Computación ubicua e intelgencia ambiental

Smiley-Killer

Universidad de Granada

José Manuel Luque Burgos

Presentación

En este documento se detallan las fases transcurridas para el desarrollo de una aplicación situada en el campo de la computación ubicua e inteligencia ambiental, más en concreto realidad aumentada.
Se explicarán las posibles ideas que me han surgido y el porqué he elegido la llevada a cabo así como su explicación técnica sin entrar en mucho nivel de detalle.

Ideas descartadas

Juego de chapas:

La idea consiste en llevar a la pantalla de nuestro smartphone/tablet el típico juego de chapas pero con el añadido de la realidad aumentada, lo que permitiría ver en el dispositivo para cada chapa con un marcador identificativo su correspondiente jugador asociado y definir de igual manera el balón.
Ésta aplicación sería de carácter lúdico y podría interesar a empresas las cuáles vendan productos con chapa y aún más si su imagen está vinculada al deporte (coca-cola, cruzcampo…)
¿Porqué no me he decantado por esta idea?
Debido a mi total desconocimiento en la programación tanto en android como con cualquier sdk de realidad aumentada me pareció algo difícil de llevar a cabo para obtener un resultado tangible a corto-medio plazo.

Visualización interactiva de gráficos:

El sistema funcionaría en dispositivos móviles y la idea básica es mostrar datos de una forma interactiva en gráficos. Imaginemos una conferencia en la que se muestra un marcador de realidad aumentada en el proyector, los asistentes podrían enfocar con las cámaras de sus móviles para ver el gráfico relacionado a ese marcador e interactuar con él, consultando datos adicionales en el mismo instante como podrían ser gráficos relacionados, fuentes de los datos…
¿Porqué no he llevado a cabo esta idea?
En primera instancia fué la idea que pensaba desarrollar pero tras darle algunas vueltas no terminé de ver una situación natural una conferencia en la que todo el mundo apuntase con su dispositivo al mismo sitio, ya que entonces esto dejaría de ser una conferencia valga la redundancia. Podría ser interesante llevarlo acabo en otro tipo de entornos donde no se requiera atención a otra cosa, por ejemplo gráficos de información en un museo.

Otro de los factores a tener en cuenta es que no fuera sabido como abordar el problema en lo que a programación respecta.

Idea final

Juego arcade: Smiley-Killer

El juego consiste en usar una etiqueta reconocida por el sdk con el que se ha desarrollado el programa, a partir de el cual y enfocándolo con la cámara de nuestro dispositivo móvil podremos iniciar una partida en la que empezaran a nacer personajes desde la posición en la que se encuentra el marcador, deberemos acabar con ellos lo antes posible disparandoles. Cuanto más lejos del centro de nacimiento esté más puntos obtendremos pero si dejamos que se alejen demasiado desaparecerán y perderemos una vida.

¿Qué aporta de novedoso este juego?
El juego que cambia la dinámica de los juegos a los que acostumbramos en los que se muestra un mundo virtual sin detalles del nuestro propio.

¿Qué problema pretende resolver?
Como la mayoría de juegos, no intenta resolver un problema determinado sino ofrecer entretenimiento de una manera novedosa y llamativa.

¿Porqué lo elegí?
En primer lugar y como he descrito en los apartados anteriores, el desconocimiento de la programación para dispositivos móviles a sido un factor decisivo y por tanto éste ejemplo era más fácil de abordar que el resto además de tener mayor documentación.

Smiley-Killer

Requisitos funcionales y no funcionales

- Desarrollo en unity3D
- Vuforia como SDK para la realidad aumentada.
- El juego debe tener un comienzo y un fin, con unos objetivos específicos.
- La plataforma sobre la que se ejecuta es Android > 4.
- La plataforma para el desarrollo es Windows 7.
- El juego está desarrollado para una pantalla de 9.7”.
- El dispositivo necesita un procesador arm7 o superior.
- Debe ser lo más ligero posible, tanto para su descarga como reproducción.
- El dispositivo necesita obligatoriamente una cámara.
- Es necesario tener el marcador 0, proporcionado por el SDK vuforia para verlo en funcionamiento.
- El juego debe pausarse en cualquier momento que se deje de enfocar al marcador.
- El lenguaje de programación usado será C#

