UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

# *Estudio, diseño e implementación de apps: "wereables"*

**autor: Saúl Micó Guerrero**

**tutor: Jordi Linares Pellicer**

**titulación: Grado de Informática**

**ERT: E.P.S.A.**

**curso:4º**

**Convocatoria de defensa: *Junio 2015***

**INDICE**

# 1.-Introducción

Hoy en día, el uso de diferentes pequeños dispositivos electrónicos se ha vuelto muy natural y cotidiano, utilizándose para todo tipo de labores, haciéndonos mas productivos, como por ejemplo nuestro teléfono móvil.

Esos dispositivos, si los incorporamos a alguna parte de nuestro cuerpo, como por ejemplo la muñeca o el cuello, y a la vez los utilizamos de un modo constante y con alguna función especifica, reciben el nombre de wereables.

La palabra wereable tiene una raíz inglesa cuya traducción significa "llevable" o "vestible" , y no es mas que una referencia al hecho que los llevamos como cualquier complemento mas.

Parece que se da por hecho que los wereables, serán utilizados en un futuro a corto plazo ya que podemos ver como en labores concretas nos hacen muy eficientes, cosa muy demandada actualmente.

Al tener la oportunidad de disfrutar de una beca de investigación en el laboratorio de informática, pude trabajar con las GoogleGlass, un wereable muy innovador, y aunque es un prototipo aun en una temprana fase, no cabe duda que dispositivos similares harán su irrupción con fuerza dentro de poco, con lo que centre mi investigación en ellas, y en sus 2 características principales (mas adelante se puede encontrar un estudio mas a fondo del dispositivo y su sistema operativo):

1.- la micro pantalla que poseen donde puedes ver la información rápidamente solo con dirigir la vista un poco hacia arriba

2.-la libertad que te da utilizar este tipo de dispositivos. Busqué la forma de interactuar con ellas sin necesidad de tocarlas directamente, solo con la vista y la voz.

Se estableció desde el laboratorio que mi aplicación final debería ser capaz de reconocer rostros, para seguidamente pintar encima un dibujo preestablecido (una máscara), con lo que se habrían conseguido los dos objetivos, escaneo de la imagen en tiempo real en busca de patrones(en este caso los patrones corresponden a caras) e inserción de un elemento virtual en un mundo real. Aparte de eso, debía adaptar mis proyectos al engine utilizado en el laboratorio, Unity3d, con lo que primero trataría de entender como funciona con ese tipo de dispositivo.

Decidí dividir mi investigación en varios campos:

.-Análisis de todos el hardware con la finalidad de establecer interfaces* funcionales e intuitivos

.-Búsqueda y reconocimiento de rostros, formas y colores a través de la cámara principal del dispositivo.

.-Utilización de la Realidad Aumentada* con el fin de mostrar información adicional sobre el mundo real en tiempo real

El objetivo final fue diseñar esa aplicación haciendo frente a los distintos retos tecnológicos que se planteaban a la vez que estudiaba la mejor manera de interactuar con ese tipo de dispositivo tan actual como son las Google Glass.

Todo los proyectos desarrollados referentes a los wereables que realice en ese periodo quedan reflejado en este proyecto, el cual cuenta ademas con varios tutoriales en el Anexo, con el que se puede reproducir todos los pasos que di para configurar los distintos entornos.

He tratado de que a la vez que reflejara todo mi estudio en este campo, fuera útil para un futuro miembro del laboratorio que tuviera que realizar una labor parecida con el mismo dispositivo. También hay que tener en cuenta que este tipo de desarrollos quedan obsoletos en muy poco tiempo, por eso esta redactado de tal manera que sea el punto de inicio prefecto para un proyecto que comparta características, reflejando solo las partes mas importantes del código, que con un poco de investigación posterior, te pueden llevar rápidamente a la solución buscada.

## 2.-Descripción de las herramientas utilizadas en el desarrollo del proyecto

GLASS

*Ilustración 1: logo de las google glass*

**Goglee Glass**

[AII]Google Glass ("GLASS") es un dispositivo de visualización tipo gafas de realidad aumentada desarrollado por Google, una empresa multinacional estadounidense especializada en productos y servicios relacionados con Internet, software, y otras tecnologías. Google Glass Explorer Edition fue lanzado por los desarrolladores de Google por mas de 1500 dolares en 2013, mientras que la versión para consumidores salió a la venta un poco mas tarde solo en Estados Unidos.

*Ilustración 2: las google glass*

AII - Referencia II en la bibliografía

Características de la Google Glass

Para poder aprovechar al máximo el potencial del dispositivo es necesario conocer todas las características del mismo. Las que tienen alguna relevancia con los proyectos posteriores son:

Hardware

Cámara: Google Glass tiene la capacidad de tomar fotos a una resolución de 5 MP y grabar vídeo en 720p.

Touchpad: El costado de las Google Glass es una superficie táctil que permite a los usuarios controlar el dispositivo mediante gestos como desplazar y tocar.

Especificaciones técnicas

No hay información oficial sobre la resolución de pantalla. Se ha sugerido la de 640 × 360 ya que se recomienda para los desarrolladores de aplicaciones

- Cámara de 5 megapíxeles, capaz de grabar vídeo 720p
- Wi-fi 802.11b/g
- Bluetooth
-Batería: hasta un día de uso, aunque indican que utilizarlas con Google Hangouts o para grabar vídeo reducirá la batería
- Almacenamiento de 16 GB sincronizados con la nube (12 GB disponibles)
- Texas Instruments OMAP 4430 SoC 1.2GHz Dual (ARMv7)
- 682MB RAM "proc"
- Giroscopio de 3 ejes.
- Acelerómetro 3 ejes
- Sensor geomagnético (brújula)

- Sensores de luz ambiente y sensor de proximidad interno.

- Sistema de inducción ósea para la transmisión del sonido. A partir de la versión 2.0 se permite la utilización de auriculares específicos.

- La montura es ajustable.

- Actualmente se basa en Android 4.0.4 modificado y adaptado al dispositivo [1].

Software

Aplicaciones

Las aplicaciones de Google Glass son aplicaciones gratuitas creadas por desarrolladores de terceros aunque también utiliza muchas aplicaciones existentes de Google, como Google Maps, Google + y Gmail.

Al ser aun un prototipo en fase de pruebas, en los términos del servicio, los desarrolladores no pueden poner anuncios en sus aplicaciones o exigir tasas(aunque esto podría cambiar en el futuro).

Muchos desarrolladores y empresas han creado aplicaciones para Glass, incluyendo aplicaciones de noticias, edición de fotografías y redes sociales como Facebook y Twitter.

Una de la atracciones de este producto es que puede ser controlado mediante "acciones de voz". Para activar Glass, los usuarios pueden inclinan la cabeza hacia arriba en un ángulo anteriormente configurado y decir "Ok, Glass"

Una vez Glass está activada, los usuarios pueden decir una acción, como "Take a picture" (Haz una foto), "Record a video" (Graba vídeo) así como "Google" para iniciar una búsqueda, "Get directions to..." (Guíame hacia...), "Send a message to..." (Enviar mensaje a...), "Make a call/videocall to..." (Hacer una llamada a...).

[AI]Todo esto nos hace pensar que en un periodo a medio o largo plazo el mercado de las aplicaciones para wereables de este tipo puede que empiece su actividad en un futuro, resultando ventajoso si asi fuera conocer el proceso de fabricación de software para el.

De todas formas hay que tener en cuenta que el dispositivo no cuenta con especificaciones tecnicas suficientes para usarlo durante periodos de tiempo largos y a pleno rendimiento, aunque puede ser muy provechoso en campos como la productividad o la movilidad reducida, aunque limitando el uso del dispositivo a periodos cortos de tiempo.

AI - Referencia I en la bibliografia

*Ilustración 3: logo Unity3D*

**Unity3d 4.6.x**

[AIII]Unity3d es un completo engine* propiedad de Unity Technologies* mediante el cual podemos desarrollar para varias plataformas, incluyendo algunas tan novedosas como WebGL* y OculusRift*, con la peculiaridad de que el mismo proyecto vale para todas (la conversión es un proceso el cual el usuario no controla).

Aquí vemos una lista completa con las 21 plataformas actuales:

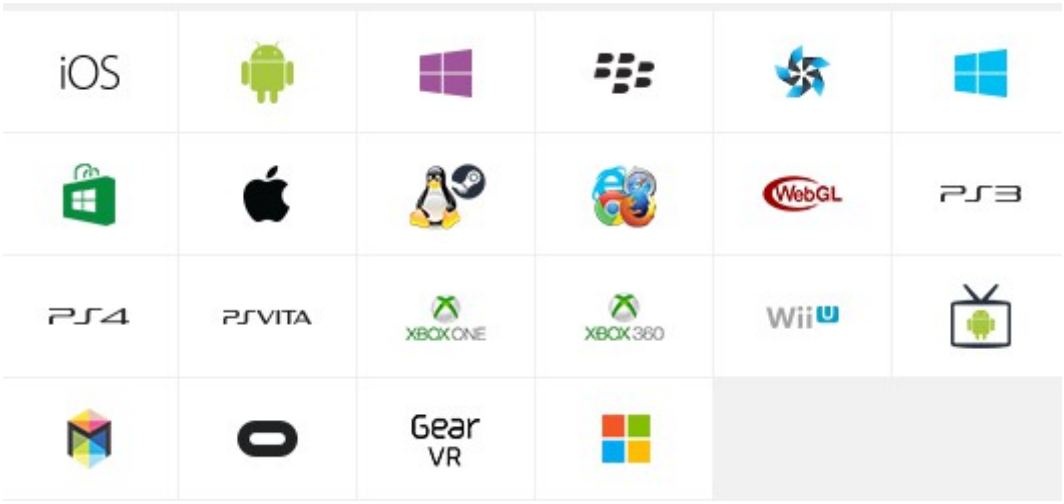


*Ilustración 4: todas las plataformas soportadas por Unity*

AIII - Referencia III en la bibliografía

[AIV]Su robustez, su facilidad de uso, y su soporte mantenido por los propios desarrolladores y los cada vez mas numerosos seguidores de Unity en la red, han hecho que se extienda rápidamente entre los desarrolladores de aplicaciones, como por ejemplo Blizzard*, que lo utilizo para implementar uno de sus últimos lanzamientos.

Principalmente lo utilizaremos para desarrollar nuestras diferentes aplicaciones, todas ellas para el sistema operativo Android.

AIV - Referencia IV en la bibliografía

*Ilustración 5: logo de eclipse*

**Eclipse**

[V]Es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java.

