**CAPITULO 3**

**ANALISIS Y EVALUACION DE TECNOLOGIAS PARA EL DESARROLLO DEL PROTOTIPO.**

**Falta introducción.**

**Estudio de Tecnologías de Realidad Aumentada:** Se llevó a cabo un análisis y evaluación de las librerías existentes para el desarrollo de aplicaciones de Realidad Aumentada para dispositivos Android. Se identificó las ventajas e inconvenientes entre cada una de las tecnologías involucradas y se optó por aquella que mejor se ajuste al desarrollo del sistema. En el estudio se consideraron cuatro librerías de código abierto:

**Vuforia:** Es un SDK desarrollado por Qualcomm, una empresa productora de chipsets para tecnología móvil. Esta librería fue publicada a finales de 2010 y está escrita en java y C++. Está disponible para Android e iOS y se basa en el reconocimiento de imágenes basado en características especiales, por lo que también soporta marcadores naturales (targets) o Realidad Aumentada sin marcadores. Además dispone de un plugin para interactuar con Unity3D y ofrece la posibilidad de crear botones virtuales para ampliar las vías de interacción con el usuario. [Mamolar, 2012].

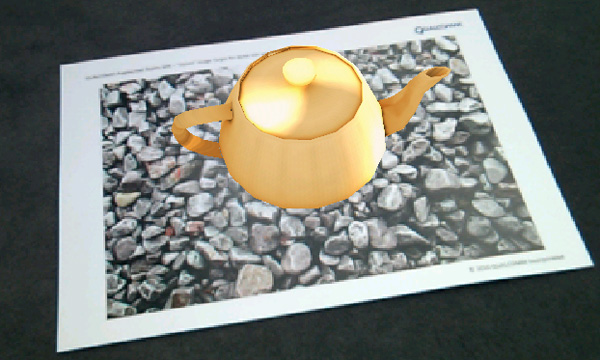


Figura 3.1: Reconocimiento de marcadores naturales o Target. [Qualcomm, 2010]

**Metaio Mobile SDK**: Es una librería escrita en Java, implementada por Metaio y liberada a finales de 2011. Hasta este año todas sus herramientas de desarrollo eran de pago, pero han liberado su SDK para móviles debido al éxito que estaba teniendo. La versión completa, de pago, incluye reconocimiento de caras y de QR (Quick Response Code). Aunque la versión gratuita no incluye estas funcionalidades y que las aplicaciones desarrolladas deben incluir una marca de agua de la empresa, este SDK incluye un potente reconocedor de marcadores naturales, es decir, se pueden desarrollar aplicaciones “markerless”. Por otro lado esta librería añade gravedad a los objetos aumentados, mejorando enormemente la experiencia del usuario. [Mamolar, 2012].



Figura 3.2: Reconocimiento del Patrón QR [Metaio, 2011].

AndAR: Es un proyecto que permite crear aplicaciones de Realidad Aumentada para la plataforma Android. Todo el proyecto se distribuye bajo la licencia GNU,  lo que significa que puede utilizarse libremente. Esta librería fue creada en 2010 por Tobias Domhan y fue escrita en java, también está basado en el proyecto ARtoolKit.

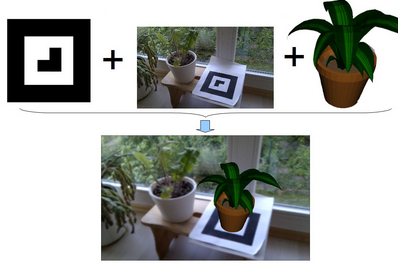


Figura 3.3: Uso de marcadores básicos [AndAR, 2010]

Esta librería funciona con marcadores básicos y maneja una plantilla de texto para gestionar los marcadores. Esta plantilla puede generarse mediante el software mk\_patt de ARToolKit, que sirve para convertir una imagen patrón en una plantilla que pueda ser reconocida por la librería. En la documentación de ARToolKit se puede encontrar la plantilla para crear nuestro propio marcador, puesto que éste debe tener unas proporciones exactas para poder ser reconocido por la librería. [Mamolar, 2012].