Detallación técnica

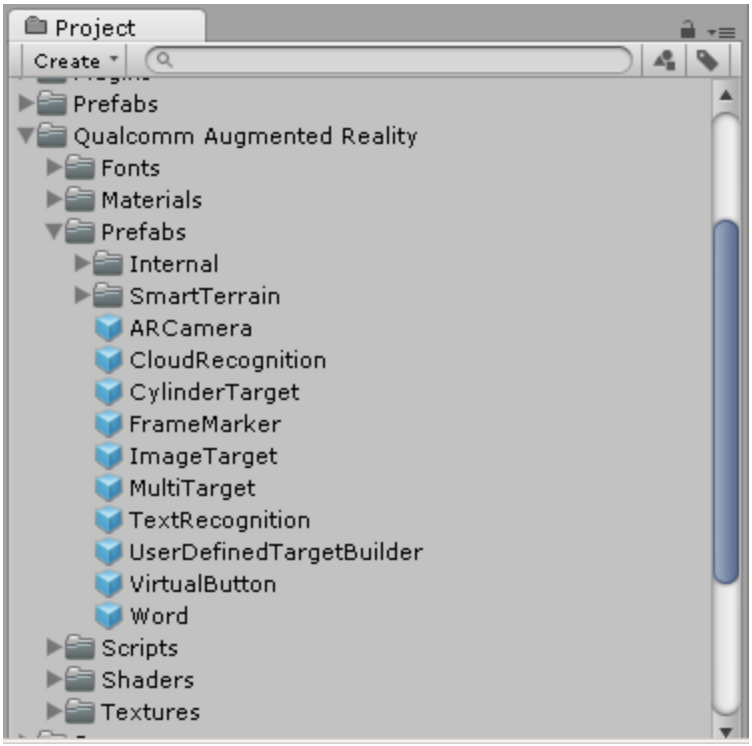
Entorno de programación y SDK's

Para el desarrollo de esta aplicación he precisado el siguiente software:

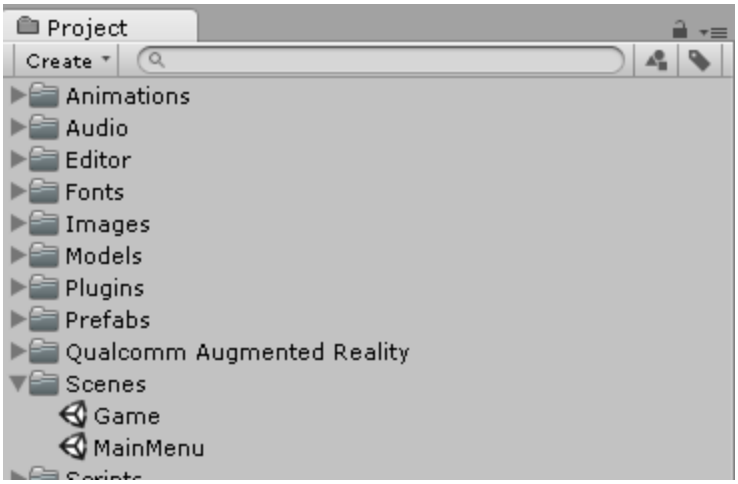
- Unity3D 4.3: plataforma de desarrollo para videojuegos.
- Mono-Develop (incluido en unity3d): programación de scripts en C#
- Android-Vuforia SDK 3.0: soporte para realidad aumentada (integración en Unity3D)
- Android SDK: necesario para la exportación desde Unity3D a android
- Gimp: edición de imágenes.
- Audacity: edición de sonidos.
- Blender: edición 3D.

Configuración del entorno

Si queremos desarrollar proyectos en unity3d de realidad aumentada lo primero que tenemos que hacer es importar en el entorno el SDK de vuforia (assets -> import package -> custom package), una vez hecho esto aparecerá en nuestro proyecto aparecerá la carpeta de Qualcomm con los ficheros necesarios de Vuforia.



