Empezando en Unity3D

Unity Spain

Contenido

En este tutorial te introducirás en el motor Unity 3D. Esta parte del tutorial contiene una introducción a que es Unity, para que se puede utilizar, que no es, una introducción básica a los conceptos de Unity y una muestra de la interfaz y sus características básicas.

Este tutorial asume que eres nuevo en Unity, acabas de descargar Unity o has comprado Unity pro; puede que hayas probado un poco el motor o hayas leído tutoriales pero casi todos están por encima de tu nivel.

Si crees que encajas en este perfil entonces este debe ser tu punto de inicio antes de leer otros tutoriales o pasar a algo más avanzado. Todo el mundo odia las guías para principiantes que hablan acerca de como funcionan las cosas y donde se encuentran, todos queremos empezar a crear algo, por lo que haré esta parte lo más breve posible para luego poder pasar a aprender sobre proyectos o material real.

Este apartado se subdivide en las siguientes secciones:

- Introduciendo Unity3D.

- La interfaz de usuario de Unity.

INTRODUCIENDO UNITY3D

Unity es un motor gráfico 3D para PC y Mac que viene empaquetado como una herramienta para crear juegos, aplicaciones interactivas, visualizaciones y animaciones en 3D y tiempo real. Unity puede publicar contenido para múltiples plataformas como PC, Mac, Nintendo Wii y iPhone. El motor también puede publicar juegos basados en web usando el plugin Unity web player. Como motor gráfico, este es posiblemente el mejor motor por debajo de los 100.000€.

El editor de Unity es el centro de la linea de producción, ofreciendo un completo editor visual para crear juegos. El contenido del juego es construido desde el editor y el gameplay se programa usando un lenguaje de scripts. Esto significa que los desarrolladores no necesitan ser unos expertos en C++ para crear juegos con Unity, ya que las mecánicas de juego son compiladas usando una versión de JavaScript, C# o Boo, un dialecto de Python.

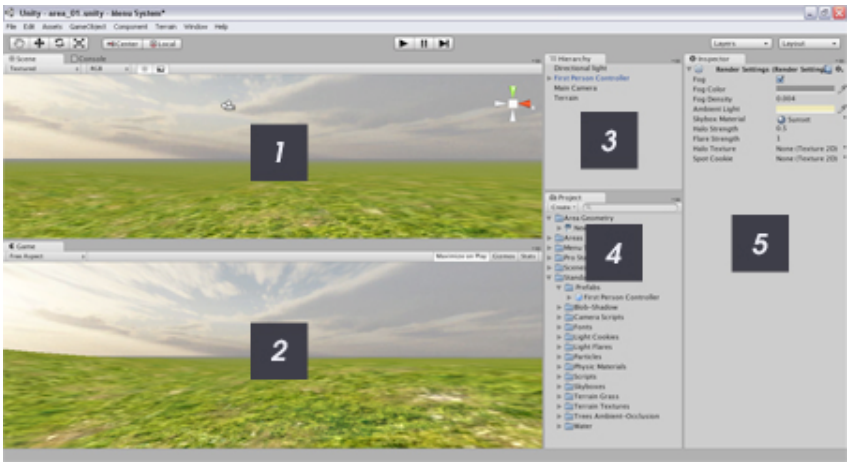
Los juegos creados en Unity son estructurados en escenas como el motor Gamebryo. En Unity una escena puede ser cualquier parte del juego, desde el menú de inicio como un nivel o área de tu juego; la elección es tuya ya que una escena es un lienzo en blanco sobre el que dibujar cada parte del juego usando las herramientas de Unity.

El motor también incluye un editor de terrenos, desde donde puedes crear un terreno (como una hoja en blanco), sobre la que los artistas podrán esculpir la geometría del terreno usando herramientas visuales,pintar o texturizar, cubrir de hierba o colocar arboles y otros elementos de terreno importados desde aplicaciones 3D como 3DS Max o Maya.

Con esto podemos dar por concluida la introducción a Unity. Pasamos al siguiente apartado en el que trataremos la interfaz de Unity.

LA INTERFAZ DE USUARIO DE UNITY

En esta parte de nuestra introducción a Unity veremos la interfaz. Existen principalmente 5 áreas de la interfaz de Unity, numeradas en la imagen de abajo.



Vista de Escena

La escena es el área de construcción de Unity donde construimos visualmente cada escena de nuestro juego.

Vista de Juego

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

Vista de Proyecto

Esta es la librería de assets para nuestro juego, similar a la librería en Flash. Puedes importar objetos 3D de distintas aplicaciones a la librería, puedes importar texturas y crear otros objetos como Scrips o Prefabs que se almacenaran aquí. Todos los assets que importes en tu juego se almacenaran aquí para que puedas usarlos en tu juego.

Ya que un juego normal contendrá varias escenas y una gran cantidad de assets es una buena idea estructurar la librería en diferentes carpetas que harán que nuestros assets se encuentren organizados y sea más fácil trabajar con ellos.

Vista de Jerarquía

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

Vista de inspector

La vista de inspector sirve para varias cosas. Si seleccionas objetos entonces mostrara las propiedades de ese objeto donde puedes personalizar varias características del objeto. También contiene la configuración para ciertas herramientas como la herramienta de terrenos si tenemos el terreno seleccionado.

Los scripts son escritos usando Unitron en Mac y Uniscite en la versión para PC. Cuando creamos un script, podemos saltar al editor de scripts desde Unity, guardar el script y usarlo en el juego.

Al igual que muchas aplicaciones podemos modificar la interfaz por defecto de Unity. Prueba a hacer clic sobre la pestaña de la vista de juego y arrastrarlo al lado de la pestaña de la vista de escena, veras que aparece un indicador de posición. Suelta el botón del ratón y la vista de juego no estará en la misma posición que la vista de escena, estará a un lado. Ahora podremos cambiar entre la vista de escena y la de juego al hacer click sobre la pestaña apropiada. Esto te dará mucho mas espacio para la vista de escena, lo que resulta muy útil.

Cuando trabajemos con la vista de escena es útil también saber que pulsar el espacio agrandara la vista de escena hasta completar el editor por completo.

En las siguientes páginas vamos a ver de forma mas detallada cada uno de estos componentes para entender exactamente para que son estos componentes.

Bien, hemos hablando brevemente de que es unity y ahora vamos a ver con algo más de profundidad la interfaz del editor. En esta parte del tutorial vamos a introducirnos más en la vista de escena.

¿Para que es la vista de escena?

Los juegos creados en Unity están divididos en "escenas" al igual que muchos motores AAA como Gamebryo y el Unreal Engine usan partes.

La vista de escena es un entorno 3D para crear cada escena. Trabajar con la vista de escena, en la forma mas sencilla, sería arrastrar un objeto desde la vista de proyecto a la vista de escena que colocara el objeto en la escena; entonces podrás posicionarlo, escalarlo y rotarlo sin salir de al vista de escena.

La vista de escena es también el lugar donde editas los terrenos (esculpiendolos, pintando texturas y colocando elementos), colocas luces y cámaras y otros objetos.

Arrancando Unity

Lo primero que debemos hacer es arrancar Unity 3D. Cuando hagas esto se cargara la demo que se incluye con Unity o un proyecto en blanco. Para este tutorial trabajaremos con la demo incluida con Unity, llamada Island Demo, por lo que si tienes un proyecto vacío deberás ir al menú "File -> Open Scene" y seleccionar "Islands" del directorio de proyecto ("Island Demo/Assets").

Modos de visualización

Por defecto la vista de escena tiene un perspectiva 3D de la escena. Podemos cambiar esto por un numero de vistas Ortográficas: top down, side y front. En la parte derecha de la vista de escena veréis un "Gizmo" que parece una caja con conos que salen de ella.

Podemos usar este Gizmo para cambiar la perspectiva:
· Hacer clic sobre la caja nos llevara al modo perspectiva.
· Hacer clic en el cono verde (y) nos llevara al modo Top-Down.

· Hacer clic sobre el cono rojo (x) nos llevara al modo Side (derecha).
· Hacer clic sobre el icono azul (z) nos llegara al modo Front(frontal).
· También tenemos 3 conos grises que nos llevaran a los siguientes modos: "Back, Left y Bottom", es castellano, Atrás, Izquierda y Abajo.

Configuración de la visualización

En la esquina izquierda de la vista de escena encontraremos un conjunto de botones para cambiar la configuración general de la visualización. Vamos a ver cada uno de ellos, de izquierda a derecha.

Render Mode

La primera opción es Render mode o modo de renderización en castellano. Por defecto aparecerá en "Textured". Si hacemos clic aparecerá una lista desplegable con un numero de diferentes opciones de renderizado:

· Textured: Las texturas se renderizan en la vista.
· Wireframe: Las superficies no se renderizan, solo vemos la malla.
· Textured Wireframe: Las texturas se renderizan, pero también vemos la malla.

Prueba a cambiar entre las tres opciones para entender como se ven realmente.

Color Modes

La segunda opción es el modo de color, que aparece como "RGB" por defecto. Si haces clic sobre este boton aparecerá una lista desplegable que mostrara los modos de color disponibles:

· RGB: Todos los colores son renderizados.
· Alpha: El modo es cambiado a "Alpha".
· Overdraw: El modo es cambiado a "Overdraw".
· Minimaps: El modo es cambiado a "Minimap".