**NyARToolkit:** Es una librería también basada en ARToolKit creada por Ryo Lizuka en 2008. En este caso no fue desarrollada específicamente para desarrollo de aplicaciones Android, sino que el proyecto NyARToolkit es compatible con plataformas Java, C#, ActionScript 3, Silverlight 4, C++ y Processing, aunque no todos al mismo nivel en términos de estabilidad. NyARToolkit para Android ha sido desarrollada por un grupo de usuarios de Android en Japón y existe muy poca documentación online, y la que hay está escrita en japonés. Por eso, para entender esta librería hay que sumergirse directamente en el código fuente de Java para Android. Esta librería se encuentra aún en fase de desarrollo, pero no tiene una comunidad grande de desarrolladores detrás, por lo que los avances son muy lentos. En la última versión de NyARToolkit se puede importar formatos como el MD2 y OBJ que son compatibles con Blender. Esta librería utiliza un motor de renderizado externo llamado MIN3D, en lugar de cargar los objetos directamente. [Mamolar, 2012].

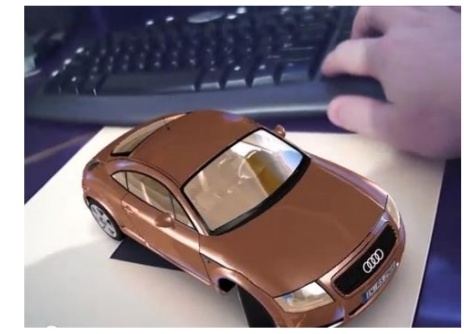


Figura 3.4: Ejemplo NyARToolkit [Diaz, 2013]

En la Figura 3.5 se presenta un resumen de las características de cada una de los SDK:

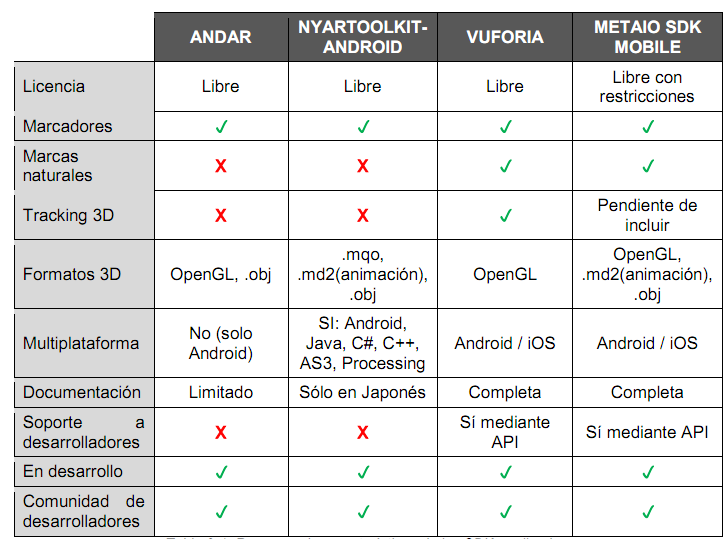


Figura 3.5: Características de las tecnologías de Realidad Aumentada [Mamolar, 2012].

**Análisis de tipos de Marcadores**

Las herramientas ANDAR y NYARTOOLKIT no permite marcadores naturales, por lo que las imágenes como fotografía no pueden ser utilizado con esta tecnología. Ambas librerías utilizan únicamente marcadores QR de color blanco y negro, limando la creatividad en cuanto al uso de los marcadores.

Por un lado Vuforia, dispone de un servicio web, en el que, previo registro, se puede crear los marcadores naturales o targets de manera fácil y sencilla. En el caso de Metaio el proceso es similar, aunque en lugar de ser un servicio on-line se puede descargar la versión demo de la herramienta “Metaio Creator”, una herramienta pensada para generar espacios de Realidad Aumentada rápidamente y sin necesidad de tener conocimientos de programación, todo funciona con una interfaz gráfica en la que se añaden marcadores y objetos 3D que la herramienta se encarga de transformar en una aplicación de Realidad Aumentada. Las aplicaciones generadas contienen una marca de agua de la empresa.

