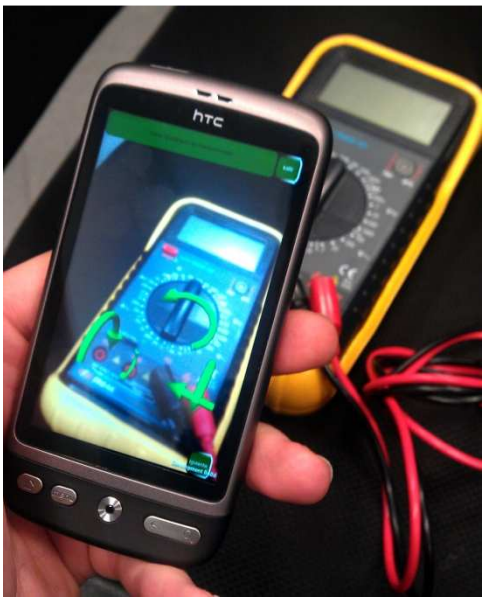
Universidad Rey Juan Carlos

Máster Oficial en Informática Gráfica,
Juegos y Realidad Virtual

PROYECTO FIN DE MÁSTER

PROpuesta de PROceso de PRODucción de Manuales en Entornos de Realidad Aumentada (PRO³MERA)



Autor: Carlos Alberto Catalina Ortega

Tutor: Marcos García Lorenzo

Septiembre de 2012

TÍTULO: PROPUESTA DE UNA ARQUITECTURA PARA LA CREACIÓN DE MANUALES EN ENTORNOS DE REALIDAD AUMENTADA.

AUTOR: CARLOS ALBERTO CATALINA ORTEGA

TUTOR: MARCOS GARCÍA LORENZO

La defensa del presente Proyecto Fin de Máster se realizó el día ____ de Julio de 2012; siendo calificado por el siguiente tribunal:

PRESIDENTE: _____

SECRETARIO: _____

VOCAL: _____

Habiendo obtenido la siguiente calificación

CALIFICACIÓN: _____

Presidente **Secretario** **Vocal**

RESUMEN

Los manuales de mantenimiento son un tema recurrente en la literatura de realidad aumentada (RA), siendo un producto muy deseable pero que aun no ha conseguido llegar a mercado. La gran mayoría de las iniciativas de manuales están dirigidas a mantenimiento de grandes maquinarias desarrolladas con software propietario.

El presente proyecto pretende crear un manual de mantenimiento con hardware (teléfono inteligente con *Android*) y software (*Unity3D* y *Vuforia*) actuales y de bajo coste. Además se propondrá un proceso de producción de manuales con una arquitectura abierta, extensible y simple.

La implementación del manual que se ha desarrollado, no se ha planificado como un simple ejemplo de manual o del proceso de producción propuesto, sino que proporciona toda la base para poder editar cualquier tipo de manual de mantenimiento de una forma simple y sin necesidad de conocimientos de programación, gracias a el editor de Unity3D.

Se espera con este proyecto proporcionar una base y una metodología para que este tipo de manuales puedan desarrollarse de forma sistemática reduciendo su coste y permitiendo que lleguen al gran público.

ABSTRACT

Maintenance manuals are a recurring topic in the literature of augmented reality (AR), being a very desirable product however it has not yet reached market. Most initiatives are aimed at maintenance manuals of large machinery with proprietary software.

This project aims to create a maintenance manual with hardware (smartphone with Android) and software (Unity3D and Vuforia) current and low cost. It also defines a manual production process with an open, extensible and simple arquitecture.

The implemented manual has not been planned as a simple example of manual or the proposed production process, but also provides all the base to edit any type of maintenance manual in a simple and without programming knowledge, thanks to the editor of Unity3D.

This project is expected to provide a base and methodology for this type of manual so it can be developed systematically reducing cost and allowing to reach all the public..

Índice de la memoria

Índice de ilustraciones

Universidad Rey Juan Carlos

Máster Oficial en Informática Gráfica,
Juegos y Realidad Virtual

PROYECTO FIN DE MÁSTER

PROpuesta de PROceso de PROducción de Manuales en Entornos de Realidad Aumentada (PRO^{3}MERA)

PRIMERA PARTE

DESCRIPCIÓN DEL PROYECTO
Y
ESTADO DEL ARTE

1 Introducción

1.1 Descripción del proyecto

La realidad aumentada (RA) nos permite decorar la realidad que podemos ver con nuestros ojos con información adicional que nos proporcione nuevos datos o ayuda sobre el entorno que vemos. Esa información debe estar posicionada y orientada con respecto al entorno de modo que sea consistente con lo que vemos.

La realidad aumentada tiene una gran cantidad de aplicaciones:

- Mantenimiento interactivo de máquinas.

Ilustración 1 - Realidad aumentada aplicada a industria [1] [2].

- Añadir en visores, fijos con imágenes de personajes históricos, reconstrucción de zonas, zona en otra épocas…

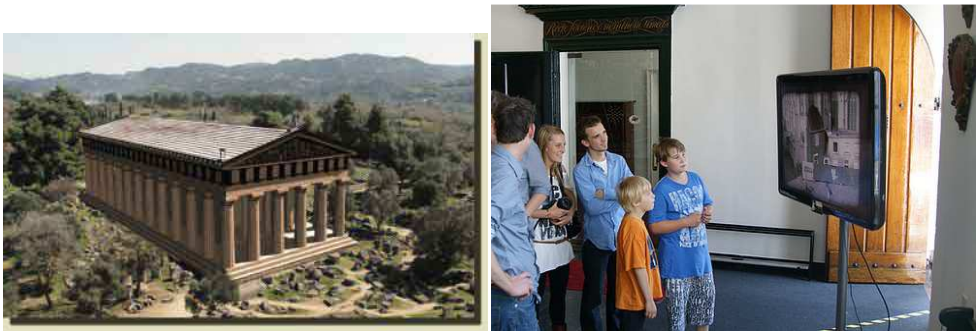


Ilustración 2 Realidad aumentada aplicada a turismo [3] [4].

- Catálogos 3D.

Ilustración 3 - Realidad aumentada aplicada a catálogos 3D [5] [6].

- Juegos interactivos.

Ilustración 4 - Realidad aumentada aplicada a juegos [7] [8].

- Publicidad.

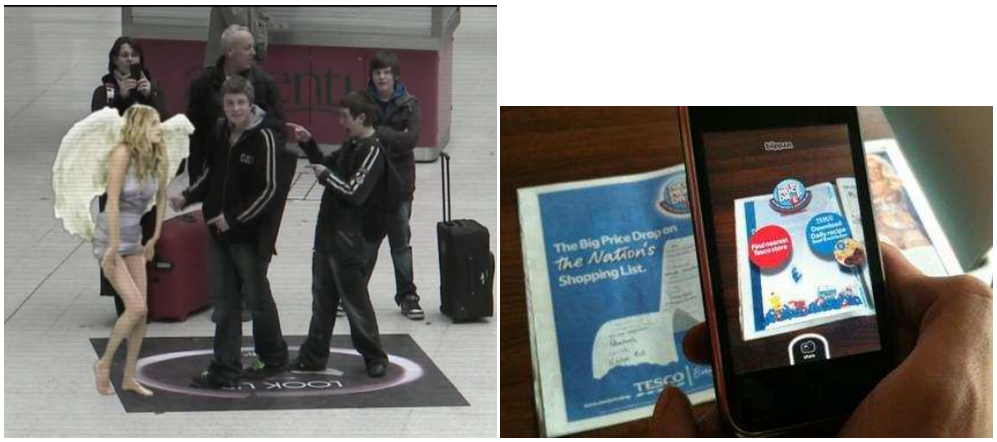


Ilustración 5 - Realidad aumentada aplicada a publicidad y marketing [9] [10].

- Formación interactiva.

Ilustración 6 - Realidad aumentada aplicada a formación [11] [12] .

- Decoración.

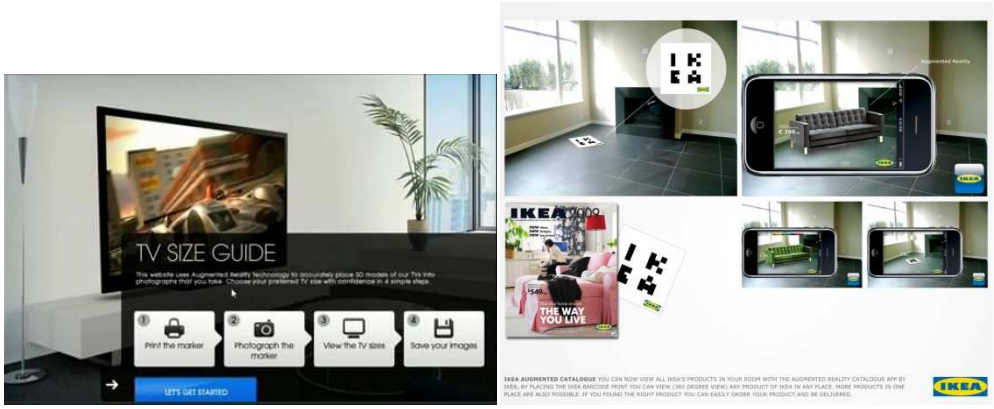


Ilustración 7 - Realidad aumentada aplicada a decoración [13] [14].

- Cirugía.

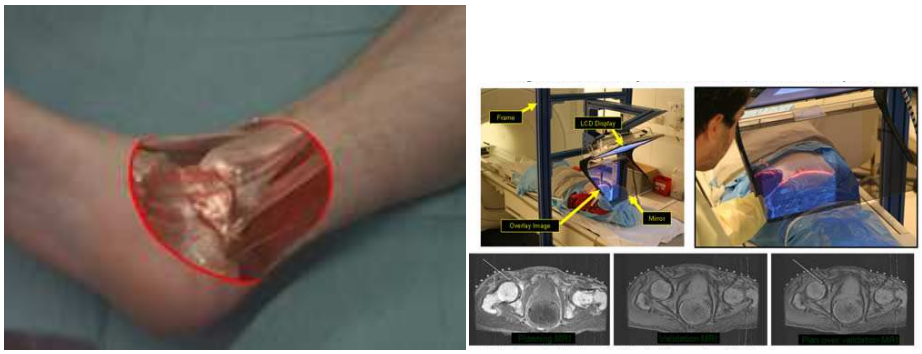


Ilustración 8 - Realidad aumentada aplicada a cirugía [15] [16].

- Localización de lugares o puntos de interés

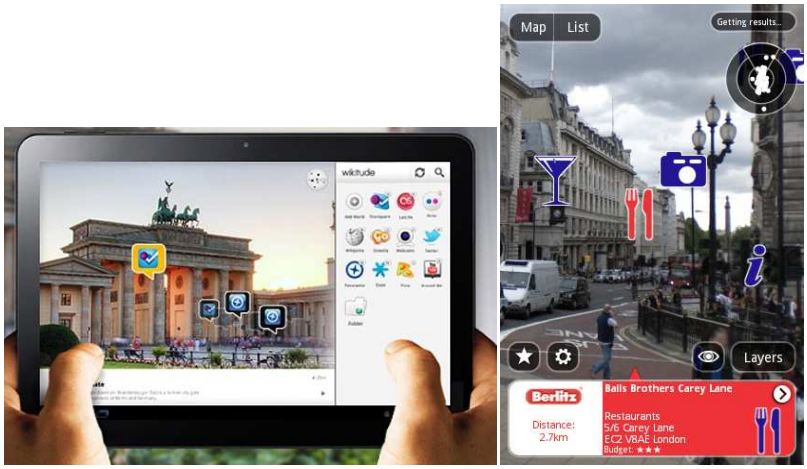


Ilustración 9 - Realidad aumentada aplicada a localización [17] [18].

- Y un largo etcétera…

En el presente proyecto se aplicará la tecnología de la Realidad Aumentada para la realización de manuales de mantenimiento y se creará una base completa de funcionamiento sobre la que se implementará un caso concreto de manual. Para la creación de dicha base, se propondrá un proceso completo de producción de manuales centrándose en las siguientes características: versatilidad, facilidad de creación, coste de desarrollo del manual y coste de los dispositivos hardware reducidos.

El proceso de producción incluye tanto una guía de realización, como una arquitectura genérica que se define, tras analizar que tipo de elementos son necesarios para cubrir la mayor cantidad posible de situaciones que puedan darse en la elaboración de un manual.

Una vez definida el proceso productivo y la arquitectura, se implementa el manual de mantenimiento. La implementación realizada no es solamente un manual concreto, sino que sirve de base para crear cualquier otro manual. El manual además se ejecutará sobre un teléfono inteligente garantizando la mayor versatilidad para demostrar cómo este tipo de productos pueden llegar al gran público.

1.2 Motivación

Los manuales de mantenimiento que utilizan tecnologías de realidad aumentada (RA) son un tópico recurrente en la literatura. El enfoque de dichos trabajos suelen estar orientados a mantenimiento de grandes maquinarias, utilizan software desarrollado a medida para el caso concreto y en muchos de ellos utilizando hardware muy costoso.

Los manuales de mantenimiento que hacen uso de la RA proporcionan muchas ventajas sobre los manuales tradicionales.

- Proporcionan información visual sobre el propio dispositivo del que queremos realizar el mantenimiento dándonos información posicionada.
- La probabilidad de cometer un error al pulsar un botón o realizar una acción disminuyen al estar las instrucciones claramente asociadas con el objeto sobre el que se realizan las operaciones.
- El manual además nos guía sobre el orden de los pasos y nos puede proporcionar advertencias de seguridad.
- Debido a la forma de presentar las instrucciones del manual este puede ser utilizado por una persona no experta.
- Si pensamos en un manual en un dispositivo móvil además podremos tener disponibles los manuales en cualquier situación (en casa para programar un video, en el coche para mirar el nivel del aceite…)

Todas estas ventajas hacen que los manuales con realidad aumentada sea un producto muy deseable. Sin embargo, presentan varios problemas técnicos y económicos, en el desarrollo de este proyecto se pretende proporcionar herramientas para poder evitar algunos de los problemas que surgen en la elaboración de estos manuales.

El primer problema y uno de los más importante, es reconocer adecuadamente el entorno. Debemos saber hacia dónde estamos mirando (o mejor dicho hacía donde mira el dispositivo de realidad aumentada) para poder proporcionar información precisa y posicionarla correctamente.

El segundo es el coste de creación de los manuales, que está formado por la base funcional y por las instrucciones del propio manual. La creación de un desarrollo de software específico y completo para la base del manual hace que la creación de los manuales sea costosa. A esto hay que unirle el coste de editar cada una de las operaciones de mantenimiento del manual.

Finalmente en muchos sistemas se utiliza hardware con un coste muy alto (sistemas de tracking externo, cascos de realidad virtual o aumentada…). Este alto coste no permite que los manuales con esta tecnología puedan llegar al público general y que por lo tanto no se abaraten sus costes.

La motivación de este proyecto es poder dar un paso en la creación de este tipo de manuales de forma que puedan llegar al mercado de consumo y para ello se estudian formas de mitigar los problemas anteriormente indicados:

- Definiendo un hardware de base asequible: Un teléfono inteligente con un Sistema Operativo Android.
- Realizando en el estudio del estado del arte un análisis de las librerías actuales que permiten realizar tracking de objetos y marcas y seleccionando aquellas que mejor se ajustan a los requisitos de nuestro sistema.
- Se define un "proceso de producción" de manuales de forma simple y se implementa el ejemplo sobre un motor gráfico creando varias utilidades que permitan que la edición de manuales pueda ser barata en términos de horas hombre.

1.3 Objetivos

El objetivo final del proyecto es crear un proceso de producción de manuales de mantenimiento e implementar un manual para un dispositivo concreto, sobre un hardware comercial y asequible. El ejemplo servirá como demostración de la viabilidad técnica y económica de crear manuales de con esta tecnología para el publico general.
Para conseguir dicho objetivo, la base del manual que se cree se realizará de una forma genérica. Igualmente la definición de proceso productivo y arquitectura serán lo más flexibles posible.

De forma concreta los objetivos del proyecto son los siguientes:

- Estudiar los proyectos y artículos sobre manuales de realidad aumentada para poder proponer mejoras en algún punto clave de estos.
- Implementar un manual sobre un dispositivo portable tipo teléfono inteligente.
- Desarrollar una arquitectura que permita gran versatilidad en los manuales.
- Definir un proceso de producción completo que sirva como guía para implementar la arquitectura anteriormente definida.

- El manual creado debe proporcionar una base para el desarrollo de otros manuales.
- Uso de hardware y software de bajo coste.

1.4 Contribución

Como se podrá ver en el apartado de estudio del arte, ya existen varias iniciativas para definir arquitecturas y aplicaciones de manuales de mantenimiento de Realidad Aumentada. En este trabajo, se pretende definir un "proceso de producción" simple y comprensible que permita diseñar y crear manuales de mantenimiento. Igualmente gracias a la implementación concreta se evaluará la facilidad de implementar, sobre dicha arquitectura, manuales mediante herramientas relativamente sencillas de utilizar y accesibles para todo el mundo. Las herramientas principales de desarrollo (Unity3D para Android y Vuforia de Qualcomm). Estas herramientas tienen un coste bajo, pudiendo ser adquiridas desde 280€.

Se ha huido de software desarrollado a medida, con gran cantidad de horas hombre, de librerías con alto coste o de hardware complejo. Aun así se ha intentado no sacrificar funcionalidad y se deja el sistema preparado para ampliaciones de funcionalidad.

Sobretodo y como visión diferenciadora este sistema puede ser utilizado para implementar manuales de mantenimiento o uso de bajo coste: cómo utilizar un video, operaciones básicas de revisión en un vehículo…

1.5 Estructura de la memoria

En el segundo capítulo se estudia la literatura existente relacionada con las distintas partes del proyecto, se analizan diversos artículos relacionados con manuales de mantenimiento que utilizan realidad aumentada y posteriormente sobre algoritmos o librerías que permiten realizar seguimiento de marcadores o de objetos.

A partir de este punto se irán describiendo las distintas partes que nos permitirán crear un manual de mantenimiento. Primero se describe el proceso de producción así como la arquitectura propuesta. Una vez definido dicho proceso y arquitectura se describe la implementación concreta del manual de mantenimiento.

Una vez vista la implementación concreta se indica cómo utilizar el desarrollo del manual como base para poder crear otros manuales de cualquier temática.

Finalmente se muestran los resultados de rendimiento del manual sobre un dispositivo real.

2 Estado del arte

A continuación se realizará un pequeño resumen del estado del arte cubriendo tres temas relacionados con el proyecto.

Primero se estudia la literatura existente sobre manuales de mantenimiento con realidad aumentada para poder tener una idea clara del tipo de implementaciones realizadas en el pasado. Se analizan de forma general sus ventajas y sus inconvenientes para intentar mantener sus puntos fuertes y mejorar sus puntos débiles.

Después se estudiarán los distintos artículos y algoritmos que existen sobre reconocimiento de imágenes u objetos 3d. Se pretende en esta primera búsqueda seleccionar un algoritmo o librería que permita que el manual tenga la mayor versatilidad posible a la hora de reconocer entornos. Esta primera búsqueda proporciona información para decidir si se decide implementar algún algoritmo genérico o si se puede portar la funcionalidad de alguna librería a un dispositivo móvil.

Tras el estudio de artículos y algoritmos genéricos de reconocimiento, se pudo comprobar, que su aplicación a un manual genérico para reconocer objetos 3d no es suficientemente versátil y en los casos en los que puede serla, la capacidad de procesamiento necesaria es mayor de la que puede proporcionar un teléfono inteligente actual. Por ese motivo y pensando en una implementación real, se realiza un estudio no exhaustivo de las librerías de realidad aumentada comerciales y disponibles actualmente para plataformas móviles, evaluándose sus características. Como desventaja todas las librerías permiten reconocer solamente marcadores por lo que, de forma general, será necesario añadirlas al dispositivo. Lo que pretende con este último punto es seleccionar la librería comercial de realidad aumentada para el caso de uso de la implementación de la arquitectura.

2.1 Manuales de mantenimiento

A continuación, se ven varios proyectos que proponen distintas soluciones arquitectónicas y modelos de trabajo:

AMRA: Augmented Reality assistance in train maintenance tasks [1] propone una arquitectura basada en XML dividiendo cada acción a realizar en pequeñas descripciones en las que pueden ser asociadas imágenes, video o modelos 3d.
La arquitectura divide el sistema en 3 partes: prerrequisitos, media disponible y pasos. Se realiza un estudio de las operaciones más comunes que consisten en *"unscrewing nuts and removing the covers/cables/connectors maintained by these nuts"* para cubrir dicho funcionamiento generan animaciones en VRML que pueden usar por separado o en conjuntos.

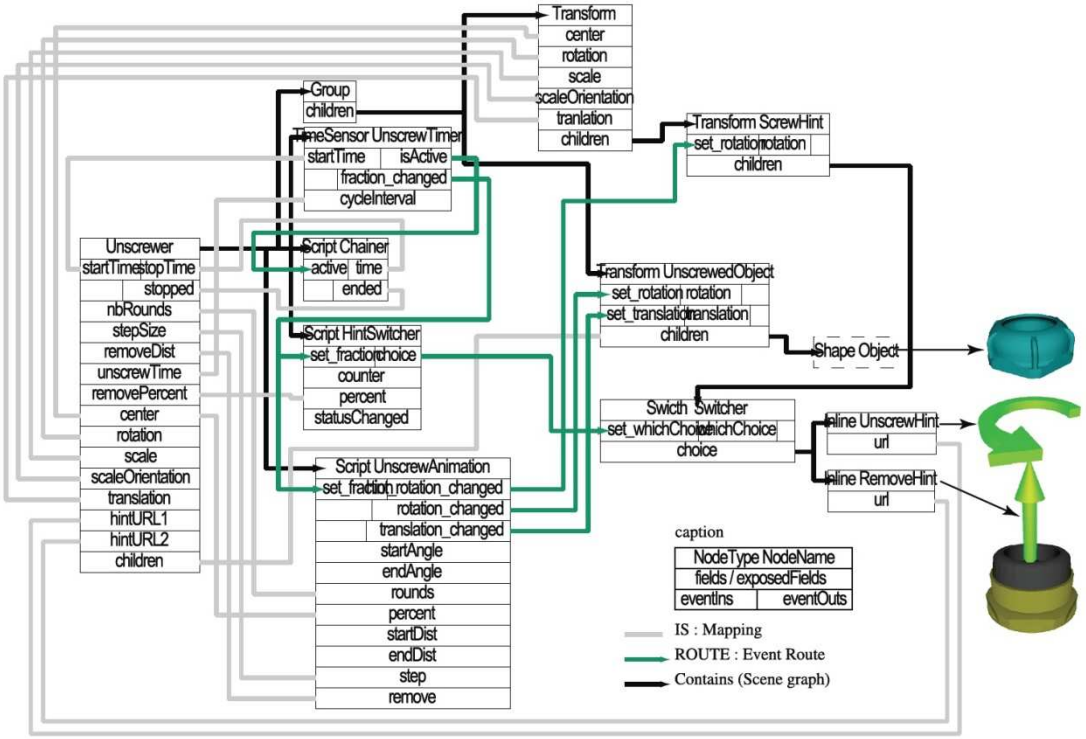


Ilustración 10 - Unscrewer photo arquitecture.

El proceso es versátil y permite agrupar funcionalidades que se puedan aplicar juntas pero la edición parece costosa. El proyecto, finalizado en 2009, pretendía utilizar un Tablet PC para la visualización de los manuales. El hardware aunque no deja las manos libres ni es tan inmersivo como un casco de realidad virtual es mucho más versátil. Por falta de capacidad de procesado el proyecto realmente se realizaba con un pc remoto y un monitor TFT.

En [19] se propone la edición de manuales mediante el paso de lenguaje natural simplificado a instrucciones en XML. Las instrucciones se dividen en muchas pequeñas partes para indicar las acciones a realizar, referencias… combina el análisis del lenguaje natural con referencias espaciales.

```
<instruction>
  <action_phrase>
    <action>
      <verb_remove/>
    </action>
  </action_phrase>
  <tool tool_id="t_001" size="PH2"/>
  <target_phrase>
    <target>
      <part part_id="p_020"/>
    </target>
    <target>
      <part part_id="p_021"/>
    </target>
    <qualifier>
      <qual_rel rel_pron="that">
        <rel_clause>
          <verb_phrase>
            <verb>
              <verb_hold/>
            </verb>
          <verb_phrase>
          <noun_phrase>
            <noun>
              <part part_id="p_010"/>
            </noun>
          </noun_phrase>
        </rel_clause>
      </qual_rel>
    </qualifier>
  </target_phrase>
</instruction>
```

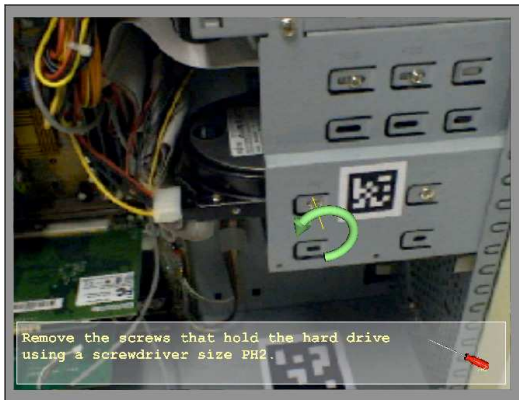


Ilustración 11 - Manual basado en análisis del lenguaje natural.

Knoepfle, Ch.; Weidenhausen, J.; L., Chauvigné; Stock, I [20] proponen un enfoque de proyecto muy similar al aquí descrito, dividiendo las operaciones igualmente en pequeñas unidades. El sistema sin embargo se basa en la creación de plantillas con operaciones que son interpretadas *"Next, the worker should remove the screw with the ratchet"* que proporcionan instrucciones.