Prueba a cambiar entre las opciones para tener claro como quedan.

Interruptor de luces

El siguiente botón enciende o apaga la iluminación del escenario. Apagar la iluminación resultara en una escena mostrada sin luces; lo que puede ser útil para el rendimiento y también si no hay luces en la escena.

Encender la luz provocara que las luces tengan efecto sobre la escena. Si no tienes luces en la escena esta será oscura, ya que no hay luz.

Interruptor de skybox, lense flare y niebla

El último botón activa y desactiva estos tres efectos. Esta opción es útil para desactivar los efectos por razones de rendimiento o visibilidad al trabajar sobre una escena.

Botones de Control

Debajo de las opciones de visualización veras una fila con 4 botones, como en la imagen de abajo.

Puedes usar Q, W, E, R para alternar entre cada uno de los controles, que detallamos debajo:

· *Hand Tool* (Q): Este control nos permite movernos alrededor en la vista de escena. Mantener ALT nos permitirá rotar, COMMAND/CTRL nos permitirá hacer zoom y SHIFT incrementa la velocidad de movimiento mientras usas la herramienta.
· *Translate Tool* (W): Nos permite mover cualquier objeto seleccionado en la escena en los ejes X, Y y Z.
· *Rotate Tool* (E): Nos permite rotar cualquier objeto seleccionado en la escena.
· *Scale Tool* (R): Nos permite escalar cualquier objeto seleccionado en la escena.

Cuando usemos las herramientas de traslación, rotación o escalad veremos un gizmo alrededor del objeto seleccionado con lineas para cada uno de los ejes. Podemos usar este gizmo para realizar las operaciones de translación, rotación y escalado.

Con esto podemos concluir la parte de la vista de escena, pasemos a la vista de proyecto.

¿Para que es la Vista de Proyecto?

La vista de proyecto es esencialmente una librería de assets para el proyecto de nuestro juego. Sus funciones son muy similares a por ejemplo la librería de Adobe Flash. Todos los componentes del juego que crees desde el editor y todos los objetos que importes como modelos 3D, texturas, efectos de sonido, música etc. se guardaran ahí.

Como este panel contiene todos los assets de tu juego, y no solo los que están en la escena actual, es importante mantener una buena estructura.

Podemos crear carpetas y colocar los objetos dentro de esas carpetas para crear un jerarquía de carpetas. Puesto que un proyecto de juego competo contendrá muchos assets; en algunos juegos una cantidad muy grande de assets, es una buena idea el decidir y crear una estructura que sea fácil de usar por el equipo con el que trabajaras antes de empezar a producir tu juego.

Ya que la vista de proyecto es muy sencilla y funciona como cualquier gestor de ficheros no será necesario mas que explicar algunas notas de uso que podéis encontrar debajo.

Notas de Uso

· No crees la estructura de tu librería o muevas assets usando Windows Explorer o Finder ya que Unity puede perder enlaces y dependencias importantes. Como practica general debemos usar las herramientas de organización de la vista de proyecto para crear la estructura y organizar los assets puesto que Unity seguirá su localización.

· En la parte superior de la vista de proyecto veras un botón “Create”, que mostrara una lista desplegable con varias opciones de creación. Podremos crear carpetas, scripts, shaders, animaciones y otros tipos de objetos usando este panel.

· Hacer dos clic sobre un ítem en la librería nos permitirá renombrarlo.

· Puedes hacer clic y arrastre de carpetas y ítems para organizar la estructura.

· Puedes importar y exportar paquetes (colecciones de assets) simplemente haciendo clic derecho sobre la vista de proyecto y seleccionando la opción apropiada.

· Puedes importar assets como archivos de audio, texturas modelos etc. con hacer clic derecho y seleccionar “Import Asset”

· Puedes buscar por la librería usando la casilla de búsqueda al lado del botón “Create”.

Bueno, con esto queda explicada la vista de proyecto; pasemos a la Hierarchy view o vista de jerarquía en castellano.

¿Para que es la Vista de Jerarquía?

La vista de jerarquía contiene una lista de todos los objetos usados en la escena actual. Cualquier objeto que coloques en la escena aparecerá como una entrada en la jerarquía.

La jerarquía también sirve como método rápido y fácil para seleccionar objetos en la escena. Si quieres por ejemplo, seleccionar un objeto de la escena puedes seleccionarlo desde la jerarquía en lugar de moverte por la escena, encontrarlo y seleccionarlo.

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.

La jerarquía también sirve para para otros propósitos, pero de estos ya hablaremos en otra ocasión ya que son conocimientos más avanzados.

Con esto concluimos la explicación de la vista de jerarquía, sigamos por el inspector, otra de las partes fundamentales que componen la interfaz de unity.

¿Para que es el Inspector?

El inspector es esencialmente un panel de propiedades; si seleccionas un objeto en la escena todas las propiedades editables aparecerán en el inspector.

Por ejemplo, si seleccionas una luz o cámara, el inspector te permitirá editar varias propiedades de la luz o de la cámara. Adicionalmente el inspector sirve como panel de herramientas para ciertos tipos de objetos. Por ejemplo, si seleccionas un terreno el inspector mostrara las opciones de terreno y también el editor con herramientas como esculpir, texturizar, etz.

Como el inspector cambia según el objeto que seleccionemos la mejor forma de aprenderlo es probarlo.

Terminamos con el inspector, por lo que ya solo nos queda la vista de juego, la última, pero no por ello la menos importante.

¿Para que es la Vista de Juego?

En Unity puedes ejecutar tu juego sin salir del editor, lo que es una bendición para los diseñadores que están construyendo niveles y los desarrolladores que están añadiendo nuevas mecánicas de juego.

La imagen sobre estas líneas muestra los controles de reproducción, que están localizados en la parte superior del editor. Puedes entrar en la previsualización del juego en cualquier momento pulsando el botón de reproducción(el primero por la izquierda), pausar usando el botón de pausa (central) o saltar adelante usando el botón derecho.

Puedes jugar desde la vista de juego o extenderla a pantalla completa.

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 el juego, lo que es ideal para probar nuestro juego en distintas pantallas y plataformas.

Con esto concluimos esta parte del tutorial, dedicada a la explicación de la interfaz de unity.

PRIMEROS PASOS

Seguimos con nuestro proceso de aprendizaje de unity. Ahora pasamos a algo más interesante, al trabajo real.

En esta parte llamada primeros pasos estudiaremos lo siguiente:

· Crear un proyecto nuevo

· Crear nuestra primera escena

· Añadir un cielo a la escena

· Crear un terreno básico

· Añadir luces

· Colocar un controlador en primera persona

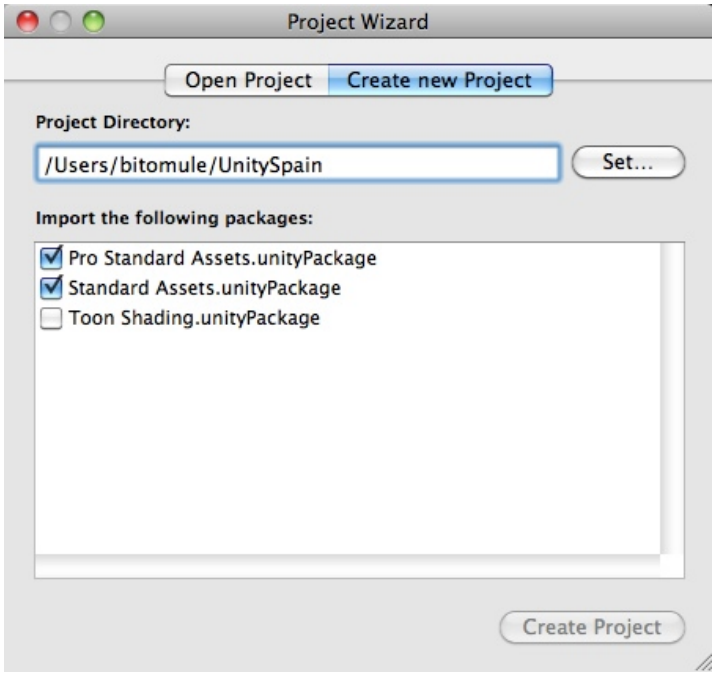
· Jugar con las diferentes configuraciones

· Crear una versión ejecutable para Windows o Mac.

Creando un proyecto

Empecemos por crear un nuevo proyecto. En Windows unity habrá creado por defecto la carpeta “Unity Project” en la carpeta de mis documentos. Puedes guardar ahí los tutoriales si quieres, pero te recomiendo que crees una carpeta llamada UnitySpain y ahí guardes el material con el que trabajemos.

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 primero que debemos hacer es elegir la localización de nuestro proyecto, para eso debéis hacer clic sobre Set.. en MacOS o Browse... en Windows, seleccionar la carpeta donde queréis crear vuestros proyectos y crear la carpeta “PrimerosPasos”.