**Reconocimiento del marcador cuando éste es ocultado parcialmente**

Los marcadores QR que utilizan ANDAR y NYARTOOLKIT, si son parcialmente tapados, el sistema no podrá reconocerlo. Sin embargo los SDK que utilizan marcadores naturales si pueden hacerlo. No obstante, Vuforia tiene un mejor comportamiento en este sentido que Metaio. Como curiosidad decir que mientras Metaio y Vuforia sí que implementan la función ScreenShot de Android, andar y NyArtoolkit no lo hacen. Esto es una desventaja, pues la acción ScreenShot añade funcionalidades a las aplicaciones de Realidad Aumentada.

**Reconocimiento del marcador según la distorsión de la perspectiva**

En este caso se analizó el comportamiento de cada SDK frente a distorsiones de la perspectiva. La distorsión de perspectiva se ha representado mediante ángulos de visión de 90º a 15º.

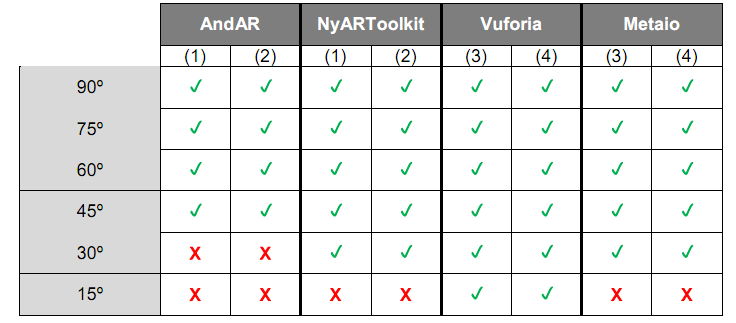


Figura 3.6: Análisis de la distorsión de la perspectiva. [Mamolar, 2012].

Se Observa que el reconocimiento de marcadores naturales es mucho más eficiente cuando se enfrenta a una inclinación pronunciada. Entre los dos SDK el más eficaz es Vuforia, que no deja de reconocer el target a diferentes ángulos de inclinación. [Mamolar, 2012].

**Tasa de frames por segundo en función del número de caras**

En este apartado se analiza el rendimiento de cada SDK en términos de fps (frames por segundo) modificando la escena aumentada. Este estudio está enfocado a la tasa de renderizado en función del número de caras del objeto, recordando que una cara del modelo es cada uno de los triángulos que lo forma. A medida que aumenta la complejidad del objeto a renderizar esta tasa suele bajar. Una de las características básicas de una aplicación de Realidad Aumentada es que sea en tiempo real, para ello la tasa de frames por segundo tiene que estar entre los 15 fps y 30 fps. [Mamolar, 2012].

Los valores recogidos de fps durante las ejecuciones con los diferentes objetos renderizado se muestran en la figura 3.7. En las pruebas, AndAR no ha sido capaz de renderizar el objeto, la aplicación reporta un error y se suspende la ejecución.

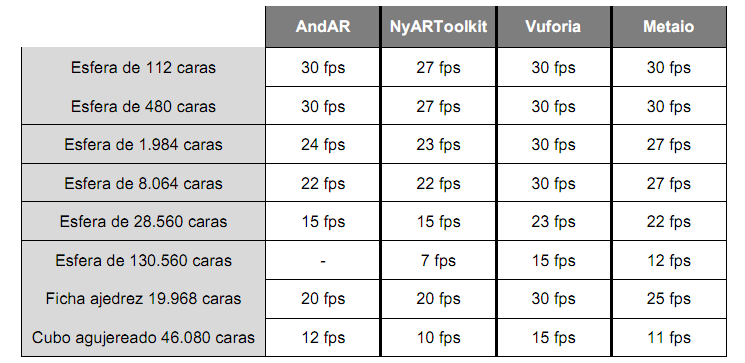


Figura 3.7: Renderizado en términos de frames por segundo según el numero de caras. [Mamolar, 2012].