Según sus propias conclusiones el proyecto permite ahorar mucho tiempo en la edición gracias a las plantillas, como desventaja la edición para posicionar los objetos es muy laboriosa. El sistema solamente promueve una arquitectua y realmente se ejecuta sobre el visualizador de realidad aumentada creado en el proyecto ARVIKA

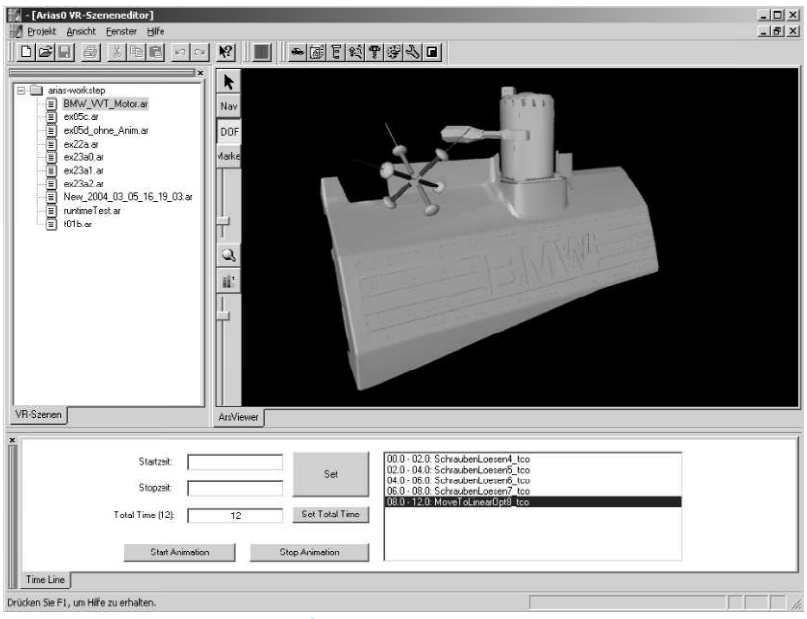


Ilustración 12 - Imagen del editor.

AMIRE [21] basa su edición en ficheros XML pero propone una edición mediante herramientas de realidad virtual que simplifican el posicionmiento de los objetos. Si bien el enfoque parece adecuado puesto que permite edición sobre el propio objeto esta requiere estar cerca del objeto del que queremos editar un manual. Si por ejemplo queremos realizar el manual de un coche o de una máquina grande deberemos transaladra a los técnicos al lugar en el que haya una unidad de dicho elemento.

Ilustración 13 - Amire en modo edición.

Hay más artículos con temáticas similares [22], [23], [24] y la mayoría proponen un modelo basado en XML como modelo de edición o almacenado de datos.

2.2 Reconocimiento de marcas u objetos 3d

Se hace un recorrido en este apartado por los distintos artículos y algoritmos que se han encontrado sobre reconocimiento de imágenes u objetos 3d. Se pretende en esta primera búsqueda seleccionar un algoritmo o librería que permita que el manual tenga la mayor versatilidad posible a la hora de reconocer entornos. Esta primera búsqueda proporciona información para decidir si se decide implementar algún algoritmo genérico o si se puede portar la funcionalidad de alguna librería a un dispositivo móvil.

Monocular Model-Based 3D Tracking of Rigid Objects: A Survey [25] es una gran recopilación de las distintas técnicas que se pueden aplicar para el reconocimiento de objetos 3d o imágenes, si bien no proporciona ninguna implementación concreta, es un recurso importante para conocer una gran cantidad de técnicas y poder estudiar las más prometedoras. En concreto los capítulos 4 *Natural Features, Model-Based Tracking* y 5 *Tracking by Detection* son los más relacionados con los intereses de este proyecto.

En [26] se propone el seguimiento de objetos 3d reconociendo sus bordes, el sistema es estable si el objeto tiene suficientes bordes característicos, si la escena esta pobremente "texturizada" y si el objeto produce bordes con alto contraste. Si no es así el algoritmo se vuelve lento y tendrá poca precisión. Debido a la falta de versatilidad no se puede utilizar este algoritmo en el proyecto.

Ilustración 14 - Resultados del seguimiento de línea productiva y de camión de juguete.

El enfoque utilizado en [27] mantiene la detección de líneas pero añade el estudio de qué elementos deben estar en el objeto y que elementos no deben estar para poder detectar mejor los objetos. Usan imágenes estéreo para el reconocimiento y la definición de la geometría que realizan parece compleja al tener que definir vértices, líneas…

Finalmente los autores avisan de que el proceso puede tener problema con la complejidad de los objetos o el número de estos que se deben reconocer.

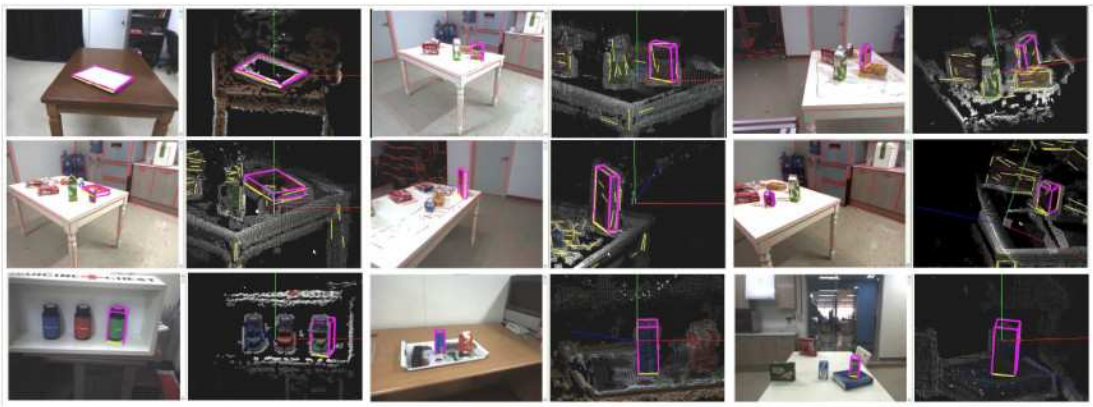


Ilustración 15 - Resultados del reconocimiento con evidencias positivas y negativas.

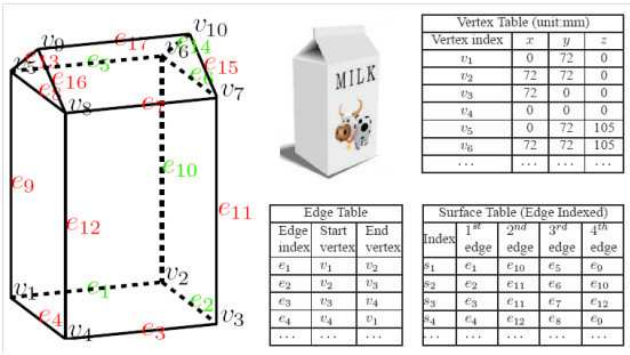


Ilustración 16 - Definición de la geometría del objeto.

El artículo *Viewpoint-Aware Object Detection and Pose Estimation* [28] presenta una aproximación, sin utilizar el extendido reconocimiento de patrones 2D, a la detección y estimación de posición. El sistema crea una representación 3D mediante una nube de puntos del modelo y luego realiza una búsqueda en un espacio de 6D para estimar la transformación del mismo.

Ilustración 17 - Construcción del modelo.

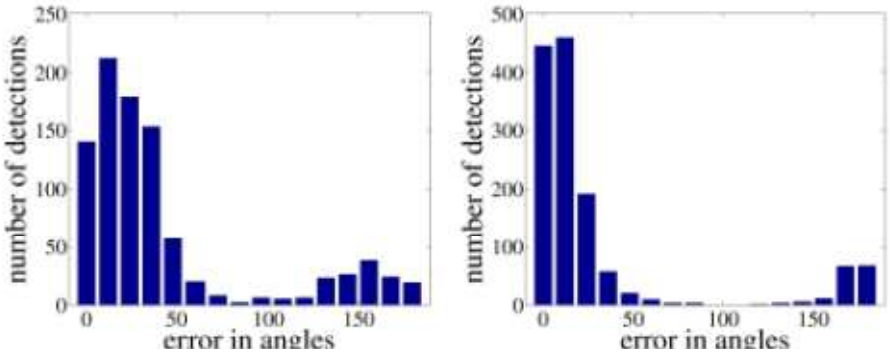


Ilustración 18 - Imágenes del modelo 3d y errores de orientación.

Pedram Azad, Tamim Asfour y Rudiger Dillmann proponen en [29] también un reconocimiento de objetos por su forma, obteniendo los datos mediante imágenes estéreo pero procesadas mediante GPU. El sistema combina la triangulación estéreo con la correlación basada en apariencia. El sistema puede

reconocer varios objetos proporcionando una buena posición y orientación. El tiempo de procesado es relativamente alto 20ms y solamente se puede utilizar con objetos de un color debido al proceso de segmentación de la imagen utilizado.

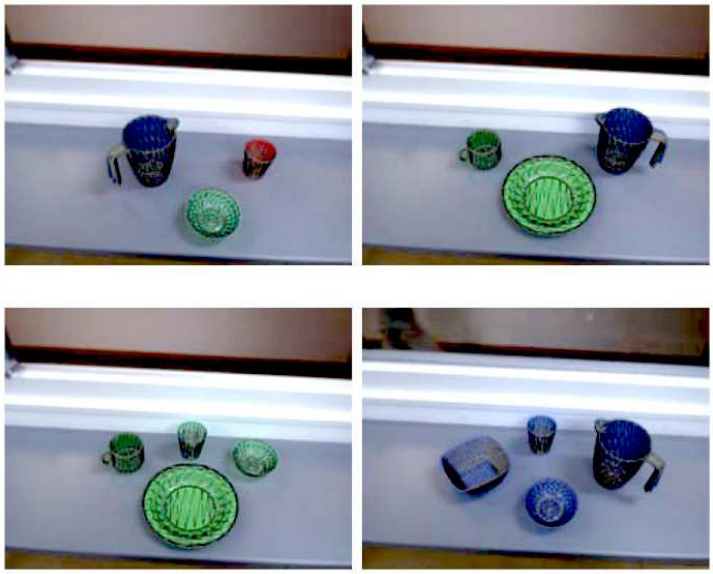


Ilustración 19 - Resultados del reconocimiento y estimación de posición.

Multi-View Object Class Detection with a 3D Geometric Model [30] utiliza el mismo enfoque de reconocer a través de imágenes 2D como los dos anteriores pero con una referencia de entrada de la geometría 3D a buscar. Esta misma idea se aplica en [31]

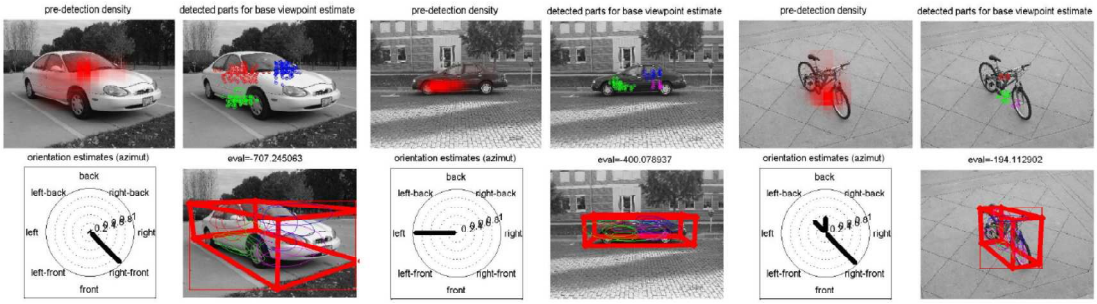


Ilustración 20 - Reconocimiento de diversos objetos.

El último artículo estudiado [32] combina reconocimiento 2d y 3d utilizando el aprendizaje de elementos invariantes sin embargo este está más orientado a reconocer clases de objetos que no objetos individuales (coches y no un coche en concreto).

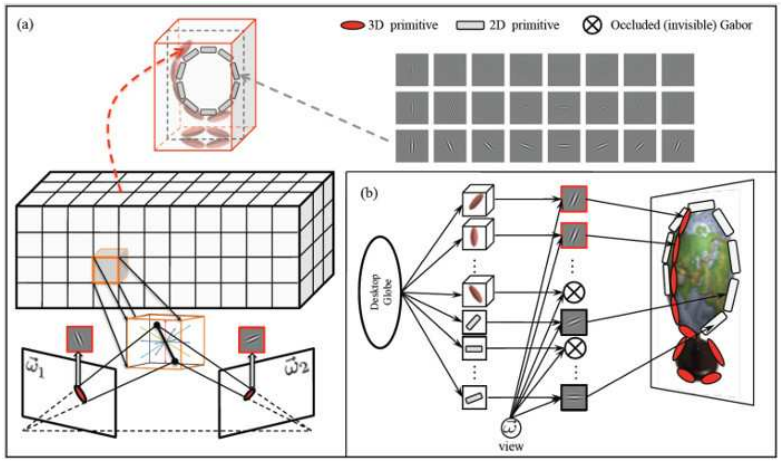


Ilustración 21 - Reconocimiento con elementos invariantes.

Finalmente se encontraron los trabajos de Vicent Lepetit con varias publicaciones en reconocimiento y seguimiento de objetos en tiempo real [33] [34] [35] [36]. Siguiendo estos trabajos se encontró los artículos de su alumno de doctorado Stefan Hinterstoißer que contaba con tres artículos muy interesantes, como ventaja adicional Stefan proporcionaba los códigos fuente de parte de sus proyectos lo que permite compilar y probar de forma simple sus investigaciones. http://campar.in.tum.de/personal/hinterst/index/downloads.html LEOPARD [37] y GEPARD [34] y DOT [38].

Como último paso del estudio del estado del arte se decidió probar estos desarrollos y evaluar su aplicación al proyecto al ser los más prometedores.

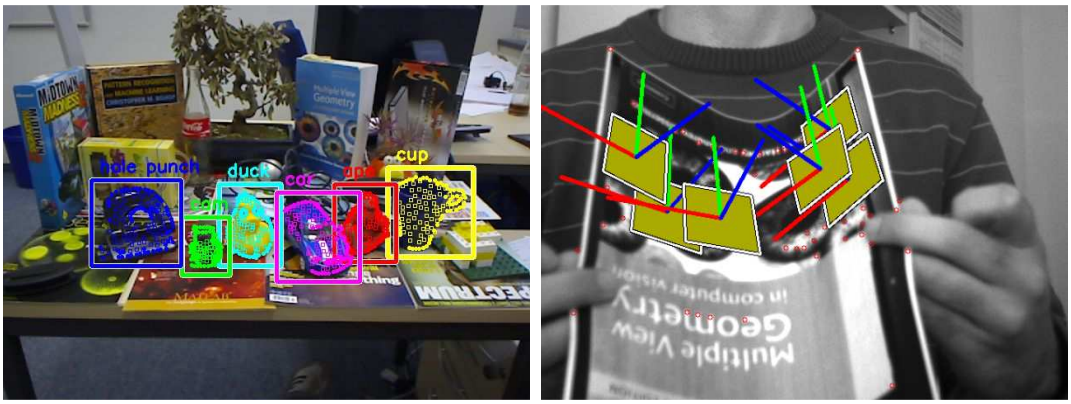


Ilustración 22 - Imágenes de los trabajos de Stefan Hinterstoisser.

GEPARD

Esta fue la primera librería a probar puesto que parece poder proporcionar un tracking de objetos adecuado para utilizar en el manual. Se compilaron las librerías de GEPARD consiguiente un funcionamiento adecuado. Sin embargo el procesado es bastante lento, si se tienen almacenadas 3 "zonas" a reconocer los

fps de la compilación disminuyen en gran medida (2-6 fps). Sin marcas a una resolución de 480*320 proporciona unos 17fps sobre un ordenador portátil con procesador core I7 a 2.2 GHz.

Ilustración 23 - Sin marcadores 15-16 fps.

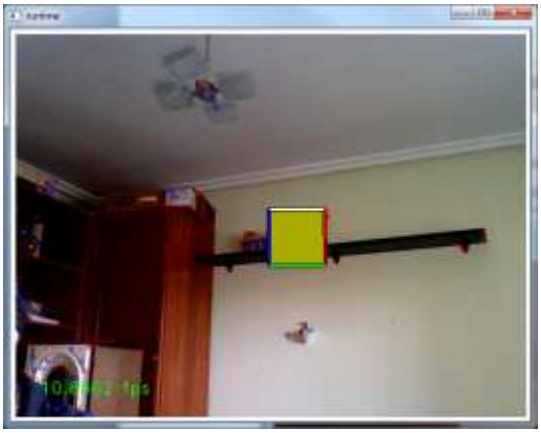


Ilustración 24 - Con un marcador 10 fps.

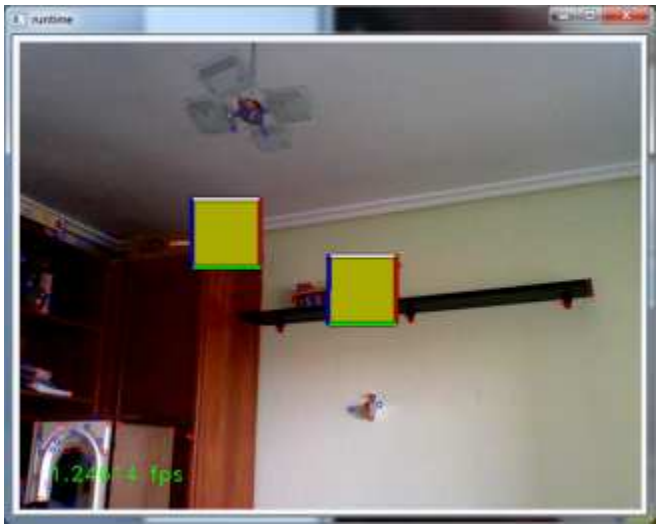


Ilustración 25 - Con dos marcadores de 2-5 fps.

El rendimiento indicado en el artículo es mayor debido a que el proyecto tiene integradas librerías de optimización IPP para varios procesadores. Dichas librerías no se han podido encontrar de forma libre por lo que solo se ha probado la versión sin optimizar que en GEOPARD es fácil de eliminar.

Las versiones mejoradas (LEOPARD, DOT) tienen muy integrados las librerías IPP dependiendo de ellas en la mayoría de las llamadas, parece complejo eliminar dichas librerías. Sin dichas librerías el rendimiento seguro que disminuirá en gran medida y puesto que el objetivo del proyecto es su uso en dispositivos móviles no se ve factible su uso.

2.2.1 Librerías de reconocimientos de marcas para dispositivos móviles

Tras el estudio de algoritmos genéricos de reconocimiento de objetos 3D no se ha conseguido encontrar uno que permita un reconocimiento robusto y que pueda funcionar adecuadamente en dispositivos móviles de bajo coste. Por ese motivo se procede a realizar un estudio comparativo con las librerías de reconocimiento de marcas que se han encontrado disponibles en el mercado que funcionen sobre dispositivos móviles y que tengan un coste bajo. El documento presentado a continuación, no es un estudio exhaustivo debido a que no es posible cubrir todas las librerías existentes en el mercado, tampoco es posible repetir con exactitud las condiciones de cada librería debido a que cada una reconoce marcadores de un tipo con sus ventajas e inconvenientes. Además en muchos casos la resolución de la imagen capturada por el dispositivo no se puede conocer ni es parametrizable.

El estudio permite escoger cual será la librería de reconocimiento de marcas que se utilizará en la implementación del manual.

Para la realización de las pruebas se establece una marca de 3cm de lado este tamaño permitiría colocar una marca relativamente discreta en muchos dispositivos de uso cotidiano (televisiones, teléfonos fijos…). Aun así el tamaño es independiente de la evaluación, se medirá la distancia máxima a la que el sistema puede reconocer la marca y la robustez en el reconocimiento como parámetros principales.

A continuación se enumeran las librerías y dispositivos utilizados para las pruebas:

- PDA Asus P535 a 520 MHz con *ArtoolkitPlus*
- HTC con *Andar* (*Artoolkit* adaptado)
- PDA Asus P535 a 520 MHz con *Studierstube Tracker*
- Iphone 4 con *StringAR*
- HTC desire con librerías *Vuforia*

Evaluación de las librerías. Con cada librería y dispositivo se evaluará la distancia la que reconoce la marca. Además se valorarán parámetros no puramente de rendimiento tales como: disponibilidad, mantenimiento, continuidad de la librería…

Varias de las librerías no están disponibles para sistemas operativos de móviles (Android, iOs, Windows Mobile Phone) aun así se evalúan como punto de comparación.

Librerías de *Artoolkit Plus* (*WM5*)

Las librerías funcionan solamente sobre *Windows Mobile 5.0*, no se conoce versión portada a Android. La resolución de la cámara es de tan solo 320*240. Aun siendo una librería relativamente antigua su rendimiento y funcionamiento es muy bueno.

Ventajas:

- Reconocimiento muy bueno.
- Gran distancia de reconocimiento de marcas.
- Buen rendimiento.
- Robusto.

Desventajas:

- No hay versión para sistemas operativos móviles actuales.
- La librería no se sigue soportando, sus desarrolladores pasan a la librería de pago *Studierstube*.

Conclusión:

Las librerías funcionan bien. Son muy similares a *Vuforia* que se verán más adelante pero con menos problema de iluminación.

No se puede utilizar en la implementación puesto que no cuenta con versión para plataformas actuales.

ANDAR (*Artoolkit*)

Esta librería es la portada a Android por parte de *Artoolworks* la empresa que comercializa las librerías de *Artoolkit*. Las librerías funcionan rápido y reconocen las marcas desde muy lejos. El problema que tienen es que reconoce casi cualquier cosa como una marca.

Ventajas:

- Gran distancia de reconocimiento de marcas.
- Buen rendimiento.
- Cuentan con versión para *Android*.

Desventajas:

- No son robustas, reconocen casi cualquier cuadrado negro como un marcador.

Conclusión:

No se puede utilizar en la implementación debido a su falta de robustez en el reconocimiento de las marcas.

Studierstube Tracker (*WM5*)

De las 5 librerías analizadas es la que mejor funciona, en cuanto a distancia de reconocimiento y robustez. Al igual que sucede con *ArtoolkitPlus* solo está disponible para *Windows Mobile 5*.

Ventajas:

- La de mayor distancia de reconocimiento de marcas.
- Buen rendimiento.

Desventajas:

- No cuenta con versión para sistemas operativos móviles actuales.

Conclusión:

No se puede utilizar en la implementación debido a que no está disponible para sistemas operativos móviles actuales.

String AR (*Iphone*)

Las que mejor funcionan junto con *Studierstube tracker*, con la diferencia de que esta librería solo reconoce imágenes. Solo está disponible para *Iphone* y no cuenta con versión libre.

Ventajas:

- Gran distancia de reconocimiento de marcas.
- Buen rendimiento.
- Robusto.
- Integrado con el motor gráfico *Unity 3D*.

Desventajas:

- No puede hacer seguimiento de marcadores, solo imágenes por lo que puede ser complejo utilizarlo en manuales que requieren seguimiento de varias partes del mismo, por ejemplo 20 marcadores.
- Solo están disponibles para *Iphone*.
- No hay versión gratuita (solo una de desarrollador a partir de 90$)

Conclusión:

No permite los marcadores con identificador por lo que no sirve para cualquier manual. Sin embargo puede ser una muy buena opción para la implementación de un caso concreto de manual.

Vuforia (marcadores)

Librerías de Qualcomm para reconocer marcadores. Permite el reconocimiento desde una gran distancia es robusto y tiene buen rendimiento. Además está desarrollado directamente para Android e Ipone y cuenta con un paquete que permite su rápida integración en el motor gráfico *Unity3D*.

Ventajas:

- Gran distancia de reconocimiento de marcas.
- Buen rendimiento.
- Bastante robusto.
- Muy optimizado para funcionar sobre dispositivos Android con procesador de Qualcomm aunque también funcionan en dispositivos que no tengan dicho procesador.
- Integrado con Unity 3D.

Desventajas:

- Problemas al reconocer varias imágenes según las condiciones de luz debido a los colores que forman sus marcas (negros, blancos y grises) necesita diferenciar 3 niveles de color en vez de 2 como el resto de librerías de marcas. Parte del equipo de trabajo de *ArtoolkitPlus y Studierstube* trabajan ahora en estas librerías pero solamente han incluido el tipo de marcas denominado *Frame-Marker*.

Conclusión:

Esta librería es la mejor opción por rendimiento, características y , aun con el problema con la luz.

Además, tal y como se verá en el siguiente punto, estas librerías están preparadas para reconocer imágenes en vez de patrones lo cual puede ser útil para ciertos manuales.

Vuforia (imágenes)

La librería *Vuforia* permite la utilización de imágenes como marcadores, las imágenes deben tener bastantes detalles y contraste para que sean reconocibles. El funcionamiento es estable y bastante rápido, una vez localizada la marca permite alejarnos mucho de la misma.

Ventajas:

- Reconocimiento rápido de imágenes.
- Muy optimizado para funcionar sobre dispositivos Android con procesador de Qualcomm.
- Integrado con Unity 3D.

Problemas:

- No puede utilizar cualquier imagen como marcador, solo aquellas con suficientes detalles.
- No puede realizar tracking de muchas marcas al mismo tiempo.

Conclusión:

Puede ser utilizado siempre que el manual necesite el seguimiento de pocas imágenes.

Imágenes comparativas de las librerías estudiadas con la máxima distancia a la que se realiza el tracking de la marca de 3cm:

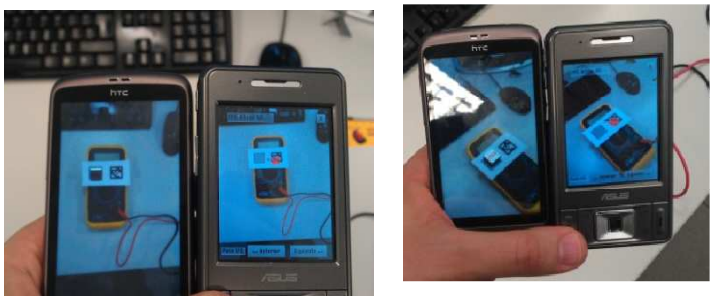


Ilustración 26 - Comparativa de *Artoolkitplus* y *Vuforia* la distancia del tracking es muy similar.