No es más que un entorno de desarrollo integrado (IDE) en el que encontrarás todas las herramientas y funciones necesarias para tu trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

Hasta hace muy poco formaba parte del ADT*, por lo que su compatibilidad con aplicaciones android es total.

Principalmente lo utilizaremos para la creación de librerías .jar* mediante el cual podremos ejecutar código nativo de dispositivos android en nuestra aplicación unity, creando plugins compatibles y totalmente funcionales.

V - Referencia V en la bibliografía


*Ilustración 6: logo de AndroidStudio*

**AndroidStudio**

[VI]AndroidStudio es un entorno de desarrollo integrado (IDE) para la plataforma Android, para Windows, Mac y Linux. Es el IDE oficial y cuenta con mejoras respecto a eclipse como:

.-Sistema de construcción basado en el flexible sistema Gradle

.-Construye distintos y multiples apk's.

.-Plantillas para generar las aplicaciones mas comunes

.-Editor de diseño tipo "drag and drop"

.-Herramientas para controlar el rendimiento, usabilidad, compatibilidad de versiones y otros problemas

.-Capacidades ProGuard y app-signing

.-El soporte integrado para la plataforma Google Cloud, hace fácil combinar mensajería y el app-engine

Al igual que eclipse lo utilizaremos principalmente para la creación de librerías .jar*.

VI - Referencia VI en la bibliografía

*Ilustración 7: logo de Android*

**SDK Android**

[AVII]SDK responde a las siglas Software Development Kit, lo que viene a ser un kit de desarrollo de software. Con él podremos desarrollar aplicaciones y ejecutar un emulador del sistema Android de la versión que sea. Todas las aplicaciones Android se desarrollan en lenguaje Java con este kit.

Con SDK de Android podremos utilizar nuestro dispositivo en modo desarrollador desde nuestro ordenador. Lo puedes descargar junto con el IDE oficial [AVIII]AndroidStudio.

**GDK Android**

[AIX]El kit de desarrollo de Cristal (GDK) es un add-on para el SDK de Android que te permite construir software para las gafas que se ejecuta directamente sobre en ellas. Las guías de esta sección describen cómo utilizar las principales APIs en el GDK

AVII - Referencia VII en la bibliografía
AVIII - Referencia VIII en la bibliografía
AIX - Referencia IX en la bibliografía

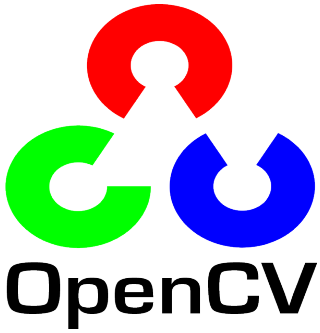


*Ilustración 8: logo de opencv*

**OpenCV**

[AX]OpenCV (Open Source Computer Vision Library) son unas librerías de código abierto basado en visión por computador. OpenCV fue construido para proporcionar una infraestructura común para aplicaciones de visión por computador y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al ser un producto con licencia BSD, OpenCV hace que sea fácil para las empresas a utilizar y modificar el código.

La biblioteca cuenta con más de 2.500 algoritmos optimizados, que incluye un amplio conjunto de clásico y algoritmos de visión por computador y aprendizaje automático con tecnología de última generación.

Estos algoritmos se pueden utilizar para detectar y reconocer rostros, identificar objetos, clasificar las acciones humanas en videos, movimientos de cámara, objetos pista pista en movimiento, extraer modelos 3D de objetos, producen nubes de puntos 3D de cámaras estéreo, unir imágenes juntos para producir una alta resolución imagen de toda una escena, encontrar imágenes similares de una base de datos de imágenes, eliminar los ojos rojos de las imágenes tomadas con flash, seguir los movimientos de los ojos, reconocer el paisaje y establecer marcadores para cubrirás de realidad aumentada, etc.

OpenCV tiene más de 47 mil personas de usuario la comunidad y la estimación del número de descargas superan los 7.000.000.

Por todo esto es idóneo para nuestros reconocimientos de la imagen en busca de los diferentes objetivos.

---

AX - Referencia X en la bibliografía


*Ilustración 9: logo C#script*

**C# Script**

[AXI]CS-Script es un CLR (Common Language Runtime) sistema de scripting que utiliza ECMA-compliant C # como lenguaje de programación. CS-Script Actualmente dirige implementación de Microsoft de CLR (.NET 2.0 / 3.0 / 3.5 / 4.0 / 4.5) con soporte completo en Mono(esa compatibilidad total con Unity lo hace idóneo para desarrollar el código de la aplicación).

CS-Script es una iniciativa de código abierto (MIT) que se distribuye bajo el acuerdo de licencia, que puedes encontrar en:

http://www.csscript.net/

CS-Script combina el poder y la riqueza de C # y FCL con la flexibilidad de un sistema de scripting. CS-Script puede ser útil para los administradores de sistemas y redes, desarrolladores y probadores. Para cualquiera que necesite una automatización para resolver diversas tareas de programación.

CS-Script ha comenzado como un artículo en CodeProject pero rápidamente ha crecido más allá de la escala de una sola publicación. Actualmente se utiliza en todo el mundo para extender la funcionalidad de las aplicaciones con las secuencias de comandos y como un entorno general scripting propósito. Es utilizado por los entusiastas y programadores profesionales. Se encontró su camino a las organizaciones sin fines de lucro (por ejemplo, institutos de educación), así como a las organizaciones comerciales. Estos son sólo algunos ejemplos: MediaPortal, FlashDevelop, API K2, SF.net ("WinTin"), bonsai, AyaNova (software de gestión de servicios).

La idea principal de CS-Script es permitir "plain vanilla" ejecución de código C # de ambos-símbolo del sistema y formar cualquier aplicación CLR aloja el motor de scripts.

---

AXI - Referencia XI en la bibliografía

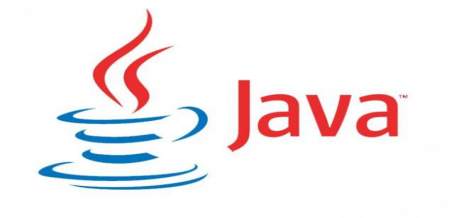


*Ilustración 10: logo java*

**Java**

[AXII]Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.

Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

Recurrimos a el para confeccionar los diferentes plugins nativos de Android compilados desde eclipse o AndroidStudio.

AXII - Referencia XII en la bibliografía

## 3.-Estudio Previo

## 3.1.-Estudio del hardware

Análisis del hardware:

Estudiando las especificaciones técnicas de las GG(anexo) y obviando el panel multitoque que tiene en un lateral y un botón encima de la cámara, llegue a la conclusión de que la mejor manera de interaccionar con ellas de un modo cómodo y productivo era utilizando los sensores de los que dispone. Dentro del chip Asahi Kasei AKM8975 incorporado en el dispositivo, podemos encontrar:

.-1: Giroscopio de 3 ejes.

-2: Acelerómetro 3 ejes

Entre estos dos sensores podemos extraer perfectamente la posición, inclinación y orientación del dispositivo en todo momento.

El acelerómetro cuenta con tres pequeños tubos en cuyo interior, colocado en la parte superior, existe un muelle en cuya otra punta hay una bola haciendo de masa. Esto quiere decir que si tenemos tres tubos simulando los tres ejes de coordenadas tridimensionales, si movemos este conjunto, la bola se desplazará dentro de los tubos. Así es como sabemos la posición de nuestro dispositivo.

El giroscopio, por su parte, Es un dispositivo mecánico que nos ayudará a medir, mantener y cambiar la orientación del dispositivo a la vez que da suficiente información para saber la posición.

Para ciertas tareas es imposible usar estos sensores como método de interacción principal, ya que usaras el dispositivo en movimiento, cambiando constantemente los datos proporcionados por el giroscopio y el acelerómetro.

Para las tareas en las que vamos a permanecer sentados, en cambio, si que podemos usarlo para saber en que posición esta la cabeza del usuario, y a partir de ahí, rastrear sus movimientos y traducirlos en interacción con el interfaz, es decir, nos ayuda a manejar el dispositivo con movimientos leves de cabeza.

Tenemos varias opciones:

.-En un scroll horizontal o vertical con varios elementos donde solo esta seleccionado el elemento central, con movimientos laterales de izquierda a derecha o de arriba a abajo, podemos recorrer el scroll en esas mismas direcciones.

.-En un menú con varios elementos interactivos donde necesitaremos un cursor, podemos traducir la posición del mismo en función de la inclinación del dispositivo.

Si el usuario esta mirando de frente la posición del cursor sera el centro de la pantalla, si mira abajo e inclina un poco la cabeza a la derecha el cursor se posicionara en la esquina inferior derecha de la pantalla.

El menú detectara si el cursor esta proyectado sobre algún elemento interactivo y esperara el disparador correspondiente para activarlo(desde un comando de voz hasta una cuenta atrás de 3 segundos).

.-3Sensor geomagnético (brújula)

Se trata de un componente electrónico capaz de medir y cuantificar la cantidad de fuerza magnética de un objeto. O para lo que muchos dispositivos lo usan, como brújula, detectando el polo norte magnético (que como curiosidad no coincide con el polo norte geográfico).

Todos los dispositivos que estén dotados con brújula deberían de llevar incorporado un magnetómetro en su interior.

También es útil combinado con elementos externos magnéticos, dotados de dar alguna función, como poner en bajo consumo el dispositivo al cerrar la tapa o como en las Google Cardboard, dotar al nuevo dispositivo de un botón adicional.

Con todos estos elementos coordinados podemos dotar al usuario de una verdadera experiencia placentera y productiva , aunque siempre teniendo en cuenta que el uso prolongado de este dispositivo en concreto, haría aumentar el estrés en el usuario.

Debe recordarse que es un dispositivo complementario a las funciones de un móvil, con lo que no se debería derivar todas las capacidades del mismo al dispositivo, si no procurar una mayor productividad en determinadas situaciones y tareas.

.-4Microfono(Acciones de voz)

Este modo de interactuar nos permite movernos libremente a la vez que interactuamos con el dispositivo. Nuestro objetivo principal es reconocer lo hablado por el usuario y trasladarlo en formato texto dentro de nuestra aplicación. A partir de ahí reaccionar ante determinadas palabras seria muy fácil.

.-5Notificaciones

[AXIII]Una notificación es un mensaje que es mostrado al usuario desde fuera de tu aplicación. Cuando se le dice al sistema que dispare una notificación, primero aparecerá un icono en el área destinada a mostrarlas. Para ver los detalles de la misma, el usuario tiene la posibilidad de abrir el menú contextual del área de notificaciones. Esas dos áreas están siempre a disposición del usuario.

