban algunos parámetros, los ejes XYZ. Para arreglar dicho error, se planteó una herencia. En esta se creó un Joint “heart”, aunque este podía tener los parámetros XYZ con valor NULL. Aun así, no se consiguió solucionar el problema.

A continuación, se intentaron implementar dos soluciones poco astutas. La primera de ellas, fue poner un parámetro extra en los Joints para introducir el dato cardíaco. Este método permitió enviar los datos correctamente (el acto de enviar), aunque surgió otro problema adicional, se asignaba a cada Joint latidos del corazón, lo cual no tiene sentido. El segundo método, en vez de enviar la información de los latidos del corazón junto con los demás Joints, se enviaba con la configuración del usuario. Por lo tanto, este método tampoco fue eficaz, ya que solamente se actualizaba una vez y, además, se enviaba al lugar incorrecto.

Posteriormente, se solucionó creando un paquete adicional, que sería enviado utilizando la misma función usada para enviar los Joints ya existentes en el programa.

Este método solucionó prácticamente el envió de datos. A pesar de ello, surgió un nuevo problema. Unity al capturar los paquetes, es capaz de escribirlos, pero no interpretarlos. Por un lado, el código de recepción de Unity no se esperaba un dato Heart, por lo que no era capaz de interpretar los datos correctamente resultando que el muñeco no se podía mover. Para arreglar este error, se exploraron 3 posibles soluciones.

En primer lugar, se intentó que corazón fuese un Joint extra, lo cual no fue la solución correcta.
En segundo lugar, se intentó crear un método para detectar y hallar exclusivamente el paquete enviado “Heart”, ya que, como se ha mencionado anteriormente, los paquetes se enviaban por separado. Así, se ha creado una función para que el paquete que llega, si empieza por “Heart”, recoja la información y elimine el paquete. Sin embargo, este método tampoco fue el correcto, debido a que si el paquete era analizado por una función no podía ser analizado por otra posteriormente.
En tercer lugar, se intentó añadir una función para extraer la información de los otros Joints. Esta consistía en hacer una condición if que al encontrarse con “Heart” lo leía con una función específica para ello, y cuando no lo era lo leía con la función ya implementada.

Finalmente, se conectó el reloj bangle.js al programa y pudo capturar y enviar los datos a Unity.

2. TÉCNICAS, TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS.

En conjunto, el programa fue desarrollado en tres entornos diferentes. La primera parte, Kinect, fue desarrollada en Microsoft Visual Studio, la segunda en Unity y la tercera en bangle.js. Las dos primeras fueron desarrolladas en lenguaje Csharp.

Se han utilizado 2 herramientas diferentes: Bangle.js y Kinect.

Visual Studio es un entorno de desarrollo que permite la integración con windows y otros dispositivos como la kinect en la que se usa en la práctica, este entorno es compatible entre muchos lenguajes entre ellos Csharp.

La Kinect es un dispositivo que controla el movimiento desarrollado por Microsoft, en un principio diseñado para ser un accesorio para la consola Xbox 360. Su principal característica es la capacidad de permitir a los usuarios interactuar con la consola y los juegos sin necesidad de usar un controlador físico, utilizando únicamente sus gestos y comandos de voz. La tecnología de la Kinect se basa en varios componentes:

- La cámara uno de los componentes claves para el desarrollo de la práctica en la que captura imagenes del entorno a color
- Sensor de profundidad es otro componente clave ya que utiliza una combinación de cámaras de infrarrojos para saber la distancia entre los objetos así pudiendo crear un mapa en 3D de donde se obtienen los datos de los ejes X Y Z.
- Microfono: este componente no es tan importante para nuestro proyecto ya que solo queremos capturar el movimiento.
- Procesador: este componente si que es clave ya que procesa los datos recogidos a tiempo real e interpreta los movimientos.

Cuando ejecutamos el código la kinect tiene varios apartados como home camera settings

Otra plataforma que se ha utilizado en práctica ha sido Unity la que es una plataforma de desarrollo de software que se utiliza principalmente para crear videojuegos, aunque también se usa en otros tipos de aplicaciones interactivas, como simulaciones, realidad aumentada y realidad virtual. Esta plataforma se escribe en Csharp lo cual es positivo con la integración con nuestro programa pero va a ser un problema con el reloj bangle.js

Bangle.js es un dispositivo wearable que se destaca por su versatilidad y su enfoque en la programación y la creación de aplicaciones para dispositivos portátiles. Se trata de un reloj inteligente de código abierto, diseñado específicamente para los entusiastas de la programación, los desarrolladores y cualquiera interesado en la creación de software para wearables.