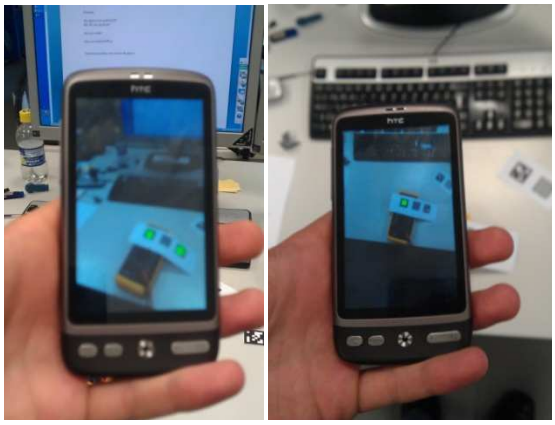


Ilustración 27 - *Artoolkit* para *Android*, permite una distancia muy grande pero no es robusto, confunde "cualquier" objeto con una marca.

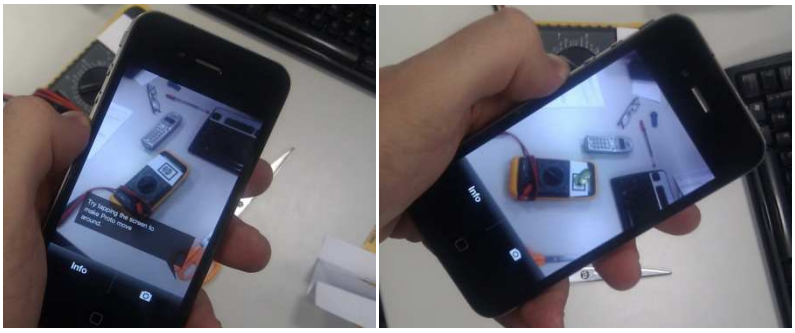


Ilustración 28 - Tracking muy bueno de *StringAR*, como desventajas, el coste y el no tener marcas con identificador.

Ilustración 29 - *Studierstube tracker par WM5* el mejor tracking pero no está portado a plataformas actuales.

Conclusión del estudio de librerías de reconocimiento para dispositivos móviles.

Solo hay tres librerías para sistemas operativos móviles actuales: *Vuforia, Andar* y *StringAR.* De esas tres librerías podemos descartar Andar debido a su baja robustez.

Entre las dos que quedan seleccionaremos *Vuforia* puesto que cubre una funcionalidad muy similar que *StringAR* pero además funciona tanto en *Iphone* como en *Android*. Además *Vuforia* permite trabajar también con marcadores lo cual puede ser una ventaja en algunos manuales y no tiene coste de licencia.

Seleccionando *Vuforia* como librería, cubrimos todas las necesidades propuestas al inicio del proyecto y por lo tanto, será la librería a utilizar para la implementación del manual.

2.3 Conclusiones del estado del arte

En la primera parte del estado del arte se han podido ver los puntos débiles de muchos manuales sobre todo en cuanto a la facilidad de edición, necesidad de conocimientos específicos (programación, Xml…), la dificultad de reconocimiento de entornos y al coste de los manuales, tanto por las horas de desarrollo como por el hardware que utilizan en algunos casos. Por lo tanto este proyecto se centra en poder mitigar algunos de dichos problemas.

En la segunda parte del estudio del arte se pudo ver como aun no hay librerías para reconocer objetos que sean suficientemente versátiles y que permitan reconocer tanto zonas completas (una máquina, un coche…) como las partes diferenciadas que pueden querer referenciarse durante un proceso de mantenimiento (filtro, rueda…). Además los algoritmos que mejores características proporcionaban tienen un consumo de recursos muy superiores a los disponibles en los teléfonos inteligentes actuales.

En la parte final se realiza un repaso por las librerías disponibles para dispositivos móviles y se escogen las librerías *Vuforia* para la implementación del manual debido a sus características (rendimiento, estabilidad, plataforma, tipo de marcas que reconoce, soporte e integración con motor gráfico).

Universidad Rey Juan Carlos

Máster Oficial en Informática Gráfica,
Juegos y Realidad Virtual

PROYECTO FIN DE MÁSTER

PROpuesta de PROceso de PROducción de Manuales en Entornos de Realidad Aumentada (PRO[3]MERA)

SEGUNDA PARTE:

PROCESO DE PRODUCCIÓN
E
IMPLEMENTACIÓN DE CASO DE USO

3 Definición del proceso de producción de manuales

Para poder desarrollar un manual de mantenimiento con una estructura adecuada, ampliable y versátil se define durante el capitulo 3 y 4 un proceso de producción y una arquitectura genérica.

Estos dos capítulos definirán de forma independiente del sistema de marcadores, motor gráfico o hardware que se utilice en el manual final. De ese modo, aunque en el presente proyecto se implemente con el motor gráfico *Unity3D* y las librerías *Vuforia* sobre un teléfono inteligente con *Android*. Esta se podría implementar con cualquier otro motor gráfico (*Ogre3D*, Coin3d…) y con cualquier librería de reconocimiento de marcas (*ArtoolkitPro*, *Metaio*…) sobre plataformas móviles o sobre ordenadores de sobremesa.

Aun siendo el objetivo del proyecto la creación de una implementación concreta de manual que permita ver su viabilidad de su aplicación en productos para el gran público, **el proceso de producción y la arquitectura propuestas pueden considerarse en sí mismas como una primera contribución de este proyecto.**

El proceso de producción propuesto está basado en tres pilares: una arquitectura, una metodología basada en la arquitectura anterior y la posibilidad de extender la arquitectura para poder lidiar con las peculiaridades de algunos manuales. Dichos elementos se explican en el presente capítulo.

Se pretende, que la arquitectura junto con el proceso de producción permitan mejorar ciertas deficiencias que se han podido observar al estudiar los manuales de mantenimiento, dichas mejoras que se irán viendo en este capítulo son las siguientes:

- Sencilla de implementar con herramientas actuales.
- Pensado para pequeños manuales no enfocándose tanto en motores, tanques, naves espaciales…
- Sin requerimiento de un hardware especial y con un alto coste.
- Normalmente no hay forma de comprobar que la operación se ha llevado a cabo por lo que puede repetir u omitir pasos. Mitigado con "Pasos pregunta" que verifican ciertas condiciones en puntos intermedios de las operaciones.

- Facilitar la edición y una primera validación gracias a una imagen de fondo de referencia que se elimine de forma automática al publicar el manual final.

3.1 Arquitectura

La arquitectura definida en este proyecto pretende cubrir varios requisitos que se han podido ver en el estudio del estado del arte. Se describirá la arquitectura interna del sistema y como se deben organizar los distintos elementos para que el sistema funcione de forma adecuada.

3.1.1 Arquitectura interna

La arquitectura pretende mantener lo visto en otras arquitecturas en el estudio del estado del arte:

- Fácil de implementar.
- Debe ser suficientemente versátil para cubrir la mayor cantidad de opciones posibles que se puedan dar en un manual.
- Preparada para ampliar sus posibles funcionalidades.

Definición de la arquitectura:

El sistema se implementa esencialmente en tres niveles que parten del concepto manual: Operación, Paso e Instrucción.

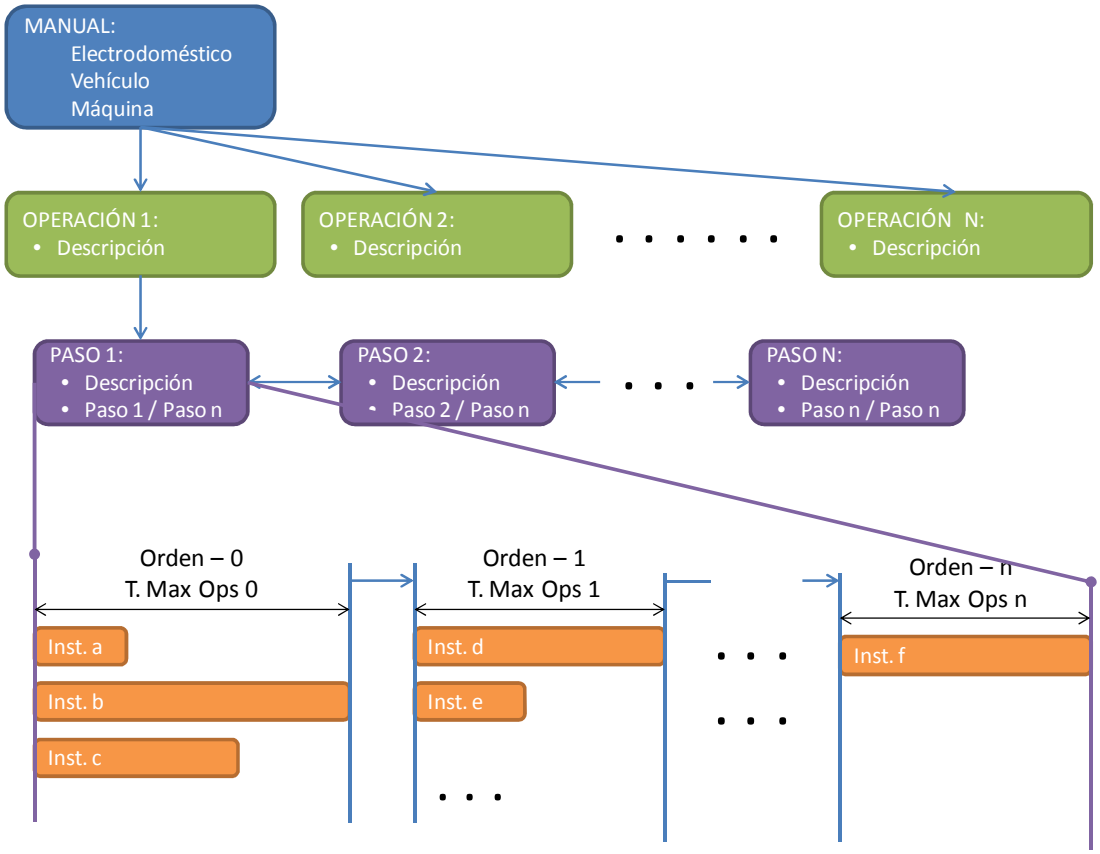


Ilustración 30 - Esquema de la arquitectura.

Operación:

En el primer nivel del manual tendremos las distintas operaciones que queremos realizar sobre el dispositivo. Si pensamos en un manual de un multímetro serían operaciones del tipo: "Medir tensión", "Cambio de pila del multímetro"…

En este nivel solamente se incluye una descripción de la operación aunque podrían añadirse otras informaciones como la complejidad de la misma, si se realización conlleva peligros… Estas operaciones siempre se seleccionan desde un menú y no tienen relación entre ellas.

La implementación del manual deberá tener un sistema para listar todas esas operaciones accediendo a su descripción y mostrándoselas al usuario para que este pueda escoger cual desea realizar.

Paso:

En el segundo nivel tenemos los pasos de la operación. Estos nivel indica los distintos pasos que se deben seguir para poder realizar la operación. Cada paso contendrá una descripción de lo que debe realizar el usuario y una indicación del paso en el que se encuentra sobre el total de pasos que tiene la operación.

El usuario puede moverse entre los pasos hacia adelante y hacia atrás. El usuario, podría repetir o saltarse alguna operación, al no ser técnicamente viable validar si el usuario realiza o no cada paso.

El funcionamiento normal de los pasos es secuencial pasando de uno en uno. Aun así hay operaciones que pueden llevarnos a tener que saltarnos, o no, algún paso. Por ese motivo se añaden operaciones de salto. Mediante la respuesta que proporcione el usuario el manual puede saltar a una u otra operación. (ej: Si el depósito tiene aceite no debe rellenarlo, pase a cerrar el depósito)

Es importante almacenar los saltos que realiza el usuario, si este realiza la siguiente secuencia de pasos 1,2,8,9 y quiere ir hacia atrás para comprobar la secuencia, al pulsar hacía atrás en el paso 8 se deberá volver al 2 y volver a preguntarle la condición de salto.

En la implementación para mitigar el problema de que el usuario pueda saltarse pasos, se pueden añadir pasos en los que se realicen preguntas de validación del tipo "¿Se visualiza texto sobre la pantalla?". Igualmente es recomendable implementar un sistema que permita al usuario saber si ha encontrado todas las marcas que son necesarias para dicho paso de modo que si el usuario ve una instrucción sobre una determinada marca sepa si ya puede avanzar de paso o si debe seguir verificando el resto de marcas en busca de nueva información antes de pulsar el botón paso siguiente.

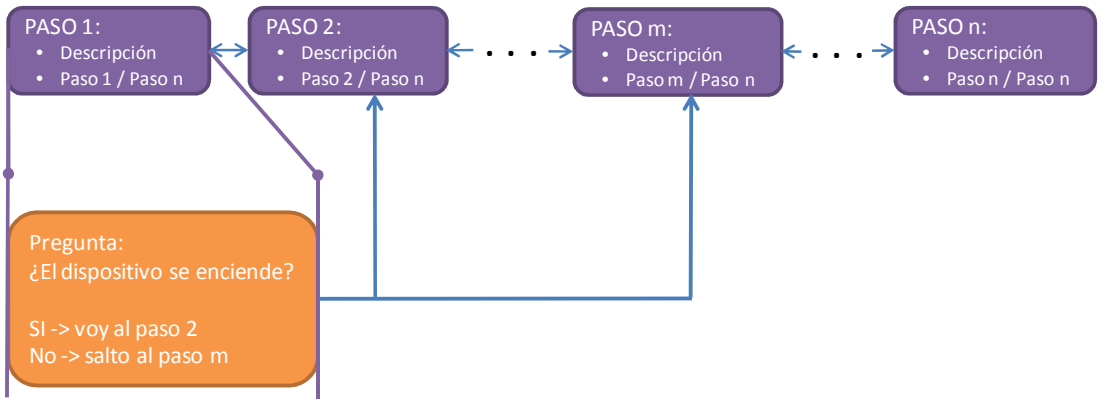


Ilustración 31 - Detalle de las instrucciones de salto.

Instrucción:

Finalmente tenemos las instrucciones. Este nivel es el que muestra más variabilidad. Y también el que le da más versatilidad al sistema.

En el último nivel en la mayoría de los casos damos información concreta y simple al usuario sobre un punto concreto del dispositivo con instrucciones del tipo: "Pulse un botón" "Retire una pieza" "Quite un tornillo"…

Las instrucciones definen la unidad mínima de información que se le puede dar al usuario. Dentro de este nivel podemos diferenciar varios casos:

- Información única sobre un solo punto de referencia.
- Varias informaciones sobre un punto.
- Informaciones al mismo tiempo sobre varios puntos.
- Una información simple seguida de otra información simple.

Estas informaciones básicas además pueden tener duraciones diferentes. La arquitectura por lo tanto debe permitir esa versatilidad.

Se definen en este caso dos propiedades:

- Orden: Todas las operaciones con el mismo valor de orden se inician al mismo tiempo. Para cada nivel de orden cuando el tiempo de la operación más larga ha terminado se finaliza ese grupo de instrucciones y se pasa a las instrucciones del siguiente orden.
- Duración: tiempo que se mantiene activa dicha instrucción, al terminar el tiempo si no se ha acabado el tiempo total del orden se para la visualización. Las instrucciones aun así deberán permitir en su tiempo de ejecución repetir la animación. EJ: programo giro de un destornillador a una velocidad adecuada, con un tiempo de 1 segundo el giro completo,

pero se repite hasta cumplir los 5 segundos, la operación de su orden puede durar aun más segundos si es necesario.

Esta arquitectura nos permite definir en un mismo paso una secuencia como la del siguiente ejemplo:

- Paso 2: "Quite la rueda"
  - Instrucción visual orden 0:
    - 4 elementos posicionados sobre los 4 tornillos nos indican que los desatornillemos. Cada giro tarda 2 segundos.
    - Al mismo tiempo vemos un cartel posicionado sobre el centro del tapacubos que indica que tenemos que usar una llave de tubo del 12. El texto se mantiene durante 5 segundos.
  - Instrucción visual orden 1:
    - Un flecha indicando que podemos sacar la rueda. Duración 3 segundos.

La implementación concreta que se realice de las instrucciones dará mayor o menor versatilidad al manual y permitirán unos tiempos de edición mayor o menor. La base mínima para un manual deberá contar con instrucciones muy básicas, **mostrar objetos 3d en puntos concretos del dispositivo, añadir algún tipo de animación y realizar preguntas para operaciones condicionales.** Un manual más completo puede contener operaciones complejas o específicas de un determinado sector (animaciones para retirar un tornillo indicando el tipo de destornillador, rellenar un deposito hasta un determinado nivel, secuencias de apriete de ruedas de un coche…) Una implementación de instrucciones más concreta es menos versátil pero consigue una edición mucho más rápida.

La ventaja de la arquitectura propuesta es que es extensible fácilmente. La funcionalidad mínima de instrucciones ya se ha indicado y si una operación se vuelve muy común podemos programarla como una "unidad de instrucción". Se puede tener el conjunto de todas las instrucciones que queramos y el esquema seguirá funcionando con la misma sencillez.

La calidad de la implementación que se realice y las facilidades que permitan las herramientas dará un tiempo y facilidad de edición mayor o menor. Además la implementación ideal no debería requerir que la persona que edita el manual necesite tener ningún conocimiento específico de programación, 3D o Realidad Aumentada tal y como se verá en el punto 4.5.

3.1.2 Organización de elementos

Para un funcionamiento óptimo del sistema se propone la siguiente organización. Con el objetivo de proporcionar la mayor versatilidad posible, está planteada para un manual con varios marcadores de referencia (ver ejemplo de organización en Ilustración 33)

Cuando hablamos de jerarquía en este apartado, nos referimos a la relación que suele existir en los motores gráficos entre objetos padre e hijo. Cuando un objeto es hijo de otro, al mover o rotar el padre todos sus hijos rotar o se desplaza siguiendo al padre.

Debemos tener un padre de toda la jerarquía de objetos. Como hijos del padre de la jerarquía deberemos tener los distintos marcadores que queremos reconocer. Además a cada marcador deberemos añadir varios hijos que sean los puntos de interes y plano de referencia, al ser hijos del marcador que vamos a detectar con las librerías de realidad aumentada mantendrán siempre una posición y orientación relativa correcta.

Ilustración 32 - Organización general de elementos.

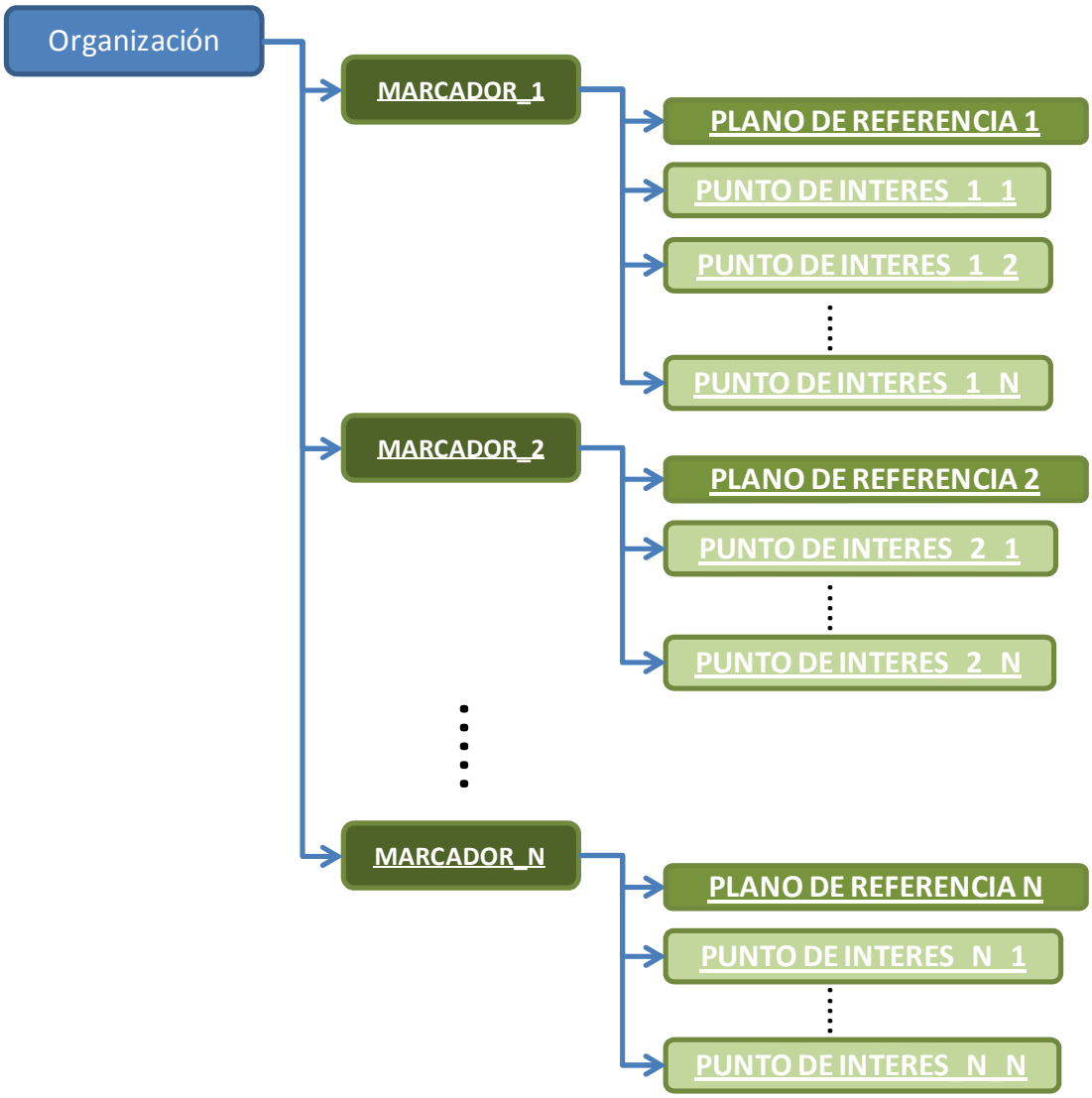


Ilustración 33 - Organización detallada de elementos.

Los hijos propuestos para cada marcador son los siguientes:

- Un hijo con un objeto vacio, es decir sin ninguna geometría 3D asociada, que a partir de ahora llamaremos *Punto de interés* en cada uno de dichos punto será donde podremos querer mostrar información. Dichos puntos deberán tener un nombre descriptivo para poder ser seleccionados con facilidad posteriormente. Al contar con un punto exacto posicionado cada punto de interés, todas las instrucciones que quieran mostrar algún objeto 3d sobre el objeto podrán posicionar este sobre el (0,0,0) relativo a dicho punto y estarán en el sitio exacto.
- Otro hijo que sea una plantilla de edición que llamaremos *Plano de referencia.* Una de las partes problemáticas con los editores es que se edita en offline y luego se prueba on-line, es decir, sobre el objeto real. Esto provoca que problemas de orientación, posicionamiento,

visualización de colores… no sean detectados hasta el momento de probar todo el manual u operación. Algunos sistemas proponen [21] editar el propio manual en tiempo real, es decir con video en tiempo real del dispositivo y editar y comprobar el manual directamente. Si bien el enfoque de tener una referencia real que facilite comprobación y edición parece adecuada, como desventaja de dicho sistema no nos permite estar sentados en una posición cómoda editando un manual durante varias horas. Además nos obliga a tener físicamente el dispositivo cerca.

En la organización propuesta, la plantilla normalmente será una imagen del entorno en el que se encuentra el marcador y nos permite realizar una primera validación del funcionamiento del manual sin utilizar realidad aumentada. Ese objeto de referencia tiene que estar marcado de algún modo especial para poder "desactivarlo" de forma rápida cuando entremos en el modo de pruebas o cuando publiquemos el manual definitivo.

3.2 Proceso de creación

Se describirá a continuación el proceso de creación propuesto de un manual según la arquitectura anteriormente definida, para ilustra mejor cada paso se utilizará un ejemplo de manual de un multímetro. Los siguientes paso se indican por el orden de ejecución que debería seguirse.

1. **Estudio del objeto sobre el que se realizará el manual.**

   Lo primero que se debe hacer es estudiar sobre que partes del dispositivo va a ser necesario mostrar información. Se deberán seleccionar todas aquellas zonas sobre las que queramos dar información al usuario. En cada una de las zonas deberemos saber si podemos utilizar una sola marca o si tenemos que colocar varias. Estas pruebas se deberán realizar con un teléfono inteligente que sirva como referencia. Se deberá colocar una marca de tamaño adecuado dependiendo del dispositivo (en un multímetro podremos colocar una marca más pequeña que en una prensa industrial) y ver la zona que podemos ver con el teléfono inteligente y que permite reconocer la marca.

   En el caso de un multímetro la mayoría de la información se va a visualizar sobre el frontal del mismo puesto que es donde se encuentran los controles por lo que se puede colocar una marca en el frontal o testear si se puede utilizar la imagen del mismo multímetro como marcador. Si se quiere dar instrucciones para, por ejemplo, cambiar la pila del multímetro se deberá proporcionar información también en la parte posterior del mismo, las partes

posteriores no tienen mucho detalle visual por lo que no es posible utilizar dichas imágenes como marcas y deberemos utilizar marcadores con identificador.

2. Selección de tipo de marcadores.

Aunque existen otro tipo de sistema de tracking de objetos, los más comunes (y los accesibles actualmente sobre teléfonos inteligentes) son de dos tipos: de marcas con identificador y de imagen Ilustración 36.

