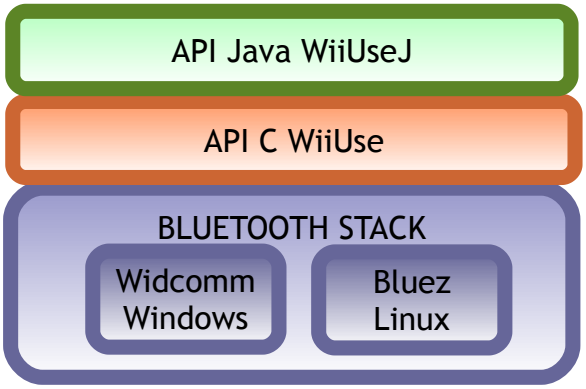
La API WiiUseJ

WiiUseJ es una API Java para el acceso al periférico Wiimote de la consola Wii de Nintendo. Se basa en una API C llamada **WiiUse** que accede directamente al bluetooth stack de nuestro sistema (como BlueSoleil, Widcomm o Bluez). La diferencia de WiiUsej con otras APIs Java reside en que no se basa en el paquete javax.bluetooth (implementación del estándar JSR-82), con lo cual resulta ser un mecanismo más eficiente para acceder al mando al estar implementada sobre un lenguaje de bajo nivel como es C.

La librería WiiUse en C está disponible para plataformas Windows y Linux (formatos .dll y .so).



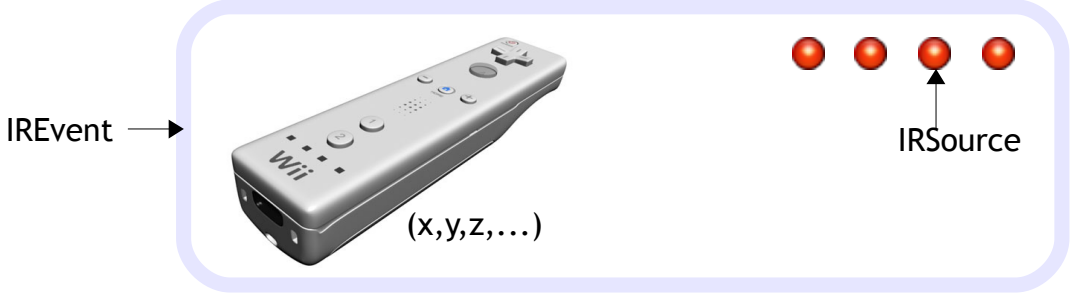
Para que WiiUseJ acceda a la API C WiiUse, tenemos que copiar los archivos de esta última librería (.dll si estamos en Windows o .so si estamos en Linux) en la carpeta principal de nuestro proyecto Java.

Funcionamiento de WiiUseJ

El aspecto que nos interesa del mando es la recepción de fuentes de luz infrarroja. El funcionamiento de WiiUseJ en este aspecto se detalla a continuación.

A un nivel bajo, podemos considerar las clases **IRSource** e **IREvent** (fuente infrarroja y evento infrarrojo, respectivamente).

- **IRSource** representa a **una luz infrarroja** detectada por el mando, vista como un punto bidimensional con un tamaño asociado.
- **IREvent** representa un evento infrarrojo formado por **todos los IRSources** (puntos de luz infrarroja) que estén dentro del radio de acción del Wiimote. Estos puntos se almacenan en un array. En los objetos de la clase IREvent se almacena también la **posición absoluta del Wiimote** (coordenadas x, y, z) **y otra información relevante**, como por ejemplo, si el mando se encuentra por encima o por debajo de los puntos de luz.



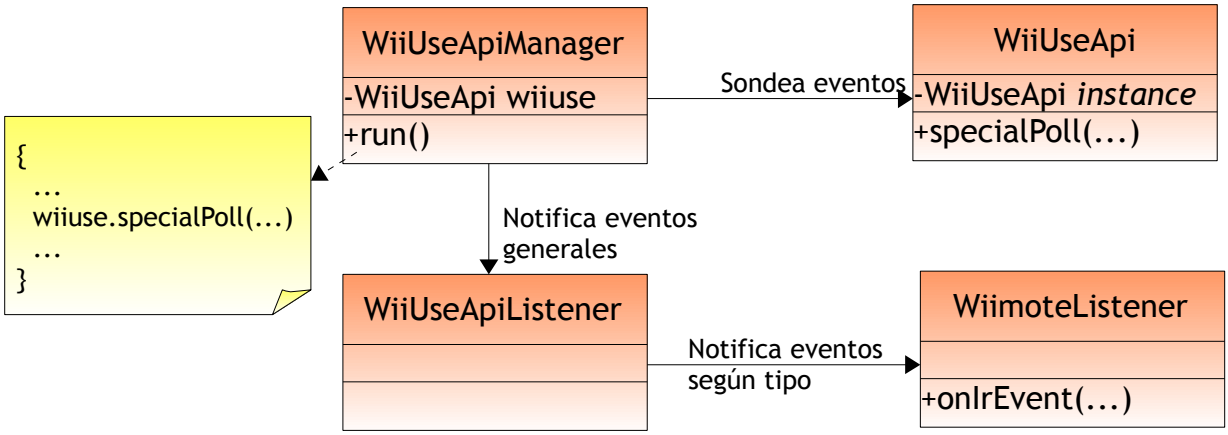
La librería WiiUse C recoge los eventos IREvent a bajo nivel y los comunica a la API Java. En esta última, la librería WiiUse se representa mediante la clase **WiiUseApi**, un singleton usado para llamar directamente a los archivos .dll o .so desde Java.

La clase que maneja a la librería WiiUse se llama **WiiUseApiManager**, que hereda de la clase Thread y se encarga de lanzar el "polling" de eventos, en su método run().