Las características de este dispositivo, y en resumen cualquier tipo de wereable, las hacen idóneas para recibir notificaciones. Las notificaciones en android también son nativas así que implementaremos un proyecto que contenga un plugin encargado de mandar notificaciones desde una actividad android a nuestra actividad principal de unity.

AXIII - Referencia XIII en la bibliografía

## 3.2.- Confeccionando plugins

.-Confección de plugins propios(motivados por la interactuación por voz y las notificaciones)

Podemos definir un plugin (o complemento) como una aplicación que se relaciona con otra para para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interaccionan por medio de la API.

En Unity normalmente se utilizan scripts para crear las funcionalidades, pero también se puede utilizar código creado fuera de Unity forma de un plugin. Hay dos tipos de plugins que puede utilizar en la Unidad: plugins administrados y plugins nativos.

Me centrare en los nativos, ya que necesito acceder a las librerías nativas de Android, y la única forma de ejecutar código no compatible con nuestro engine es aprovechando esta ventaja.

Para construir un plugin para Android, debe en primer lugar obtener el NDK de Android y familiarizarse con los pasos para construir una librería compartida(Anexos X e Y)

### 3.2.1.- Actividades Nativas

Uso de plugins Java

El mecanismo de plugins Android también permite que Java sea usado para permitir la interacción con el sistema operativo Android.

Hay varias formas de crear un plugin Java, pero en cualquier caso el resultado será un archivo .jar que contiene los archivos .class para el plugin. En el Anexo podemos encontrar la forma de hacerlo tanto con eclipse como con AndroidStudio.

Una vez ha sido construido el plugin Java (.jar), hay que copiarlo a la carpeta:

Assets->Plugins->Android en el proyecto.

Unity empacará los archivos .class junto con el resto del código Java y luego accede al código usando el Java Native Interface (JNI). JNI es usado tanto para llamar a código nativo desde Java, como para interactuar con Java (o con JavaVM) desde el código nativo.

Usar un plugin Java con clases ayudantes.

AndroidJNIHelper y AndroidJNI se pueden usar para solventar algunas de las dificultades al utilizar el JNI raso.
AndroidJavaObject y AndroidJavaClass automatizan una gran cantidad de tareas y también realizan cache para hacer llamados a Java más rápido. La combinación de AndroidJavaObject y AndroidJavaClass se construye encima de AndroidJNI y AndroidJNIHelper, pero también tiene una considerable cantidad de lógica por sí mismo (para manejar la automatización). Estas clases tambien vienen en una versión 'estática' para acceder a miembros estáticos de clases Java.

Puedes elegir cualquier enfoque que prefieras, ya sea JNI crudo a través de métodos de clase de AndroidJNI, o AndroidJNIHelper junto con AndroidJNI y al final con AndroidJavaObject o AndroidJavaClass para una máxima automatización y conveniencia.

UnityEngine.AndroidJNI es un envoltorio para los llamados JNI disponibles en C (como fue descrito antes). Todos los métodos en esta clase son estáticos y tienen transformación uno a uno con la JNI.UnityEngine.AndroidJNIHelper proporciona funcionalidad de ayuda que es utilizada por el próximo nivel, pero está expuesta a través de métodos públicos dado que pueden ser útiles en algunos casos especiales.

Las instancias de UnityEngine.AndroidJavaObject y UnityEngine.AndroidJavaClass tienen transformación uno a uno hacia una instancia de java.lang.Object y java.lang.Class (o hacia sus subclases) en el lado de Java, respectivamente. Proporcionan en esencia 3 tipos de interacción con el lado de Java:

1.-Llamar a un método (Call)

2.-Obtener el valor de un campo (Get)

3.-Colocar el valor de un campo (Set)

El Call se divide en dos categorías: Hacer Call a un método 'void', y hacer Call a un método que retorne un tipo que no sea void. Un tipo genérico es usado para representar el tipo que retornan estos métodos, el cual devuelve un tipo que no es void. Los Get y Set siempre toman un tipo genérico que representa al tipo del campo.

La clase Java com.unity3d.player.UnityPlayer ahora tiene un método estático UnitySendMessage, equivalente a la funciónUnitySendMessage de iOS desde el lado nativo. Puede ser usada en Java para pasar datos al código de script..

Escribir extensiones para el código de UnityPlayerActivity en Java

Con Unity Android es posible extender la clase estándar de UnityPlayerActivity (la clase primaria en Java para el Unity Player en Android, similar a AppController.mm en Unity iOS).

Una aplicación puede sobrescribir todas y cada una de las interacciones básicas entre el sistema operativo Android y Unity Android. Puedes hacer esto creando una nueva intancia de Activity la cual sea una extensión de UnityPlayerActivity (el archivo UnityPlayerActivity.java se encuentra en:

Windows:
C:\Program Files\Unity\Editor\Data\PlaybackEngines\AndroidPlayer\src\com\unity3d\player

Mac:
/Applications/Unity/Unity.app/Contents/PlaybackEngines/AndroidPlayer/src/com/unity3d/player

Para hacer esto, primero hay que localizar el classes.jar que viene con Unity Android. Se encuentra en el directorio de instalación:

Windows:
C:\Program Files\Unity\Editor\Data

Mac:
PlaybackEngines/AndroidPlayer/bin.

Luego se agrega classes.jar al classpath utilizado para compilar el nuevo Activity. El archivo (o archivos) .class resultante debe ser comprimido en un archivo .jar y colocado en la carpeta:

Assets->Plugins->Android.

Dado que el manifiesto determina cuál actividad es lanzada, es necesario también crear un nuevo AndroidManifest.xml. El archivo AndroidManifest.xml debe ser colocado también en la carpeta:

Assets->Plugins->Android

Al colocar un manifiesto personalizado, se omite completamente el manifiesto por defecto de Unity Android. A modo de ejemplo, podemos ver la definición de un .class:

*public class OverrideExample extends UnityPlayerActivity* {

Y su correspondiente entrada en el manifest:

*<activity android:name=".OverrideExample"*

Una vez entendido el concepto ya pudiendo elaborar plugins propios, crearemos varios que nos ayuden a plasmar los conceptos desarrollados anteriormente.

## 3.2.2.-Explotando OpenCV en Android

Junto con la librería OpenCV-Android podemos encontrar varios ejemplos de su uso incluido la detección de patrones en frames. No creare uno de 0, si no que modificare uno existente para hacerlo compatible a nuestro dispositivo.

En líneas generales, nuestro método de detección de rostros funciona de la siguiente manera:

1.-Recogeremos cada frame captado por la cámara

2.-Lo escanearemos

3.-Los resultados obtenidos los contrastaremos con nuestros datos en busca de patrones

4.-Cuando detectemos una coincidencia, habremos localizado un rostro.

5.-En ese mismo frame añadiremos encima del rostro un elemento externo a la imagen.

Con OpenCV podemos hacer una actividad nativa en Android que se ocupe de todo lo mencionado anteriormente, a la vez que gestionamos esa actividad desde nuestro entorno Unity, si la convertimos en una librería y la integramos a nuestro proyecto.

La actividad nativa implementara el interface CvCameraViewListener2, con lo que sus métodos públicos pueden ser modificados para poder acceder a ellos. Estos son:

onCameraViewStarted()

El método se invoca cuando la prevista de la cámara ha empezado. Después de invocar al método se empezaran a entregar al cliente frames vía onCameraFrame() callback.

onCameraViewStopped()

Este método es invocado cuando la prevista de la cámara ha sido detenida por alguna razón. Después de la llamada de este método no se entregaran mas frames vía onCameraFrame() callback.

OnCameraFrame()

El método es invocado cuando la entrega del frame necesita ser hecha. Los valores que devuelve es un frame modificado el cual debe ser mostrado por pantalla. El formato del frame se puede pasar especificando parámetros(RGB, BPP, etc).

También debemos crear el comportamiento de nuestra aplicación para interactuar con los diferentes estados(create, pause, resume, etc) y crear el método callback que sera el encargado de disparar la aplicación. Esto nos sirve para no perder la coherencia e ir reaccionando ante los distintos eventos correctamente.

Por ultimo, para poder apuntar a la cámara de la actividad y tratar sus datos, deberemos utilizar la clase CameraBridgeViewBase, ya que es la clase que implementa toda la interfaz de interacción con la misma.

# 4.-Analisis y diseño de las aplicaciónes

## 1.- Proyecto Input

Para empezar a trabajar con Unity para las google glass debemos encontrar la manera de poder interactuar con todos los sensores y paneles analizados anteriormente.

Este proyecto servirá para entender como trabajan todos esos sensores, muestreando los datos por pantalla en tiempo real, como cualquier tipo de interacción por parte del usuario.

Nos serviremos del propio GUI de Unity, dividiendo la pantalla en tantos campos como sea necesario, para tratar de capturar y mostrar por pantalla toda la información posible relevante.

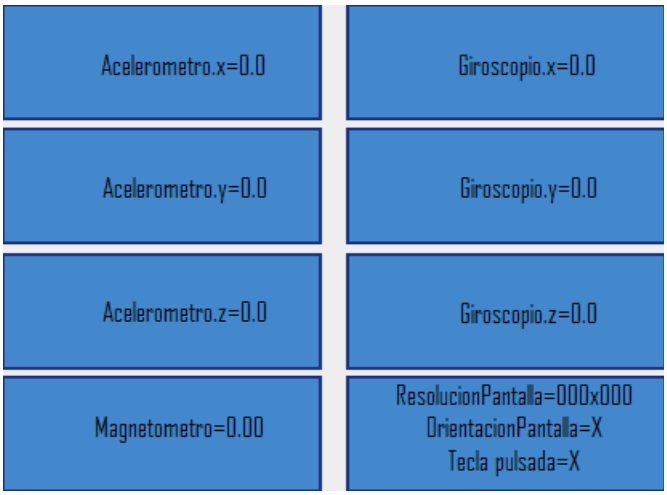


*Ilustración 11: diseño de interfaz*

Ademas dotaremos la aplicación de la activación por voz para poder arrancarla como una aplicación nativa de las glass.

El proyecto será capaz de rastrear:

.-Los datos del giroscopio, acelerometro y magnetómetro.

.-La datos de la cámara como la orientación y la resolución.

.-El panel multitoque colocado en la patilla derecha.

.-El botón de la cámara.

Para todo ello haremos uso de la clase Input de Unity enigne.