Lo siguiente que tienes que hacer es seleccionar los paquetes que quieres incluir en el proyecto. Los paquetes son conjuntos de assets, puedes importar paquetes en cualquier momento desde unity si quieres usar paquetes adicionales que has descargado o no has seleccionado en este punto.

Como en la imagen superior selecciona los paquetes solo los paquetes estándar y luego haz clic en “Create” en Windows y “Create Project” en MacOS.

Cuando hagas esto Unity se reiniciara, creara la estructura del proyecto en la carpeta especificada y importara los paquetes seleccionados al proyecto. Una vez termine se mostrara una escena en blanco con una cámara.

Como veras en la vista de escena, tenemos una escena en blanco ya que no hay nada en ella, pero Unity nos ayuda y ha incluido una cámara inicial por nosotros, que podemos ver en la vista de escena y en la jerarquía.

También veras los assets estándar en la vista de proyecto, listos para ser usados.

La primera escena

Unity ha creado nuestra primera escena automáticamente al crear el proyecto. Si quieres crear una escena nueva, sencillamente ve a "File -> New Scene".

Esta nueva escena es un espacio sin titulo, realmente necesitamos guardarlo. Para ello 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 hagas esto. Puedes llamar la escena "escena_1".

Con el proyecto creado y nuestra primera escena delante, estamos listos para empezar a construir nuestro primer nivel.

Añadiendo un terreno

Ya tenemos nuestro proyecto y la escena en blanco, ahora vamos a añadir un terreno para empezar a crear nuestro nivel. Mientras creamos el terreno veremos algunas características y opciones básicas de los terrenos.

Unity maneja los terrenos de la misma forma que muchos otros motores, como una malla plana que podemos texturizar y esculpir sin salir del editor.

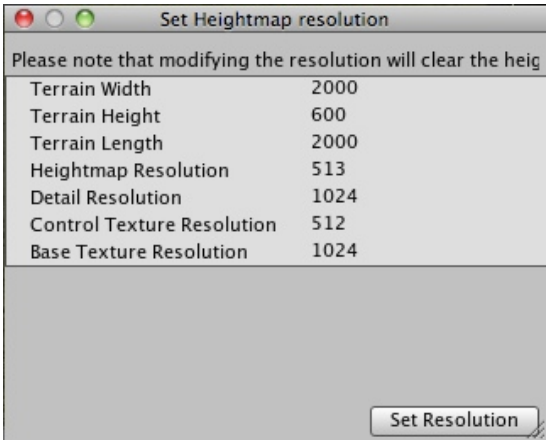
Para insertar un nuevo terreno vamos a "Terrain -> Create Terrain" desde el menú principal.

Lo que tenemos ahora no es particularmente bonito. Si no puedes ver el terreno, desactiva las luces en la vista de escena. También podrás apreciar que el terreno aparece en la jerarquía y un asset se ha añadido a la librería en la vista de proyecto.

Configuración básica de un terreno

Ahora que tenemos nuestro terreno en la escena, debemos definir algunas propiedades importantes, como la longitud del terreno y algunas propiedades que controlar como de detallado debe ser el terreno.

Para modificar estas propiedades iniciales debemos ir a la ventana "Set Heightmap Resolution", a la que podemos acceder desde "Terrain -> Set Resolution".



Expliquemos que es cada una de las propiedades de la ventana "Set Heightmap Resolution" :

· Width: El ancho en metros de nuestro terreno.

· Lenght: La longitud en metros del terreno.

· Height: La máxima altura en metros del terreno.

· Heightmap Resolution: La resolución del heightmap. Debe tenerse en cuenta que debe ser potencia de 2 + 1. (Ejemplo: 129,513)

· Detail Resolution: La resolución del mapa de detalles, cuanta más resolución, más precisión a la hora de dibujar los detalles sobre el terreno y colocar objetos.

· Control Texture Resolution: La resolución de las texturas pintadas sobre el terreno, más resolución = más detalle, menor solución = más rendimiento.

· Base Texture Resolution: Esta es la resolución base de la textura que se renderiza desde distancia (LOD).

Son bastantes cosas que recordar, especialmente si no has usado herramientas de creación de terrenos de otros motores antes. Tomarán sentido después de que trabajes un poco con terrenos, experimentar siempre es importante.

Yo usare estos valores; pero no significa que debáis tomar los mismos, podéis personalizar vuestro propio terreno.

· Width: 500m

· Lenght: 500m

· Height: 500m

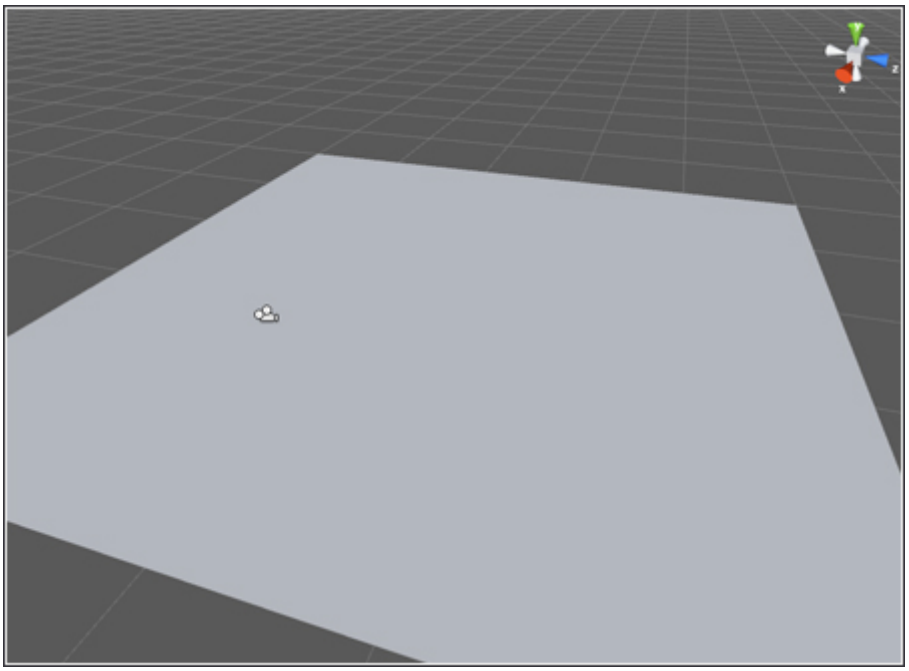
· Heightmap Resolution: 513

· Detail Resolution: 1024

· Control Texture Resolution: 512

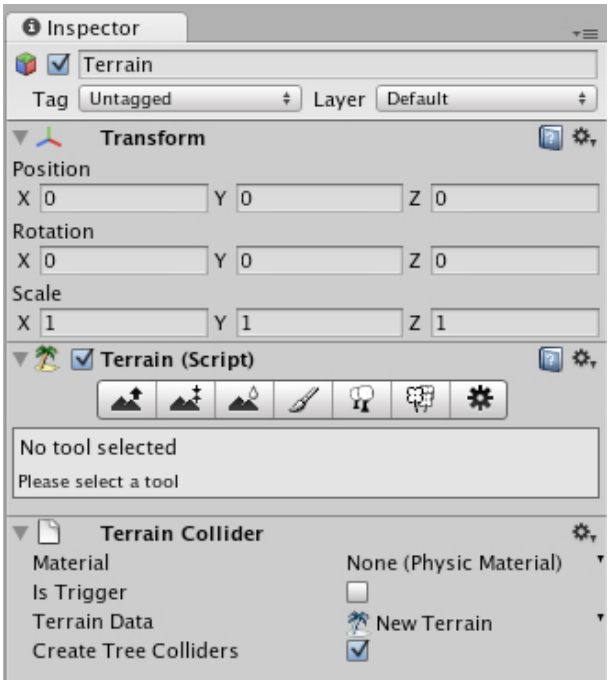
· Base Texture Resolution: 1024

Ahora deberías tener un terreno plano y gris en tu vista de escena, como en la imagen a continuación.



Herramientas de terreno

Como puedes ver nuestro terreno es fino y no muy bonito, para modificar nuestro terreno haremos uso de las herramientas de edición de terrenos. Para acceder a las herramientas necesitamos seleccionar el terreno en la jerarquía, por lo que haz clic sobre el. Cuando hagas esto notaras que el inspector cambia para mostrar las herramientas de edición de terrenos como en la imagen a continuación.



El inspector estará dividido en 3 áreas:

· Transform: Permite mover y escalar el terreno sobre los ejes x,y,z.

· Terrain Script: Contiene varias herramientas y propiedades para el terreno, sobre las que hablaremos ahora.

· Terrain Collider: Contiene las propiedades de colisión para el terreno.

En el panel Terrain Script veremos una fila de botones; estos son los botones editores de terreno, cada uno permite realizar diferentes tareas. Aquí tenéis una descripción de lo que permite hacer cada uno de los botones, en orden de izquierda a derecha:

· Raise / Lower: Nos permite levantar y hundir la geometría del terreno usando un pincel.

· Set Height: Pitamos el terreno con una altura límite.

· Smooth: Nos permite suavizar un terreno para eliminar esquinas, por ejemplo.

· Paint Texture: Nos permite pintar texturas sobre la superficie del terreno.

· Place Trees: Nos permite colocar arboles.