A la vista de los resultados lo más llamativo es el bajo rendimiento de las librerías NyARToolkit y AndAR frente a las otras dos. Entre ellas NyARToolkit es la que peor resultado ofrece, incluso con el objeto más simple. Ello se debe a que NyARToolkit utiliza un motor de renderizado externo, min3D, lo cual ralentiza esta tarea enormemente. Esto es más llamativo cuando los objetos son más complejos. [Mamolar, 2012].

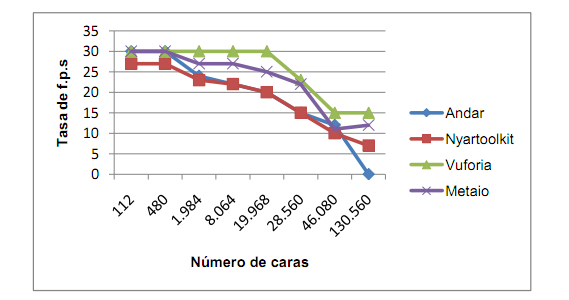


Figura 3.8: Fps en función de números de caras. [Mamolar, 2012].

El SDK que ofrece mejores resultados en todos los casos es Vuforia como lo expresa la figura 3.8. Una explicación es que Vuforia utiliza OpenGL directamente, que sólo contiene funciones de bajo nivel para mostrar y representar gráficos. El hecho de definir el objeto directamente desde primitivas, resulta mucho más eficiente computacionalmente. Además, el módulo de renderizado de Vuforia se encuentra en las funciones nativas en C++, a diferencia de Metaio que está escrito íntegramente en Java, lo que logra mayor eficiencia en cuanto a velocidad de renderizado. [Mamolar, 2012].

Por otro lado, señalar que Qualcomm, la empresa que está detrás de este SDK, es la desarrolladora de la familia de System on Chip (SoC) Snapdragon para la mayoría de dispositivos móviles. Es por ello que el SDK de Qualcomm Vuforia, genera los gráficos de manera óptima para sacar el máximo rendimiento a las prestaciones de Snapdragon. [Mamolar, 2012].

Según lo visto anteriormente, Vuforia ofrece mejores prestaciones que los demás SDK para el desarrollo de aplicaciones interactivas. Metaio al ser el segundo que mejores prestaciones, tiene una licencia libre pero con restricciones, lo que significa limitaciones en cuanto al desarrollo. Vuforia a diferencia de Metaio está completamente liberado. Por otro lado hay que señalar que, para el mismo target y el mismo objeto a renderizar, Metaio se comporta peor en algunas ocasiones produciéndose momentos de jitter y parpadeo. Esto se aprecia especialmente cuando el target se mueve rápidamente. Además hay que señalar que Andar y NyARToolkit utilizan marcadores QR de contraste blanco y negro, limitando la creatividad en cuanto al diseño de marcadores. En cambio Vuforia ofrece marcadores naturales. Se concluye que la mejor herramienta para el desarrollo de aplicaciones móviles con realidad aumentada es la librería Vuforia.

**Elección del Entorno de Desarrollo para el prototipo**

Debido a que Vuforia es la mejor opción para el desarrollo de aplicaciones con realidad aumentada, es necesario un entorno de desarrollo donde se integre ésta librería. Unity 3D y Eclipse ofrecen un ámbito de desarrollo con estas características.

**Unity 3D**

Unity es un motor gráfico 3D para PC y Mac que viene empaquetado como una herramienta para crear aplicaciones interactivas, juegos, visualizaciones y animaciones en 3D en tiempo real. El editor de Unity ofrece un completo editor visual para crear aplicaciones y juegos. El contenido de las aplicaciones es construido desde el editor y se programa usando un lenguaje de scripts. Esto scripts se compiladas usando una versión de JavaScript, C# o Boo, un dialecto de Python. [Collado, 2012].