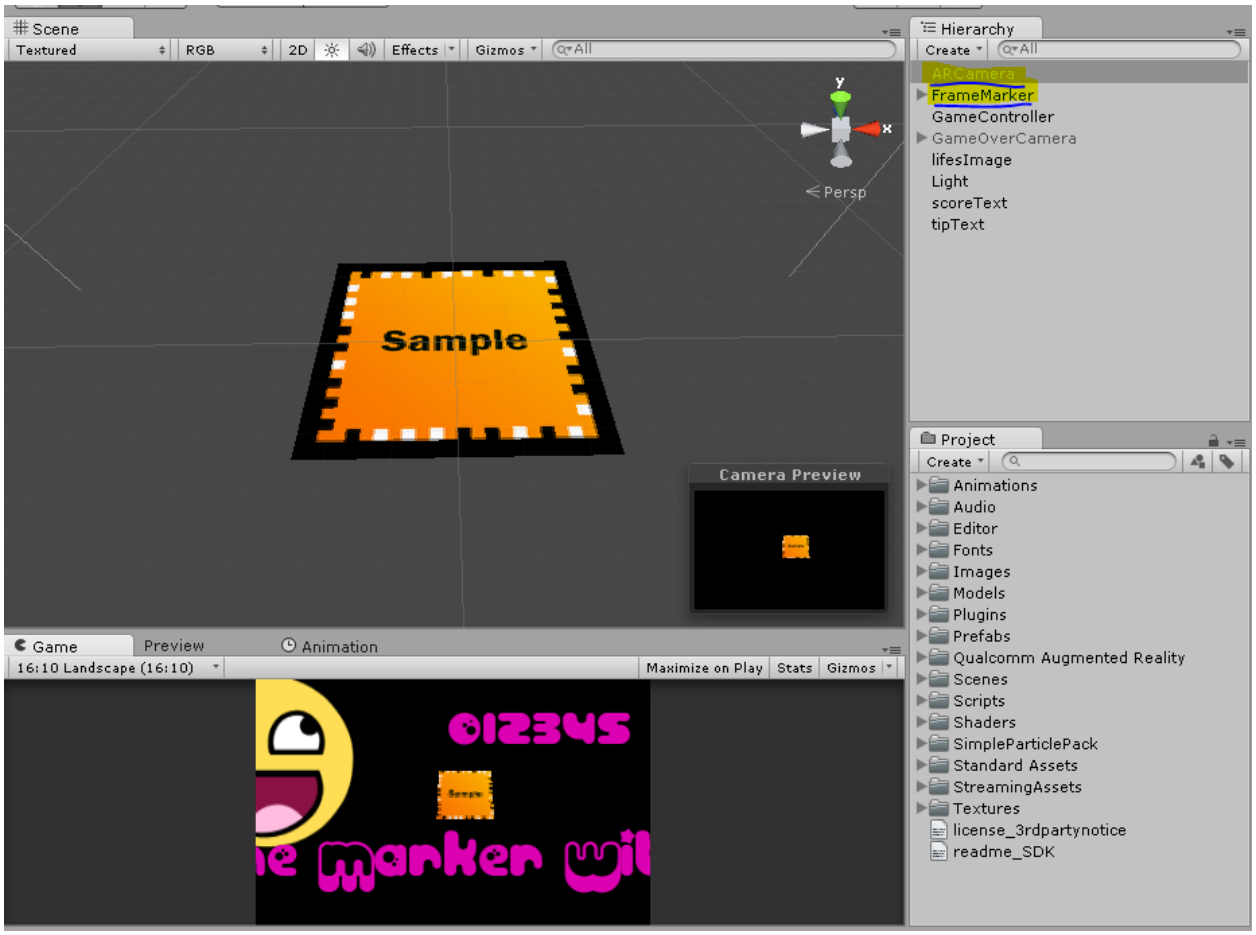
A continuación deberemos crear una escena, que contendrá el juego y otra que contendrá la pantalla de título.



Para terminar la configuración nos dirigimos a File -> Build Settings y marcamos android como plataforma lo que permitirá exportarlo para android.

Primeros pasos

En primer lugar, añadimos un marcador desde la carpeta de Vuforia, esta será la representación de nuestro marcador en unity y apartir de el cual queremos generar el contenido. Tambien necesitamos añadir un nuevo tipo de cámara del SDK Vuforia llamada ARCamera que nos permitirá visualizar el contenido de realidad aumentada con la nuestra del dispositivo móvil.