Marcas con imagen: estas marcas reconocen patrones en imágenes, en los manuales pueden ser muy útiles al permitir aprovechar la propia imagen del dispositivo como marca de referencia. Este caso es el ideal, puesto que no tenemos que añadir ningún elemento externo al dispositivo. Las condiciones de dicha imagen dependerán de cada librería utilizada, pero normalmente requieren que sea una superficie plana (como suelen ser los paneles de control de muchos dispositivos, electrodomésticos…) y además es necesarios que dicha imagen o imágenes que usemos como referencia tenga suficientes detalles y que no sea demasiado parecidas entre ellas. Debido a todas esas restricciones no siempre será factible utilizar imágenes del propio objeto como marcador.

En el ejemplo del multímetro la parte frontal del mismo nos sirve como imagen de referencia y podemos proporcionar una gran cantidad de instrucciones sobre el mismo. Sin embargo la parte trasera e interiores del mismo no tienen suficiente detalle por lo que no pueden ser usadas. Por lo tanto en el caso concreto del multímetro se podría utilizar reconocimiento de imágenes, si solo quisiéramos proporcionar información de la parte frontal. Otra opción es añadir una imagen pegada en la parte trasera del multímetro, aunque esto altera la estética del propio dispositivo lo cual puede no ser deseable.

Ilustración 34 – Imágenes del frontal y partes traseras del multímetro.

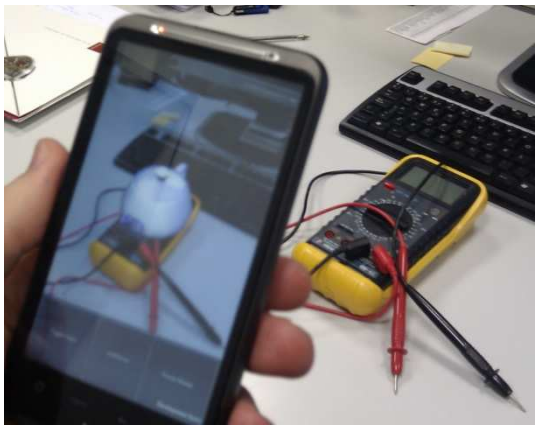


Ilustración 35 - Ejemplo tracking parte frontal multímetro.

Marcas con identificador: las marcas con identificador nos proporcionan un sistema de localización generalmente más estable, además es posible añadir tantas marcas como queramos, podríamos colocar "50" marcadores en el motor de un coche y el sistema sería estable para proporcionarnos datos sobre él.

La gran desventaja en este caso es que debemos añadir elementos externos al propio dispositivo lo cual altera su estética y puede no ser deseable en muchos de los casos.

Ilustración 36 - Ejemplo de marcador con identificador de Vuforia (izquierda) y marcador imagen con suficiente detalle (derecha).

El siguiente esquema puede servir de guía para la selección del tipo de marcadores que vamos a utilizar:

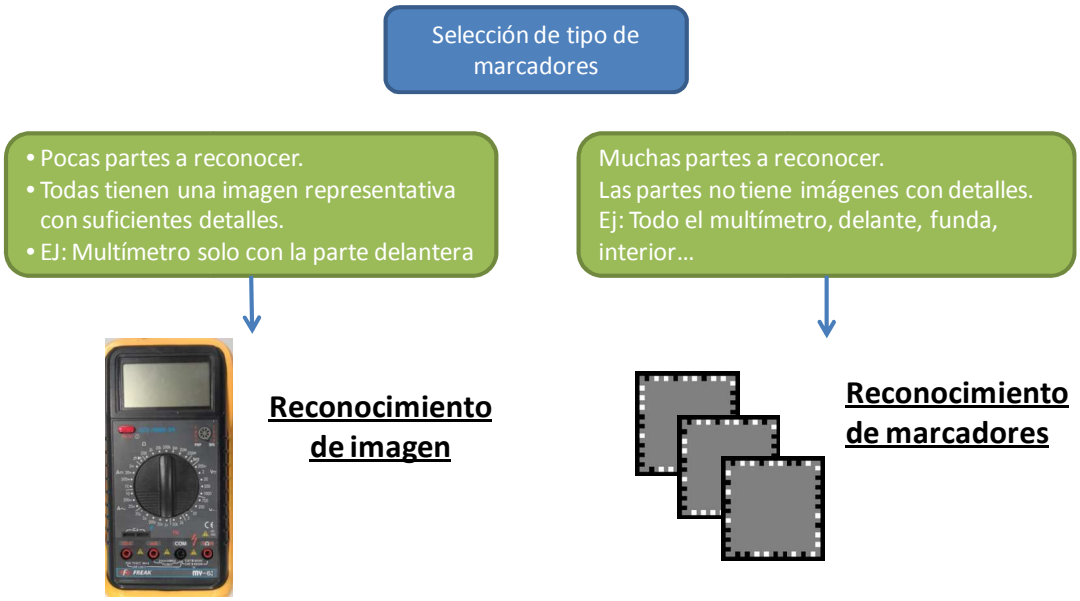


Ilustración 37 - Selección del tipo de marcadores.

3. Colocación de marcas

Una vez seleccionado el tipo de marcadores dependiendo del tipo de dispositivo se debe buscar donde colocar los marcadores y seleccionar los puntos de interés.

Lo ideal es colocar el menor número de marcadores posibles. Se deberá colocar una marca por cada zona en la que queramos mostrar información, además se deberá comprobar con la librería que al visualizar la marca con el dispositivo podemos visualizar toda la zona sobre la que queremos mostrar información.

En el caso concreto del multímetro se pueden colocar 4 marcas, una marca en la parte frontal, otra sobre la funda trasera, la tercera sobre la parte trasera del multímetro y la última en el interior del mismo. Con esas 4 marcas se cubren todas las zonas sobre las que puede ser necesario proporcionar información.

4. Plantilla de fondo.

Llegados a este punto tenemos una idea clara de donde vamos a mostrar información al usuario. Para facilitar la posterior edición del manual debemos tomar imágenes de las zonas sobre las que mostraremos información y crear unos planos dentro del motor gráfico que seleccionemos que nos sirvan como referencia.

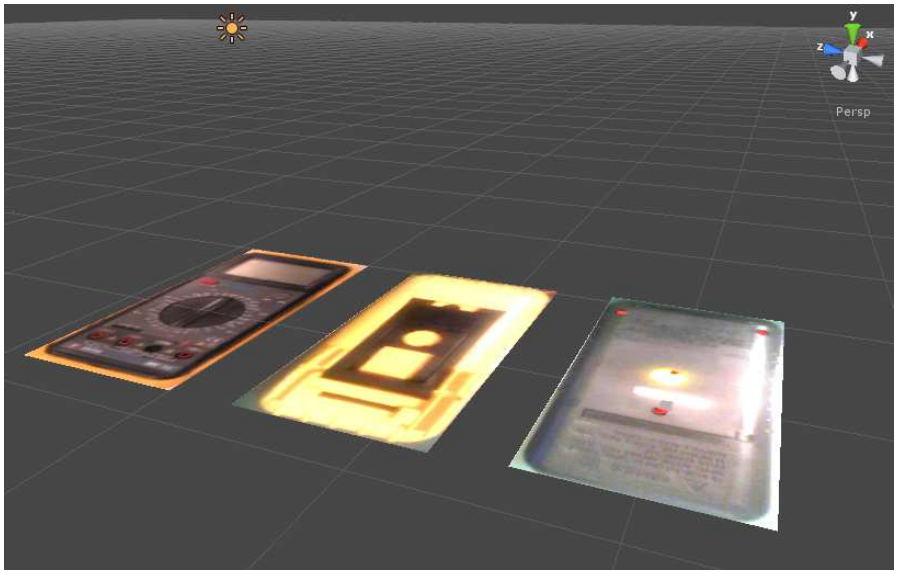


Ilustración 38 - Planos de referencia para la edición del manual.

5. Selección de puntos de interés

Según la arquitectura definida deberemos colocar un punto 3D de referencia sobre cada posición en la que queramos mostrar posteriormente información al usuario. Tal y como se indica en punto 3.1.2 se deberá generar una jerarquía de objetos partiendo de la marca del multímetro a la que hace referencia esa zona de modo que la posición de dichos puntos se mantenga.

En el caso del multímetro, en la parte frontal se añade indicadores en todas las posiciones que puede tener el selector, en las bornas, en el botón de encendido y en la pantalla de datos.

Ilustración 39 - Puntos de referencia del frontal del multímetro.

6. Edición de primera operación.

Una vez realizado todo este proceso se tendrá preparado el sistema para realizar la edición de las operaciones.

No es posible describir la edición de operaciones puesto que depende en gran medida de las facilidades que proporcione para la edición el motor gráfico seleccionado y las herramientas que haya desarrollado el programador. En este proyecto se expone un caso concreto en el punto **4.5**

3.3 Extensibilidad

La extensibilidad del sistema viene proporcionada por la propia arquitectura y proceso:

- La arquitectura del sistema es independiente del sistema de seguimiento de marcas por lo que según avancen los sistemas de reconocimiento de marcas u objetos se pueden sustituir dichas librerías en el manual. Todas las librerías de reconocimiento de marcas u objetos proporcionan la posición y orientación del objeto a reconocer por lo que dicha posición y orientación debe ser asignada al objeto padre de cada marcador.

- La organización en operaciones, pasos e instrucciones permiten cubrir prácticamente cualquier funcionalidad necesaria en un manual.
- En el nivel más bajo, las instrucciones nos dan la mayor posibilidad de extensibilidad posible. Es posible implementar cualquier funcionalidad que sea necesaria, lo cual hará la base del manual más versátil. Mostrar imágenes o videos, guiar hacia una marca si esta no tiene instrucción, acceso a manuales digitales en papel… este último nivel asegura que se pueda ampliar el manual según surjan nuevas necesidades y que dichos módulos estén disponibles para la edición de futuros manuales.
- Todos los contenidos nuevos generados en cada manual (objetos 3D, animaciones, nuevos scripts, materiales…) pueden ser reutilizados en los siguientes manuales, de ese modo, para cada nuevo manual, se contará con más recursos mejorando y abaratando los desarrollos.

Universidad Rey Juan Carlos

Máster Oficial en Informática Gráfica,
Juegos y Realidad Virtual

PROYECTO FIN DE MÁSTER

PROpuesta de PROceso de PROducción de Manuales en Entornos de Realidad Aumentada (PRO^{3}MERA)

CUARTA PARTE

IMPLEMENTACIÓN DE CASO DE USO

4 Implementación de un manual para un multímetro

A continuación se describe la implementación concreta de la arquitectura descrita en el proyecto. La implementación realizada no pretende ser un simple manual de uso de un multímetro sino, sobre todo, proporcionar un primer ejemplo base de implementación sobre un motor gráfico. Parte de **los desarrollos software realizados pueden ser utilizados para la realización de futuros manuales para otro tipo de dispositivos**.

Los apartados 4.2 y 4.3 pueden ser reutilizados fácilmente, pero pueden requerir que un programador adapte la funcionalidad.
El apartado 4.4 muestra las funcionalidades que se pueden reutilizar directamente sin necesidad de cambios para otros manuales. La funcionalidad de este apartado es el núcleo del manual con realidad aumentada.

Para realizar un manual se ha implementado la arquitectura completa descrita en este proyecto pero se han añadido algunos elementos más para proporcionarle un aspecto final de producto:

- Menú inicial
- Una descripción de controles del multímetro
- Opciones generales (foco, modo de enfoque…)
- Y un botón de "Acerca de"

Durante la memoria se utilizarán tres roles diferenciados:

- Programador: La persona que ha creado la base del manual, los menús, integración con realidad aumentada, scripts de la arquitectura, instrucciones visuales…
- Modelador: Persona encargada de crear contenidos 3D para el manual, tales como planos con imágenes del dispositivo, flechas, destornilladores…
- Creador de manual: es el encargado de utilizar las diversas herramientas que le proporcionan el programador y modelador para crear las instrucciones que guiaran al usuario en la realización de las operaciones de mantenimiento.
- Usuario: Persona que descarga e instala la aplicación en su teléfono inteligente y la utiliza para aprender a utilizar un dispositivo, en este caso concreto un multímetro.

4.1 Herramientas seleccionadas

Para el desarrollo del proyecto se ha seleccionado como motor gráfico *Unity3D* y como librerías de reconocimiento de marcas *Vuforia* de *Qualcomm*. A continuación se detalla el por qué de dicha selección.

Unity3D

Este motor gráfico nos proporciona una gran cantidad de ventajas para la realización del manual, se detallan continuación las características por las que se ha seleccionado:

- Facilidad de edición. A parte de la facilidad para crear juegos o aplicaciones con él, su sistema de edición nos permite que el *Programador* cree scripts configurables desde el editor facilitando la creación de nuevas operaciones o pasos por parte del *Creador del manual* que no tiene por qué tener conocimientos de programación, motores gráficos o realidad aumentada.
- *Unity3d* permite de forma nativa el despliegue de sus aplicaciones sobre teléfonos inteligentes con *Android*.
- Integración con *Vuforia*, *Qualcomm* ha creado un paquete que nos permite tener una fácil y rápida integración con las librerías de realidad aumentada.
- De forma más general la propia capacidad y rendimiento del motor gráfico.

Vuforia* de *Qualcomm

Como ya se indicó en el estudio del estado del arte esta librería nos permite mucha funcionalidad de la que necesitamos para el manual de mantenimiento. Permite reconocer imágenes y marcadores codificados. Tiene un rendimiento muy bueno en teléfonos inteligentes además de ser simple de utilizar. Su coste es bajo permitiendo incluso aplicaciones comerciales sin coste de licencia.

Al iniciar la implementación con esta librería surgió un problema a la hora de utilizar marcadors con identificador. Debido a la gran cantidad de texto que tiene el multímetro, los cuales no debemos tapar con la marca, solo había espacio disponible en la parte frontal para marcas muy pequeñas 2-3cm, que no permitían realizar un seguimiento adecuado. Para solucionar el problema y poder usar dichos marcadores se decidió colocar una marca sobre la pantalla recortando la parte interior que no se utiliza para el tracking, de este modo se puede hacer un seguimiento desde una distancia muy grande. Para el

seguimiento del resto de partes del multímetro podremos utilizar marcas de un tamaño adecuado sin problemas de espacio.

Solución propuesta:

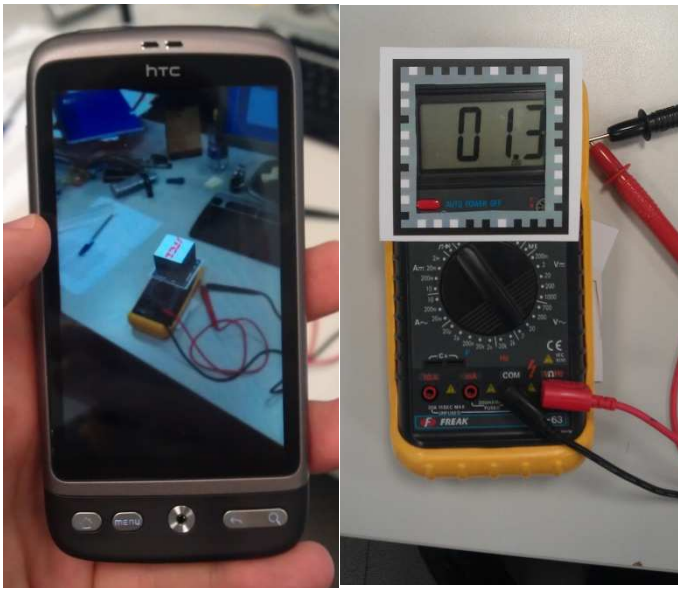


Ilustración 40 - Librerías de Qualcomm con reconocimiento de marcadores.

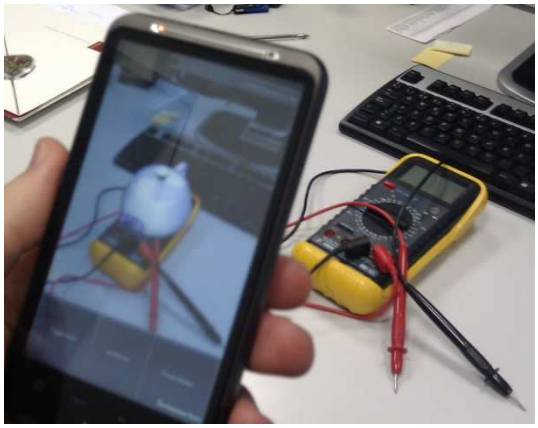


Ilustración 41 - Librerías de Qualcomm con reconocimiento de imágenes.

4.2 Utilidades genéricas

El desarrollo realizado cuenta con dos utilidades genéricas para facilitar el uso, la primera está dirigida al programador del manual para facilitar el depurado en posibles ampliaciones o extensiones de funcionalidad. La segunda utilidad proporciona información al usuario para saber las marcas que debe encontrar en cada paso.

Consola "Debug" sobre el teléfono inteligente :

La clase "*GlobalVar*" contiene una función "*SetError(mensaje)*" que nos permite ver en tiempo real sobre la pantalla del SmarthPhone los últimos mensajes asignados con dicha función. Dichos mensajes se visualizan de fondo en la aplicación permitiendo al desarrollador realizar labores de depurado de la aplicación que tan complejas resultan en este tipo de plataformas.

La funcionalidad se ha implementado en la clase *GlobalVar* para que los mensajes puedan incluirse desde cualquier clase de la aplicación.

La depuración puede ser activado o desactivado desde las opciones de la aplicación, una vez liberada la versión final de la aplicación solamente se deberá comentar la llamada a dicho botón en *OnGUI*.

Ilustración 42 - Ejemplo mensajes debug durante la ejecución del manual.

Información proporcionada al usuario sobre las marcas:

El usuario de un manual con realidad aumentada normalmente visualiza información solamente sobre aquellas marcas que en el paso actual contienen

información. Esta forma de presentar la información al usuario tiene dos problemas.

El primer problema es que el usuario cuando visualiza una marca a través de la pantalla no puede estar seguro de si el sistema no reconoce la marca por estar lejos, mal iluminada… o si dicha marca no tiene información. Por ese motivo se añade una funcionalidad, en cada paso en la que se debe indicar cuáles son las marcas que no se utilizan. De ese modo siempre que se visualice dicha marca mostraremos un objeto genérico que nos indicará que dicha marca se ha reconocido pero que no contiene información relevante en este paso.

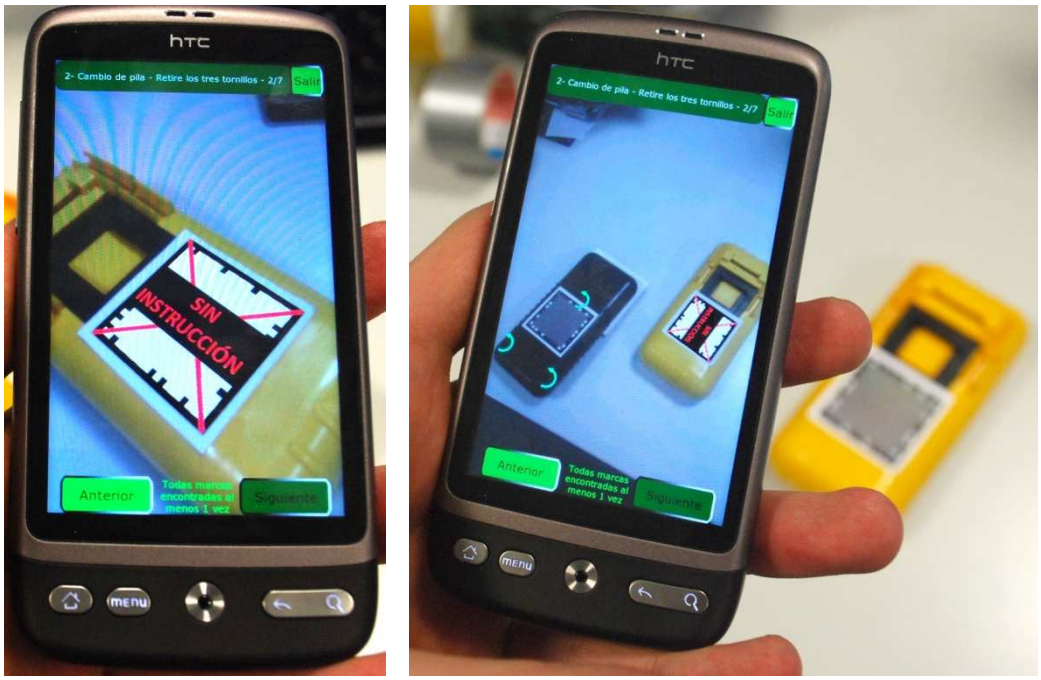


Ilustración 43 - Visualización de marca sin instrucción asociada.

Este sistema asegura que el usuario sepa si una marca contiene información o no. El segundo problema es que el usuario con la información sobre las marcas no sabe si ha "encontrado" todas las marcas que contienen información relevante para dicho paso. Para solventar dicho problema se añade información al lado de los botones de navegación entre pasos. Se le avisa al usuario del número de marcas que debe encontrar para dicha operación, la información se muestra en verde o rojo para reforzar la información visual proporcionada. Además se inhabilita el botón de siguiente paso mientras el usuario no encuentre todas las marcas.

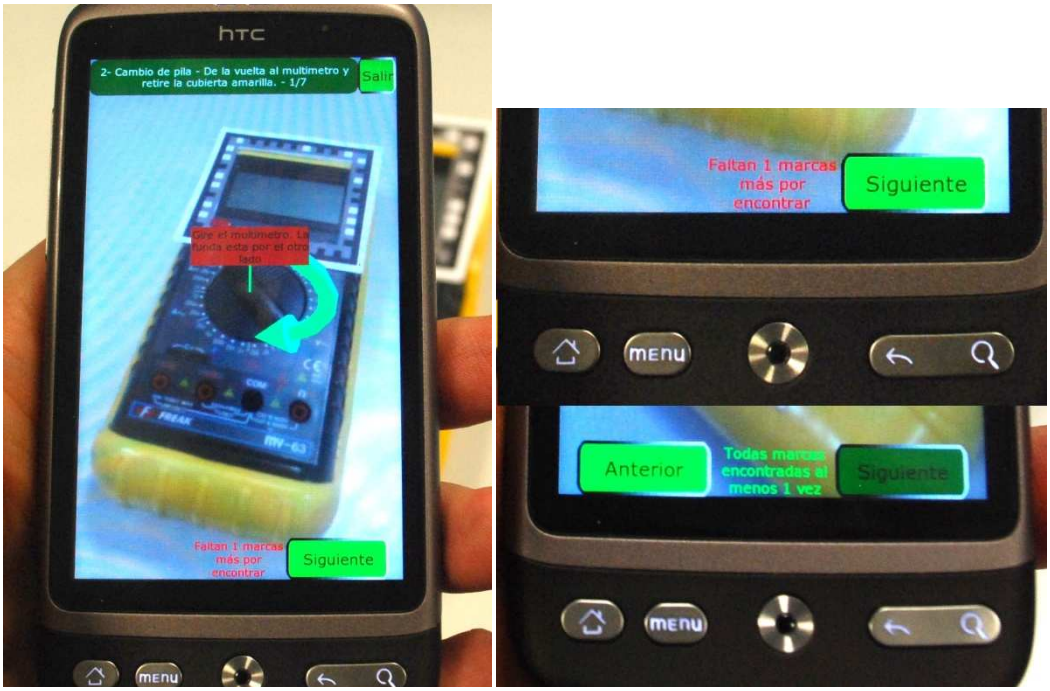


Ilustración 44 - Texto informativo para el usuario del número de marcas que le faltan por encontrar en el paso actual.

4.3 Partes específicas del manual implementado

Interfaz y menú:

Se ha creado un interfaz simple que permite ver y seleccionar opciones básicas que puede tener un manual aunque estás pueden variar mucho dependiendo del tipo de dispositivo.

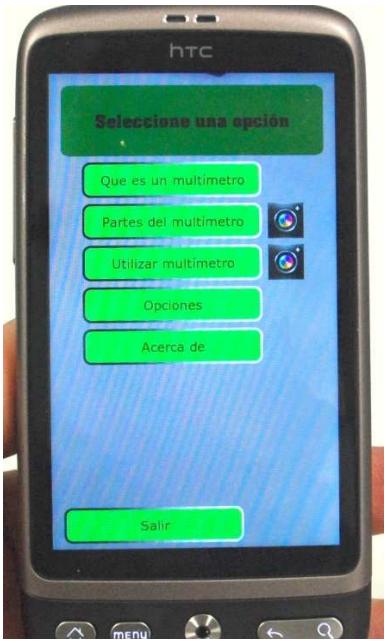


Ilustración 45 - Menú inicial de la aplicación.

La opción "Qué es un multímetro" muestra un simple scroll con texto indicando que es un multímetro y para que puede ser usado. "Acerca de" nos da información sobre el proyecto en el que se ha creado el multímetro.

En las opciones permitimos activar una opción de "Debug" activar o desactivar el foco y una opción para mostrar siempre los objetos aunque no veamos la marca. Estás opciones se desactivarían en la versión final.

Módulo controles del multímetro

Se ha añadido una utilidad no incluida en la arquitectura, que sirve para proporcionar información inicial al usuario sobre cuáles son los controles o puntos de interés sobre una marca. En el caso del multímetro se han incluido información de interés en los controles y opciones de medición frontales del multímetro.