Es el interfaz en el sistema de entrada. Es la clase que se utiliza, por ejemplo, para leer los ejes establecidos por el InputManager, o para acceder a los datos del acelerometro en un dispositivo móvil.

También es capaz de rastrear multitoques en un panel de entrada, así que la usaremos para recoger todos los datos posibles en la implementación del proyecto.

## 2.-Proyecto Ray Cursor

Una vez rastreados y entendidos los datos recogidos en el proyecto anterior, usaremos el acelerometro para recoger los movimientos de cabeza del usuario y transformarlo en el movimiento de un cursor situado sobre un panel en la pantalla.

Para ello necesitaremos 2 objetos [AXIV]"cámara" de unity en nuestra escena. Un objeto cámara en unity simplemente es un dispositivo a través del cual el jugador puede ver el mundo(en este caso nuestra escena).

Una cámara se ocupara de mostrar al usuario el panel y el cursor, y la otra, sera la encargada de mover el cursor.

En realidad moveremos la cámara esta lanzando un rayo constante hacia el panel. Ese punto es el punto de destino del cursor. Si al mover ligeramente la cabeza, movemos ligeramente la cámara de un modo natural, el rayo impactara en otra parte del panel y el cursor se desplazara suavemente hacia su nueva posición.

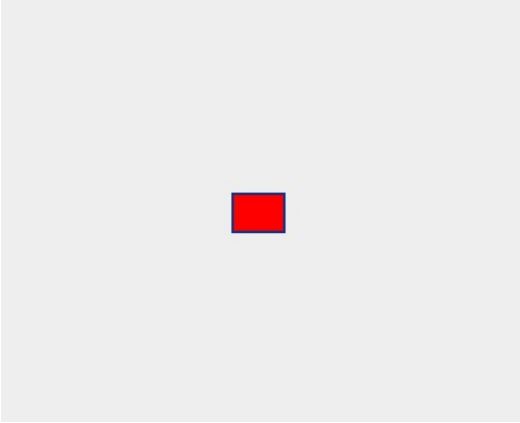

*Ilustración 12: mirada al centro*

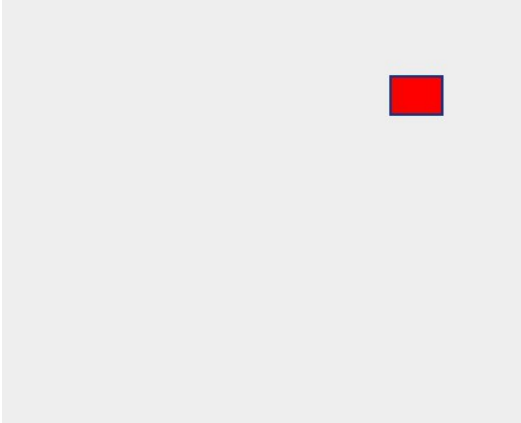

*Ilustración 13: mirada arriba derecha*

AXIV - Referencia XIV en la bibliografía

No hay que olvidar que en unity un punto de espacio en la pantalla se define en píxeles. La parte inferior izquierda de la pantalla es (0,0); el derecho-superior es (pixelsAncho, pixelHeight). La posición z está en unidades mundo de la cámara.

De este modo crearemos nuestro prototipo de cursor para poder utilizarlo en un menú donde solo tienes que usar la cabeza para interactuar con el. Este tipo de interfaz también se podría utilizar sin problemas en dispositivos de realidad virtuales como las Oculus Rift o con smartphones usados con CardBoards.

## 3.- Proyecto Micro Glass

El siguiente objetivo es realizar una aplicación que reconozca lo que decimos.

El interfaz simplemente constara de una etiqueta con un mensaje indicando que para activar el evento hay que tocar el panel multitoque, y de un contenedor de texto, llamado label, que nos mostrará la información recogida para comprobar que efectivamente corresponde con lo dicho.

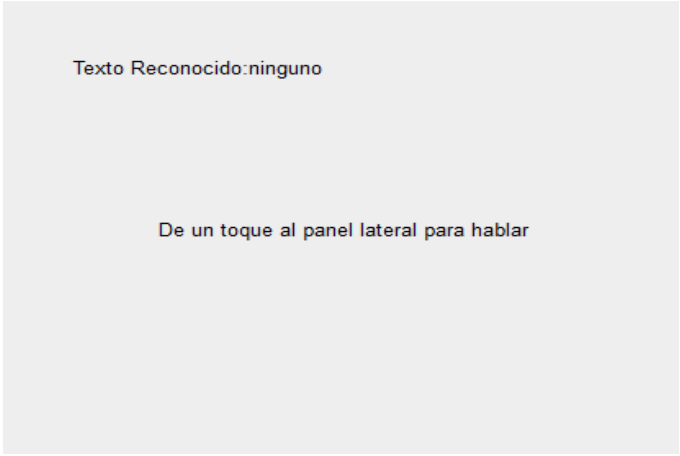


*Ilustración 14: diseño interfaz*

Al presionar el panel nos dará paso a la pantalla de recogida de voz.

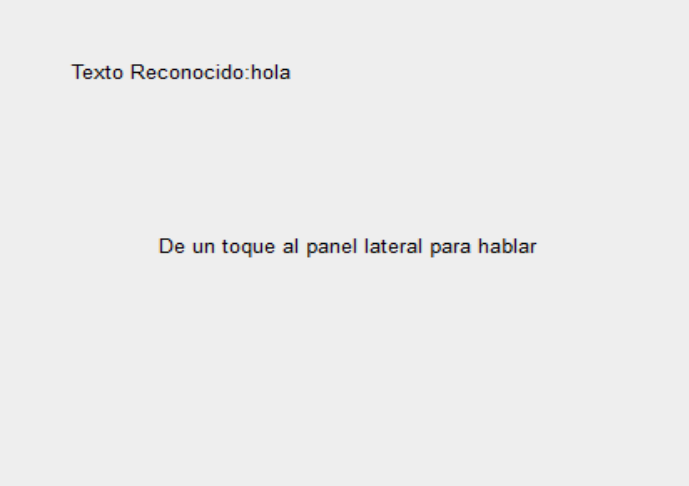


*Ilustración 15: arriba a la izquierda se muestra lo captado*

Debemos tener en cuenta que este servicio solo está disponible conectado a Internet, aunque sin duda es una de las características a explotar mas deseables.

## 4.- Proyecto LlamarFD

El proyecto debe: Dar la posibilidad de llamar desde Unity a una actividad nativa Android (por medio de un disparador, en este caso un toque en el panel multitoque), con la funcionalidad de rastrear y cotejar los datos recogidos en la cámara del dispositivo.

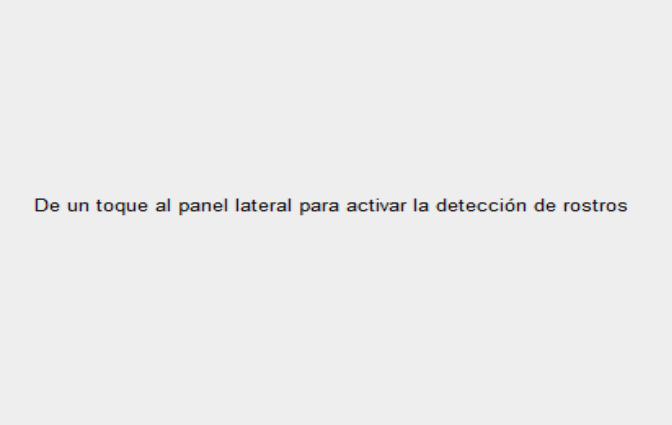


*Ilustración 16: diseño interfaz*

En caso que detecte un rostro en algún frame, deberá añadir un rectángulo perfectamente adaptado al tamaño del rostro, y mostrarlo por pantalla.

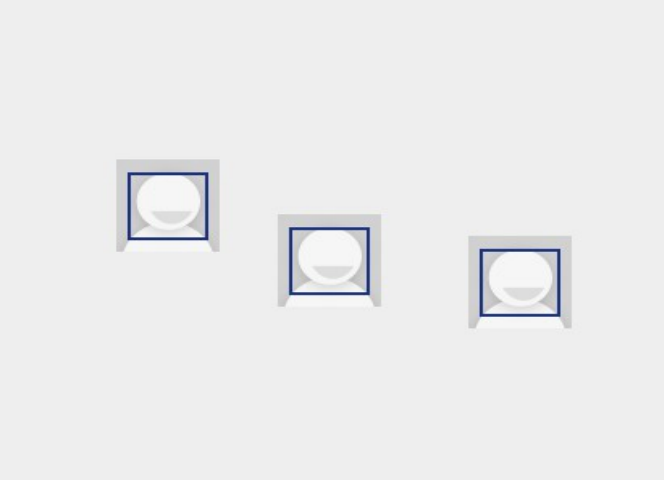

*Ilustración 17: simulación de la lógica final*

Todo ello será en tiempo real, mostrando continuamente por pantalla lo que capta la cámara y las modificaciones posteriores.

## 5.-Proyecto LLamarFD Mask

Sera exactamente que el proyecto anterior, pero en vez de ajustar un rectángulo al rostro, buscaremos una imagen de una máscara debidamente preparada para dejar ver los ojos del rostro captado.

*Ilustración 18: mask.png*

La mascara también deberá adaptarse apropiadamente al tamaño del rostro detectado.

## 6.-Proyecto Notification

Aquí programaremos una notificación para que se dispare en un rango de tiempo dado, y con unos parámetros como el título y el contenido. Esta aplicación no debería instalarse en las google glass, si no en el dispositivo "padre" vinculado a ellas.

Deberemos ser capaces de recoger desde Unity esos 3 campos, asignarles un valor, e invocar a la clase nativa encargada de la notificación con esos 3 parámetros.

Con un simple botón que haga de disparador tendremos suficiente para verificar su funcionamiento.


*Ilustración 19: diseño interfaz*

La notificación, aparte de en el mismo dispositivo donde se programo, también se activara en los wereables que tengas anexados a tu smartphone o tablet, en nuestro caso con las google glass, aunque los smartwatch compatibles con Android(que no necesariamente con SO Android) también son capaces de recibirlos.

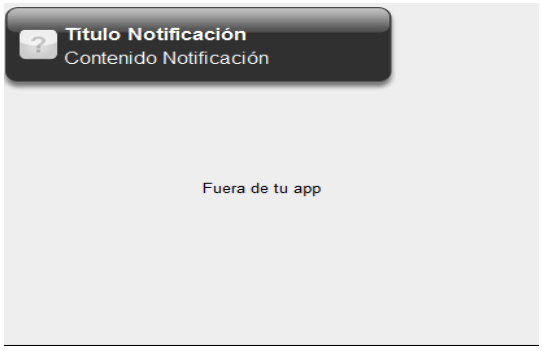


*Ilustración 20: simulación de la muestra de la notificación*

# 5.-Implementación de la solución

## 5.1.-Pasos Previos

En primer lugar analizaremos los diferentes componentes que toman un papel fundamental en nuestras aplicaciones mas generales, para luego ir en el siguiente punto proyecto a proyecto explicando sus características mas relevantes.