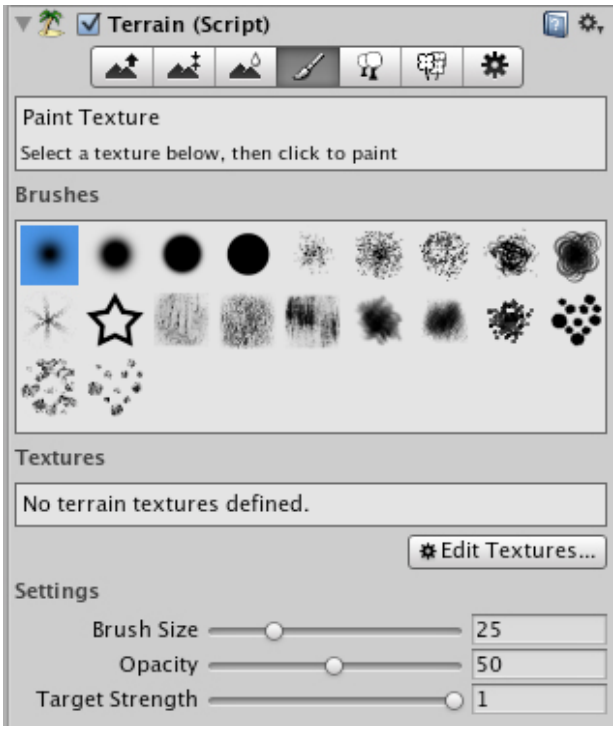
· Paint Detail: Nos permite dibujar los detalles del terreno como hierba.

· Terrain Settings: Accede a las propiedades del terreno donde podemos cambiar varias propiedades.

Definiendo una textura base

Lo primero que debemos hacer es definir una textura base para el terreno. La primera textura que añadas será aplicada al terreno entero, cualquiera que añadas después será pintada en capas superiores.

Cuando seleccionas la herramienta Paint Textures veras que aparecen algunas preferencias nuevas, como en la imagen a continuación.



La primera sección permite seleccionar el tipo de pincel, muy similar a los pinceles de Photoshop.

Debajo de los pinceles tienes el espacio de selección de texturas, donde podemos seleccionar la textura con la que queremos pintar.

Por último encontramos las propiedades del pincel que nos permiten modificar el tamaño, la opacidad y la fuerza del pincel.

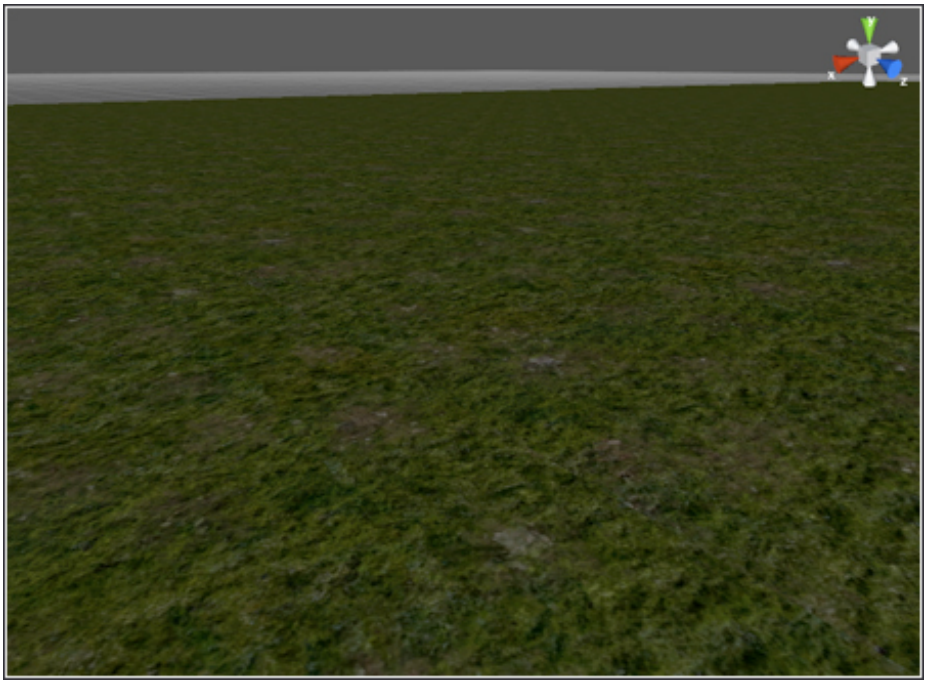
Debemos seleccionar una textura para pintar antes de hacer cualquier cosa, haz clic en “Edit Textures” y selecciona “Add Textures” para mostrar la ventana de selección de texturas.

Esta ventana nos preguntara por tres cosas distintas:

- La textura en si misma. Podemos seleccionar una textura del cuadro desplegable. Selecciona “Grass (Hill)” para nuestro ejemplo.

- El ancho y el largo de la textura. Deja estos como están.

Cuando hayas hecho clic sobre “add” veras que la textura aparece ahora en el panel sobre el botón “Edit Textures”; también notaras que la textura se ha pintado sobre el terreno. Debería parecerse a la imagen a continuación (haz zoom sobre el terreno para conseguir una mejor vista de la textura).



Si rotas la cámara alrededor lo bastante lejos del terreno veras las “juntas” de la textura por el terreno. Si quieres añadir mas texturas, estas se pintaran en el terreno sobre esta textura.

Geometría del terreno

Nuestro terreno sigue siendo plano y aburrido. Ya que el propósito de este tutorial es que tengas los conocimientos para hacer algo simple no vamos a tomarnos mucho tiempo en que nuestro terrenos quede espectacular, veamos las herramientas básicas de geometría que serán suficientes para el cometido de este tutorial.

Si seleccionas la herramienta "Raise / Lower" veras que las propiedades cambian. Las opciones serán similares a las que hemos visto con la herramienta de pinceles; estilo de pincel, tamaño y opacidad (fuerza). Una opacidad baja creara la geometría despacio, una opacidad mayor hará una geometría mas intensa.

Prueba a usar la herramienta dibujando sobre el terreno. Si mantienes shift al hacer clic hundirá el terreno, pero no por debajo de 0.

Consejo:

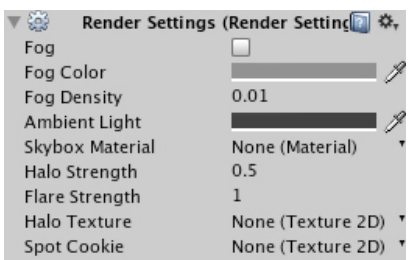
Es recomendable esculpir por pasos, primero las partes grandes y bruscas, como montañas, ríos, caminos y luego refinamos los detalles desde los más grandes a lo más pequeños. De esta forma conseguiremos un resultado mucho mejor.

Con esto terminamos la explicación de los terrenos, como siempre lo mejor que podéis hacer es trastear por vosotros mismos esculpiendo y texturizando un terreno.

Añadiendo un cielo

En Unity tenemos varias opciones para crear un skybox. La primera es adjuntar un skybox a la cámara o adjuntarla a la escena directamente, haremos esto más tarde.

Vayamos a "Edit -> Render Settings" para mostrar las propiedades de renderizado que se mostraran en el inspector.

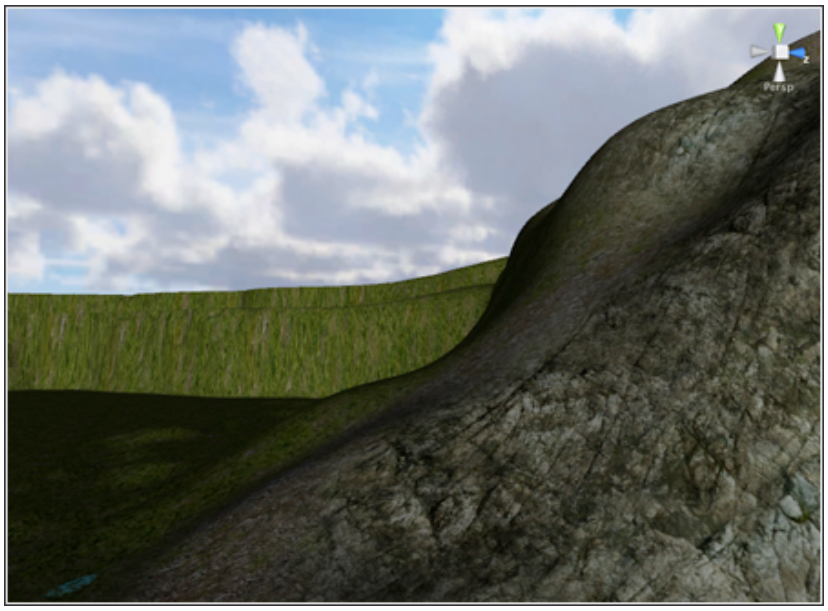


En las propiedades de renderizado, la imagen superior, veremos una entrada llamada Skybox Material. Si hacemos clic sobre esta entrada podremos seleccionar un material para el cielo. El paquete standard de assets incluido con unity trae dos materiales; “Blue Sky” y “Sunset”. Podéis encontrar más skybox en la tienda de unity Spain, por ejemplo. Las que vienen con unity no son las mejores y si quieres algo mas profesional tendrás que realizarlas tu mismo o comprarlas ya hechas por profesionales.