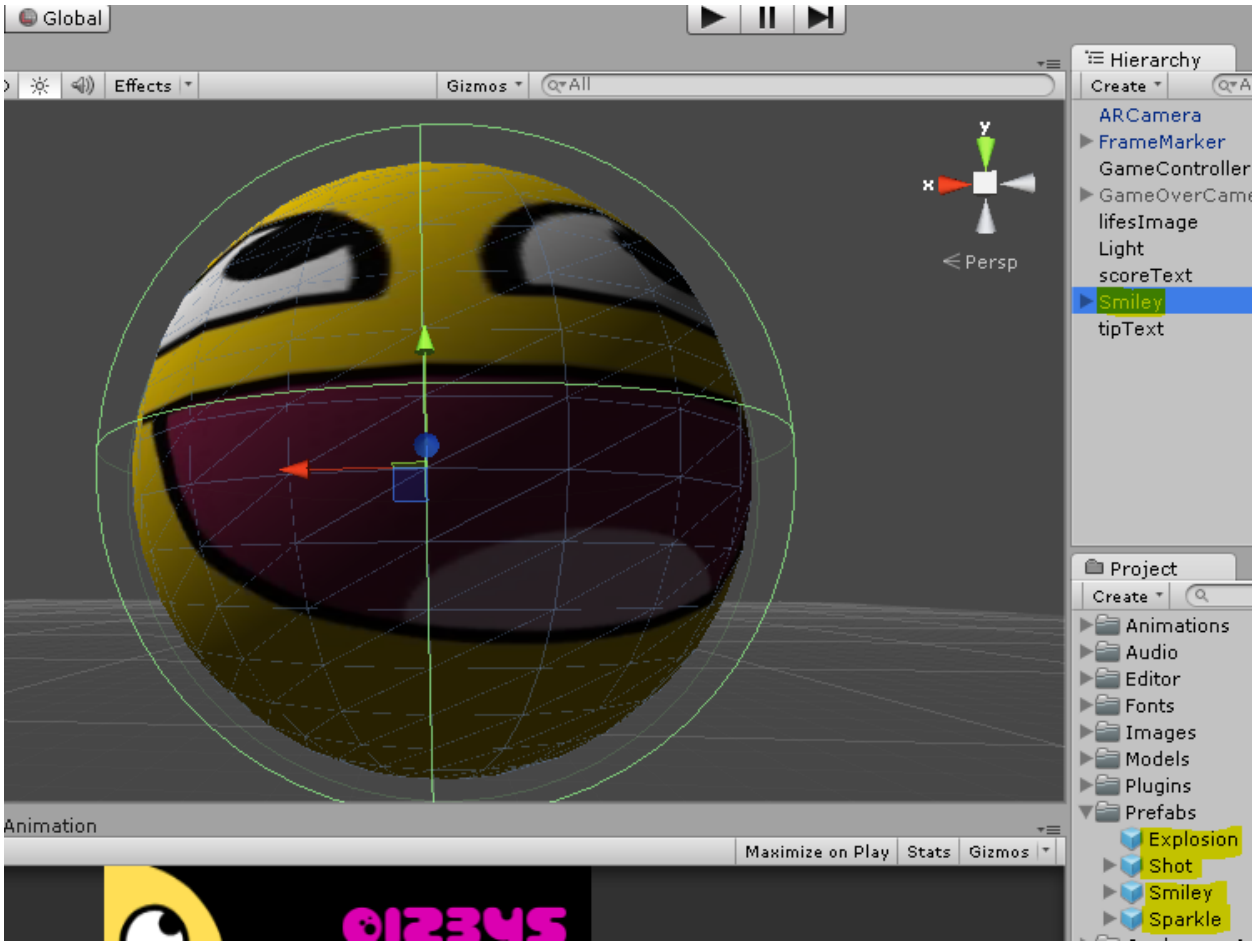


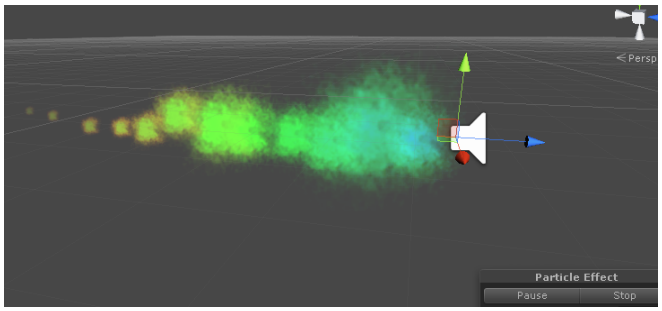
Algunos pasos no están detallados por ser considerados algo trivial como puede ser la importación de fuentes ó texturas, en éste caso la única textura que se usa es la que se le aplica a una esfera a fin de obtener el personaje central del juego.