### 1.-AndroidManifest.xml

[AXV]Situado en la raíz de nuestras aplicaciones como AndroidManifest.xml, es un archivo de configuración donde podemos aplicar las configuraciones básicas de nuestra app. Además es obligatorio, y si no generamos nuestro propio fichero de configuración unity usara el suyo propio. Entre otras cosas el fichero hace lo siguiente:

.-Nombra el paquete Java para la aplicación. El nombre del paquete sirve como identificador único para la aplicación.

.-En él se describen los componentes de la aplicación - las actividades, servicios, receptores de radiodifusión, y proveedores de contenido que la aplicación se compone de. Nombra a las clases que implementan cada uno de los componentes y publica sus capacidades (por ejemplo, qué mensajes Intención que puede manejar). Estas declaraciones permiten el sistema Android sabe lo que los componentes son y en qué condiciones pueden ser lanzados.

.-Declara los permisos que la aplicación debe tener para acceder a partes protegidas de la API y para interactuar con otras aplicaciones.

.-Declara el nivel mínimo de la API de Android que la aplicación requiere.

.-Lista las bibliotecas necesarias para su vinculación con la aplicación.

---

AXV - Referencia XV en la bibliografía

[AXVI]Para poder compatibilizar nuestra aplicación con las google glass debemos introducir en el AndroidManifest varias entradas adicionales a las que vienen en el fichero de configuración generado por Unity. Estas son:

.-Dentro de <manifest…> definiremos los permisos necesarios. En este caso acceso a Internet y el uso de la cámara.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CAMERA"/>
```

Nota:Dependiendo de tu entorno de trabajo, quizás también necesites la entrada:

```
<uses-permission android:name="com.google.android.glass.permission.DEVELOPMENT" />
```

.-Para configurar el disparador por voz de nuestra aplicacion debemos expresar que queremos hacer uso de el, y ademas diremos en que fichero de recursos esta la cadena asociada a el. Lo pondremos dentro de <activity… >

```
<intent-filter>
    <action android:name="com.google.android.glass.action.VOICE_TRIGGER" />
</intent-filter>

<meta-data
    android:name="com.google.android.glass.VoiceTrigger"
    android:resource="@xml/my_voice_trigger" />
```

AXVI - Referencia XVI en la bibliografía

.-Dentro de la definición de activity introduciremos la orientación de la pantalla. En este caso horizontal:

*android:screenOrientation="landscape"*

.-Dentro de <manifest…> podemos poner la version de andorid que nuestro dispositivo utiliza. En este caso usare:

*<uses-sdk android:minSdkVersion="13" android:targetSdkVersion="13" />*

Teniendo en cuenta que los SDKs corresponde a las versiones de Android siguientes:

13:Android 3.2

19:Android 4.4.2

## 2.-MonoBehaviour

Es la clase, de la que cada clase hereda.

El uso de Javascript hace que cada script herede automáticamente de MonoBehaviour. Al utilizar C# o Boo, explícitamente tenemos que definirlo.

*public class ExampleBehaviourScript : MonoBehaviour*

Las funciones públicas de MonoBehaviour que vamos a utilizar en nuestras aplicación son:

[AXVII]Awake();

.-Se llama cuando la instancia del script se carga.

.-Se utiliza para inicializar las variables o el estado del juego antes de que comience.

.-Se llama una sola vez durante la vida de la instancia .

.-Se llama después de que se inicialicen todos los objetos para que puedas contactar con seguridad con ellos.

.-Siempre se llama antes de la función Start();

.-No puede actuar como un corutina.

AXVII - Referencia XVII en la bibliografía

[AXVIII]Start();

.-Es llamado en el mismo frame que un script es habilitado y justo antes de cualquier Update que se vaya a ejecutar en ese mismo frame.

.-Igual que el Awake, solo se le llama una vez, solo que Awake se llama una vez que el objeto se inicializa, sin importar si el script esta habilitado o no, no como Start.

[AIXX]Update();

.-Se llama cada frame, si MonoBehaviour esta habilitado.

.-Es lo mas utilizado para implementar cualquier tipo de sistema de juego.

AXVIII - Referencia XVIII en la bibliografía
AIXX - Referencia IXX en la bibliografía

## 3-GUI

[AXX]La clase GUI es el interfaz usado  para diseñar el GUI de Unity.

Los controles de unity hacen uso de una function especial, que como el Update, se ejecuta a cada frame(siempre que su secuencia de comando este activada).

Los utilizaremos sobre todo para mostrar información por pantalla a través de objetos label (contenedores de texto) y para lanzar eventos a través de objetos Button (disparadores de eventos en forma de botón).

AXX - Referencia XX en la bibliografía

**4.-Codigo especifico contenido en los C# Scripts**

Hay partes de código que solo usaremos solo por el hecho de que estamos programando para un wereable.

.-Si queremos determinar que una accion solo se produzca en nuestro terminal android basta con introducirlo bajo esta condicion.

```
if (Application.platform == RuntimePlatform.Android)ir en el Awake();
```

.-Para evitar que el dispositivo apague la pantalla miesntras utilizamos la aplicación introduciremos en en un Awake de inicialización.

```
Screen.sleepTimeout = SleepTimeout.NeverSleep;
```

.-Para no perder al objeto entre actividades añadir en el Awake():

```
DontDestroyOnLoad (transform.gameObject);
```

.-Para detectar si el panel lateral ha sido presionado(también puedes detectar el numero de toques simultáneos):

```
if (0 < AndroidInput.touchCountSecondary){
```

## 5.2.-Aplicaciones Desarrolladas

**.-aplicación con todos los sensores activos -> Proyecto Input**

Proyecto Input

Este proyecto tiene como finalidad la muestra por pantalla de los valores que pueden ir tomando los diferentes sensores con los que cuenta el dispositivo los cuales necesitamos rastrear y cuantificar para poder usarlos posteriormente en nuestro software.

En la carpeta assets encontraremos:

Assets>

Carpeta Plugins>Android>AndroidManifest.xml

Adicionalmente deberemos introducir:

Carpeta Plugins>Android>res>strings.xml

Fichero xml con el nombre de la app(app_name) y el nombre que veremos en el launcher (glass_voice_trigger) propio del dispositivo que tambien nos servira para arrancarlo por voz

```
<?xml version="1.0" encoding="UTF-8" ?>
    <resources>
    <string name="app_name">@string/app_name</string>
     <string name="glass_voice_trigger">test input</string>
</resources>
```

Carpeta Plugins>Android>xml>my_voice_trigger.xml

en este fichero xml configuramos el disparador de voz asociándolo a una cadena definida en nuestro fichero string de recursos(en este caso “test input”)

```
<?xml version="1.0" encoding="UTF-8" ?>
<trigger keyword="@string/glass_voice_trigger">
</trigger>
```

Carpeta Scenes>Test.unity

La escena a ejecutar contenedora de nuestra actividad

Carpeta Scripts>Giros.cs

Script que muestra con la funcionalidad explicada anteriormente. A destacar:

.-Para todo tipo de wereables Android estos valores pueden ser usados en caso de que dispongan de los sensores adecuados

.-Deberemos activar el acelerometro si queremos mostrar sus valores

```
Input.gyro.enabled = true;
```

.-Accederemos a los valore de los sensores y los trataremos para su posterior muestreo, POr ejemplo valores como la inclinación y la aceleración, para luego usarlos en el GUI propio de Unity

```
posicion = Input.gyro.attitude;
aceleration = Input.gyro.userAcceleration;

GUI.Box (new Rect (10, 10, 60, 50), "Gx" + posicion.x);
GUI.Box (new Rect (70, 10, 60, 50), "Gy " + posicion.y);
GUI.Box (new Rect (130, 10, 60, 50), "Gz " + posicion.z);
GUI.Box (new Rect (250, 60, 200, 50), "Acel: " + aceleration);
```

.-Cada cierto tiempo debemos calibrar el acelerómetro usando para ello la función ResetGyro(float waitTime), que recibirá como parámetro el tiempo máximo que durará la operación.

```
ResetGyro(float waitTime);
```

.-Botones con correlación directa al teclado. Las GG solo responden a una señal producida por una pulsación de tecla(la tecla ESC). Registraremos la pulsación con la siguiente linea

```
if  (Input.GetKey(KeyCode.Escape))
        boton = "escape key is held down";
```

.-La orientación de pantalla la podemos obtener con

```
if(Screen.orientation == ScreenOrientation.LandscapeRight)
        orientation ="LR";
```

**.-aplicación con el cursor que sigue tu mirada-> Proyecto Ray Cursor**

En este proyecto ponemos en practica lo aprendido sobre los sensores del dispositivo e implementamos el sencillo cursor diseñado anteriormente que usa gestos mínimos para desplazar su posición.

Assets>

Carpeta Plugins>Android>AndroidManifest.xml

Fichero de configuración de Android

Adicionalmente deberemos introducir:

Carpeta Plugins>Android>res>strings.xml

Fichero xml con el nombre de la aplicación (app_name) y el nombre que veremos en el launcher (glass_voice_trigger) propio del dispositivo que también nos servirá para arrancarlo por voz

```
<?xml version="1.0" encoding="UTF-8" ?>
<resources>
    <string name="app_name">@string/app_name</string>
    <string name="glass_voice_trigger">test menu</string>
</resources>
```

Carpeta Plugins>Android>xml>my_voice_trigger.xml

en este fichero xml configuramos el disparador de voz asociándolo a una cadena definida en nuestro fichero string.xml de recursos(en este caso "test menu").

```
<?xml version="1.0" encoding="UTF-8" ?>
<trigger keyword="@string/glass_voice_trigger">
</trigger>
```

Carpeta Scenes>1.unity

Escena con un Panel encargado de ser el fondo, un cubo rojo que hace las funciones de cursor, y las dos cámaras necesarias. La que nos muestra la escena, Camera, y la que mueve con su orientación el cursor por la pantalla en función de nuestro gestos CameraRay.

El Cursor tendrá el script asociado a toda la actividad

Carpeta Scripts>CursorMovement.cs

Este es el encargado de recoger la orientación de la cámara CameraRay y transformarla en un movimiento fluido del cursor.

Cabe destacar:

.-Usaremos un objeto publico donde arrastraremos la CameraRay para poder manejarla desde el propio scripts, llamado camera1.