Lo que distingue a Bangle.js de otros relojes es su capacidad de ejecutar JavaScript directamente en el dispositivo. Esto significa que los desarrolladores pueden escribir y cargar código directamente en el reloj así proporcionando más versatilidad, lo que lo convierte en una plataforma ideal para experimentar con el desarrollo de aplicaciones, juegos y otros proyectos interactivos.

Además, Bangle.js viene con una amplia gama de sensores ya instalados, como acelerómetro (nos ayuda a medir el movimiento en el espacio), monitor de frecuencia cardíaca, GPS y muchos más. Estos sensores permiten a los desarrolladores aprovechar diversas funcionalidades para crear aplicaciones que pueden rastrear la actividad física, medir la frecuencia cardíaca, registrar datos de movimiento etc.

3. PROBLEMAS PLANTEADOS Y PROCEDIMIENTOS SEGUIDOS PARA SU RESOLUCIÓN.

En cuanto a problemas surgidos, han sido mencionado en el apartado uno y, a continuación, van a ser desarrollados en profundidad.

En primer lugar, fue necesario entender el programa, ya que no había suficiente información sobre el funcionamiento del programa. Esto fue debido a que en todas las memorias anteriores, de las personas que han trabajado sobre esta práctica, se explicaba el procedimiento y los problemas surgidos, pero no el funcionamiento del código. Para solucionar esto, contacté personalmente vía Teams a Philip, una de las personas que trabajó en este proyecto. Esto me ayudó a entender más el programa escrito en Microsoft visual studio para la parte de kinect. Consecuentemente, elaboré un diagrama de flujo que representa los pasos del programa.

Después de terminar el diagrama de flujo, se procedió a enviar un dato a Unity. Inicialmente, solo quería enviar cualquier información a Unity, ya que el objetivo era poder enviar un dato para posteriormente sustituirlo.

El primer intento fue introducir “corazón” como Joint. Sin embargo, dio error, ya que al Joint “heart” (como se llama mi objeto) le faltaban algunos parámetros, los ejes XYZ. Para arreglar dicho error, se planteó una herencia. En esta se creó un Joint “heart”, aunque este podía tener los parámetros XYZ con valor NULL. Aun así, no se consiguió solucionar el problema.

A continuación, se intentaron implementar dos soluciones poco astutas. La primera de ellas, fue poner un parámetro extra en los Joints para introducir el dato cardíaco. Este método permitió enviar los datos correctamente (el acto de enviar), aunque surgió otro problema adicional, se asignaba a cada Joint latidos del corazón, lo cual no tiene sentido. El segundo método, en vez de enviar la información de los latidos del corazón junto con los demás Joints, se enviaba con la configuración del usuario. Por lo tanto, este método tampoco fue eficaz, ya que solamente se actualizaba una vez y, además, se enviaba al lugar incorrecto.

Posteriormente, se solucionó creando un paquete adicional, que sería enviado utilizando la misma función usada para enviar los Joints ya existentes en el programa.

```
Task.Run(() => {

        universalKinect.frameArrived += (sender, args) => {

            string json = JsonConvert.SerializeObject(args.dictionary, Formatting.Indented);

            universalKinect.Bindings.LastJSON = json;

            udpSender.messageQueue.Add(json);

           // udpHeart.messageQueue.Add("Heart: 123"); //mi segundo paquete de datos

        };

    });
```

Este método solucionó prácticamente el envió de datos. A pesar de ello, surgió un nuevo problema. Unity al capturar los paquetes, es capaz de escribirlos, pero no interpretarlos. Por un lado, el código de recepción de Unity no se esperaba un dato Heart, por lo que no era capaz de interpretar los datos correctamente resultando que el muñeco no se podía mover. Para arreglar este error, se exploraron 3 posibles soluciones.

En primer lugar, se intentó que corazón fuese un Joint extra, lo cual no fue la solución correcta.
En segundo lugar, se intentó crear un método para detectar y hallar exclusivamente el paquete enviado “Heart”, ya que, como se ha mencionado anteriormente, los paquetes se enviaban por separado. Así, se ha creado una función para que el paquete que llega, si

empieza por “Heart”, recoja la información y elimine el paquete. Sin embargo, este método tampoco fue el correcto, debido a que si el paquete era analizado por una función no podía ser analizado por otra posteriormente.
En tercer lugar, se intentó añadir una función para extraer la información de los otros Joints. Esta consistía en hacer una condición if que al encontrarse con “Heart” lo leía con una función específica para ello, y cuando no lo era lo leía con la función ya implementada.

```
public IEnumerator receiveData()
{
    string data;
    while (true)
    {
        // Si la cola de mensajes ha sido creada con éxito y tiene mensajes pendientes
        if (messageQueue != null && messageQueue.Count > 0)
        {
            // Desencola el mensaje
            data = messageQueue.Dequeue().ToString();
            // Si el mensaje comienza con "Heart:", procesa el ritmo cardíaco
            if (data.StartsWith("Heart:")) // encuentra corazon
            {
                ProcessHeartData(data);
            }
            // Si no, procesa los datos de las articulaciones
            else
            {
                ProcessJointData(data);
            }
        }
        // Pausa la ejecución y continúa en el próximo frame
        yield return null;
    }
```