Por otro lado, debemos usar uno de esos materiales para nuestro tutorial, seleccionemos cualquiera de los dos “Blue Sky” o “Sunset”.

Una vez hayas seleccionado un material para el cielo lo veras aparecer en la vista de escena inmediatamente. Si no puedes verlo en la vista de escena, asegurate de que tienes “Sky, Fog y Lense Flare” activados. Recuerda que el botón para activarlos se encuentra en la parte superior de la vista de escena, al lado del botón que activa y desactiva la iluminación.

Ahora deberías de ver algo como la imagen inferior.



Con el cielo en la escena hemos avanzado bastante, ahora parece un mundo para un juego.

Añadiendo niebla

Unity incluye un sistema de niebla, que también se encuentra en las opciones de renderizado. Si activas la niebla en el checkbox veras que la escena se vuelve brumoso. Debajo del checkbox tenemos algunas propiedades para nuestra niebla.

La primera es el color, por defecto gris. Podemos cambiarlo según nuestras necesidades, por ejemplo si estamos trabajando en un área desértica podemos cambiar el color a amarillo o naranja, en un mundo mágico, quizás verde.

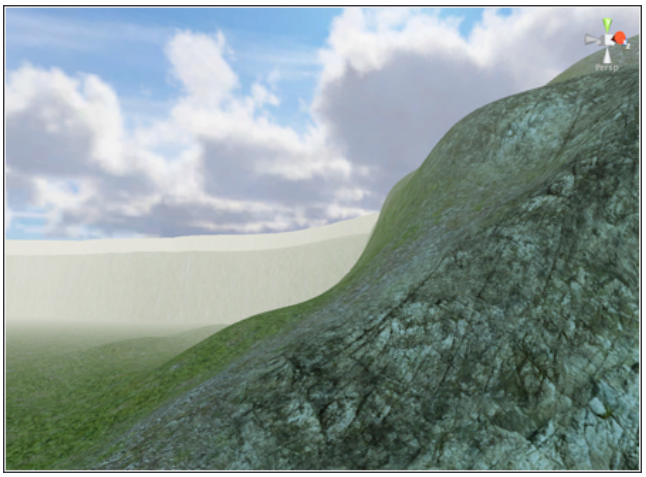
Debajo del color podemos definir la densidad de la niebla. Una densidad mayor disminuirá la visibilidad y una densidad menor la aumentara.

Color de ambiente

Aunque Unity no incluye un paquete AAA de iluminación global si tiene capacidades para efectos de iluminación globales.

Debajo de las propiedades de la niebla podemos ver “Ambient Light”, por defecto como gris. Si queremos dar a la escena un esquema diferente de iluminación, cambiar la luz ambiental es un buen principio.

Nota: Recuerda que debes tener la iluminación activada en la vista de escena para que la luz ambiental se muestre. Fijate en la captura bajo estas lineas para ver la última escena con niebla y luz ambiental.



Añadiendo luces

Por último debemos añadir una luz a nuestra escena, llamada luz solar.

Usando la luz de forma creativa podremos crear efectos sorprendentes, pero incluso con algo tan simple como la luz del sol podemos hacer nuestra escena mucho mejor.

Para añadir una luz vamos a "GameObject -> Create Other -> Directional Light".

Para que la luz tenga efecto sobre la escena es importante tener activado el botón en la vista de escena. Tan pronto como coloquemos la luz direccional veremos su efecto.

No importa donde coloquemos la luz direccional, su efecto es global a toda la escena. Lo que importa es la dirección, por lo que debes ir a la herramienta de rotación y rotar la luz para que afecte a la escena como queremos.

Podemos crear múltiples luces para crear diferentes entornos o iluminar zonas donde la fuente de luz actual no llega.

Si seleccionas la luz, veras varias propiedades en el inspector. Podemos ajustar el color de la luz para, por ejemplo, proyectar un color amarillo o naranja sobre la escena, o dar una ambientación fría con un color azul.

Cuando creemos una luz solar, queremos que esta luz se renderice como un sol. Para hacer esto, haz clic en el checkbox "Draw Halo" y selecciona "Sun" del menú desplegable Flare.

Entonces expande la sección de sombras y selecciona "Soft Shadows" del menú desplegable Type, para que esta luz realice sombras sobre el terreno.

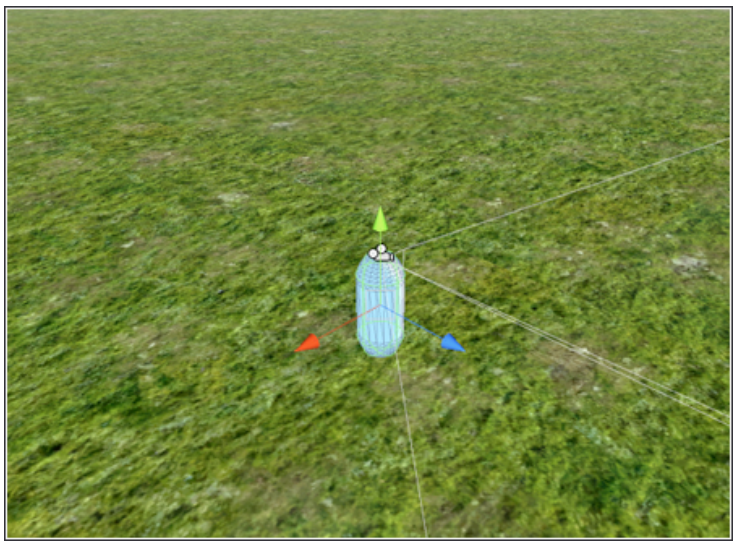
Para un mejor rendimiento y también la mejor calidad en iluminación debes considerar hacer un bake de los lightmaps, de hecho esta debe ser una practica estándar en la producción de cualquier juego.

A estas alturas ya deberías tener una escena bonita, o quizás no tanto. Te recomiendo que modifiques el terreno con bastante variación, unas colinas suaves y otras mas bruscas, incluso alguna vertical ya que ahora vamos a introducir un controlador en primera persona que caminara por nuestra escena.

Añadiendo un controlador en primera persona

Unity incluye un controlador estándar en primera persona que hace que los juegos con vista en primera persona sean realmente sencillos de configurar. Para colocar este controlador en la escena, debemos ir a la vista de proyecto y seleccionar “Standard Assets -> Prefabs”. Dentro de esa carpeta veras un prefab llamado “First Person Controller”.

Arrastra este objeto a la vista de escena y posicionalo de forma que el cilindro toque el terreno, pero no este por debajo.



Cuando el controlador esté en la escena, haz clic en el botón de reproducir para jugar en tu escena con el controlador.

Los controles son:

Flechas: Movimiento
Ratón: Control de la cámara
Espacio: Saltar

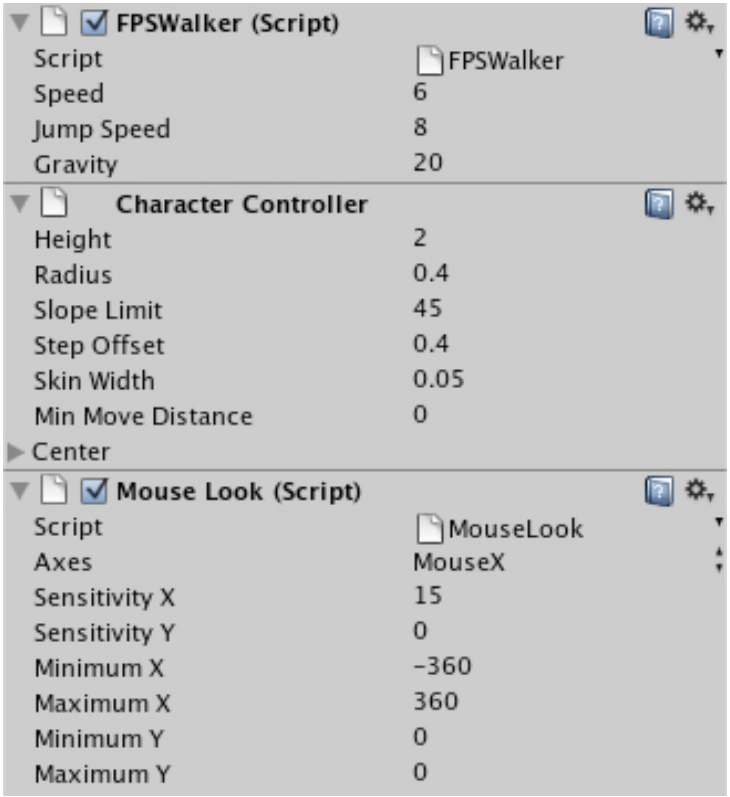
Si en este punto tu controlador cae al vacío atravesando el plano asegurate de que no este colocado en la escena atravesando el terreno.

Si te mueves alrededor te encontraras con que ciertas características básicas están incluidas en el controlador, como por ejemplo zonas donde el jugador no pueda pasar. La mejor forma de probar esto es probar a subir una inclinación vertical o bastante vertical. Para salir del modo de juego simplemente hacemos clic sobre el botón de reproducción de nuevo para volver a la vista de editor.