**Eclipse:**

Se realizo una comparación entre los dos entornos para seleccionar la que mejor ajuste al desarrollo del proyecto:

|  |  |
| --- | --- |
| Unity 3D | Eclipse |
| Dispone de una editor grafico para configurar el modelos 3D. Usa archivos con extensión obj, fbx, etc. | No dispone de un editor, se debe trabajar con el modelo 3D mediante código, haciendo la tarea más compleja. Maneja los archivos xml. |
| Vuforia dispone de un plugin para trabajar con Unity 3D y es de fácil integración. | Se debe realizar una serie de configuraciones y pasos para dejar el entorno de desarrollo listo. |
| Es multilenguaje para crear aplicaciones Android: C#, javascript o BOO. Se puede utilizar más de dos lenguajes simultáneamente. | Para crear aplicaciones Android, utiliza el lenguaje java y código nativo C++. |
| Existe es versión gratuita. | Es libre |
| Es un motor dedica para crear aplicaciones interactivas y juegos. | Es un entorno de desarrollo, pero no está dedica o especializado para aplicaciones interactivas. El desarrollo es mas difícil. |

Tabla 4.1: Comparación entre entornos de desarrollo

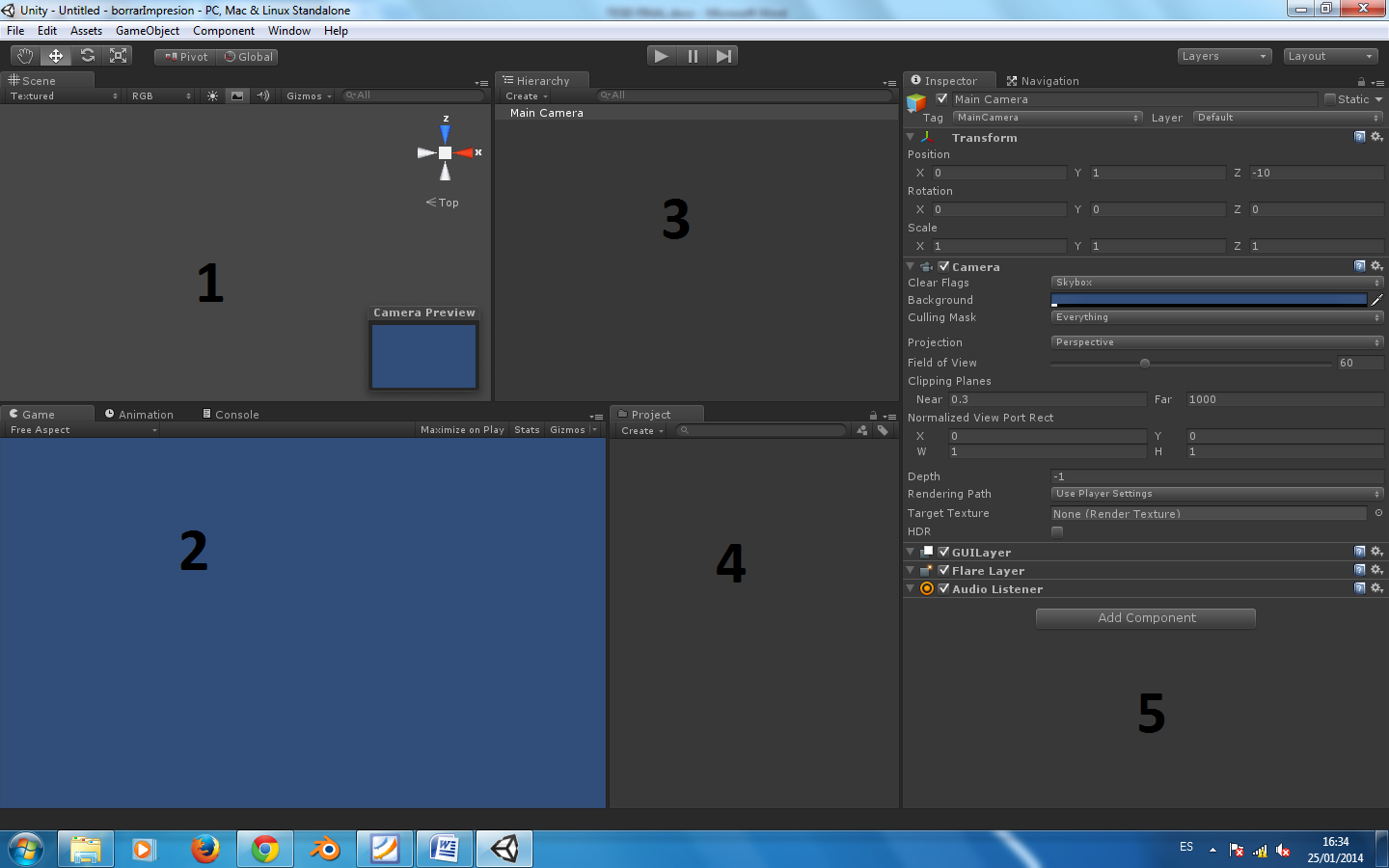
El análisis de las características de ambas herramientas, inclina la balanza a favor de Unity 3D. Como [Unity3D](http://unity3d.com/) es una plataforma de contenidos 3D interactivos y [Vuforia](https://www.vuforia.com/) es el sistema de desarrollo (SDK) de realidad aumentada de Qualcomm, juntos forman un binomio muy potente que permite crear desde simples visualizadores de realidad aumentada hasta complejos videojuegos.