Finalmente, se logró encontrar la solución al envío de datos, y solamente faltaba obtener los datos del reloj para poder enviarlos. Este último objetivo fue el más complicado, de hecho, no se ha podido cumplir.

Para intentar cumplir dicho objetivo se intentó abarcar de cuatro maneras diferentes.

En primer lugar, se intentó integrar 32feet.Net, una librería diseñada para facilitar comunicación entre dispositivos inalámbricos. Este método falló, debido a que no detectaba al reloj bangle.js, mientras que mi dispositivo móvil sí. Una de las razones que se encontró fue que bangle.js es un dispositivo LE (Low Energy).

En segundo lugar, se probó con InTheHand.BluetoothLE, una librería especializada para dispositivos de Low Energy, pero tampoco fue capaz de detectar el dispositivo.

En tercer lugar, se intentó integrar mi proyecto hecho en HTML durante el anterior semestre en el menú de Kinect. Esta vez se detectaba el dispositivo y, una vez integrado en una

pestaña nueva se podía conectar con mi dispositivo, pero esto generó otro problema. Al implementar mi programa, recibía los datos correctamente en JavaScript, pero no se podía transformar a Csharp. Esto impedía enviar los datos a Unity. Para solucionarlo, se intentó usar el WebViewer, pero sin éxito.

Por último, al no lograr solucionar el problema anterior, se intentó abordarlo de otra manera. Se encontró que HTML tiene una interacción innata con Unity, por lo que se trató de resolver de esta manera. Sin embargo, no se pudieron obtener resultados, ya que había que transformar el programa Unity en web, lo cual resultó en dos problemas. Por un lado, debido que el programa no está hecho para web, no fue posible. Por otro lado, al intentar transformarlo daba errores desconocidos.

4. APORTACIONES EN MATERIA DE APRENDIZAJE.

4.1. RESPECTO A LOS CONOCIMIENTOS ADQUIRIDOS.

En el transcurso de mi práctica con Chsarp Java, HTML y Bangle.js he adquirido conocimientos valiosos más allá de la programación pura. He mejorado mi capacidad de resolución de problemas, aprendiendo a abordar desafíos complejos y a encontrar otras maneras de afrontar los problemas y a pensar out of the box. La experiencia en el desarrollo web me ha enseñado sobre prácticas de diseño y accesibilidad.
A lo largo de mi trayecto, he mejorado mi habilidad para buscar información de manera efectiva y eficiente. También he aprendido a trabajar un poco en un proyecto que se trabaja en diferentes entornos (lenguajes)

4.2. RESPECTO A LAS COMPETENCIAS ADQUIRIDAS.

Durante mi experiencia trabajando con CSharp Java, HTML y Bangle.js, he adquirido una sólida comprensión de la programación orientada a objetos a través de Java y Csharp debido a la compresión de los Joints. Mi capacidad para desarrollar aplicaciones y soluciones utilizando estos lenguajes me ha permitido solucionar las dificultades que ha habido.

5. CONCLUSIONES.

Trabajar con Csharp Bangle.js, HTML y JavaScript ha sido una experiencia enriquecedora que ha ampliado mis habilidades en el desarrollo de aplicaciones interactivas y dispositivos de bluetooth. Esta experiencia ha fortalecido mi base de conocimientos y mi capacidad para adaptarme a entornos cambiantes, dándome una base sólida para seguir creciendo en un campo apasionante y en constante evolución como es el desarrollo de aplicaciones y dispositivos tecnológicos.

He aprendido que pasar de entornos puede parecer fácil, pero no lo es.

6. BIBLIOGRAFÍA

https://docs.unity3d.com/es/530/Manual/HTMLcodetoloadUnityWebPlayercontent.html

https://banglejs.com/

https://es.wikipedia.org/wiki/Serializaci%C3%B3n

https://stackoverflow.com/questions/22668912/get-data-from-bluetooth-device-in-c-sharp

https://learn.microsoft.com/es-es/dotnet/csharp/