En la API WiiUseJ existen dos tipos de "listeners", representados por las dos interfaces siguientes:

- **WiiUseApiListener**: se encarga de escuchar los eventos que recoge WiiUseApiManager. La clase que implemente esta interfaz se encargará de llamar a los métodos adecuados dependiendo del tipo de evento escuchado. Si detecta que el evento escuchado es infrarrojo, llamará al método onIrEvent(IREvent arg) definido en la siguiente interfaz.
- **WiimoteListener**: la clase principal del proyecto tiene que implementar esta interfaz, que es la que define las acciones a llevar a cabo en la aplicación concreta cuando se producen los eventos recogidos por WiiUseApiListener. Define el método onIrEvent(IREvent arg).

Todo esto queda reflejado en el siguiente diagrama:



El método **onIrEvent(IREvent arg0)** accede a la información del evento infrarrojo, pudiendo tratar directamente las coordenadas del mando con respecto a las luces infrarrojas para usarlas como queramos.

Guía de usuario

¿Cómo realizar finalmente nuestra aplicación? Los pasos a seguir son los siguientes:

1. **Copiar** los archivos de la librería WiiUse (.dll si estamos en Windows o .so si estamos en Linux) en la carpeta principal de nuestro proyecto Java.
2. **Crear** una clase (**MyListener**) que implemente el interfaz WiimoteListener (en concreto, lo que nos interesa es el método onIrEvent).
3. En la clase **main**: **Wiimote[] wiimotes = WiiUseApiManager.getWiimotes(X, true);**
   - Esto detecta los mandos que haya. Se encarga de lanzar la hebra de WiiUseApiManager. Wiimote es una clase que implementa el interfaz WiiUseApiListener, así que lo que devuelve es un array de listeners del WiiUseApi.

- El parámetro booleano "true" hace que se produzca un sonido la primera vez que se detectan los Wiimotes.

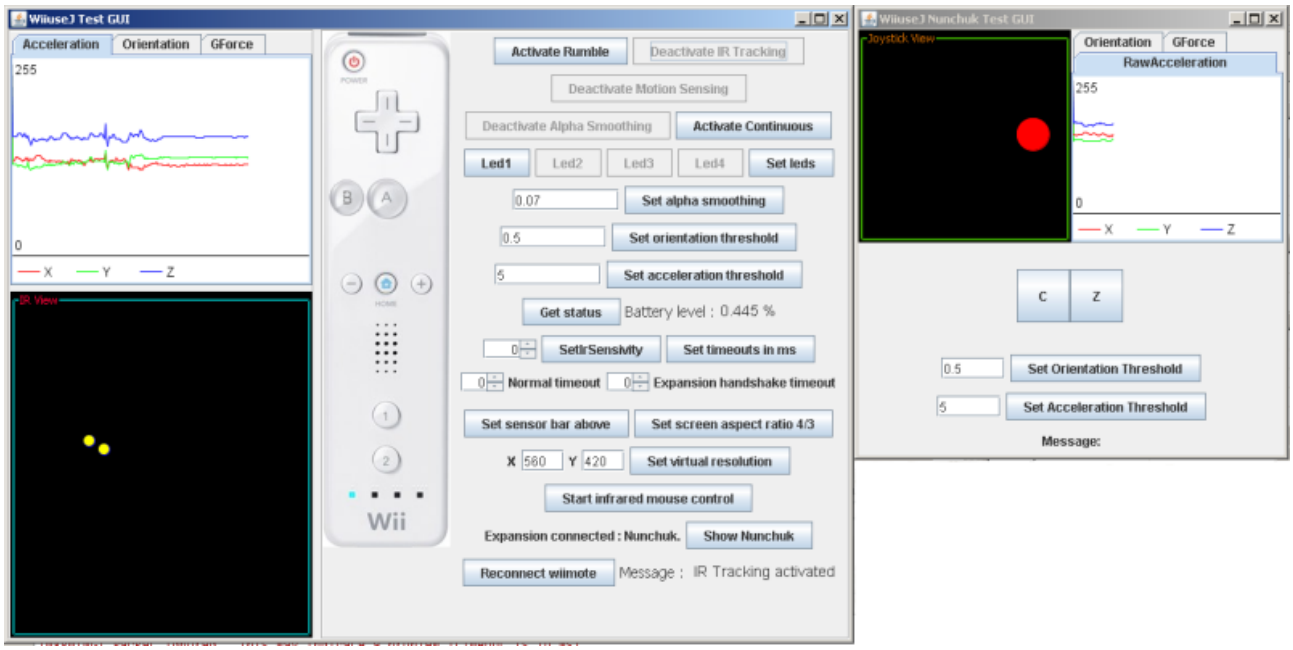
4. En la clase **main: wiimote[0].addWiiMoteEventListeners(new MyListener());**
   - Registra el primer Wiimote con nuestro listener implementado en el primer paso. Cuando el mando lance un evento infrarrojo, sucederá lo que hayamos implementado en el método onIrEvent(IREvent arg0) de esta clase.

Finalmente, se detallan algunos atributos de la clase IREvent para ilustrar la información más relevante que se puede obtener de ella.

IREvent
-IRSource[] IRPoints -short indexPoints -int x -int y -float z ... -short *WIIUSE_IR_ABOVE* -short *WIIUSE_IR_BELOW* -short *NB_POINTS* ...
+getXXX();

Pruebas

Para probar la API, ésta incluye clases que implementan una interfaz gráfica. Esta GUI se lanza desde la clase Main que viene con la API, o ejecutando **wiiusej.jar**.



Referencias

Todas las fuentes de las APIs, wiki, información general, etc. se puede encontrar en **http://code.google.com/p/wiiusej/**.