La información se visualiza por medio de cuadros de texto sobre cada una de las partes del multímetro con una breve descripción de lo que son. El sistema actual cuenta con 4 secciones principales que pueden ser activadas o desactivadas con unos botones. Algunas de dichas secciones principales tienen subsecciones o subdatos a mostrar:

- Pantalla de datos
- Encendido
- Opciones selector (se activan todos los subdatos al mismo tiempo)
    - hFe
    - Tensión alterna
    - Tensión Continua
    - Mhz
    - Faradios
    - …
- Opciones bornas
    - Corriente alterna alta
    - Común.
    - …

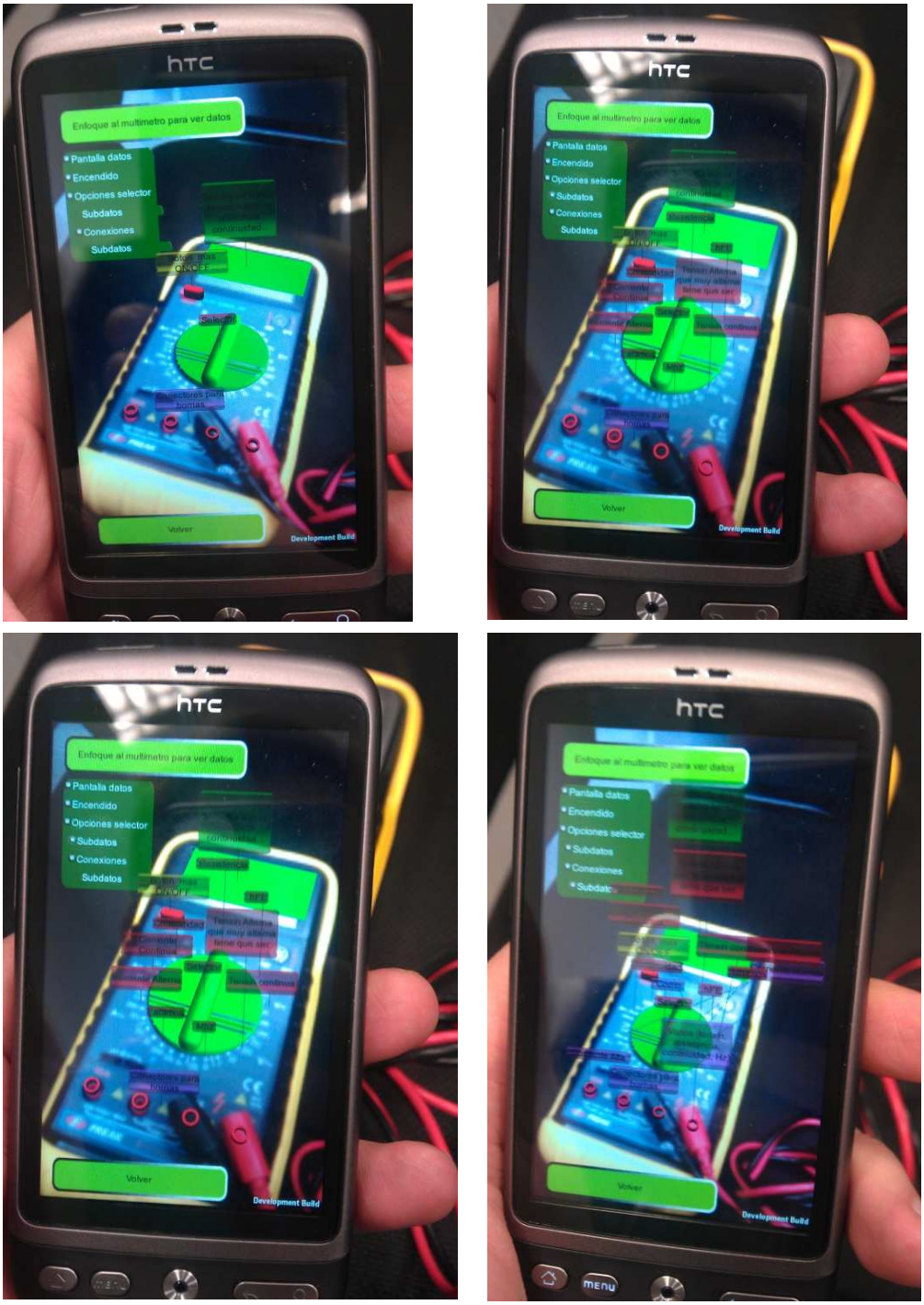


Ilustración 46 - Imágenes de carteles sobre las partes del multímetro.

Todos los carteles de información se generan y posicionan en tiempo real a una distancia determinada sobre la parte adecuada del multímetro. El cartel tiene una línea que le une con la posición "exacta" del elemento que describe.

Al tener muchos carteles de texto (hasta 17 en este caso) de tamaños variables y generarse estos en tiempo real sobre posiciones aleatorias, que pueden estar muy juntas si estamos lejos del multímetro. El sistema hace una reordenación de

los carteles de modos que se puedan ver todos los carteles, controlando que ninguna esquina de ningún cartel este dentro de otra y reubicándolo.

El sistema no asegura que siempre se vean todos los carteles activos por ser un proceso complejo y que puede ser computacionalmente costoso, además, por la resolución de la pantalla podrían no caber toda la información que deseamos mostrar. Por otra parte, esta información no es esencial sino meramente informativa por lo que no es esencial visualizar todos los carteles todo el tiempo.

El sistema de visualización que se ha programado es genérico, se añade sobre cada objeto vacio hijo de una marca un tag "*infoVer*" (para acelerar su búsqueda) que nos indica si es un objeto del que queramos ver la descripción. También es necesario añadirle un script con el Texto a mostrar en dicha posición, un color, un grupo y un subgrupo.

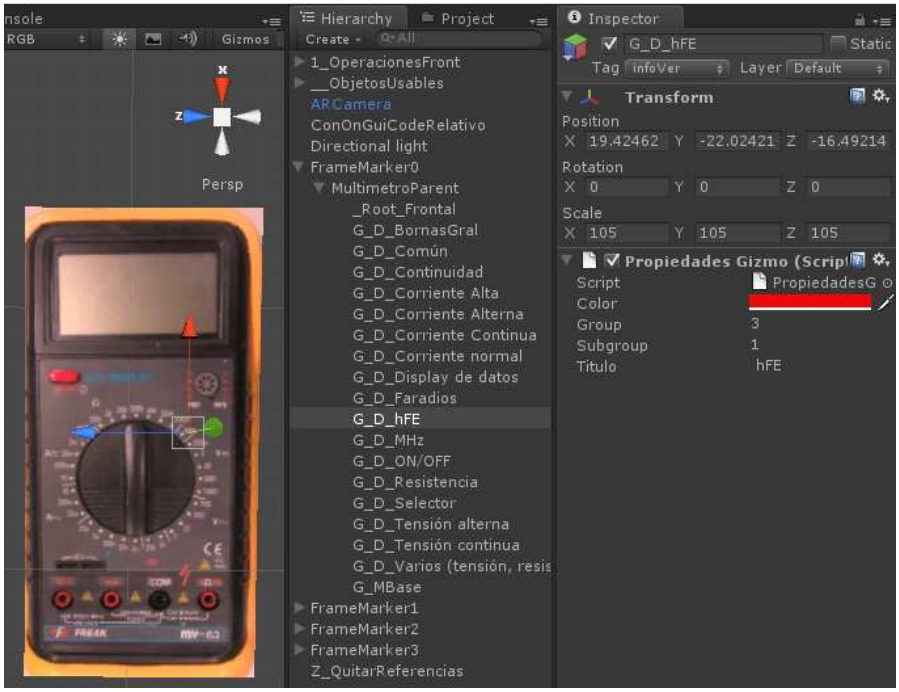


Ilustración 47 - Script visualización de textos seleccionad hFE (comprobación de transistores).

En el *script* general de funcionamiento se buscan todos los "*Tag*" de Unity3D etiquetados como "*infoVer*" y en la posición del *Gizmo* se muestra el texto siempre que el grupo y subgrupo estén activos. Los grupos y subgrupos se controlan con varios botones tipo *toggle* (al pulsar una vez se activan y al volver a pulsarlos se desactivan):

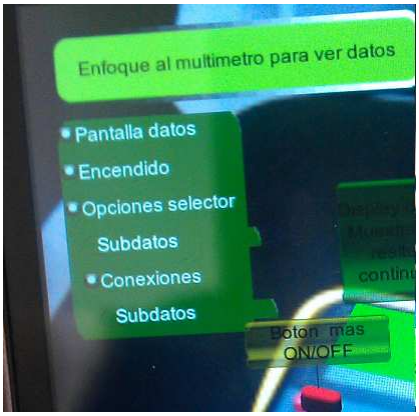


Ilustración 48 - Botones para visualizar datos y subdatos.

Gracias a estos *scripts* es muy fácil añadir nuevos carteles, modificar los textos o crear nuevas jerarquías de objetos. Aun así es necesario crear el menú de botones a mano.

4.4 Implementación de arquitectura

A continuación podremos ver la implementación concreta de la arquitectura propuesta en el proyecto.

4.4.1 Diagrama de clases

En este apartado veremos el diagrama de las clases involucradas en la arquitectura del manual y su representación sobre el motor gráfico seleccionado.

Debido a la forma de trabajar del motor gráfico, muchas de las relaciones entre clases no son visibles directamente en los atributos del diagrama de clases. Para poder editar de forma sencilla el manual los scripts se añaden a *GameObjects* (objetos 3D de Unity3D que pueden contener o no geometría) vacios y se buscan componentes en los hijos de dichos objetos.

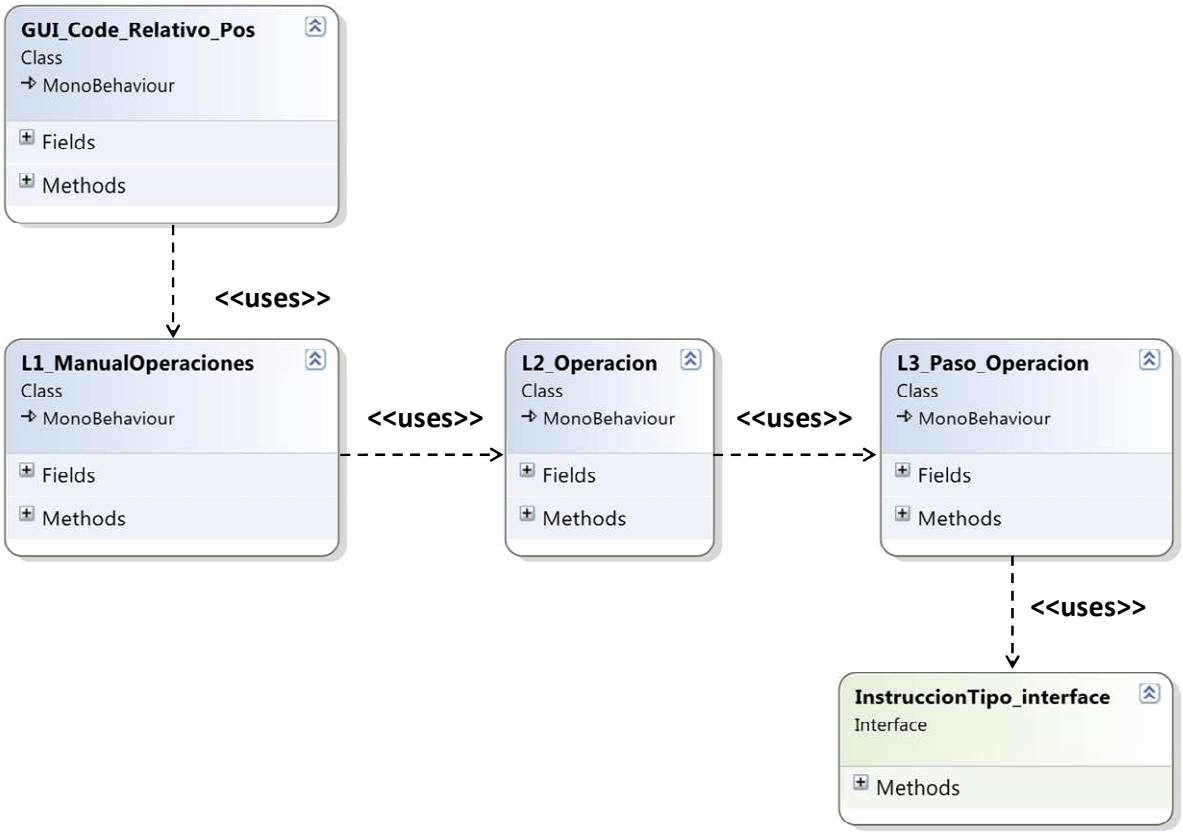


Ilustración 49 - Diagrama de clases.

El detalle de las clases involucradas en el proyecto se puede ver a continuación.

- ListaOperaciones está asociada a un GameObject que contiene la lista de las operaciones del manual este será el punto de inicio de los datos desde el interfaz de usuario.
- DatosOperacion nos proporciona un título de cada paso de la operación y nos da acceso a los pasos (DatosPaso).
- DatosPaso El GameObject al que está asociado el script incluye las "InstruccionesTipo" que nos permitirán visualizar las acciones que tenemos que hacer en cada caso
- InstruccionTipoInterface, interfaz para el acceso desde DatosPaso a todas las implementaciones de instrucciones visuales, de ese modo se pueden buscar todas las instrucciones que tiene asociado el objeto que sean de esta interfaz para ejecutarlas.
- InstruccionTipo X 5 implementaciones de pasos visuales que nos permiten la forma más versátil posible la implementación de acciones visuales sobre el manual, todas ellas implementan el interface InstruccionTipoInterface. Las instrucciones tienen varios campos fijos: orden y tiempo hasta siguiente paso. Además todas cuentan con el

método "doAction" que será llamado en cada frame cuando la instrucción este activa y será la función encargada de ejecutar la operación visual.

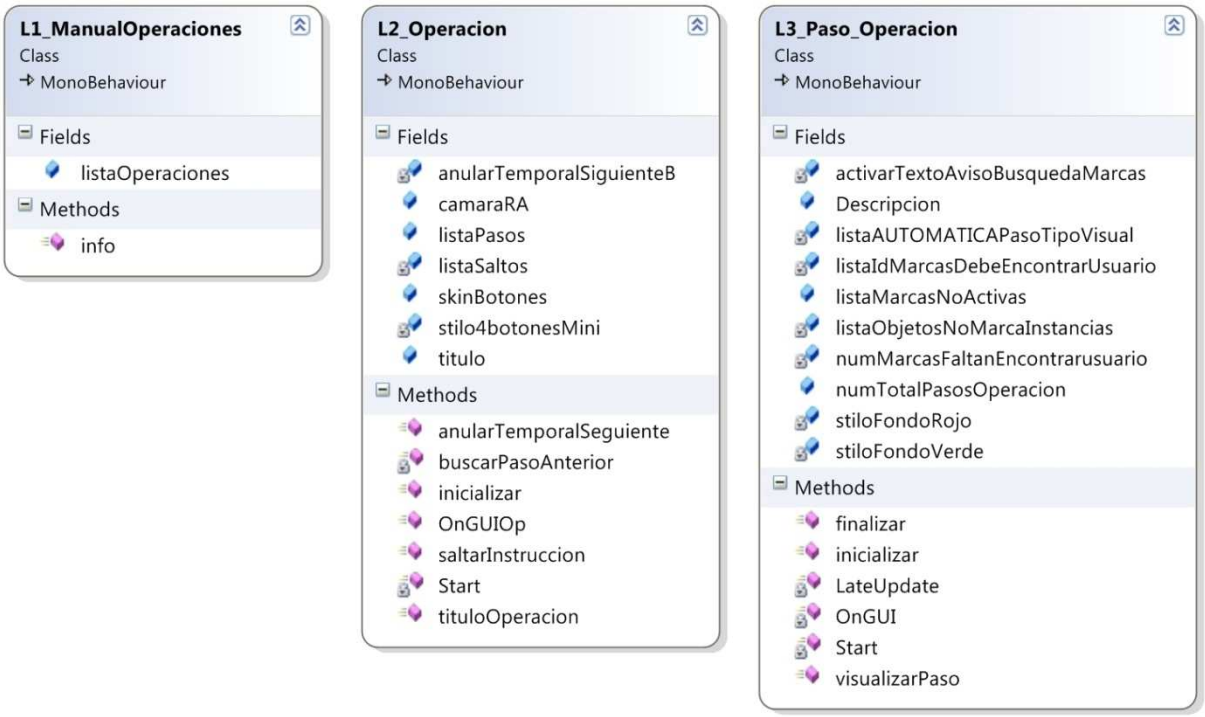


Ilustración 50 - Clases base de la arquitectura del manual.

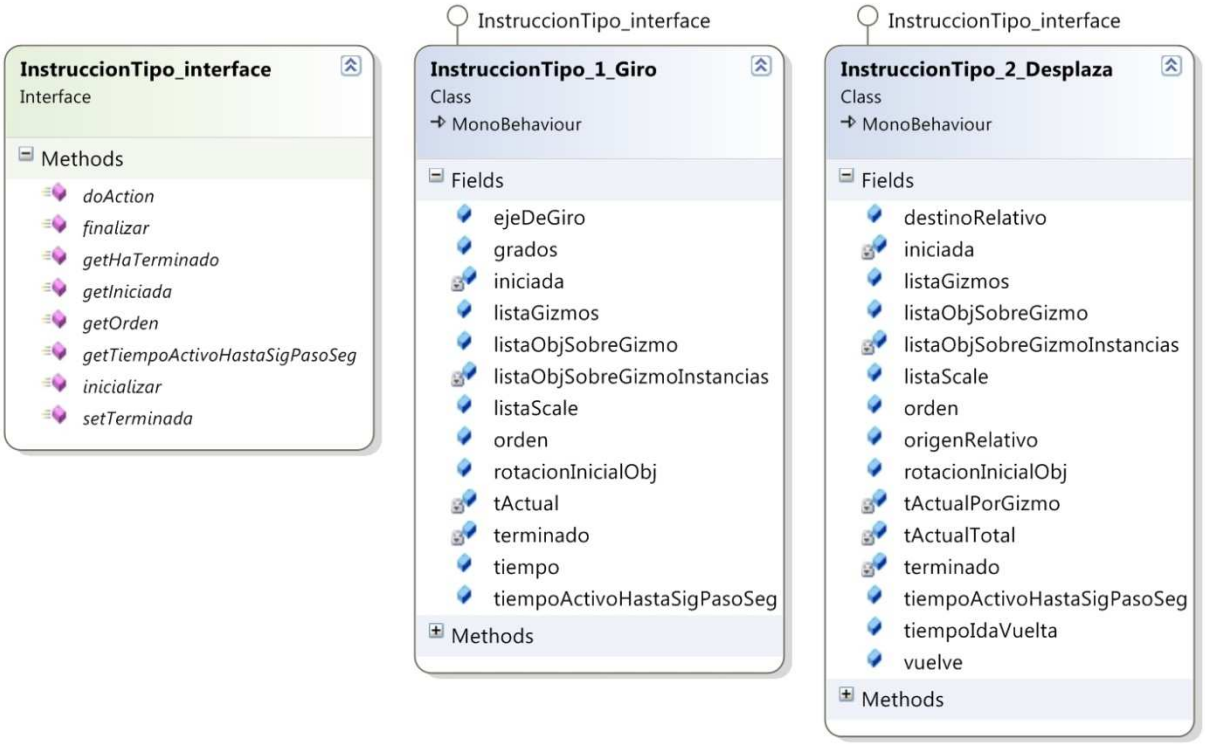


Ilustración 51 - Pasos tipo 1.

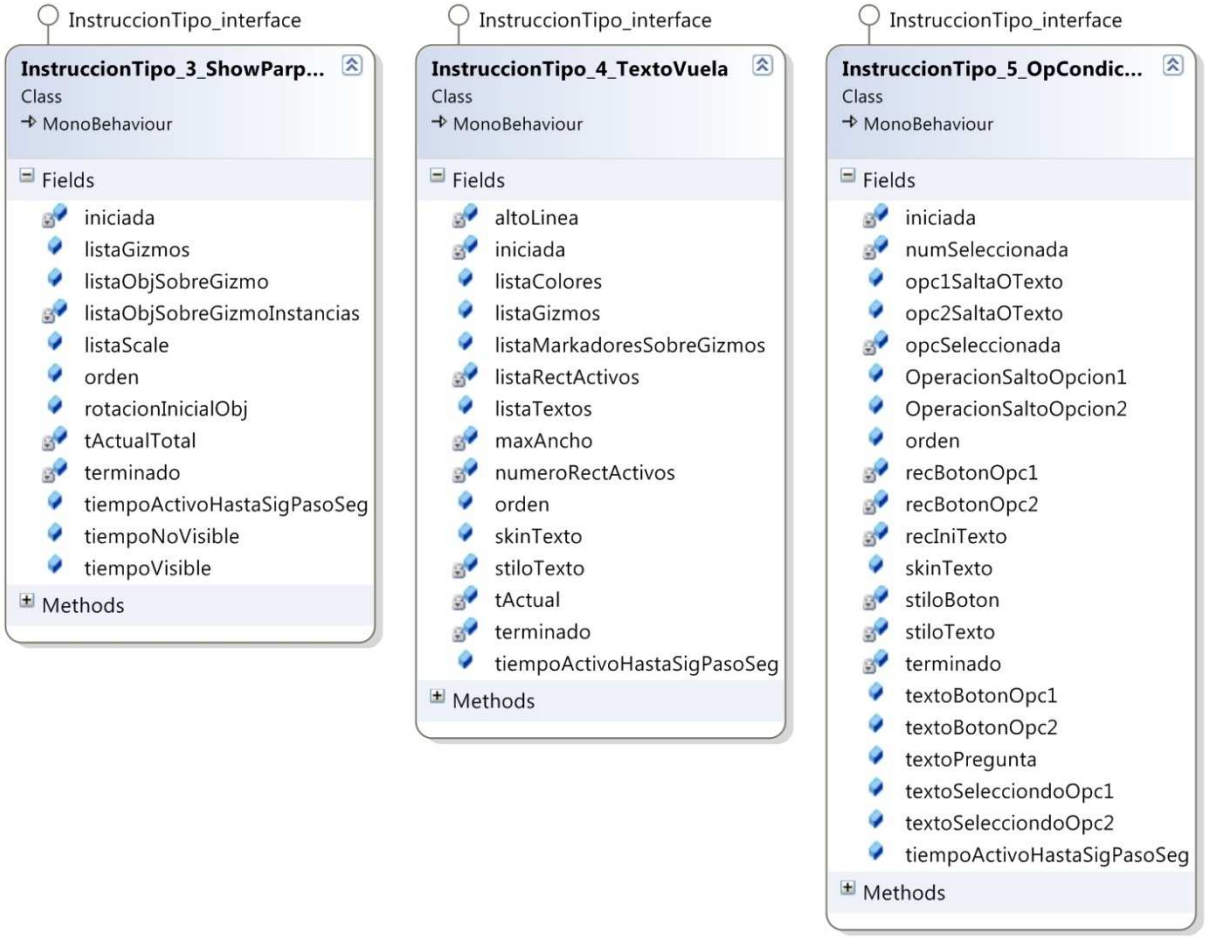


Ilustración 52 - Pasos tipo 2.

A parte de las clases que implementan la arquitectura hay otras 5 clases en el proyecto que se utilizan para diversas funcionalidades. Estás clases no son esenciales para la implementación de un manual genérico, son las clases que proporcionan otro tipo de funcionalidades, acceso a opciones, interfaces…

- ControlRa Proporciona funcionalidades genéricas que pueden ser utilizadas en el manual, ocultación de todos los objetos hijos de otro objeto, encender o apagar el foco, activar modos de enfoque, opciones para visualizar las posiciones de todos los objetos estén activas o no sus "marcas padre"
- GlobalVar: Almacena una gran cantidad de variables generales registradas como static, el script da acceso rápido a otros script para actualizar o validar el estado de pasos, acciones…
- Ordenar_Msg: Utilidad para ordenar los mensajes de texto que se muestran al visualizar información sobre el polímetro. El sistema intenta ordenar todos los cuadros de texto que se posicionan sobre el multímetro teniendo en cuenta su posición. El sistema no asegura que se puedan ver

todos, sobretodo para mantener un script eficiente que no retrase los frames por segundo.

- PropiedadesGizmo: Script añadido a ciertos *GameObjects* de posición que permiten mostrar textos sobre ellos al visualizar
- Z QuitarReferencias: El script mantiene una lista de objetos que se eliminan de la jerarquía del manual al iniciar su ejecución. Dichos objetos son las plantillas que se utilizan para edición y al eliminarlos en tiempo real de la jerarquía nos aseguramos mantener la facilidad de edición pero no tenemos que preocuparnos de eliminar dichos objetos cada vez que queramos realizar una prueba.