Añadiendo modelos prefabricados

Los modelos del juego son prefabricados y gratuitos, lo que significa que podemos importarlos en unity una vez los tengamos descargados en nuestro sistema.
En el juego se van a usar 4:

- Explosion para cuando matamos una carita sonrriente
- Disparo para representar la bala que disparamos
- Smiley que es nuestra carita
- Efecto de partículas para mostrar sobre el marcador

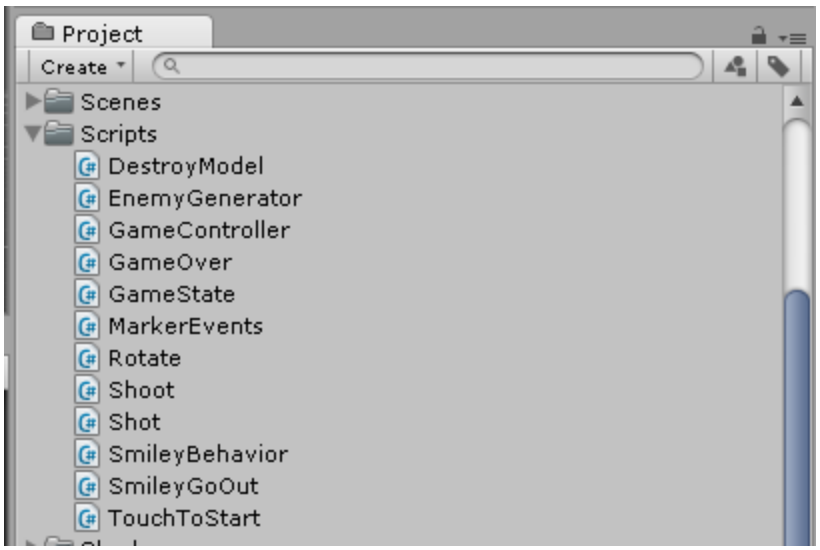




Hora de programar

En modo de funcionamiento de unity3d es simple, tenemos objetos a los que asociamos comportamientos. Estos comportamientos los definimos en scripts los cuales serán añadidos a los modelos prefabricados de tal manera que tengamos un mismo elemento tantas veces como queramos con el mismo comportamiento

Para el desarrollo del juego se han precisado los siguientes scripts (todos ellos en C#)



Vamos a ver la funcionalidad básica de cada uno de ellos y a qué modelo está asociado, se trata de un juego “fácil” y la separación en diversos scripts hace que el comportamiento de cada uno de ellos sea simple y específico.

Asociados a el personaje

Rotate.cs

Se encarga de hacer que el modelo al que está asociado gire en torno a un punto a la velocidad indicada. Irá aumentando el radio poco a poco gracias a la función *transform.RotateAround*.

SmileyGoOut.cs

Su función es eliminar el propio personaje al que está asociado una vez se cumpla un radio establecido definido en la variable *radioToRemove.* Tiene una función llamada *smileyOut()* la cual se encarga de contabilizar una vida menos, reproducir el sonido cuando un personaje se escapa y finalmente destruir el propio personaje.

Asociados a la explosión (muerte de un personaje)

DestroyModel.cs
Éste script es muy simple, define el tiempo de vida de la explosión para eliminarla de memoria una vez el personaje ha muerto.

Asociados a un disparo

Shot.cs
Es el encargado de comprobar las colisiones entre objetos. Si el disparo choca contra el suelo desaparecerá sin más pero si choca con una carita le enviará un mensaje para que ésta se destruya

DestroyModel.cs
El mismo script de la explosión, para que los disparos no duren eternamente.

Asociados a la cámara

Shoot.cs
Es el script encargado de detectar cuando pulsamos la pantalla para generar un disparo gracias al método *Instantiate.* Se ha establecido un delay entre cada disparo para que el jugador no disfrute de “poder-infinito”.

Asociados a el objeto GameController
Este es un objeto normal (sin representación gráfica), al que se le han añadido los comportamientos de la lógica del programa.

EnemyGenerator.cs
Es el encargado de ir generando personajes continuadamente gracias al método *Instantiate*, se le indica mediante un parámetro sobre que objeto van a girar que en nuestro caso es el marcador de realidad aumentada.

MarkerEvents.cs
Gestiona los eventos que se desencadenan cuando la cámara encuentra o pierder el marcador, aquí es donde se sitúa el código para que el juego se pause cuando se pierde y se reanude cuando volvemos a tenerlo en el objetivo gracias a las funciones *FoundMarker() y LosedMarker()*

GameState.cs

Es el encargado de llevar la lógica del juego y su función principal es la de contabilizar las vidas, si hemos perdido o no y de actualizar la puntuación tanto internamente como gráficamente gracias a las funciones *updateScore() y LooseLife()*

GameOver.cs

Es el encargado mostrar la pantalla final, en la que se indica la puntuación y cambiar a la escena de inicio.

Referencias

- http://forum.unity3d.com/
- http://wiki.unity3d.com/index.php/Main_Page
- https://developer.vuforia.com/forum
- http://www.youtube.com/user/juande