El editor de Unity ofrece un completo editor visual para crear aplicaciones y juegos. El contenido de las aplicaciones es construido desde el editor y se programa usando un lenguaje de scripts. Esto scripts se compiladas usando una versión de JavaScript, C# o Boo, un dialecto de Python. [Collado, 2012].

Las aplicaciones creados en Unity son estructurados en escenas, donde cada escena puede ser cualquier parte de la aplicación. En un juego, una escena puede ser el el menú de inicio, la configuración del juego o un nivel del juego.

Entorno de Desarrollo de Unity

La interfaz de unity 3D está dividida en 5 áreas principales, numeradas en la imagen siguiente.



1. Vista de Escena

Es el área de visualización de Unity donde construimos visualmente cada escena de nuestro aplicación. La vista de escena es un entorno 3D para crear cada escena desde el inicio. Trabajar con la vista de escena, en la forma más sencilla, sería arrastrar un objeto desde la vista de proyecto a la vista de escena, entonces podrás posicionarlo, escalarlo, rotarlo y agregarle textura.

Esta vista cuenta con una herramienta de modo de visualización. [Collado, 2012].



Por defecto esta herramienta tiene un perspectiva 3D de la escena. Podemos cambiar esto por un numero de vistas Ortográficas: top down, side y front.

Vista de Aplicación o de juego.

En la vista de aplicación obtendremos una previsualización de nuestro aplicación. En cualquier momento podemos reproducir nuestro aplicación y probarlo en esta vista.

Cuenta con los siguientes botones:



Puedes entrar en la previsualización del juego en cualquier momento pulsando el botón de reproducción, pausar usando el botón de pausa o saltar adelante usando el botón derecho.

1. Vista de Jerarquía

La vista de jerarquía contiene todos los objetos en la escena actual.

Cualquier objeto que coloques en la escena aparecerá como una entrada en la jerarquía. Cuando un objeto es seleccionado en la jerarquía también lo es en la vista de escena, donde puedes moverlo, escalarlo, rotarlo, borrarlo o editarlo. El inspector también mostrara las propiedades del objeto seleccionado; de esta forma la jerarquía sirve como una herramienta útil para seleccionar rápidamente objetos y editar sus propiedades. [Collado, 2012].

1. Vista de Proyecto

Todos los componentes de aplicaciones que creamos desde el editor y todos los objetos que importes como modelos 3D, texturas, efectos de sonido, música etc. se guardaran en esta vista. Como este panel contiene todos los assets de una aplicación, es importante mantener una buena estructura de directorios.

Cuenta con un botón “Create” que mostrará una lista desplegable con varias opciones de creación. Podremos crear carpetas, scripts, shaders, animaciones y otros tipos de objetos usando este panel. [Collado, 2012].