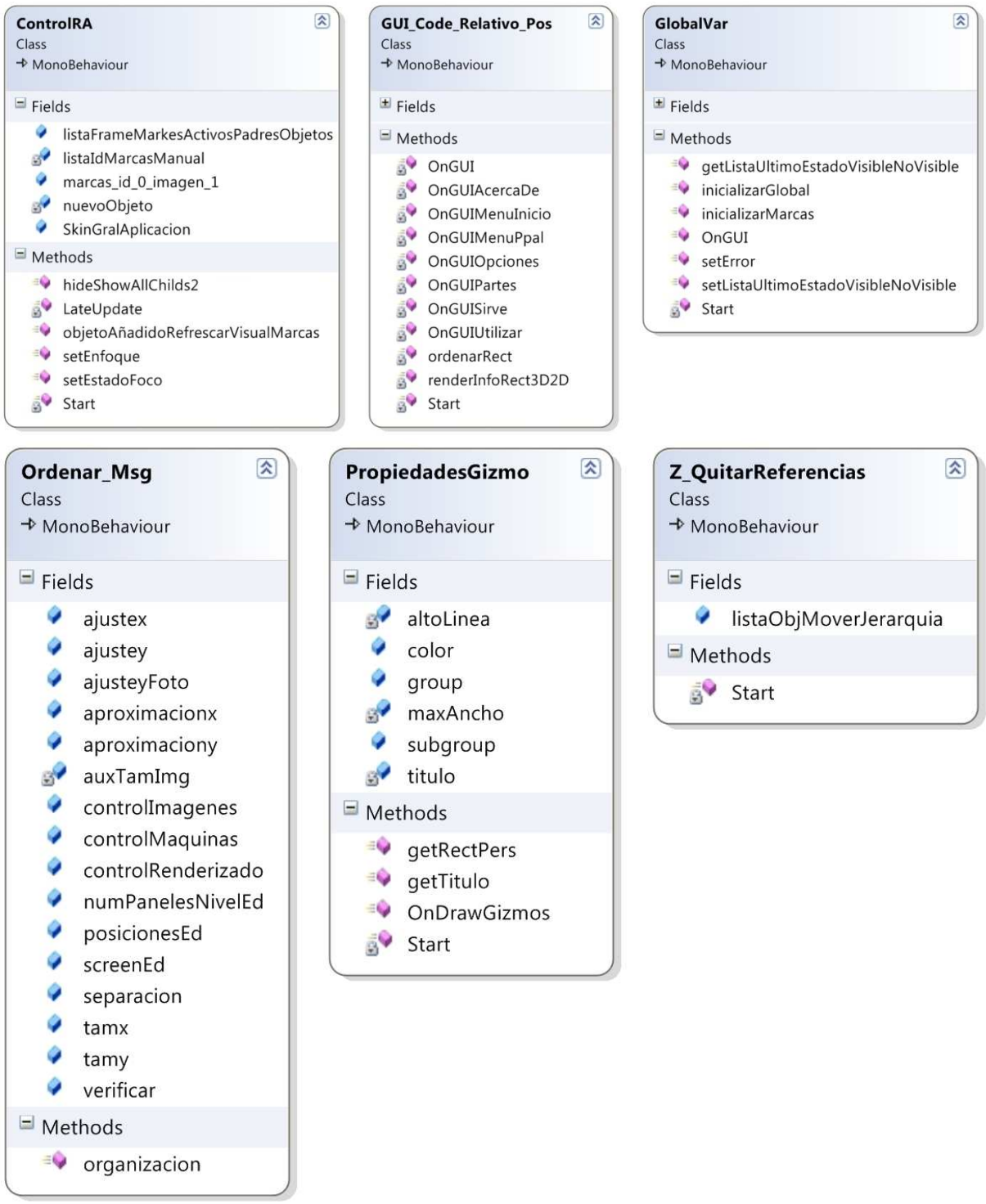


Ilustración 53 - Clases no directamente relacionadas con la arquitectura.

4.4.2 Diagrama de secuencia

Aquí podemos ver el diagrama de secuencia principal de la interacción del manual entre operaciones, pasos e instrucciones.

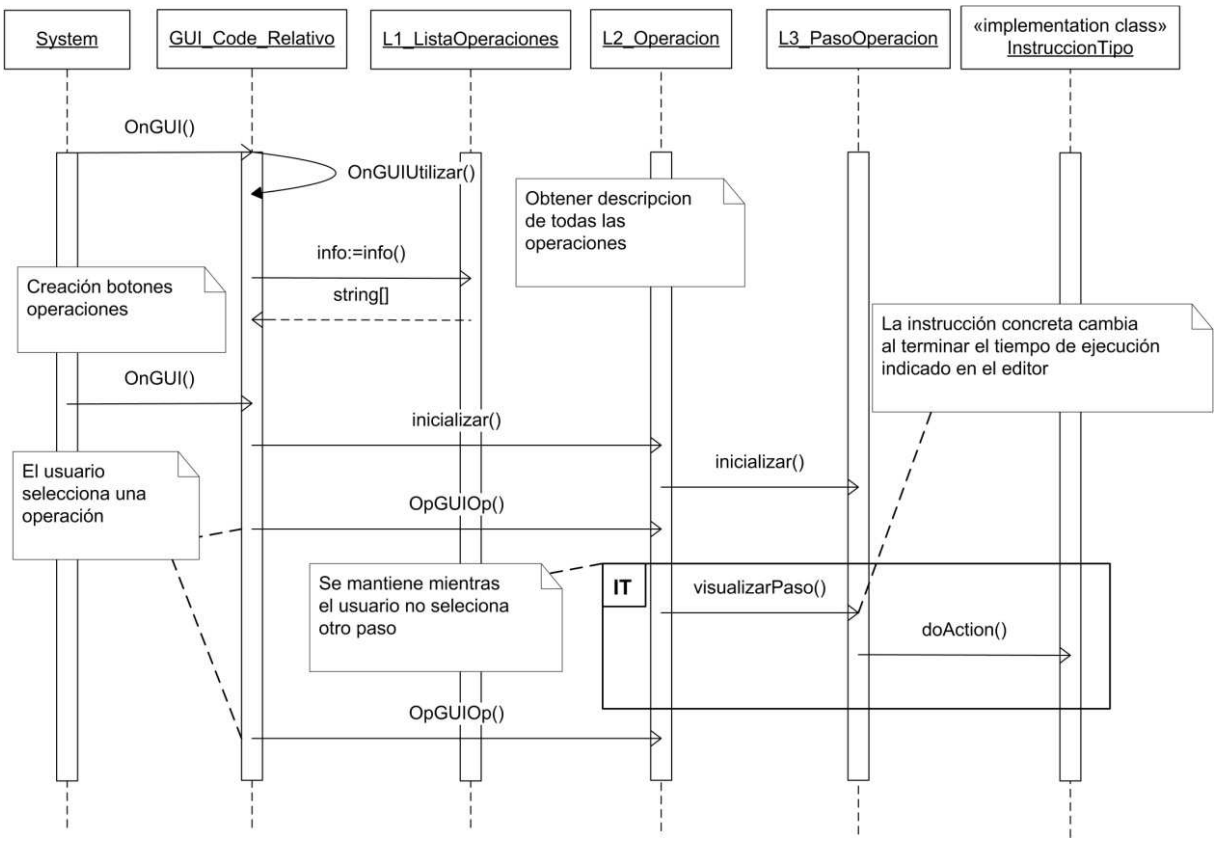


Ilustración 54 - Diagrama de secuencia.

Como se puede ver, el esquema general de secuencia del manual es bastante simple.

4.4.3 Tipos de instrucciones reutilizables implementadas

Se han implementado 5 tipo de instrucciones, cada una de ellas implementan el interfaz InstruccionTipo_Interfaz, las clases inicialmente implementadas son las que se han podido ver que son más comunes a todos los manuales de mantenimiento. La funcionalidad de la base de manuales puede ser ampliada fácilmente creando nuevas clases de InstruccionTipo.

A continuación veremos el funcionamiento de cada una de las instrucciones implementadas que se han preparado para ser lo más versátiles posible. Todos

los scripts que permiten realizar acciones sobre un *Punto de interés* y están preparados para que con el mismo script esa acción se pueda visualizar sobre varios *Punto de interés* sin necesidad de añadir más instancias de dicho script.

Instrucción de giro:

Útil para poder mostrar objetos que giran sobre puntos determinados. El script permite indicar "n" *puntos de interés* sobre los que mostraremos objetos rotando, cada uno de ellos puede tener una orientación inicial, un eje de giro una escala y unos grados de giro por tiempo.

- Lista Gizmos: Lista de *Punto de interés* marcados sobre el multímetro sobre los que mostraremos objetos rotando.
- Lista Obj Sobre Gizmos: Lista que contiene los objetos 3D que se mostraran rotando.
- Rotación inicial obj: Se puede definir la rotación inicial de cada objeto.
- Eje de Giro: eje de giro del objeto
- Lista Scale: escala de los objetos a mostrar
- Grados
- Tiempo: La unión de grados y tiempo nos proporciona la información sobre la velocidad de giro del objeto

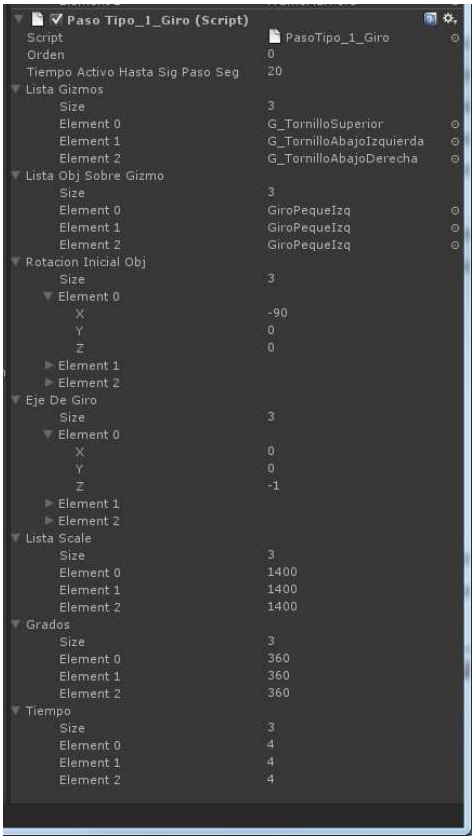


Ilustración 55 - Menú Unity3d ejemplo instrucción giro.

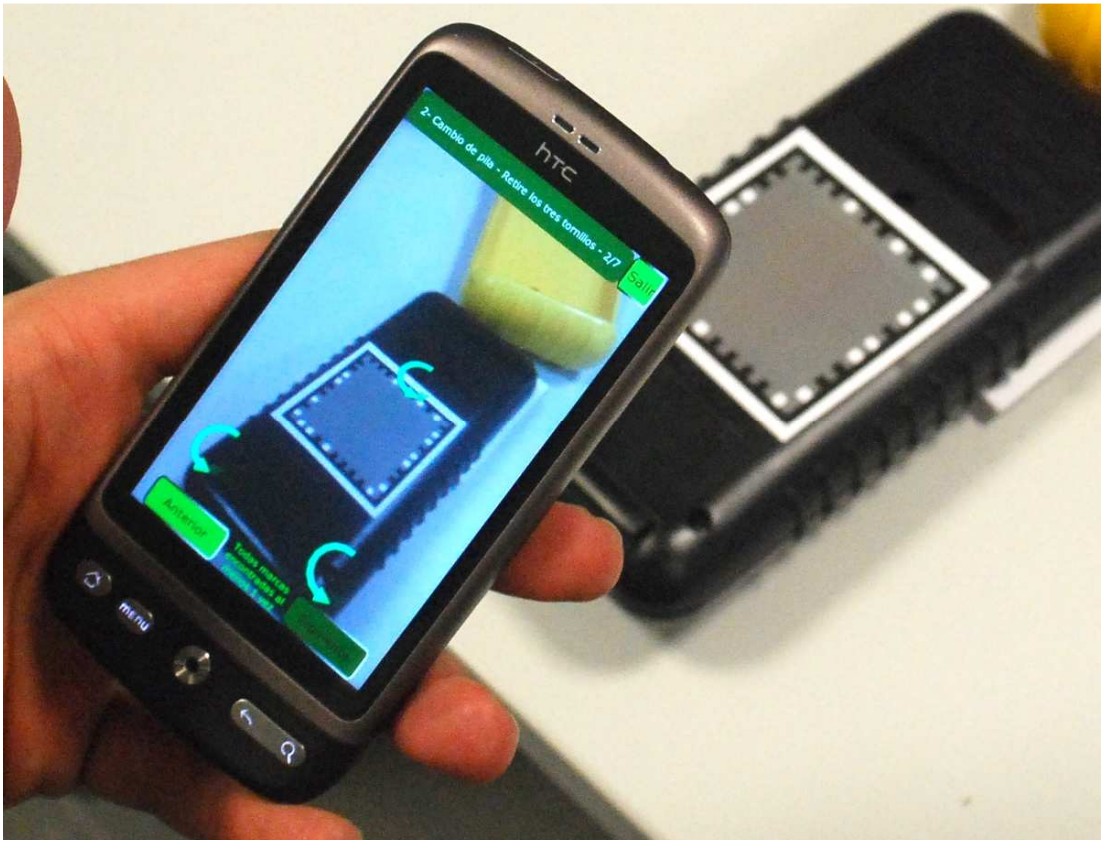


Ilustración 56 - Ejemplo de visualización de un solo script de giro con tres objetos.

Instrucción de desplazamiento:

El script muestra un objeto desplazándose de un punto a otro en un tiempo determinado, el objeto puede ir y volver al objeto o moverse solo en una dirección.

Al igual que "Giro" se cuenta con una lista de *Punto de interés* de posición, de objetos y de rotación inicial.

- Origen Relativo: Posición de origen del desplazamiento de cada objeto respecto al *Punto de interés* de referencia.
- Destino relativo: Posición de destino del desplazamiento.
- Vuelve: Boolean que indica si el objeto va y vuelve o si solo realiza una y otra vez el camino de ida.
- Tiempo Ida Vuelta: Tiempo total de ida y vuelta.

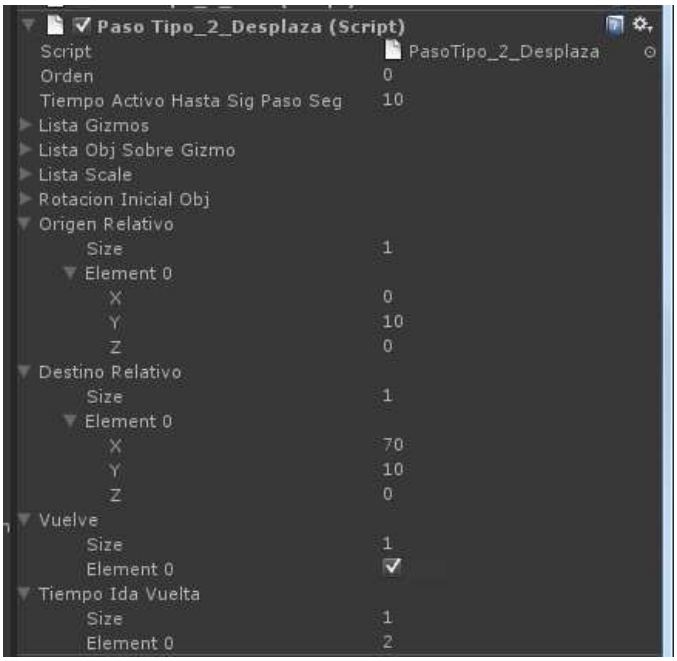


Ilustración 57 - Menú Unity3d ejemplo instrucción desplazamiento.

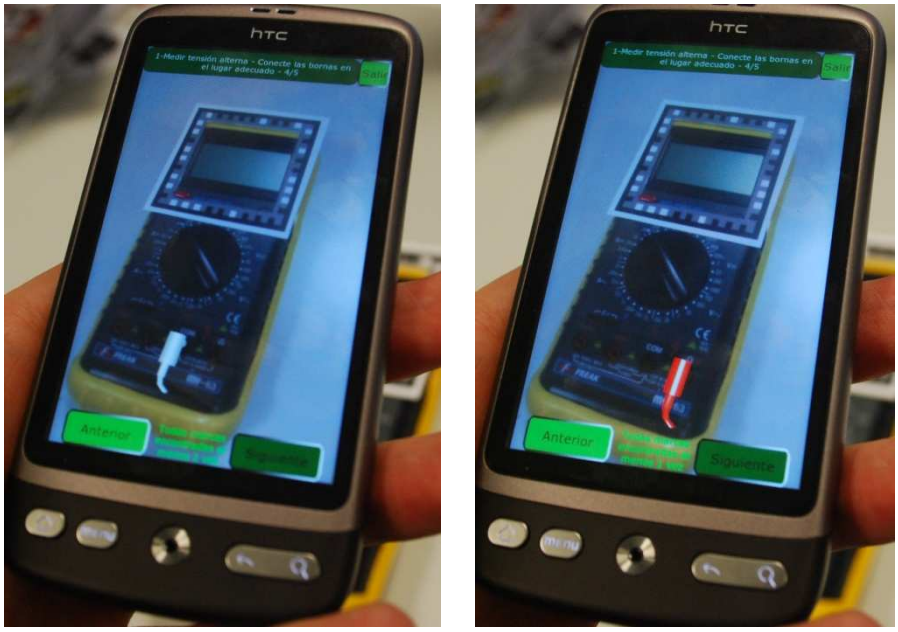


Ilustración 58 - Ejemplo desplazamiento objetos.

Instrucción de parpadeo:

Se muestra un objeto que parpadea, se muestran a continuación los campos distintos a los dos scripts anteriores.

- Tiempo visible
- Tiempo no visible

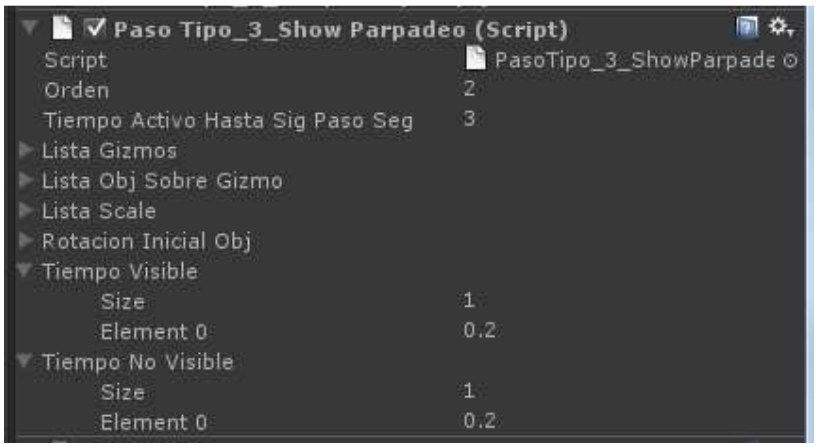


Ilustración 59 - Menú Unity3d ejemplo de instrucción de parpadeo.

Instrucción que muestra un texto que vuela:

Se muestra un texto sobre un *Punto de interés* determinado, el texto se une al punto exacto al que se refiere mediante una línea.

- Skin Texto: Define el tipo de letra que queremos utilizar, es más importante al estar en plataforma Android que no permite fuentes

dinámicas y no es aconsejable crear muchos tamaños de letra distinta puesto que ocupan bastante espacio.

- Lista Gizmos: al igual que en casos anteriores es la posición en la que se colocará la línea que va hasta el campo de texto.
- Lista Marcadores Sobre Gizmos: Para poder mostrar o ocultar el texto se debe indicar sobre que marcador está este.
- Lista Textos: Los textos que se mostrarán sobre cada uno de los gizmos.
- Lista Colores: Color de fondo de los textos, puede usarse, si así se define, como una regla de avisos: fondo rojo peligro, fondo verde todo correcto…

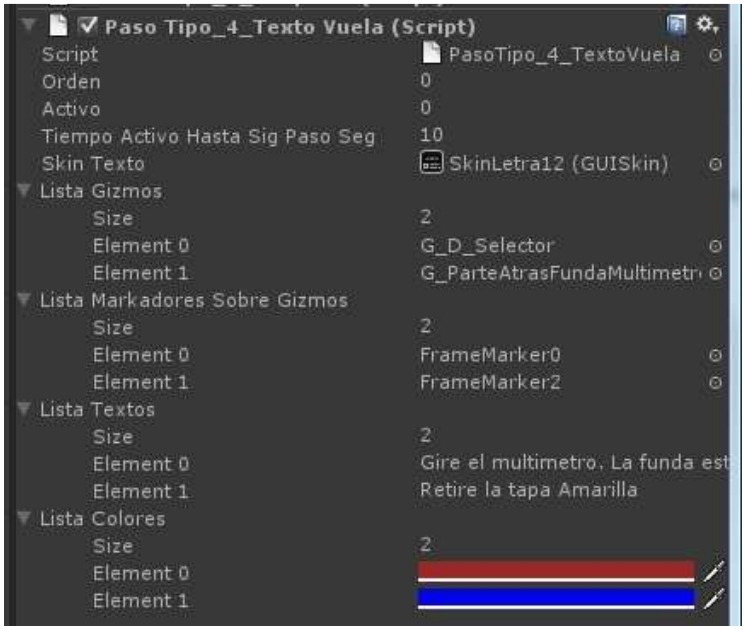


Ilustración 60 - Menú Unity3d editor instrucción texto vuela.

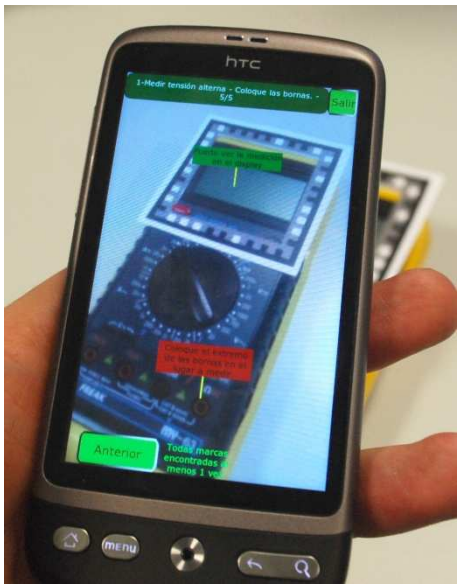


Ilustración 61 - Visualización de textos sobre multímetro.

Instrucción de operación condicional:

Esta operación es algo distinta de las anteriores, en ella se muestra un mensaje de texto, el sistema permite añadir dos botones con texto configurable.

Al pulsar el botón existen dos opciones, mostrar un nuevo texto con un botón de continuar o saltar a una instrucción.

Esta operación nos permite añadir validaciones de usuario y generar saltos en los pasos de la operación. Podemos incluir en este punto también validaciones de funcionamiento para guiar al usuario a otras operaciones, si por ejemplo pedimos al usuario que pulse el botón de encendido del sistema, le podemos preguntar si ve algún dato sobre la pantalla, si no es así podemos sugerirle ir a la instrucción de cambio de pila o cambio de fusible que son las dos situaciones más posibles por las que la pantalla no se enciende.

- Skin Texto: Al igual que en el caso anterior especificamos el tipo de letra.
- Texto Pregunta: Primer texto que verá el usuario al llegar a este paso y que debería ser una pregunta.
- Texto Boton Opc X: El texto que tendrá el botón que se muestra debajo del texto, si no hay texto el botón no se pinta.
- Opc XSalta O Texto: si esta marcado es una operación de salto y al pulsar en el botón correspondiente a dicha opción saltaremos a la instrucción asignada. El sistema debe almacenar todo estos saltos por si el usuario querer ir adelante o atrás en los pasos.
- Operación Salto Opción X: operación a la que se salta.
- Texto Seleccionado OpcX: Si no es una operación de salto mostraremos al usuario un nuevo mensaje con la única opción de dar a continuar.

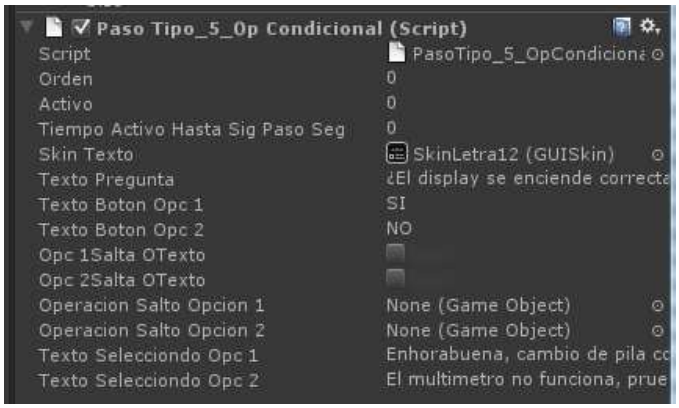


Ilustración 62 - Menú Unity3d de la instrucción operación condicional.

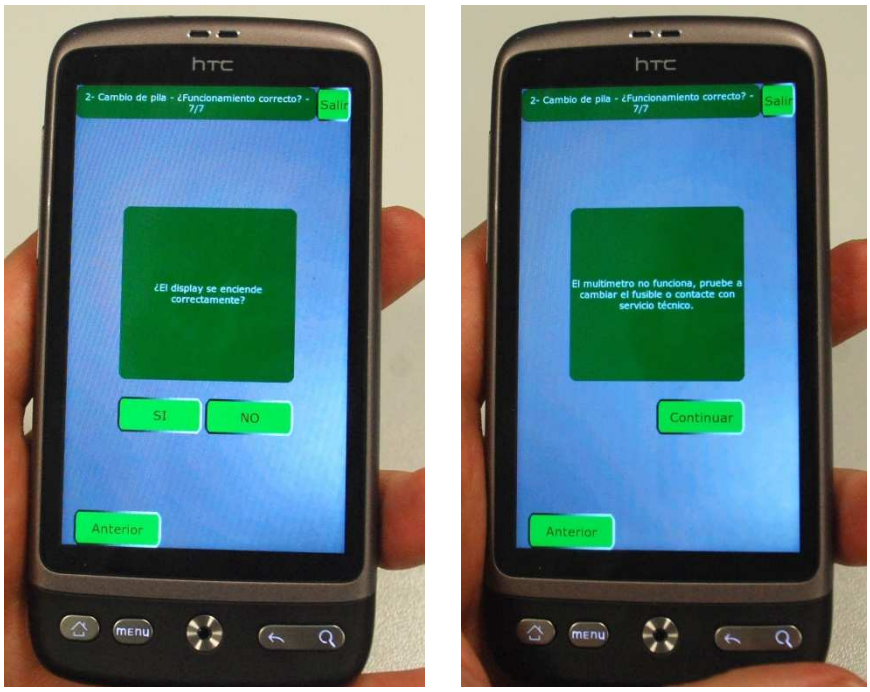


Ilustración 63 - Operación condicional, pregunta y mensaje de aviso.

4.5 Pasos para la edición de manuales sobre la base creada con Unity3D

A continuación veremos el proceso de edición que se ha seguido para crear el manual del multímetro, estos pasos se muestran de forma genérica de modo que el documento pueda servir como guía para la creación de manuales para otros dispositivos.

Esta descripción supone que se han seguido los pasos descritos en **3.2** para configurar toda la escena inicial.