```
public Camera camera1;
```

.-Proyectaremos un rayo usando como dirección la orientación de la camera1 en el método Update().

```
Physics.Raycast(camera1.transform.position,camera1.transform.forward,out hit,1000);
```

.-Cuando el rayo detecta el plano mueve el cursor a la posición de la colisión y vuelve a re colocar la camera en un función de nuestros movimientos

```
if(hit.collider.tag == "Plane")
        gameObject.transform.position = posicion;
        RotateCamera ();
```

.-Cuantificamos los datos del acelerómetro y los convertimos en la nueva posición de la cameraray mediante la conversión de un Quaternion a una transformada.

```
Quaternion offsetRotation = Quaternion.Inverse(gyroInitialRotation) * Input.gyro.attitude;
camera1.transform.localRotation = initialRotation * Quaternion.Inverse(offsetRotation);
```

**.-aplicación con el plugin de voz-> Proyecto Micro Glass**

En este proyecto se utilizaremos un plugin para poder usar el reconocimiento de voz online de google.

Para ello crearemos una simple escena con un evento que dispare la petición de recogida de voz, activando el micrófono y dando oportunidad al usuario a pronunciar unas palabras o una frase para luego mostrar lo captado por el interprete como una cadena de texto por pantalla.

En la carpeta assets encontraremos:

Assets>

Carpeta Plugins>Android>AndroidManifest.xml

Fichero de configuración de Android

Adicionalmente deberemos introducir:

Carpeta Plugins>Android>micro_plugin.jar

Plugin con el código necesario para activar el interprete. Se creara desde eclipse. A destacar:

.-En la definición de la clase vemos que extiende UnityPlayerActivity(la clase primaria en Java para el Unity Player en Android, similar a AppController.mm en Unity iOS).

*public class MainActivity extends UnityPlayerActivity{}*

.-Para ello debemos importar la libreria correspondiente

*import com.unity3d.player.UnityPlayerActivity;*

.-Activaremos el reconocimiento de voz. En vez de inciar la actividad de modo normal con starActivity, haremos un startActivityForResult. De esta forma cuando tenga un resultado lo devolverá a la actividad que lo disparo. Gracias al método onActivityResult() podemos recoger la respuesta, analizarla, y comprobar si ha sido correcta.

.-Una vez tengamos el resultado correcto lo pasaremos a nuestra escena de Unity con la linea:

```
UnityPlayer.UnitySendMessage("Caller", "OnGUI", spokenText);
```

La función tiene 3 parámetros:

.-El nombre del gameobject objetivo. En este caso "Caller".
.-El método a llamar en ese object.
.-El mensaje a pasar al método dentro una cadena, en este caso dentro de "spokenText"

Carpeta Plugins>Android>res>strings.xml

Fichero xml con el nombre de la app(app_name) y el nombre que veremos en el launcher (glass_voice_trigger) propio del dispositivo que también nos servirá para arrancarlo por voz

```
<?xml version="1.0" encoding="UTF-8" ?>
<resources>
    <string name="app_name">@string/app_name</string>
    <string name="glass_voice_trigger">test micro</string>
</resources>
```

Carpeta Plugins>Android>xml>my_voice_trigger.xml

en este fichero xml configuramos el disparador de voz asociándolo a una cadena definida en nuestro fichero string de recursos(en este caso "test micro")

```
<?xml version="1.0" encoding="UTF-8" ?>
<trigger keyword="@string/glass_voice_trigger">
</trigger>
```

Carpeta Scenes>1.unity

Escena con un gameobject llamada "Caller" el cual controlara la petición de ejecución de la actividad nativa de Android a la vez que recibirá los datos y los mostrara por pantalla.

Carpeta Scripts>Test.cs

Script complemento del objeto Caller en la escena 1. A destacar:

.-Para evitar que la pantalla entre en modo sleep necesitaremos decirle en nuestro Awake() que no lo haga. Para ello utilizaremos

```
Screen.sleepTimeout = SleepTimeout.NeverSleep;
```

.-Para evitar que nuestro objeto Caller encargado de manejar los datos sea destruido entre escenas, añadiremos el codigo

```
DontDestroyOnLoad (transform.gameObject);
```

.-Para ejecutar la actividad micrófono usaremos el panel multitoque del lateral del dispositivo. Para ello dentro de un Update() incluiremos:

```
if(InputAndroid.touchCountsecondary > 0))
        AndroidCallNonStatic();
```

.-La llamada a la actividad esta compuesta por:

1.-creacion de nuestra clase contenida en el .jar en nuestro proyecto

```
AndroidJavaClass javaClass = new AndroidJavaClass("com.android.microglass.MainActivity")
```

2.-creacion del contexto para poder intercambiar datos

```
AndroidJavaObject activity = javaClass.GetStatic<AndroidJavaObject>("mContext"))
```

3.-llamada a la actividad

```
activity.Call("displaySpeechRecognizer");
```

.-La recepción de lo reconocido por el interprete lo recibirá el método AndroidMessage, el cual lleva como parámetro la cadena transmitida

```
void AndroidMessage(string dataReceived)
{
    dataReaded = dataReceived;
}
```

**.-aplicación que detecta tu rostro y pone un rectángulo>LlamarFD**

**Proyecto LlamarFD**

Se encarga de detectar todas los rostros en tiempo real que la cámara va captando añadiéndoles un rectángulo de proporción correcta.

Assets>

Carpeta Plugins>Android>AndroidManifest.xml

Fichero de configuración de Android. A destacar:

Debe ser preparado con los permisos propio que necesita OpenCV para funcionar en Android y ademas las características de las google glass.

Carpeta Plugins>Android>AndroidPugin.jar

Plugin con el código java con la funcionalidad. Contiene los ficheros MainActivity.class y DetectionBasedTracker.class. Se creara desde Eclipse. A destacar:

-.Extiende la catividad nativa de Unity como de costumbre pero esta vez hay que definir tambien que implementa CvCameraViewListener2

*public class MainActivity extends UnityPlayerActivity implements CvCameraViewListener2* {}

.-La aplicación funciona de una manera asíncrona con el OpenCVManager. Cuando la inicialización acabe sera llamado el callback OnManagerConnected. Aprovecharemos ese momento para cargar todas nuestras librerías de rostros y crear una vista compatible con nuestro dispositivo.

Dentro de onManagerConnected, después de definir el callback y recibir una conexión correcta cargamos nuestras librerías:

```
public void onManagerConnected(int status) {
    switch (status) {
        case LoaderCallbackInterface.SUCCESS:
        {
                System.loadLibrary("detection_based_tracker");...}}
```

ademas antes de habilitar nuestra vista, hay que adaptarla a la resolución de la pantalla del dispositivo, En este caso:

```
mOpenCvCameraView.setMaxFrameSize(320, 240);

@Override
Public void onResume(){
    super.onResume();
    OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_6, this, mLoaderCallback);
}
```

.-Dentro del método onCameraFrame, trataremos de dibujar un rectángulo en caso de que el frame contenga la información que buscamos.

Todos los frames con rostros detectados van a ir a un array, donde se les añadirá la máscara antes de ser mandado para ser mostrado.

```
Rect[] facesArray = faces.toArray();
 for (int i = 0; i < facesArray.length; i++)
  Core.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(), FACE_RECT_COLOR, 3);
        return mRgba;
```

.-DetectionBasedTracker.class. Es la clase propia de OpenCv para reconocer patrones en frames, en este caso rostros. Accederemos a sus funciones creando en nuestra clase principal invocándolo en mNativeDetector, pasando como parámetro la ruta con el fichero de rostros reconocibles.

```
mNativeDetector = new DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);
```

Carpeta Plugins>Android

Todos los plugins necesarias en la integración Unity-OpenCV-Android, reflejados en el anexo.

Carpeta Plugins>Android>obj

La misma que la de nuestro proyecto de eclipse

Carpeta Plugins>Android>res

La misma que la de nuestro proyecto de eclipse(a destacar):

Assets\Plugins\Android\res>raw>lbpcascade_frontalface.xml

El fichero de rostros de frente donde se buscarán coincidencias en nuestros frames.

Carpeta Scripts>Call.cs

Si recibimos un toque llamar a la actividad Android no estática como en los ejemplos anteriores. Se realizará en nuestro método Update().

```
if (i < AndroidInput.touchCountSecondary){
        if (Application.platform == RuntimePlatform.Android) {
                        AndroidCallNonStatic ();
        }
```

Carpeta Scenes>1.unity

Escena con un objeto caller activo, con el script Call adjunto.

**.-aplicación que detecta tu rostro y pone una mascara>LlamarFD**

**Proyecto LlamarFD**

El anterior proyecto insertaba un rectángulo encuadrando los rostros encontrados. Este detectara los rostros de igual manera aunque esta vez encuadrara un archivo .png a modo de mascara.

La diferencia con el proyecto anterior se encuentra en el recurso que tendremos que añadir dentro de nuestra carpeta Assets,

Plugins>Android>res>drawable>mask.png

Este es el fichero que contendrá la mascara que insertaremos en el frame en tiempo real a modo de mascara en el rostro

Ademas, en nuestro proyecto de eclipse, en nuestra clase principal deberemos cambiar la funcionalidad para que inserte nuestra imagen:

.- Antes de fusionar nuestro jpg con la vista de cámara, debemos adaptarlo con el tamaño correcto:

```
Size sz = new Size(width,height);
Mat resizeimage = new Mat();
Imgproc.resize(img, resizeimage, sz);
```

.-En vez de nuestra función rectángulo, debemos llamar a overlayImage,

```
overlayImage(mRgba.submat(ROI),resizeimage,mRgba.submat(ROI));
```

.-Donde pixel a pixel se detectara si se debe introducir nuestro pixel externo o no:

```
if(alpha>0)
 {
  double infof[] = dst.get(y, x);
  output.put(y, x, infof);
 }
```

**.-aplicacion notificacion->Proyecto Notificaction**

**Proyecto notificación**

En este proyecto implementaremos una aplicación que programe una notificación en un dispositivo Android, para que se dispare dentro de un rango de tiempo dado, tanto en él mismo, como en todos los wereables que tenga anexados.

Assets>

Carpeta Plugins>Android>AndroidManifest.xml

Fichero de configuración de Android.. A destacar:

.-Esta vez nuestra actividad principal sera del tipo UnityPlayerProxyActivity.