Personalizando el controlador

No es muy divertido ni común arrastrar los prefabricados a la escena y dejarlos como están, al menos querremos personalizarlo para que funcione como nosotros queremos que lo haga. El controlador FPS que hemos colocado en la escena puede ser personalizado a través del inspector, por lo que vamos a seleccionar el controlador en la jerarquía y echar un vistazo en el inspector, veremos que hay bastantes opciones de personalización.

Aquí tenéis una captura de pantalla:



Puedes cambiar estas propiedades con la intención de cambiar como funciona el controlador. Veamos algunas de las propiedades que te puede interesar cambiar:

· Speed: La velocidad de movimiento.

· Jump Speed: La velocidad de salto.

· Gravity: La fuerza de la gravedad.

· Height: La altura del controlador.

· Radius: El radio del controlador.

· Slope Limit: Limita el ángulo por el que puede caminar el jugador. Si por ejemplo indicamos 30 el jugador no podrá subir por superficies cuyo ángulo respecto al plano será mayor que 30.

Editando el Script

Mientras que editar las propiedades básicas debería cumplir con los requisitos de tu juego la realidad es que no es así. En bastantes ocasiones, particularmente con elementos como los controladores de personajes, necesitamos hacer mas cambios.

Estos cambios se realizan editando el script mismo.

Para editar el script del controlador, seleccionamos el controlador en la jerarquía, entonces, en el inspector buscamos la parte de FPSWalker (Script). Estas propiedades son para el FPS, un script que puedes editar.

Una forma rápida de acceder al código es hacer clic en el botón cerca de FPS Walker (Script) llamado “Edit Script”. Esto nos mostrara el editor de scripts, donde podremos hacer los cambios sobre el script.

Para empezar haremos un cambio muy simple, eliminaremos la capacidad de saltar del script. Si buscas la linea donde dice:

```
if(Input.GetButton(“Jump”)){

moveDirection.y = jumpSpeed;

}
```

Este bloque de código reacciona cuando el jugador pulsa el botón predefinido como Jump, o saltar en castellano. Podemos eliminar esto comentando o simplemente borrando el bloque entero. Para comentarlo, coloca /* antes del código y */ después. Ahora guarda el script y vuelve al editor.

Si ejecutas tu juego ahora veras que ya no puedes saltar. Deja el código sin comentar de nuevo, guarda y vuelve al editor. Ejecuta tu juego y la función de saltar estará ahí de nuevo.

Ultimos detalles

Te habrás dado cuenta de que el script del controlador acepta comandos de entrada, pero, ¿de dónde?

Si vas a "Edit -> Project Settings -> Input" veras el controlador de entrada para tu juego. Puedes configurar el numero de entradas usando el comando size y editar cada una de ellas según tus necesidades. Cambiar el nombre cambiara el nombre de la entrada, por lo que tu juego tendrá su esquema único de entrada.

Seguramente no quieras tener tu juego solo en el editor, querrás poder jugar desde un navegador web o desde un ejecutable. Para hacer esto necesitar crear una build. Ve a "File -> Build and Run". En el cuadro de dialogo haz clic en "Add Current" para añadir la escena actual y selecciona la plataforma. Tras esto el juego se compilara en la carpeta especificada. Sobre esto hablaremos más a continuación, dedicando un parte completa del tutorial a la compilación del juego.

PREPARANDO LA BUILD DE TU JUEGO

Pasamos a la tercera parte del tutorial, dedicada a compilar nuestro juego. Para crear nuestro juego y poder distribuirlo tendremos que crear una distribución de nuestro juego que funcione fuera del editor de unity.

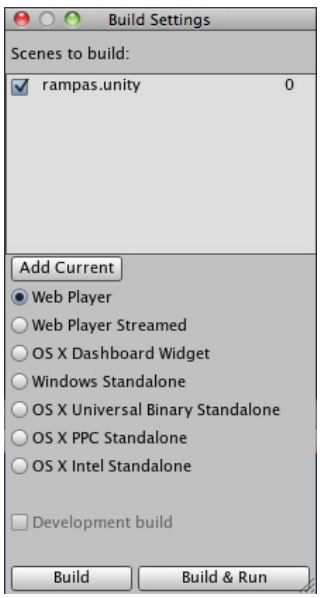
Este proceso se conoce como "building" del proyecto. Crea un ejecutable del juego y organiza los assets usados en el juego en una estructura de carpetas (sin paquetes) de las que un ejecutable puede cargar el contenido, ademas de crear otros ficheros que usa el juego. Esta build es esencialmente como se estructurará el juego el el disco duro del usuario cuando se instale el juego.

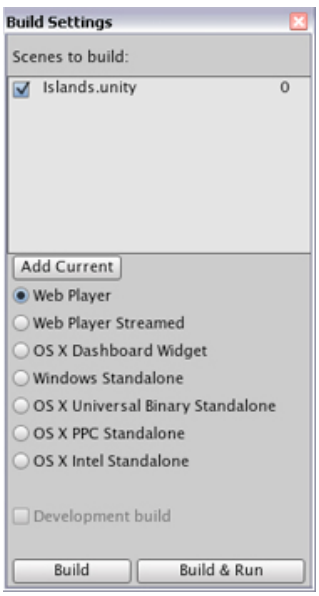
Unity no solo permite crear build para Pc y Mac, también permite llevar nuestros juegos a la web; versiones en streaming de nuestros juegos distribuidas mediante el uso del Unity Web Player. También podemos exportar para Wii si tenemos una licencia para desarrollar en esa plataforma.

Propiedades de Build

Para poder hacer la build de nuestro juego primero necesitamos elegir las propiedades de esa build. Para hacer esto debes ir a "File -> Build Settings".

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, cualquier escena que no quieras publicar debe ser demarcada. La primera escena en la lista será la escena de entrada, será la que se muestre cuando el juego 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. Veamos que es cada una:

· Web Player: Construye el juego en un formato que puede ser reproducido en el Unity Web Player.

· Web Player Streamed: Construye tu juego en un formato que puede ser reproducido en el Unity Web Player, pero el contenido se servirá mediante streaming; es decir, el juego se ira cargando dinámicamente.

· OSX Dashboard Widget: Construye el juego como un Widget para el dashboard de Mac.

· Windows Standalone: Construye el juego como una distribución para Windows.

· OSX PPC Standalone: Construye el juego como una aplicación independiente para Macs basados en PPC.

· OSX Intel Standalone: Construye el juego como una aplicación independiente par Macs basados en intel.

· OSX Universal Binary Standalone: Construye el juego para Macs PPC y Intel, por lo que da soporte a procesadores anteriores pero aumenta el tamaño de la build.

Si haces clic en "Build" el juego se construirá según la opción que hayas elegido. Pero primero veamos estas distribuciones con mas detalle.

Web player

La distribución Web Player permite al juego usarse en el Unity Web Player; lo que permite a los usuarios jugar a través de sus navegadores web desde tu sitio web usando el plugin Unity Web Player.

Cuando publiques para Web Player, los usuarios tendrán que instalarse el plugin, al igual que los usuarios necesitan instalarse flash para los contenidos flash.

Creara un fichero .unity3d y el código html.

Web Player Streamed

Esta versión del Web Player no necesita que el juego sea cargado por completo para jugar. En lugar de esto el juego se reproduce mediante streaming lo que repercute en una carga mucho más rápida.

OSX Dashboard Widget

Esta opción construirá tu juego como un widget para el dashboard de MacOS. Esta opción solo es recomendable para juegos pequeños como puzzles.

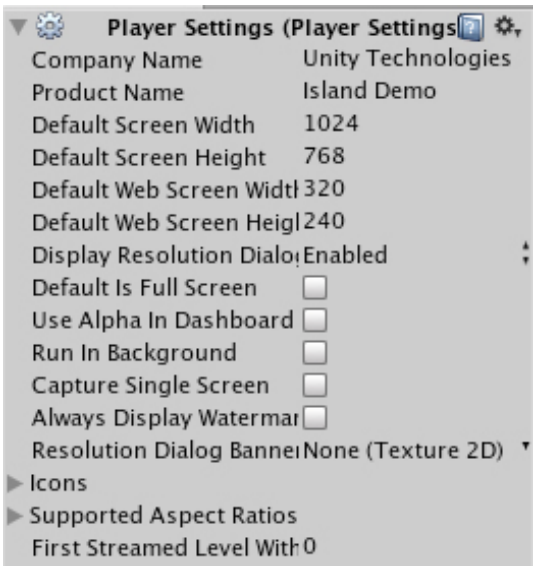
Standalone Builds

Existen varias distribuciones individuales para Mac y Pc.

Estas builds crearan un ejecutable para tu juego, comprimirán los assets y los organizarán como deben ser guardados en los discos duros de los usuarios. Esta opción es la más tradicional para juegos en descarga o vendidos en CD/DVD ROM.

Propiedades de la distribución

Para ajustar la configuración para una distribución debes ir a "Edit -> Project Settings -> Player", lo que mostrara Player Properties en el inspector.