Para mantener la forma de nombrar las distintas partes de *Vuforia* las marcas indicadas en el proceso de producción como Marcador_X se llamarán *FrameMarkerX.* Los *Puntos de referencia* se nombran como G_xx puesto que son *Gizmos* en *Unity3D*

1. Añadir y configurar prefabas

Para llegar al resultado de la **Ilustración 64** deberemos arrastrar a la escena los siguientes prefabs "ARCamera" y "MenuPrincipal"

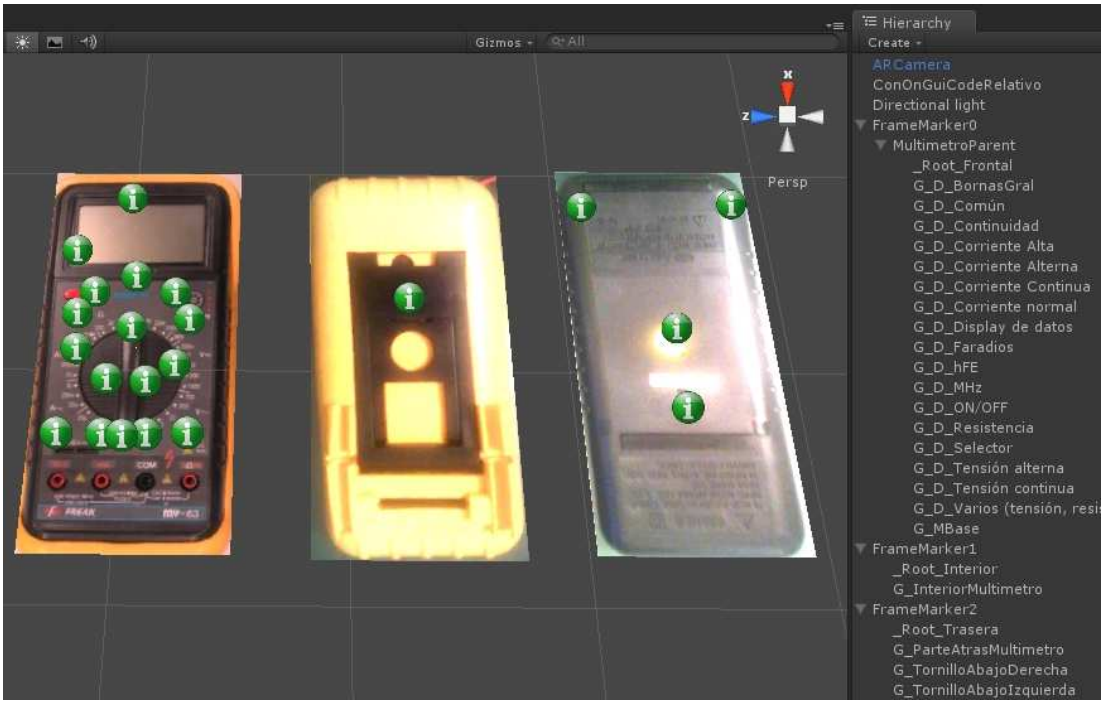


Ilustración 64 - Estado inicial para comenzar edición.

ARCamera:

Es la cámara de la aplicación contiene el script QCARBehaviour que permite configurar las opciones generales del sistema de traking, también cuenta con otros dos scripts de Vuforia (DataSetLoadBehaviour y DefaultInitializacionErrorJandler)

El script ControlRA contiene una lista de los marcadores involucrados en el manual, dicha lista es para asegurar que todos los objetos que depende de un marcador estén visibles o no visibles cuando lo está su marca. Este proceso en la implementación se realiza en el LateUpdate puesto que durante los pasos se añaden de forma dinámica objetos como hijos.

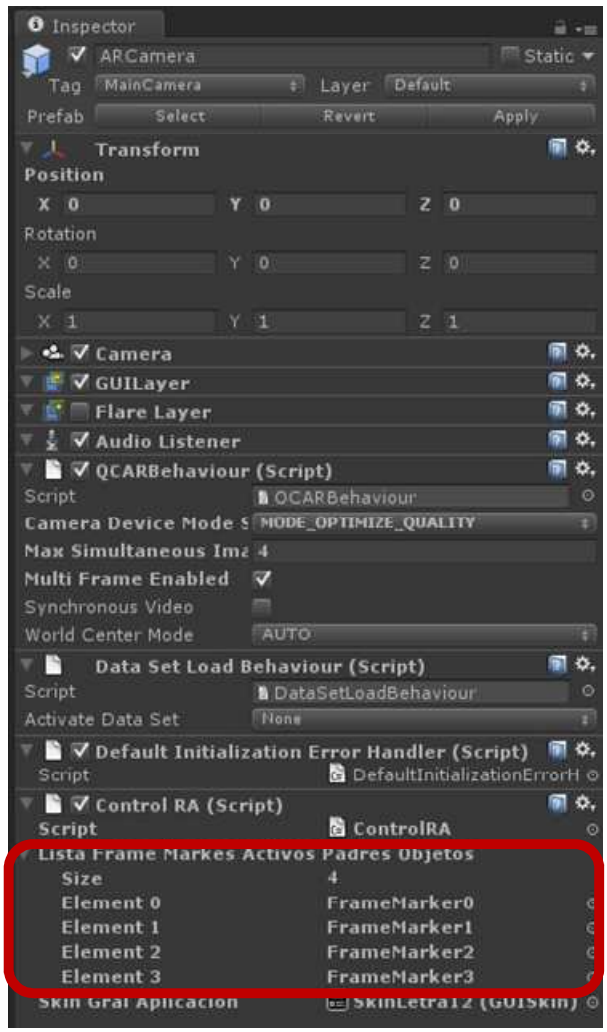


Ilustración 65 - Configuración de ARCamera.

2. Crear la estructura del manual

Arrastre el prefab "ListaOperaciones" se creará un GameObject vacio con el script "L1_ManualOperaciones" y dos operaciones iniciales, cada una de ellas tiene asociadas dos "PasosTipo".

Para crear operaciones nuevas se debe duplicar la "Operación_X" y ampliar el array de Lista de operaciones arrastrando la nueva operación. De este sencillo modo se crearán las distintas operaciones en el menú del manual.

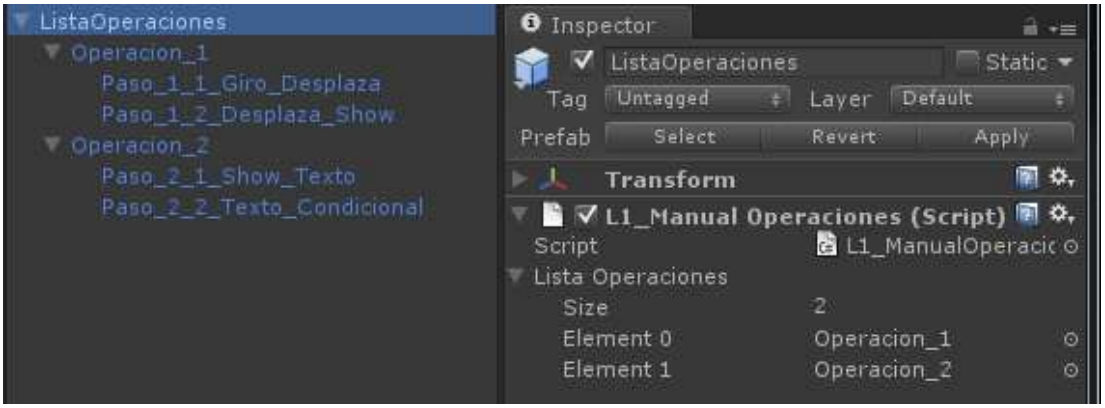


Ilustración 66 - Prefab con la base de lista de operaciones, pasos e instrucciones.

En el paso deberemos editar el titulo de la operación que será el nombre que tomará el botón del menú de selección. Igualmente deberemos asignar la ARCamera que hemos configurado en el paso dos y un skin para los textos.

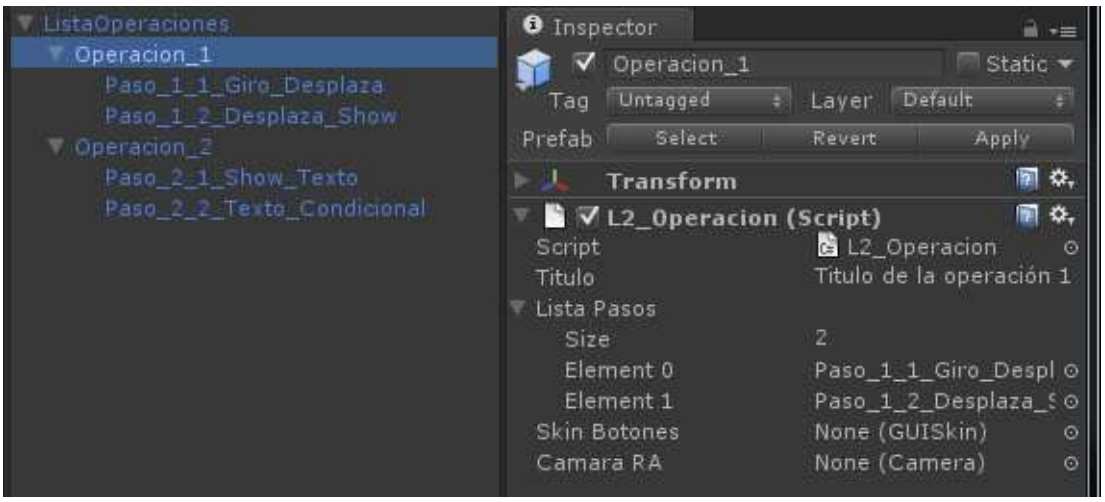


Ilustración 67 - Configuración de la operación.

En el prefab de ejemplo la operación 1 tiene dos pasos "Paso_1_1_Giro_Desplaza" y "Paso_1_2_Desplaza_Show" el nombre en este caso es solamente descriptivo para saber qué tipo de instrucciones de ejemplo contienen.

En el primer paso "Paso_1_1_Giro_Desplaza" **Ilustración 68** podemos ver las dos instrucciones de ejemplo que tiene dicho paso, una instrucción de Giro y otra de Desplazamiento.

Además el script L3_Paso_Operacion contiene un array de marcas no activas en el paso. Dicho array nos permite mostrar un objeto genérico que indica que dicha operación no tiene información asociada en este paso.

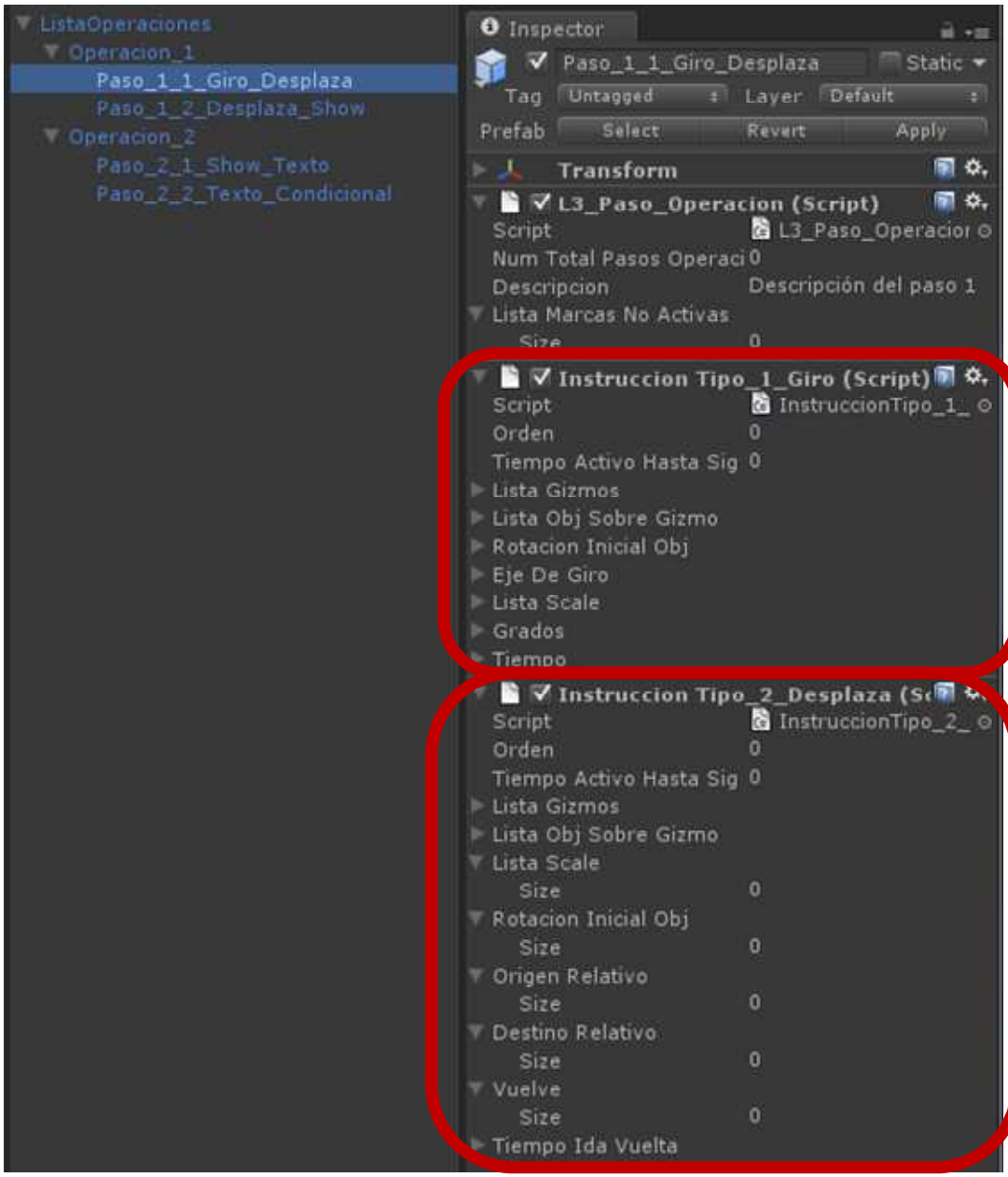


Ilustración 68 - Ejemplo paso con dos instrucciones.

Estos son los elementos necesarios para crear todos los pasos e instrucciones de un manual, todas las operaciones estarán asociados al script L2_Operacion todos los pasos estarán asociados a L3_Paso_Operacion y el objeto que contiene dicho script debe tener asociados los "InstruccionTipo_X_" necesarios para visualizar elemento.

3. Configuración de instrucciones

Se verá ahora un ejemplo práctico para editar las instrucciones tipo y para configurar los tiempo de funcionamiento de cada una de ellos.

Cada script que se añade sobre un paso tiene unos campos idénticos que son el orden y el tiempo hasta el siguiente paso. En la **Ilustración 69** podemos ver las posibilidades de configuración de ejecución de las distintas instrucciones, tenemos 4 operaciones que se ejecutan en el siguiente orden:

- La primera instrucción tiene "Orden 0" por lo que se ejecutará la primera, tiene además un tiempo de ejecución de 3 segundos. Al no haber más instrucciones con dicho orden, una vez terminada se visualizarán las instrucciones de "orden 1".
- Las instrucciones 2 y 3 tiene "Orden 1" por lo que se ejecutarán simultáneamente. La instrucción 2 tiene un tiempo de ejecución de 15 segundos y la operación 3 tiene un tiempo de 3 segundos. La operación 2 se ejecutará durante los 15 segundos, la instrucción 3 se ejecutará 3 segundos. Aun así la operación 3 puede contener múltiples "ciclos", es decir, el parpadeo de objeto puede mostrarse las veces necesarias durante esos 3 segundos.
- Finalmente se ejecutará la instrucción 4 con "Orden 2" durante 3 segundos.

Como se puede ver podemos tener instrucciones secuenciales o simultaneas dependiendo de la configuración de orden y tiempo.

Ilustración 69 - Ejemplo de configuración de tiempos.

Los campos que nos permiten configurar cada tipo de instrucción son bastante similares. Como ejemplo veremos un script configurado de giro de la **Ilustración 70**. En el campo 1 tenemos una "Lista de Gizmos" que serán las posiciones sobre las que se colocarán los objetos que queremos mostrar. Para asignarlos solamente tenemos que arrastrarlos desde Hierarchy hasta el inspector.

En el campo 2 "Lista de Objetos Sobre Gizmos" que serán los objetos que mostraremos girando sobre cada uno de los anteriores gizmos. En el campo 3 podemos ver la "Rotación Inicial de Objetos" en el indicaremos el giro inicial que tendrá el objeto sobre cada gizmo, tendremos un eje de rotación por cada uno. En el 4 se indica el eje de giro de cada uno de los objetos. El campo 5 indica la escala del objeto. El 6 y 7 nos muestran los grados de giro y el tiempo que tarda en realizarlos de modo que con ellos se puede variar la velocidad de giro.

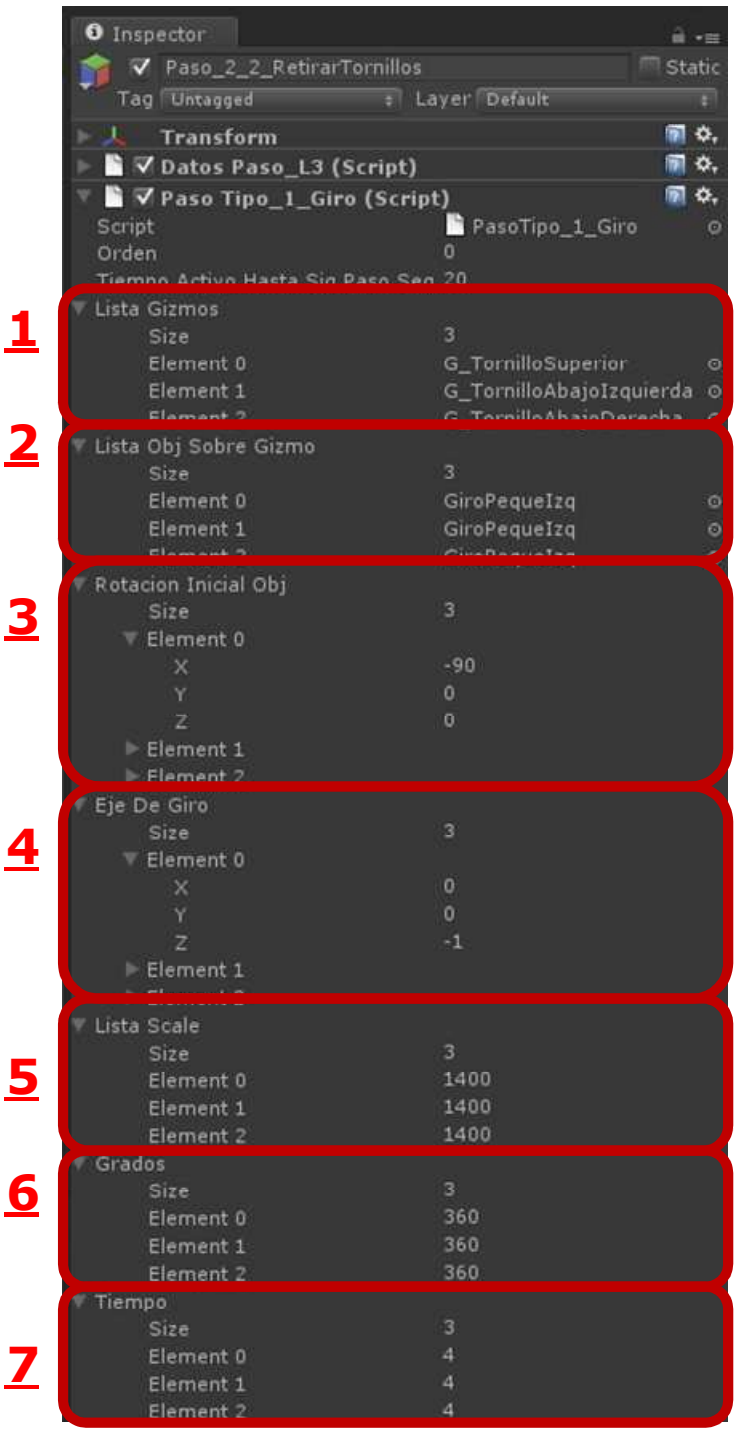


Ilustración 70 – Ejemplo de configuración de instrucción giro.

Gracias a la versatilidad del script podemos mostrar rotando todos los objetos que queramos sobre todas las referencias que queramos configurando los datos en una sola ventana.

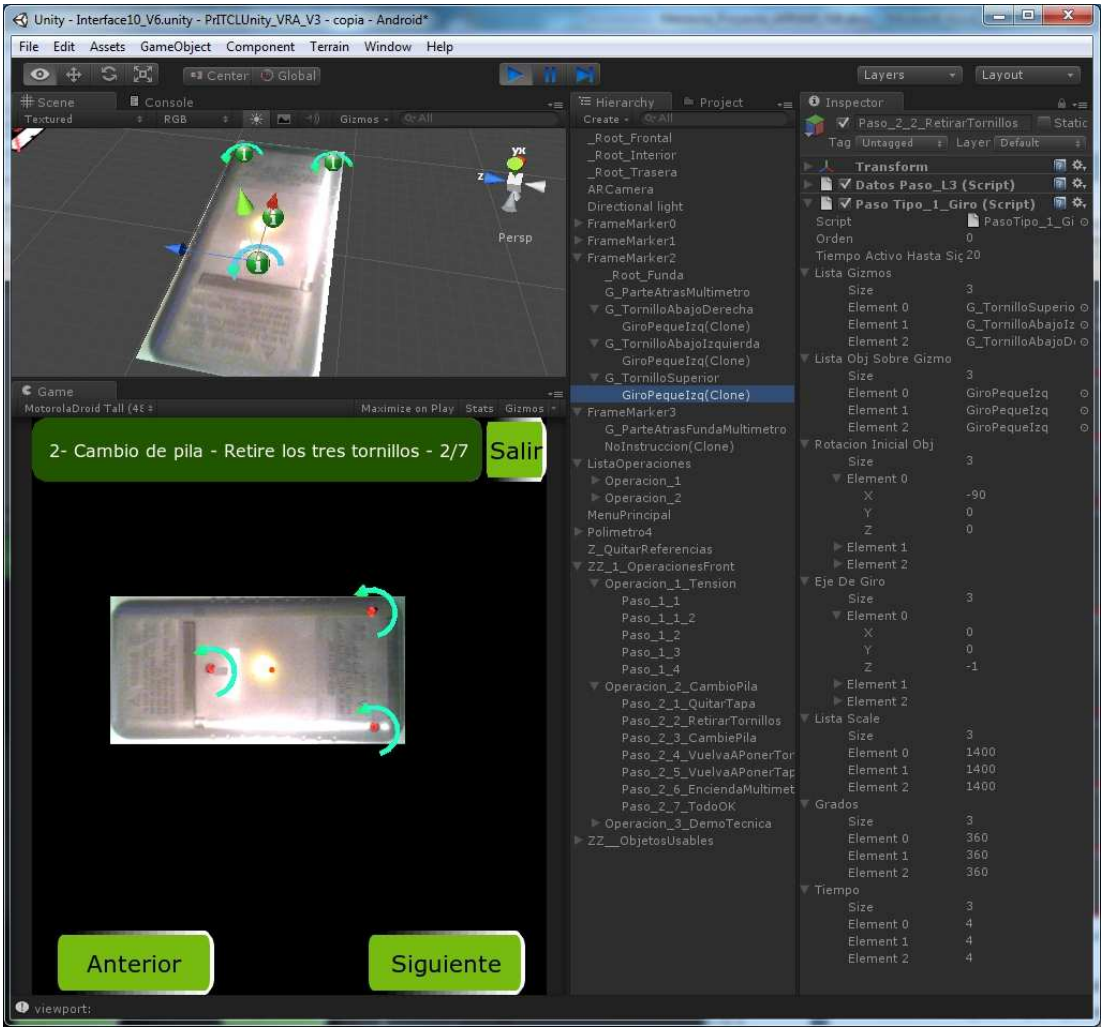


Ilustración 71 - Ejemplo giro visualización en modo edición.

La mayoría de los scripts de instrucciones creadas para la base del manual mantienen la filosofía aquí indicada. La única instrucción diferente es la instrucción de operación condicional. Dicha instrucción se deberá utilizar cuando se le quiera preguntar o avisar de algo al usuario.

En la **Ilustración 72** podemos ver la configuración de dicha instrucción. En el 1 deberemos introducir la pregunta que queremos hacerle al usuario, en el punto 2 incluiremos los textos de los botones que queremos mostrar. Si no deseamos que se visualice un botón dejaremos el campo vacio.

En la opción 3 indicaremos si la operación muestra otro texto al usuario o si por el contrario salta a una nueva instrucción. En 4 indicaremos a que instrucción debemos saltar en caso de que se pulse la opción de salto. Finalmente en el 5 indicaremos que texto se mostrará al usuario si no se realiza un salto.

Esta operación nos permite por ejemplo preguntar al usuario si ve texto en una pantalla, si es así le indicaremos que todo es correcto, si no lo ve podemos indicarle que realice alguna operación de reparación. También podemos preguntarle al usuario si el nivel de aceite es correcto y si lo es saltarnos varias instrucciones que le indiquen como se cambia el aceite.

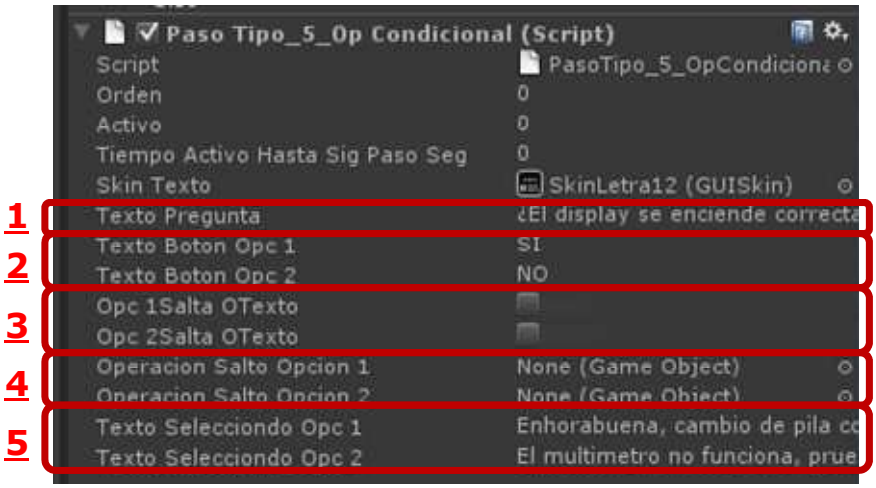


Ilustración 72 - Configuración de la instrucción tipo OpCondicional.