```
<activity android:name="com.unity3d.player.UnityPlayerProxyActivity" android:label="@string/app_name"
android:configChanges="fontScale|keyboard|keyboardHidden|locale|mnc|mcc|navigation|orientation|screenLayout|
screenSize|smallestScreenSize|uiMode|touchscreen" >
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
```

Carpeta Plugins>Android>AndroidPugin.jar

Plugin con el código necesario para programar una notificación con los parámetros dados. Contiene el fichero NotificationView.class. Se creara desde AndroidStudio. A destacar:

.-En la definición de la clase vemos que extiende BroadcastReceiver. Esta es la clase que se usa cuando vas a hacer un Intent Broadcast en Android.

```
public class NotificationView extends BroadcastReceiver{}
```

.-El nombre, el titulo y el contenido de la notificación se pasaran junto con el tiempo al que queremos que se active (en segundos) como parámetros a la clase startAlarm:

*public static void startAlarm(String name, String title, String label, int secondsFromNow)*

.-Crearemos un intento de notificación en broadcast y programaremos el alarmmanager con los parámetros recogidos:

```
AlarmManager am = (AlarmManager)act.getSystemService(Context.ALARM_SERVICE);
    Intent ii =new Intent(act, NotificationView.class);
    ii.putExtra("name", name);
    ii.putExtra("title", title);
    ii.putExtra("label", label);
    am.set(AlarmManager.RTC_WAKEUP, alarmTime, PendingIntent.getBroadcast(act, 0, ii, 0));
```

La clase AlarmManager proporciona acceso a los servicios de alarma del sistema. Estos le permiten programar la aplicación para ejecutarse en algún momento en el futuro

.-Cuando la notificación se recibe, si la activas, la aplicación "padre" se arranca de nuevo, pudiendo volver a programar la misma notificación si fuera necesario.

Carpeta Plugins>Android>android-support-v4.jar

Es una Librería de soporte de Android que permite utilizar algunas de las APIs más recientes de Android en tu aplicación aunque esta se ejecute en versiones antiguas. En nuestro caso es necesario.

Carpeta Plugins>Android>res>view_notification.xml

Xml con los detalles de la notificación:
dentro de <LinerLayout..>

```
<TextView android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:text="Aqui detalles de la notificacion"/>
```

Carpeta Scenes>Demo.unity

Escena con el Canvas que contiene el botón que programa la alarma, junto con el objeto eventsystem(que hace que el UI sea interactivo).

Carpeta Scripts>LanzaNotificacion.cs

Script complemento con el método lanzado al presionar el botón. A destacar:

.-Crearemos nuestro AndroidJavaObject de la clase NotificationView contenida en nuestro .jar

```
AndroidJavaObject nativeObj =new AndroidJavaObject("com.rooftapps.demonotification.NotificationView");
```

.-Llamaremos a su función startAlarm con los parámetros necesarios para su funcionamiento. De esta forma se convierte en algo realmente simple programar una noticiario en Android:

```
nativeObj.CallStatic("startAlarm", new object[4]{"THIS IS NAME"
                                                 ,"THIS IS TITLE"
                                                 , "THIS IS LABEL"
                                                 , 10});
```

## 6.-Conclusiones y trabajo futuro

Después del trabajo desempeñado en el laboratorio puedo concluir que:

1º)Todas mis metas, tanto las mías propias como las del trabajo realizado, han sido alcanzadas y sobrepasadas. En caso de que fuera necesario el desarrollo de una aplicación por parte de la Universidad para las Google Glass, tendría las herramientas necesarias para confeccionar cualquier tipo de software para ellas, incluido la que integra Unity-Android-OpenCV.

2º)El dispositivo estudiado es un prototipo y no va a integrarse a corto plazo en nuestro dia dia, aunque ha sido un paso previo muy importante para afianzar la tecnología en un futuro. Se ha desarrollado bastante software para ellas por parte de muchas desarrolladoras independientes y ha permitido analizar su comportamiento, y más importante, el de los usuarios.

3º)De todas formas a contribuido a que la próxima generación de wereables entre con mas fuerza. Los smartwatch, e incluso las smartbands(pulseras inteligentes) han hecho su irrupción con fuerza gracias en parte a la contribución de Google poniendo ese dispositivo en boca de todos durante una temporada. También hay que tener en cuenta que el wereable para algunas tareas y para personas con ciertas restricciones físicas puede ser extremadamente útil.

Entiendo que el wereable está aquí para quedarse definitivamente, pero todavía adoptara muchísimas formas mas, y aunque en la actualidad solo cuente con un pequeño grupo de usuarios, siempre tendrá la posibilidad de ir probando nuevas funcionalidades hasta que una representa un avance significativo en nuestra productividad diaria.

Para finalizar me gustaría añadir que los trabajos que podría desempeñar actualmente tienen bastante presencia en el mercado presente, como son las aplicaciones Android, y mas teniendo en cuenta que después de disfrutar la beca podría añadirles funcionalidades tan innovadoras como son la compatibilidad total con wereables y realidad aumentada. Dentro de Unity también existe un pequeño mercado propio, Assets Unity Store, donde se pueden encontrar todo tipo de herramientas y recursos para el engine, incluido plugins Android, con lo que pienso que realmente ha sido muy provechosa la experiencia en todos los campos.

## 7.-Bibliografia


I-https://developers.google.com/glass/

Documentación oficial con las características de desarrollo para las google glass.

II-http://es.wikipedia.org/wiki/Google_Glass

Características generales de las google glass

III-http://unity3d.com/unity

Documentación oficial con las características del engine Unity3d

IV-http://es.wikipedia.org/wiki/Unity_(software)

Características generales de Unity

V-(http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado)

Características generales de Eclipse

VI-https://developer.android.com/tools/studio/index.html

Documentación oficial con las características de AndroidStudio

VII-http://www.androidpit.es/sdk-android

Características generales del sdk de android

VIII-https://developer.android.com/sdk/index.html

Documentación oficial con las características del sdk de android

IX-https://developers.google.com/glass/develop/gdk/

Documentación oficial con las características del gdk de las google glass

X-http://opencv.org/about.html

Documentación oficial con las características de OpenCV

XI-http://www.csscript.net/

Documentación oficial con las características de C#Script

XII-https://www.java.com/es/download/faq/whatis_java.xml
Documentación oficial con las características de Java

XIII-http://developer.android.com/guide/topics/ui/notifiers/notifications.html
Documentación oficial con las características de las notificaciones android

XIV-http://docs.unity3d.com/ScriptReference/Camera.html
Documentación oficial con las características del objeto Camera de Unity3d

XV-http://developer.android.com/guide/topics/manifest/manifest-intro.html
Documentación oficial con las características del AndroidManifest

XVI-http://developer.android.com/guide/topics/manifest/uses-sdk-element.html
Documentación oficial con los elementos que contiene el AndroidManifest

XVII-http://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html
Documentación oficial sobre el método Awake().

XVIII-http://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html
Documentación oficial sobre el método Start().

IXX-http://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html
Documentación oficial sobre el método Upodate().

XX-http://docs.unity3d.com/Manual/gui-Basics.html
Documentación oficial sobre el GUI básico de Unity3d.

XXI.https://fossies.org/dox/opencv3.0.0/interfaceorg_1_1opencv_1_1android_1_1CameraBridgeViewBase_1_1CvCameraViewListener2.html
Documentación sobre OpenCV

XXII-
http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/dev_with_OCV_on_Android.html
Documentación oficial sobre OpenCV-Android

XXIII-http://docs.opencv.org/java/org/opencv/android/BaseLoaderCallback.html
Documentación oficial sobre OpenCV-Android

XXIV-http://docs.opencv.org/java/org/opencv/android/NativeCameraView.html
Documentación oficial sobre OpenCV-Android

# A.-Anexos

## A.1.-Preparando el entorno Unity

Preparando el entorno
Introducción al Desarrollo en Android

Construir juegos para un dispositivo que tenga el sistema operativo Android requiere de un enfoque similar al del desarrollo para iOS. Sin embargo, el hardware no está completamente estandarizado a través de todos los dispositivos, y esto lleva a problemas que no suceden en el desarrollo para iOS. Existen algunas diferencias de características en las versiones Android de Unity de igual forma que con el caso de las versiones de iOS.

Configurar tu Ambiente de Desarrollo Android

Necesitarás tener arreglado tu ambiente de desarrollo Android antes de poder probar tus juegos sobre el dispositivo. Esto implica descargar e instalar el SDK de Android que contenga cada una de las diferentes versiones y plataformas de Android que serán trabajadas, y agregar su dispositivo físico al sistema (esto se hace de forma un poco diferente dependiendo de si el equipo tiene Windows o Mac).

Este proceso de configuración es explicado en la página web para desarrolladores de Android, y puede exista información adicional proporcionada por el fabricante de tu dispositivo. Dado que es un proceso complejo, hemos proporcionado un esquema básico de las tareas que deben ser completadas antes de que puedas ejecutar código en tu dispositivo Android o en el emulador de Android.

Sin embargo, lo mejor que puedes hacer es seguir las instrucciones paso a paso en el portal para desarrolladores Android.

1.-Descarga el SDK de Android

Ve a la página web de Android Developer SDK. Descarga y descomprime el último SDK de Android. http://developer.android.com/sdk/index.html

2.-Instalación del SDK de Android

Sigue las instrucciones.

(NOTA)->En el paso 4 de Installing the SDK asegúrate de agregar al menos una plataforma de Android que tenga un API level igual o superior a 9 (Plataforma 2.3 o superior), los Platform Tools y los USB drivers si estás utilizando Windows.

3.-Hacer que el dispositivo sea reconocido por el sistema

.-Activar el modo debugging en el dispositivo: usb debuggin on
.-Activar el adb

Esto podría ser complicado, en especial en sistemas Windows en donde los drivers tienden a ser un problema. También, tu dispositivo puede venir con información adicional o drivers específicos procedentes del fabricante.

Para Windows:

Si el dispositivo Android es automáticamente reconocido por el sistema, aún podrías estar necesitando la actualización de los drivers con los que vienen con el SDK de Android. Esto se hace a través del Administrador de Dispositivos en Windows. —>Si el dispositivo no es reconocido automáticamente, usa los drivers del SDK de Android, o cualquier driver específico proporcionado por el fabricante. —>Se puede encontrar información adicional en: Controladores USB para Windows

Para Mac:

Si estás desarrollando en Mac OSX, por lo general no se requieren drivers adicionales.
Nota: No olvides habilitar la opción "USB Debugging" en tu dispositivo, en Settings -> Developer options. A partir de Android Jelly Bean 4.2 las opciones de desarrollador están escondidas por omisión. Para habilitarlas, haz tap varias veces en Settings -> About Phone -> Build Version. Luego serás capaz de acceder a Settings -> Developer options.

Si no estás seguro si tu dispositivo está apropiadamente instalado en tu sistema, favor leer la página de resolución de problemas para más detalles.

4.-Agregar la ruta del SDK de Android a Unity

En la primera vez que construyes un proyecto para Android (o si Unity posteriormente falla en localizar el SDK), se te pedirá ubicar el directorio en donde se halla instalado el SDK de Android (debes seleccionar el directorio raíz donde está instalado el SDK). La ubicación del SDK de Android también puede ser modificada en el editor seleccionando Edit > Preferences y luego dando clic en External Tool sobre el cuadro de diálogo preferences.

Fuentes
http://docs.unity3d.com/es/current/Manual/android-GettingStarted.html

http://developer.android.com/sdk/index.html

## A.2.-Preparando el entorno para las GoogleGlass: GDK

Con estos pocos pasos instalaremos el GDK

Si no tienes experiencia previa con Android te recomendamos instalar el AndroidStudio.

1.Instalar el SDK Platform y el el Glass development kit preview para android 4.4.2(API 19). Lo demás es opcional.

Con AndroidStudio llegaras al SDK Manager con  Configure > SDK Manager.

Si no solo tienes que encontrar el SDK Manager en el ADK de Android

2.-En el dispositivo, ve a Settings > Device Info > Turn on debug para habilitar adb, lo que permite al sistema de desarrollo comunicarse con el.

3.-Conecta las gafas a tu sistema y da un toque en el pad para autorizar el "modo debug" necesario para instalar nuestro  propio software en un dispositivo Android.

## A.3.-Preparando el entorno para OpenCV-Android-Unity

OpenCV-Android

1.Descargar el SDK de OpenCV.Android(OpenCV-2.4.11-android-sdk)
http://sourceforge.net/projects/opencvlibrary/files/opencv-android/2.4.11/

Usando eclipse:
2. Desde File -> Import -> Existing project in your workspace.
y localiza nuestra instalación de OpenCV

3. Desde Project -> Properties -> Android -> Library -> Add selecciona OpenCVLibrary - 2.4.11 e indica que es una librería

4.En nuestro AndroidManifest debemos introducir:
los permisos necesarios:

```
<uses-permission android:name="android.permission.CAMERA"/>

<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
<uses-feature android:name="android.hardware.camera.front" android:required="false"/>
<uses-feature android:name="android.hardware.camera.front.autofocus" android:required="false"/>
```

y la configuracion de nuestro Theme, en este caso pantalla completa:

```
<application
 android:icon="@drawable/icon"
 android:label="@string/app_name"
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
```

Esto se deberá añadir a ademas en nuestro AndroidManifest en Unity3d.

Ahora ya podemos implementar una actividad que utilice todas las funcionalidades de OpenCV-Android.

Una vez comprobado en este punto que el código funciona, ya puede ser exportado como librería nativa para Unity3D.

OpenCV-Android-Unity

En nuestro nuevo proyecto de Unity3D:

1.-Introducir el jar resultante en Assets>Plugins>Android junto con nuestro nuevo AndroidManifest adaptado con lo visto anteriormente.

2.-Introducir las carpetas que contiene la libreria OpenCV-2.4.11-android-sdk. Estas son:
opencv-sdkjar
libopencv_java
libnative_camera_r2.2.0
libnative_camera_r2.3.3
libnative_camera_r3.0.1
libnative_camera_r4.0.0
libnative_camera_r4.0.3
libnative_camera_r4.1.1
libnative_camera_r4.2.0

3.-Introducir las carpetas res y obj de nuestro proyecto eclipse

## A.4.-Exportando plugin .jar para Unity Eclipse

1-Descargar Eclipse e instalarlo.

Link: https://eclipse.org/downloads/

2-Crear Nuevo AndroidProject

3-Implementar tu código y comprobar que funciona

4-Pulsa con el botón derecho sobre nuestro proyecto y seguir la ruta Properties/Android, donde marcaremos la casilla "is library"

5-Crea la carpeta libs e introducir el fichero classes.jar de nuestra propia instalación de Unity.

En Windows se encuentra en
C:\Program Files\Unity\Editor\Data\PlaybackEngines\androidplayer\bin\classes.jar.

En MAc y Linux
/Applications/Unity/Editor/Data\PlaybackEngines/androidplayer\bin\classes.jar.

6-Exportar el fichero jar clicando sobre el proyecto con el boton derecho y siguiendo la ruta Export>Java>JAR file.

7-El fichero .jar generado en la ruta especificada ya puede ser introducido en Unity como un recurso del proyecto en la carpeta Assets/Plugins/Android

## A.5.-Exportando plugin .jar para Unity AndroidStudio

Android Studio para generar plugins android

1-Descargar Android Studio e instalarlo.

https://developer.android.com/sdk/index.html

2-Crear Nuevo Proyecto

3-Seleccionar el build.grandle

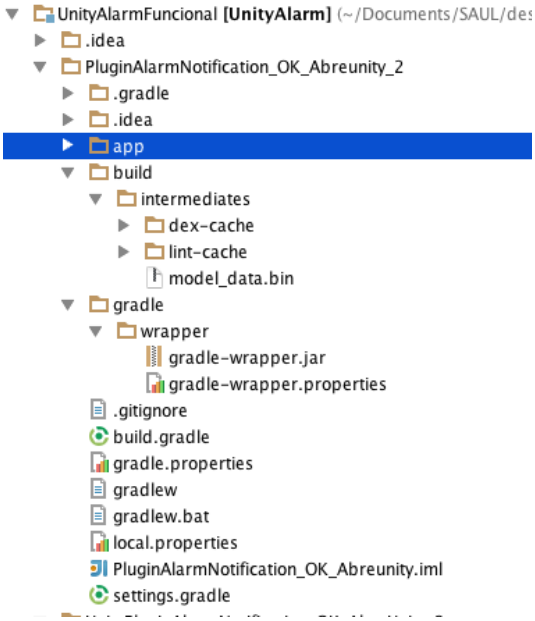


*Ilustración 21: dentro de la carpeta gradle*

4- Modificarlo con el código adjunto para añadir la propiedad exportar jar (con comentarios para facilitar su comprensión)

```
//indica que es una libreria
apply plugin: 'com.android.library'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"
    sourceSets {
        main {
            //Ruta hasta tu codigo fuente
            java {
                srcDir 'src/main/java'
            }
      }
}
    defaultConfig {
        minSdkVersion 13
        targetSdkVersion 21

    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.txt'
        }
    }
    lintOptions {
        abortOnError false
    }
}
```

```
//añadimos las 4 librerias de compatibilidad para evitar incompatibilidades en dispositivos //antiguos
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile 'com.android.support:support-v4:21.0.3'
    compile 'com.android.support:support-annotations:21.0.3'
    compile 'com.android.support:support-v13:21.0.3'
    compile files('libs/classes.jar')
}
//tarea para eliminar el plugin generado anterior
task deleteOldJar(type: Delete) {
    delete 'release/AndroidPlugin.jar'
}
//tarea para exportar los contenidos como un jar
task exportJar(type: Copy) {
    from('build/intermediates/bundles/release/')
    into('release/')
    include('classes.jar')
    ///le damos el nombre final, en este caso 'AndroidPlugin.jar')
    rename('classes.jar', 'AndroidPlugin.jar')
}
exportJar.dependsOn(deleteOldJar, build)
```

5-Añadir la libreria classes.jar de nuestra propia instalacion de Unity en nuestra carpeta libs.

En Windows se encuentra en
C:\Program Files\Unity\Editor\Data\PlaybackEngines\androidplayer\bin\classes.jar.

En MAc y Linux
/Applications/Unity/Editor/Data\PlaybackEngines/androidplayer\bin\classes.jar.

6-Implementar Plugin

Dentro de la ruta especificada anteriormente generar el código

7-Exportar .jar. Activar con un doble click la tarea "export jar", dentro de las gradle task del build.Gradle modificado anteriormente:

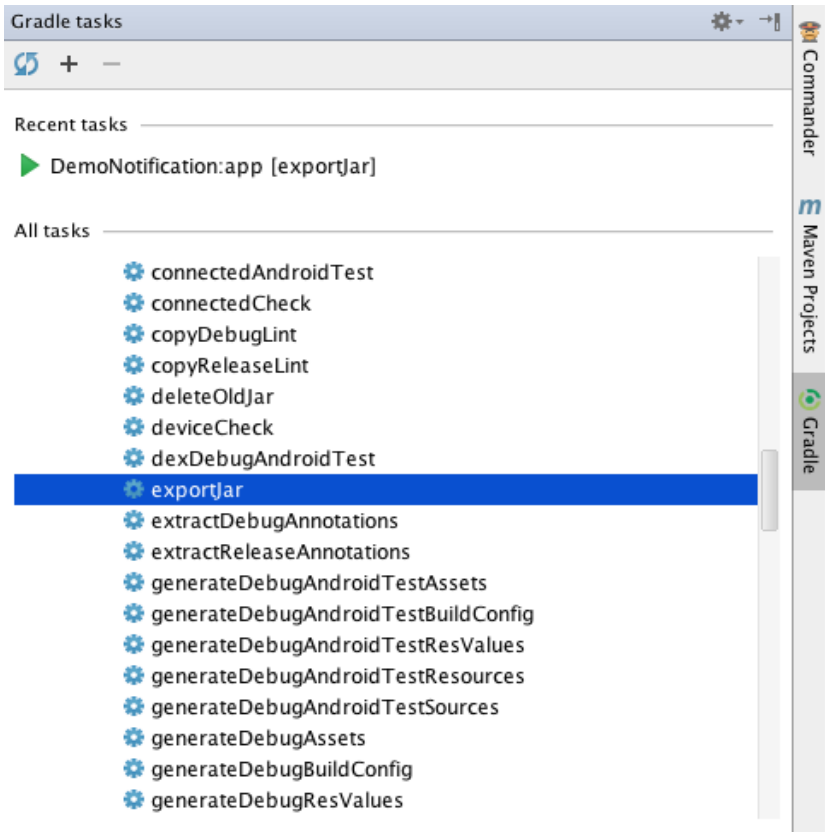


*Ilustración 22: posición de la nueva función*

8.El fichero 'AndroidPlugin.jar' se puede encontrar en la carpeta release y ya puede ser introducido como un recurso del proyecto de Unity en la carpeta Assets/Plugins/Android.