Aquí podremos definir las propiedades para la distribución del juego. Cada una de las propiedades importantes las vemos a continuación:

· Company Name: Este debe ser el nombre del publisher, normalmente tu.

· Product Name: El nombre del juego.

· Default Screen Width: El ancho de la pantalla por defecto.

· Default Screen Height: El alto de la pantalla por defecto.

· Default Web Screen Width: El ancho por defecto para las distribuciones web.

· Default Web Screen Height: El alto por defecto para las distribuciones web.

· Display Resolution Dialogue: Activa la opción de mostrar una pantalla que permite al usuario cambiar la resolución al inicio del juego. Esta pantalla puede ser personalizada para encajar con el juego.

· Default is Full Screen: El juego se ejecutara en pantalla completa por defecto.

· Always Display Watermark: Marca cuando se debe mostrar la marca de agua, es decir el logo de Unity (Solo en Pro).

· Resolution Dialogue Banner: Este es el banner mostrado en la ventana de selección de resolución. Debe tener un tamaño de 232x163 pixels.

· Icons: Nos permite seleccionar los iconos para el dashboard, escritorio y el menú de inicio para nuestro juego.

· Supported Aspect Ratios: Nos permite seleccionar que proporciones soporta el juego.

· First Streamed Level with Resources: El primer nivel en la distribución en ser cargado.

También tenemos otras opciones que personalizar: la calidad y la entrada. Para ello vamos a "Edit -> Project Settings -> Quality" para mostrar las propiedades de calidad en el inspector.

Si vamos a "Edit -> Project Settings -> Input" podremos cambiar las opciones de entrada.

Distribuir en Nintendo Wii

Como habrás notado en el menú con las distintas opciones de distribución no aparece la opción de Wii, sin embargo producir juegos para wii es una de las opciones interesantes de unity. Esto ocurre porque para poder desarrollar en Wii necesitamos la licencia de nintendo primero y luego podremos ponernos en contacto con unity para solicitar que se nos active la opción.

Como en cualquier motor que permita publicar en consolas necesitaras pagar a Nintendo para que te habiliten un kit de desarrollo.

Installer para Windows

Si has creado una distribución para mac te habrás dado cuenta de que el juego completo esta empaquetado en un solo archivo, ya que es como MacOs lo espera. Por otro lado si quieres distribuir tu juego en Pc tendrás un ejecutable y una serie de directorios que contienen paquetes con assets y otros archivos necesarios para funcionar sobre PC.

Comprimir estos archivos en un .zip o .rar y hacer que el usuario los descargue y los extraiga es una solución sencilla pero nada deseable de cara al usuario por una serie de razones:

· En la naturaleza del hombre están los errores, por lo que el usuario puede cometerlos muy fácilmente.

· El juego no se instalara y registrara, por lo que no existirá la opción en "Añadir/ Desinstalar Programas".

· No habrá icono en el escritorio

· No habrá acceso directo en el menú de inicio

· En vista y Windows 7 no habrá entrada en la carpeta de juegos.

· El sistema no comprobara las dependencias como la versión correcta de Direct-X.

· Si el juego es distribuido en CD o DVD el usuario tendrá que copiar los archivos a su disco duro para jugar desde el disco.

· En juego no se guardara en archivos de programa a no ser que el usuario lo especifique.

· Los que solo juegan y no tienen conocimientos técnicos (una gran cuota de mercado) se echaran atrás por el proceso.

Todo hace parecer que este proceso es muy poco profesional de cara a distribuir nuestros juegos. Por eso vamos a explicar como podemos mejorar este proceso de cara al usuario y así mejorar la distribución de nuestro juego.

El installer o instalador

Para crear una rutina de instalación necesitaremos usar un sistema de instalación scriptable (como InstallShield o NSIS) para crear un paquete installer que empaquetará los archivos del juego y creara un proceso de instalación que llevara el juego al directorio especificado, creara las entradas en Windows, los iconos y comprobará los requisitos.

Este proceso es independiente del motor del juego. El motor se usa para desarrollar el juego y entonces usamos un sistema de instalación como InstallShield para crear el installer.

Cada paquete funciona de forma diferente, por lo que lo recomendable es buscar y investigar uno mismo cual es el mejor en nuestro caso.

¿Que sistemas de instalación están disponibles?

Existen bastantes sistemas de instalación, algunos muy conocidos y otros no tanto. Algunos son de pago y otros son gratuitos. Aquí tienes tres de ellos que deberías tener en cuenta:

· InstallShield

· NSIS

· InstallAware

El estándar Games for Windows

Games for Windows es un estándar para como deben ser instalados los juegos, como deben accederse y jugarse en la plataforma Windows. Los desarrolladores registrados adheridos al estándar Games for Windows pueden incluir el logo "Games for Windows" en sus productos y son promocionados en la web de Games for Windows.

Lo que significa Games for Windows es que un juego se ciñe a ciertos estándares de instalación y cumple con ciertos requisitos de compatibilidad y calidad.

Puedes saber mas acerca de Games for Windows en el sitio web: http://www.microsoft.com/GFWCertification/EN/US/default.aspx , en esta página podrás encontrar como certificar tu juego, cuales son los estándares actualmente y saber cuales son los beneficios de cumplir con Games for Windows.

Consideraciones acerca de la distribución

Hasta ahora en esta parte dedicada a la distribución de nuestro juego hemos visto como distribuir para una plataforma, como llevar el proceso de instalación e incluso cumplir con algunos estándares. Con todo esto vamos a ver unas consideraciones finales sobre este tema:

· Recuerda crear un icono para el juego, no querrás el icono de unity en el escritorio de tus usuarios.

· Prueba en profundidad en tu plataforma objetivo.

· Si estas autodistribuyendo tu juego de forma gratuita, busca en distintas plataformas web para promocionarlo.

· Si estas autodistribuyendo un titulo de pago, recuerda que el marketing es el rey y hay muchas herramientas para promocionar tu juego.

· Si estas publicando para PC; incluso si no consigues la acreditación Games for Windows, intenta seguir sus lineas ya que conseguirás mejorar la experiencia de los usuarios con tu juego.

· Asegurate de que las dependencias como DirectX son fácilmente accesibles durante la instalación de tu juego.

SCRIPTING EN UNITY

En esta parte vamos a aprender como se crean los scripts de Unity y como están estructurados. Como el resto del tutorial esta principalmente enfocado a los que son nuevos en Unity y necesitan una guía sobre como lleva Unity los scripts de forma que puedan empezar a trabajar con ellos.

Unity soporta 3 lenguajes script diferentes:

· Una versión de JavaScript

· C#

· Boo, un derivado de Python

Muchos motores gráficos utilizan lenguajes script, por nombrar dos grandes: Hero Engine y Unreal Engine ambos utilizan un lenguaje integrado de scripting (HeroScript y UnrealScript respectivamente). Estos lenguajes están basados en JavaScript o similares. Aplicaciones como Flash también integran lenguajes de scripts (ActionScript) que también es similar a JavaScript.

Hay más de una razón para basar estos lenguajes en JavaScript; por ejemplo JavaScript tiene un bajo nivel de iniciación, tiene una estructura sólida y es un lenguaje comúnmente conocido.

El Javascript que usa Unity no es JavaScript puro, es una compilación que lo hace más rápido y ha sido extendido para incluir funciones específicas para el desarrollo de un juego.

Antes de empezar

Antes de empezar con este tutorial debemos crear un proyecto nuevo, también nos será útil tener un nivel básico para añadir scripts.

1. Crea un proyecto nuevo, incluyendo el paquete estándar de assets.

2. Añade un terreno plano y dale una textura.

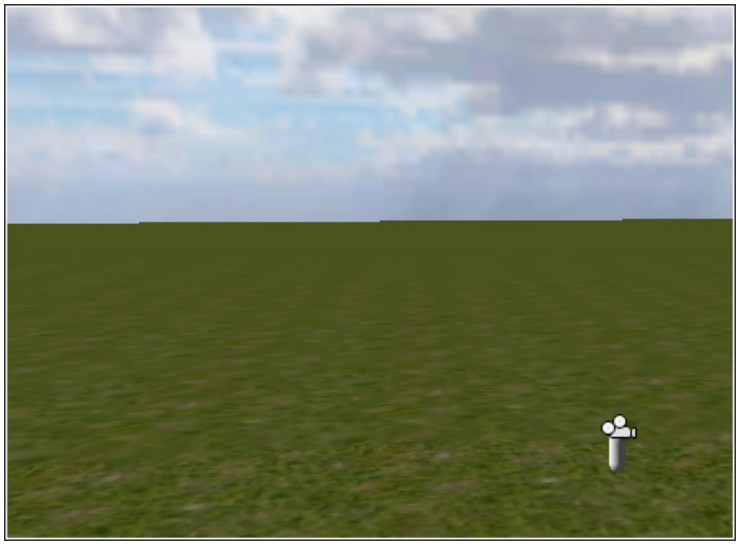
3. Añade una luz direccional.

4.Añade el controlador en primera persona de la carpeta de prefabs en los assets estándar.

Si no estas seguro de como hacer esto vuelve atrás en el tutorial ya que todo ha sido explicado.