1. Vista de inspector

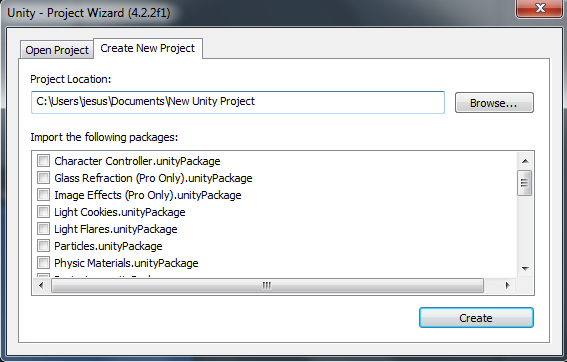
Cuando se selecciona un objetos en la vista de jerarquía, se mostrara las

propiedades de ese objeto donde puedes personalizar varias características del mismo. Por ejemplo, si seleccionas una luz o cámara, el inspector te permitirá editar varias propiedades de la luz o de la cámara.

El panel también tiene un menú contextual en la esquina izquierda que aparece como “Free Aspect” por defecto, de esta lista podremos seleccionar un número de proporciones para nuestra aplicación, lo que es ideal para probar en distintas pantallas y plataformas. [Collado, 2012].

Crear un proyecto nuevo

En Windows unity habrá creado por defecto la carpeta “Unity Project” en la carpeta de mis documentos. Para crear un nuevo proyecto vamos a File -> New Project, esto hará que se muestre el cuadro de dialogo “Create New Project” como en la imagen a continuación.



Lo siguiente es seleccionar los paquetes que quieres incluir en el

proyecto. Los paquetes son conjuntos de assets que se puede importar en cualquier momento desde unity si quieres usar paquetes adicionales que has descargado o no has seleccionado en este punto. Para finalizar se preciona el botón “Create” que reiniciara a unity, creara la estructura del proyecto en la carpeta especificada e importara los paquetes seleccionados al proyecto. Una vez termine se mostrara una escena en blanco con una cámara. [Collado, 2012].

Crear primera escena

Unity ha creado nuestra primera escena automáticamente al crear el proyecto. Para crear una escena nueva, nos vamos a “File -> New Scene”.

Esta nueva escena es un espacio sin título, por lo que para darle un nombre, nos vamos a “File -> Save Scene” y guarda la escena en algún lugar en la carpeta de asset para este proyecto. La escena aparecerá en la vista de proyecto una vez realizado este paso. [Hernández, 2011].

Script en unity

Las secuencias de comando en Unity se logran a traves de la ejecución de scripts simples hechos en JavaScript, Boo (variación de phyton) o C #. Se pueden utilizar uno o todos los lenguajes de script en un solo proyecto, no hay restricción para el uso de más de uno. Para crear un nuevo script, realizamos el siguiente: Assets-> Create-> C# Script en el menú principal. Esto creará un nuevo script llamado NewBehaviourScript ubicado en la carpeta seleccionada por el usuario. Si no hay ninguna carpeta seleccionada, el script se creará la carpeta raíz.

Se puede editar el script haciendo doble clic sobre este desde el visor de proyecto. Esto abrirá Unitron, el editor de scripts de Unity. Toda laprogramación se hace en un editor de texto externo como Unitron, y no en Unity directamente. Para configurar el editor de scripts por defecto, vaya a Unity-> Preferences-> External Script Editor. [Hernández, 2011].

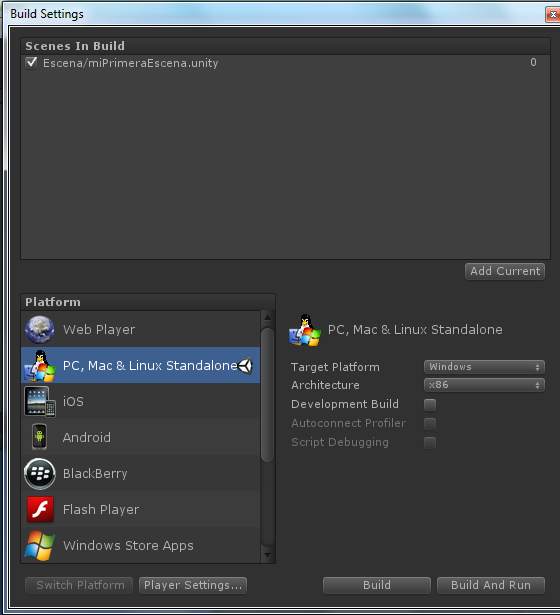
Construyendo la Aplicación

Para crear nuestra aplicacion y poder distribuirlo tendremos que crear una distribución que funcione fuera del editor de unity. Este proceso se conoce como “building” del proyecto. Crea un ejecutable de la aplicación y