Otros:

Dentro del proyecto se han creado dos escenas "MultimetroImagenes" y "MultimetroMarcadores" esas dos escenas muestran la facilidad para cambiar entre marcadores de imágenes y marcadores con identificador. Toda la funcionalidad y scripts están preparados para admitir los dos tipos de marcas. Lo único que debemos hacer es importar marcas de un tipo u otro de *Vuforia* en nuestro proyecto y colocar toda la jerarquía de *Planos de referencia* y *Puntos de interés* como hijos de una u otras. Igualmente en los scripts de pasos e instrucciones podremos arrastraremos las marcas sean de un tipo u otro.

Siguiendo los pasos aquí indicados podemos editar un manual de cualquier tipo, sin tener ningún conocimiento de programación, xml, motores gráficos… Si queremos que el manual sea visualmente más atractivo podemos contar con un modelador en 3d para que nos cree geometrías acorde con nuestro manual pero estas no son esenciales para editar un manual.

Universidad Rey Juan Carlos

Máster Oficial en Informática Gráfica,
Juegos y Realidad Virtual

PROYECTO FIN DE MÁSTER

PROpuesta de PROceso de PROducción de Manuales en Entornos de Realidad Aumentada (PRO³MERA)

CUARTA PARTE

CONCLUSIONES Y TRABAJO FUTURO

5 Resultados

A continuación se muestran los resultados de rendimiento del manual implementado, este se medirá con el número de imágenes por segundo (fps) que consigue renderizar el teléfono inteligente en distintas situaciones.

Las pruebas se han realizado sobre un teléfono inteligente *HTC Desire* con *Android 2.2* por ser el hardware del que se dispone. Este hardware además es una buena referencia ya que es un modelo salido en Febrero de 2010 con un procesador de 1 GHz, por lo tanto, si el manual funciona adecuadamente en él se puede asegurar que tendrá un buen funcionamiento en la mayoría de los nuevos teléfonos inteligentes.

Se realizan pruebas para **comprobar el rendimiento del manual en los escenarios típicos de funcionamiento del mismo: menús, carteles de texto y renderizado de objetos3D.** Todas las pruebas se han realizado con una carga normal y con una carga más extrema. Todas las pruebas se realizan en un entorno con buena iluminación y visibilidad de la marcas.

El objetivo es tener una medida de las capacidades de reconocimiento y renderizado del manual implementado.

Los valores proporcionados se han medido en todos los casos durante 5 segundos de ejecución dejando quieto del teléfono inteligente. Las pruebas se han realizado en un entorno bien iluminado para evitar problemas de reconocimiento. A continuación se describe en detalle cada uno de ellos:

Pruebas realizadas:

1. Visualización de menús estándar:

   Se han seleccionado para esta prueba dos menús representativos, el primero de ellos es el menú principal que cuenta con 6 botones y 2 2Dtextures y un label. El segundo es el manu de "Acerca de" con un botón y un scrool de texto.

2. Visualización de carteles de texto.

   Los carteles de texto se utilizan para ver las partes del multímetro y para proporcionar información al usuario. Los carteles de texto se ordenan para intentar que unos no tapen a otros siendo este un proceso costoso.

3. Renderizado de objetos 3D.

   En todo manual se mostrarán objetos 3D aunque normalmente estos suelen tener geometrías simples. Se han realizado dos tipos de pruebas la primera con geometrías sencillas un objeto de 350 triángulos y la segunda con objetos más complejos (uno de 7.200 triángulos y otro de 1.300)

4. Renderizado objetos 3D y texto de forma conjunta.

   Se han realizado pruebas de rendimiento de los dos elementos en conjunto situación que puede darse en muchos casos.

A continuación se muestra la tabla de estadísticas de los datos medidos:

		Min fps	Max fps	Media fps	Triángulos
MENUS	Menu principal.	6,88	20,15	14,02	
	Menu "Acerca de".	9,55	18,42	14,18	
Texto Vuela	Mostrar y ordenar textos 4.	5,24	8,4	7,27	
	Mostrar y ordenar textos 17.	3	3	3	
	1 marca 1 texto.	5,88	11,22	8,98	
	1 marca 4 textos.	6,37	11,55	8,23	
	1 marca 16 textos.	3,06	3,06	3,06	
	4 marcas 4 textos.	6,8	11,3	8,31	
	4 marcas 16 textos.	3	3,4	3,18	
Objetos 3D	1 marca 1 objeto simple.	7,49	17,14	10,32	350
	1 marca 1 objeto complejo.	5,98	10,29	8,5	7.200
	1 marca 4 objetos simples.	6,7	12,49	9,08	1.400
	1 marca 4 objetos complejos.	6,8	9,85	8,12	28.800
	4 marcas 4 objetos simples.	6,04	9,27	8,5	1.400
	4 marcas 4 objetos complejos.	6,4	10,77	8,01	28.800
	4 marcas 16 objetos simples.	5,7	9,03	7,86	5.600
	4 marcas 16 objetos complejos.	5,5	8,31	6,8	68.000
Objetos y textos	1 marca 1 texto 1 objeto 3D simple.	7,8	12,86	9,19	350
	1 marca 4 textos 4 objetos 3D complejos.	4,8	8,2	6,88	28.800
	4 marcas 4 textos 4 objetos 3D simples.	6,84	10,98	7,9	1.400
	4 marcas 16 textos 16 objetos 3D complejos.	2,8	2,8	2,8	68.000

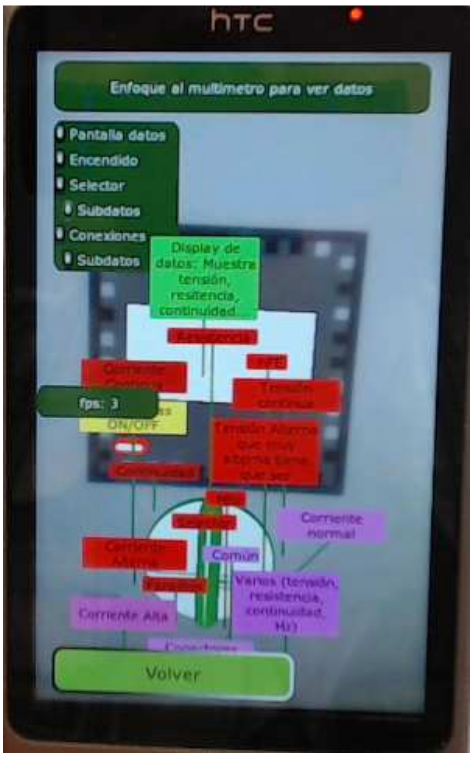
Ilustración 73 - Resumen de rendimientos.

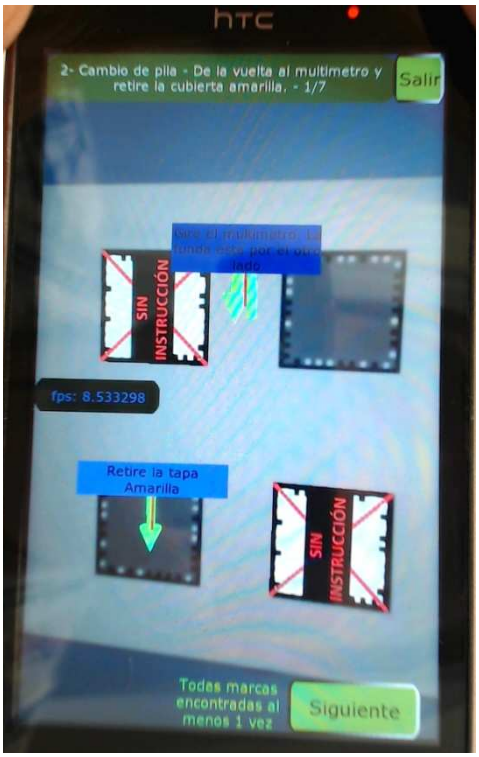
De forma general, el rendimiento observado es muy bueno, se pueden ver con fluidez prácticamente todas las pantallas y opciones implementadas.

Unity3D maneja de forma muy eficiente los modelos 3D aun aumentado el número de triángulos muy por encima de lo que es probable que sea necesario visualizar en un manual de mantenimiento, la aplicación nunca baja de los 6,8 fps.

El proceso de mostrar y ordenar textos 3d sin embargo si que puede llegar a ralentizar la visualización. Se pueden observar bajadas de rendimiento cuando hay muchos textos y cuando estos están muy cerca. Aun así no es probable que se le quiera mostrar al usuario demasiados carteles de texto al mismo tiempo.

No se podido apreciar diferencia de rendimiento usando marcas con identificador y marcas con imagen. Se ha podido ver que el tiempo necesario para reconocer la marca es menor en las marcas con identificador que en las marcas con imagen. Una vez reconocida por primera vez se pierden en condiciones similares. No se ha encontrado una manera exacta de poder medir dichos tiempos de respuesta. La propia librería de *Vuforia* tampoco proporciona datos de procesado que se puedan utilizar como referencia.





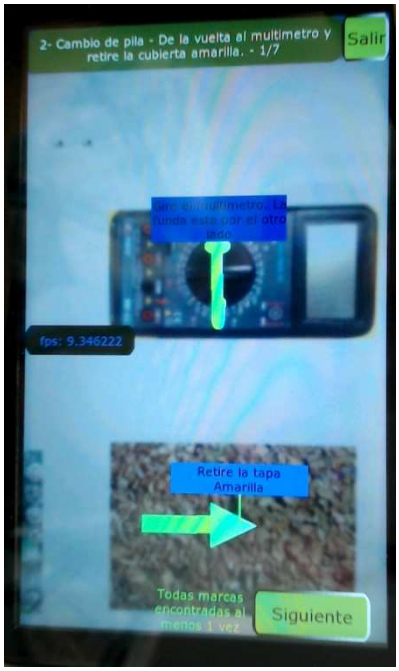


Ilustración 74 – Imágenes de la medición de rendimiento con marcadores con identificador e imagen.

El reconocimiento de marcadores tiene un buen rendimiento, se puede destacar las siguientes características de cada una de los tipos de marcas:

- El seguimiento de marcadores con identificador es muy estable en todas las posiciones. La distancia a la que se reconocen las marcas permite tener una buena amplitud en la imagen que nos proporciona la cámara y ver todo el multímetro.

  Cuando se pierde la marca la recuperación es rápida.

  El mayor problema con estas marcas se produce cuando hay varias marcas con diferente iluminación, esto se debe a que utiliza tres tonos para reconocer las marcas.

- El seguimiento con imágenes también es muy estable una vez reconocida, sin embargo, es necesario algo más de tiempo para reconocerla por primera vez. Estás marcas permiten que nos acerquemos mucho más sin perder el seguimiento para, por ejemplo, poder ver el selector del multímetro con más detalle.

  En el caso del multímetro la imagen frontal es aceptable para el reconocimiento y, aunque, al procesarla con el programa de *Vuforia* da una calificación de seguimiento *"poor"* permite realizar un seguimiento aceptable.

6 Conclusiones y líneas futuras

En términos generales se han cumplido los objetivos que se marcaron al inicio del proyecto, se ha creado un manual de mantenimiento de un multímetro sobre un dispositivo móvil con software de bajo coste. El manual creado proporciona además una base para la realización de otros manuales. Se han detallado un proceso productivo y una arquitectura para la creación de mantenimiento que puede ser implementado con otro tipo de herramientas a las aquí seleccionadas. Durante el proceso se han generado varios scripts reutilizables que permiten, de una forma sencilla, crear la arquitectura base para crear manuales, así como las instrucciones tipo básicas necesarias.

Además al estar actualizadas tanto herramientas como librerías y hardware se ha conseguido realizar un sistema con altas capacidades en un tiempo de desarrollo mucho menor que muchos de los proyectos estudiados en el estado del arte.

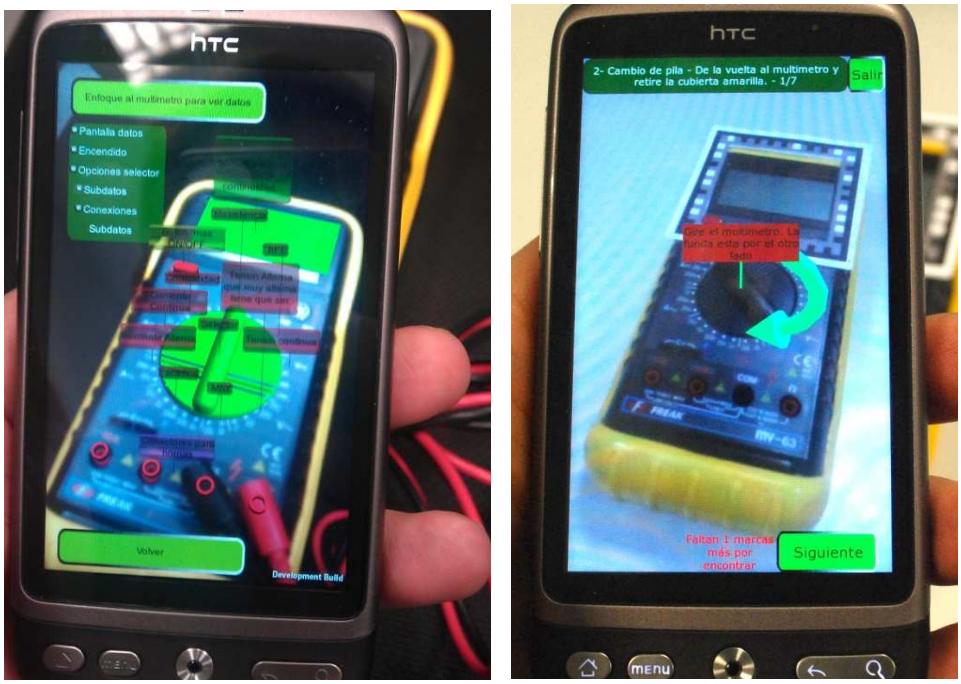


Ilustración 75 - Resultado de la implementación del manual.

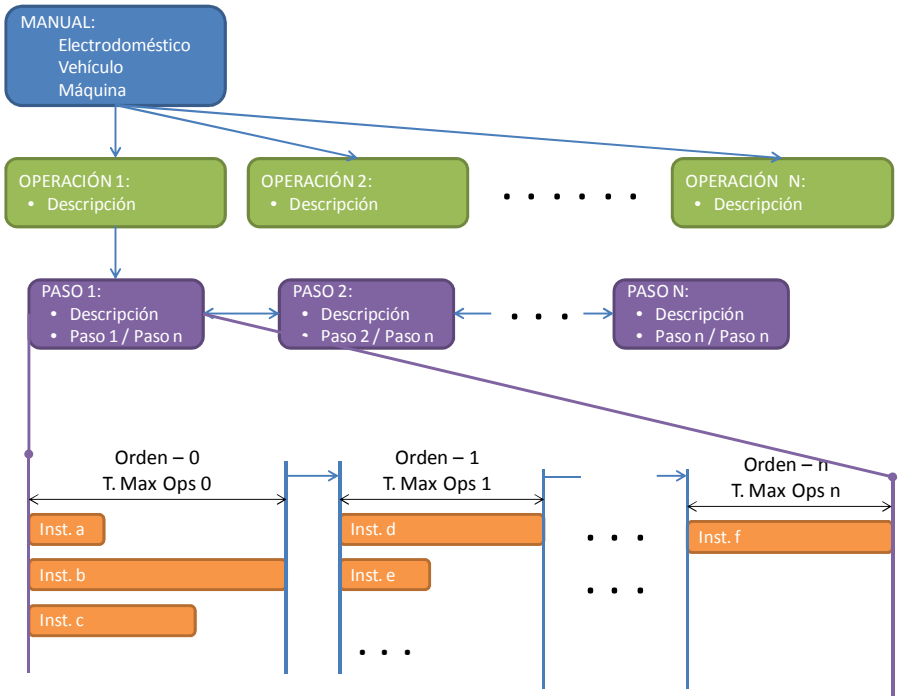


Ilustración 76 - Resumen resultados arquitectura.

El enfoque del proyecto orientado a mantenimiento o operaciones comunes en equipos disponibles para el gran público (multímetro, microondas, fotocopiadora, …) y no tanto a grandes productos (motor de un coche, Airbus, tanques, fábricas…) puede permitir popularizar este tipo de manuales. Con una baja inversión de las casas oficiales de dichos productos, se pueden crear manuales gratuitos de uso que pueden ser un elemento diferenciador para esa marca respecto a su competencia.

Se pueden destacar las siguientes mejoras que se han podido implementar respecto a otros proyectos:

- Mejoras de posicionamiento en la edición al utilizar planos de referencia para colocar puntos de interés.
- Plano base útil para la verificación del manual antes de tener que probarlo sobre la máquina real.
- Edición simple una vez creados los puntos de interés, el sistema solo requiere arrastrar referencias.
- Coste bajo de edición de manuales
- Coste bajo del hardware utilizado así como del software.
- Edición de las operaciones del manual, que es la parte que más horas necesita, no necesita de un especialista.
- Extensibilidad, un programador puede ampliar fácilmente la versatilidad o funcionalidades del manual.

- Desarrollado para teléfonos inteligentes, lo que permite una expansión y portabilidad de los manuales alta.
- Sistemas de asistencia al usuario del manual: Identificar que marcas no tienen información asociada, información de cuantas marcas faltan por encontrar.

Mejoras que quedan pendientes que no se han podido solucionar en el presente proyecto y que por lo tanto pueden quedar como líneas futuras:

- El reconocimiento y seguimiento de los objetos sobre los que se muestra la realidad aumentada siguen requiriendo normalmente añadir marcadores externos, lo que no es deseable en la mayoría de los casos. Este es uno de los mayores problemas de este tipo de productos.
- El usuario no tiene las manos libres. Utilizar un teléfono inteligente permite una gran versatilidad, portabilidad y bajo coste, pero el uso es incomodo puesto que las manos no están libres para trabajar.
- La edición del manual aun estando muy mejorada por las herramientas y facilidades de Unity3d requiere tiempo de edición y comprobaciones.

Durante el desarrollo general la parte de implementación ha sido relativamente amigable gracias a la facilidad de uso de la herramienta Unity3d y de la librería *Vuforia* de *Qualcomm*.

En cuanto a la implementación el despliegue sobre Android ha sido el que mayores problemas ha provocado. Inicialmente se produjeron problemas para la conexión y volcado de datos en el dispositivo físico. Posteriormente se produjeron problemas con el depurado, siendo estos más importantes, ya que en el dispositivo no es posible realizar una depuración y por lo tanto el desarrollo se retrasa. Se desarrollo para mitigar dicho problema un sistema simple de depurado que permitía ver valores y mensajes en tiempo real sobre la pantalla del dispositivo.

Se experimentaron mejoras en el cambio de la versión 3.4 a la 3.5 de Unity3d. Unity 3.4 al ejecutar el programa en Android se quedaba bloqueado en muchas situaciones en las que el editor no daba problemas. Estos problemas que solo eran visibles sobre el propio dispositivo con Android hacen muy complejo el depurado de la misma.

También se ha podido apreciar mejoras en la librería de reconocimiento de Qualcomm de la versión 1.7 a la versión *Vuforia*. Estas mejoras residen tanto en la robustez del reconocimiento de marcadores como en el sistema de enfoque de la cámara que es esencial en estos casos.

7 Bibliografía

[1] J.-Y. Didier, D. Roussel, M. Mallem, S. Otmane, S. Naudet, Q.-C. Pham, S. Bourgeois, C. Mégard, C. Leroux y A. Hocquard, «AMRA: Augmented reality assistance in train maintenance tasks» de *ISMAR*, 2005.

[2] «Vtt.fi» 2009. [En línea]. Available: http://virtual.vtt.fi/virtual/proj2/multimedia/index_dm.html.

[3] N. loannidis, *Augmented reality-base cultural heritage on-site guide,* 2002.

[4] «Imagilab» 2010. [En línea]. Available: http://www.imagilab.org/research.htm.

[5] «Metaio» [En línea]. Available: http://www.metaio.com/projects/kiosk/lego/.

[6] Tissot, [En línea]. Available: http://www.tissot.ch/?mod_redirection/id_reality.

[7] «Estarteco» Instituto Tecnológico de Castilla y León, 2012. [En línea]. Available: www.estarteco.com.

[8] «Qualcomm» Qualcomm, [En línea]. Available: https://developer.qualcomm.com/mobile-development/mobile-technologies/augmented-reality.

[9] «Oceanoutdoor» Oceanoutdoor, [En línea]. Available: http://www.oceanoutdoor.com/ocean-news/news/ocean-says-halo-birmingham-with-augmented-reality-angels/.

[10] «Blippar» Blippar, [En línea]. Available: www.blippar.com.

[11] «Ar-books» Bienetec, [En línea]. Available: http://www.ar-books.com/interior.php?contenido=publicados.php.

[12] «Learngears» Learngears, 2011. [En línea]. Available: http://learngears.com/2011/08/03/molecular-structure-1-0/.

[13] «Sony» Sony, [En línea]. Available: http://www.sony.co.uk/article/tv-size-guide.

[14] «Ikea» Ikea, [En línea]. Available: www.ikea.com.

[15] «TUM» Fakultät für informatik, [En línea]. Available: http://www.in.tum.de/en/research/research-highlights/augmented-reality-in-medicine.html.

[16] «Aimplab» [En línea]. Available: http://aimlab.wpi.edu/research/projects/MRI_Image_Overlay.

[17] «Wikitude» [En línea]. Available: www.wikitude.com.

[18] «Layar» [En línea]. Available: www.layar.com.

[19] S. Mader y B. Urban, «Creating Instructional Content for Augmented Reality based on Controlled Natural Language» de *Controlled Natural Language Concepts Virtual Reality Society of Japan: ICAT* , 2010.

[20] C. Knoepfle, J. Weidenhausen, C. L. y I. Stock, «Template based Authoring for AR based Service Scenarios» de *In Proceedings of IEEE VR*, 2005.

[21] P. Grimm, M. Haller, V. Paelke, S. Reinhold, C. Reimann y R. Zauner, «AMIRE - Authoring Mixed Reality» de *The First IEEE International Workshop on Augmented Reality Toolkit*, 2002.

[22] F. U., C. Brecher y Possel-Dölken, «Advanced Augmented Reality-based Service,» de *Technologies for Production Systems Smart Machining Systems*,

2007.

[23] B. Schwald, B. D. Laval, T. O. Sa y R. Guynemer, «An Augmented Reality System for Training and Assistance to Maintenance in the Industrial Context» de *The 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2003.

[24] G. Klinker, O. Creighton, A. H. Dutoit, R. Kobylinski, C. Vilsmeier y B. Brügge, «Augmented maintenance of powerplants: A prototyping case study of a mobile AR system» de *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*, 2001.

[25] V. Lepetit y P. Fua, Monocular Model-Based 3D Tracking of Rigid Objects: A Survey, Foundations and Trends in Computer Graphics and Vision, Vol. 1, Nr. 1, pp. 1-89, October 2005.

[26] H. Wuest y D. Stricker, «Tracking of industrial objects by using CAD models» de *Journal of Virtual Reality and Broadcasting. Online journal 4 (2007)*.

[27] S. Lee, Z. Lu y H. Kim, «Probabilistic 3D Object Recognition with Both Positive and Negative Evidences» de *Computer Vision (ICCV), IEEE International Conference*, 2010.

[28] D. Glasner, M. Galun, S. Alpert, R. Basri y G. Shakhnarovich, «Viewpoint-Aware Object Detection and Pose Estimation» de *ICCV 2011*..

[29] A. Pedram, A. Tamim y D. Rudiger, «Accurate Shape-based 6-DoF Pose Estimation of Single-colored Objects» de *International conference on Intelligent Robots and Systems*, 2009.

[30] L. Schmid y Cordelia, «Multi-View Object Class Detection with a 3D Geometric Model» de *IEEE Conference on Computer Vision & Pattern Recognition. 2010*.

[31] S. Michael, G. Michael y S. Bernt, «Back to the Future: Learning Shape Models from 3D CAD Data» de *Proceedings of the British Machine Vision Conference*, 2010.

[32] W. Hu y S. C. Zhu, «Learning a probabilistic model mixing 3D and 2D primitives for view invariant object recognition» de *CVPR*, 2010.

[33] H. Stefan, K. Oliver, N. Nassir, P. Fua y L. Vincent, «Real-Time Learning of Accurate Patch Rectification» de *CVPR Computer Vision and Pattern Recongnition*, 2009.

[34] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua y V. Lepetit, «Real-Time Learning of Accurate Patch Rectification» de *CVPR*, 2009.

[35] C. Michael, L. Vincent, F. Pascal, K. Kurt, B. James, M. Patrick, G. Willow y P. Menlo, «Compact Signatures for High-speed Interest Point Description and Matching» de *International Conference on Computer Vision ICCV*, 2009.

[36] P. Youngmin, L. Vincent y W. Woontack, «Multiple 3D Object Tracking for Augmented Reality» de *International Symposium on Mixed and Augmented Reality*, 2008.

[37] H. Stefan, B. Selim, N. Nassir, F. Pascal y L. Vincent, «Online Learning of Patch Perspective Rectification for Efficient Object Detection» de *Conference on Computer Vision and Pattern Recognition*, 2008.

[38] H. Stefan, L. Vincent, I. Slobodan, F. Pascal y N. Nassir, «Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects» de

CVPR Computer Vision and Pattern Recongnition, 2010.

[39] J. Liebelt, C. Schmid y K. Schertler, «Viewpoint-independent object class detection using 3d feature maps» de *In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2008*.

[40] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolidge, N. Navab y V. Lepetit, «Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes» de *In Proceedings of the International Conference on Computer Vision*, 2011.