organiza los assets en una estructura de carpetas (sin paquetes) de las que

un ejecutable puede cargar el contenido, ademas de crear otros ficheros que usa nuestra aplicacion. Esta build es esencialmente como se estructurará nuestra aplicacion el el disco duro del usuario cuando se lo

instale.Para poder hacer la build de nuestra aplicación primero necesitamos elegir las propiedades de esa build. Para hacer esto debes ir a “File -> Build Settings”. [Hernández, 2011]. Con esto cargaremos una ventana como la de las capturas bajo estas lineas.



.

En la parte superior de la ventana veremos la lista de escenas en nuestro juego, cualquier escena que quieras que sea publicada debe ser marcada. La primera escena en la lista será la escena de entrada, será la

que se muestre cuando la aplicación se cargue por lo que normalmente será un menú o una pantalla de inicio.

Debajo de la lista de escenas veremos varias plataformas para nuestra build, en este trabajo final se utilizara como plataforma Android. Para construir nuestra distribución, precionamos “Build”, esto generará un apk para ser instalado en cualquier sistema operativo android. [Collado, 2012].

* Gimp: significa Image Manipulation Program, y es un software de distribución gratuito para trabajos de retoque fotofráfico, creación y composición de imágenes. “No dispone de todas las ventajas que posee Photohop”, aunque siempre se están añadiendo nuevas funcionalidades que facilitan las tareas de diseño. Aún así, no se debe considerar como un reemplazo de Photoshop ya que depende en parte de las tareas requeridas.
* Photoshop: viene con muchas características de edición de gráficos, y es bien recibida por los aficionados y profesionales de edición de fotos. Podemos decir que Photoshop es el más potente software de edición de fotos en el mundo hasta hoy. Por lo general, la gente sólo tendrá que utilizar una pequeña parte de las funciones de edición para lograr grandes efectos fotográficos. Photoshop es una aplicación que dispone de calidad comercial, y como tal debe dar mejor rendimiento por el dinero que pagas

El siguente cuadro muestra las características principales de los editores de imágenes:

|  |  |
| --- | --- |
| GIMP | Photoshop |
| Es un software gratuito | Es pago. Pero existe su versión trial. |
| Bajos requerimientos de almacenamiento | Los requisitos de almacenamiento pueden ser excesivo. |
| Está disponible en Windows, Mac y Linux | Sólo está disponible en Windows y Mac |
| No tiene un equipo de apoyo central. Pero puede realizar sus consultas en foros. | Apoyo de Adobe es caro, pero tiene una mayor disponibilidad. |
| Es de código abierto. Eso significa que usted es libre de ver su código y hacer algunas modificaciones si lo desea. | No es de código abierto |
| GIMP 2.6 es compatible con color de 8 bits | Photoshop CS4 soporta hasta 32 bits. |
| Tiene las funciones básicas. Se pueden encontrar multitud de plugins en la web para cubrir esta carencia. | Tiene mayor conjunto de funciones |
| Su interfaz puede resultar difícil para novatos debido a las potentes funciones para la edición de fotos | Tiene una interfaz mucho más bonita y se ve más profesional. |

Conclusión

Elegimos Photoshop ya que es una herramienta más completa que Gimp y cuenta con una versión trial, la cual es suficiente para nuestro proyecto.

<http://reinspirit.com/gimp-o-photoshop/>

<http://www.wondershare.es/photo-editing-tips/gimp-vs-photoshop.html>

consultada: 16- ene-2014

Blender: es un programa informático multiplataforma, dedicado especialmente al modelado, animación y creación de gráficos tridimensionales.

El programa es un software libre que actualmente es compatible con todas las versiones de Windows, Mac OS X, Linux, Solaris, FreeBSD e IRIX.

La instalación de este programa es necesaria para que Unity reconozca los archivos 3D y sus texturas

http://www.blender.org consultada: 16- ene-2014