Ahora vamos a “Edit -> Render Settings” y seleccionamos “Blue Sky” en el inspector como material para el skybox.

Deberías tener algo así:



Antes de empezar a programar haz clic en el boton Play y comprueba que todo funciona correctamente, deberías ser capaz de andar sobre el plano.

Creando un nuevo script

En la vista de proyecto haz clic en el botón “Create” y selecciona “JavaScript” para crear un nuevo archivo JavaScript. Si quieres trabajar en C# o Boo selecciona uno de estos, pero este tutorial se centrara en el uso de JavaScript.

Cuando hagas esto veras un archivo nuevo en nuestro proyecto llamado “NewBehaviourScript”. Este es un script en blanco. Para renombrarlo haz dos clic sobre el y introduce el nombre nuevo.

Para editar el script simplemente haz doble clic sobre el para arrancar el editor de código.

Tras esto, veras que el script tiene una sola función en el, llamada “Update()”, esta es una función especial llamada con cada frame.

Nuestro primer script

Para nuestro primer script vamos a crear un código que rote un cubo. Este ejercicio te enseñara como crear un script, como adjuntarlo a un objeto y como manipular el objeto al que esta adjuntado.

1. Crea un nuevo script y llamado “RotateCube”.

2. Abrelo en el editor de scripts.

3. Borra la función Update del script.

Ahora introduce el siguiente código y guarda el script:

```
function Update(){
        transform.Rotate(0,5*Time.deltaTime,0);
}
```

Ahora para conseguir que este script se ejecute necesitamos adjuntarlo a un objeto que hará que el script se ejecute.

1. Ve a “Game Object ->Create Other -> Cube” para crear el cubo.

2. Con la herramienta de transformación seleccionada mueve su posición en y a 1 para que este sobre el terreno.

3. Mueve el cubo para que este cerca de nuestro controlador en primera persona.

4. Arrastra el script “RotateCube” en la vista de proyecto a “Cube” en la jerarquía.

El cuarto paso, en el que arrastramos el script al cubo ha adjuntado nuestro script a ese objeto.

Si seleccionas el cubo en la jerarquía i miras el inspector veras las propiedades del script en la parte de abajo, lo que confirma que el script esta adjuntado a nuestro cubo.

Con este script adjunto al cubo, haz clic en Play y mira al cubo, quizás tengas que moverte para hacerlo. Veras que el cubo esta rotando suavemente.

Pero, ¿que hace exactamente el código que hemos introducido?

· La función Update() es llamada una vez por frame, por lo que cada frame se ejecuta. Esta es una función reservada especial de Unity.

· transform.Rotate() es otra función especial de unity que direcciona a la función transform (para movimiento) y selecciona la clase Rotate (una de las funciones de transform).

Si vemos la función transform.Rotate ahora, veras que hemos pasado un número de variables. La función transform.Rotate acepta tres valores.

transform.Rotate(x,y,z);

Nosotros hemos marcado la 'x' como 0, 'y' a un valor calculado y 'z' a 0.

Como puedes ver, no hemos introducido un valor estático a 'y', en su lugar hemos usado una función que calcula 'y'.

time.DeltaTime es una función que mueve los eventos sobre el tiempo, en lugar de sobre el ratio de frames. Si lo basamos en el ratio de frames el cubo se movería velocidades diferentes en diferentes ordenadores y la velocidad no sería constante ya que los frames suben y bajan.

Usando 5*Time.deltaTime estamos indicando que el cubo gira 5 grados por segundo.

Ahora juguemos con el script, aceleremos el cubo ya que se esta moviendo muy lento.

Editamos la función transform.Rotate a:

transform.Rotate(x,60*Time.deltaTime,0);

Cambiando de 5 a 60, ahora estamos rotando el cubo 60 grados cada segundo.

Variables

Ahora que tenemos un cubo rotando, añadamos variables a la mezcla.

En la parte superior del script (antes de la función Update()) añade la siguiente linea de código:

var speed = 60.0;

Lo que hemos hecho aquí es crear una nueva variable llamada "speed" (velocidad) y inicializarla con un valor de "60.0". Ahora podemos usar esta variable para controlar la velocidad de nuestro cubo.

Para que la función de rotación utilice esta nueva variable en lugar del código que usamos antes, simplemente debemos sustituir el "60" por "speed" así:

transform.Rotate(0,speed*Time.deltaTime,0);

Usando la variable ahora podemos modificar la velocidad usando scripts. Cuando trabajemos en el gameplay, podremos tener eventos que cambien esta variable. Pongamos el caso de que el personaje muere, la velocidad del cubo podría disminuir.

Además, creando esta variable hemos creado la variable para el Cubo. Si seleccionas el cubo en la jerarquía y revisas el inspector veras que la variable "Speed" aparece como una variable del objeto. Si creas una variable en un script adjunto a un objeto, el objeto heredara esa variable que será accesible desde el inspector.

Podemos entonces editar la variable desde aquí; especialmente para artistas que estén usando los objetos para construir el mundo, podrán cambiar las propiedades desde el inspector, sin tener que enfrentarse a un script; lo que puede ser peligroso si no son programadores. Esto permite a los programadores crear scripts y a los diseñadores usar los objetos de forma segura.

Prueba a cambiar la velocidad desde el inspector y fijate en lo que ocurre; la velocidad original (60) se mantendrá en el script, pero habremos cambiado la velocidad desde el inspector.

Moviendo el cubo

En el ejercicio anterior hemos hecho rotar al cubo, ahora lo vamos a mover. Sin complicarnos, hagamos que el cubo se aleje hacia el cielo suavemente.

En el script RotateCube, vamos a añadir una función translate. La función Translate es una forma sencilla de movernos usando la función transform. Entonces, añadamos este código debajo de la función de rotación:

```
transform.Translate(Vector3.up*Time.deltaTime,Space.World);
```

Si ejecutas el juego ahora, veras que el cubo rota y al mismo tiempo se va hacia el cielo.

Esta función esta usando algunos parámetros nuevos que no hemos visto. Veamoslos pues:

· Vector3.up : Vector3 es usado para referenciar posiciones en un espacio 3d acompañados por una dirección. Entonces Vector3.up hace referencia a la dirección vertical en el espacio 3D.

· Space.World: Aplica el movimiento en el sistema de coordenadas globales. Space.Self utiliza un espacio local. Prueba a cambiar Space.World a Space.Self; nada cambiara ya que simplemente hemos indicado que la operación de translación debe usar las coordenadas locales.

Imprimiendo por consola

Unity tiene una consola de soporte, por la que podemos imprimir. Esta consola ofrece una plataforma simple para crear un script “Hola Mundo”, ya que podemos imprimir “Hola Mundo” o cualquier mensaje que queramos.

Esto es útil ya que podemos escribir errores personalizados, mensajes y información de depurado por la consola mientras desarrollamos nuestro juego.

Añadamos una nueva linea de código a nuestro script para hacer que imprima un mensaje por pantalla. Coloca la siguiente linea después de la función translate:

print("¡El cubo se mueve!");

Ahora si ejecutas el juego veras que "¡El cubo se mueve!" ha aparecido en la consola. Cuando salgas del juego, muestra la consola (doble clic sobre ella) ,que se encuentra en la parte inferior de Unity, y verás el texto listado ahí.

Ahora que tenemos una base aprendamos los 4 principios básicos del scripting, que son:

· Variables

· Funciones

· Condicionales

· Bucles

Si ya has programado antes todos estos términos te resultaran familiares y probablemente sabrás como funcionan. Si este es el caso esta última parte del tutorial te servirá como una guía para conocer la sintaxis de las variables, las funciones, condicionales y los bucles en Unity.

Para los que no sepan lo que es cada cosa, veamoslo:

- Las variables son contenedores, podremos almacenar información en ellos para usarla mas tarde.

- Las funciones son partes reutilizables de código que pueden ser ejecutadas llamando al nombre de la función.

- Los condicionales(como IF,ELSE IF,ELSE) nos permiten seleccionar de un número de condiciones.

- Con los bucles podemos repetir continuamente el mismo bloque de código mientras se cumplan unas condiciones.

Vamos a ver cada uno de estos 4 puntos en más profundidad.

Primero tenemos que crear un nuevo script, haz clic en "Create -> JavaScript" en la vista de proyecto para crearlo y llamalo "PrimerScript". Ahora haz doble clic sobre el para abrirlo en el editor. Cuando el script este abierto borra la función Update() { }, para tener un script en blanco.

Introducción a las variables

Las variables son espacios en memoria donde puedes guardar todo tipo de información como números y palabras.

Antes de poder guardar nada en una variable debemos declarar que existe, esto se hace así:

var NOMBRE_VARIABLE : TIPO = "VALOR POR DEFECTO";

Puedes llamar a la variable como quieras, pero siempre cumpliendo que no tenga espacios en blanco ni contenga caracteres especiales como @,#...

También debes decir a Unity que tipo de variable es. El tipo simplemente indica que clase de datos vamos a guardar en la variable. Existen varios tipos:

· int: Entero, contiene números enteros. Por ejemplo: 5

· float: Flotante, contiene números con decimales: Por ejemplo: 5.2

· String: Cadena, contiene palabras por lo que puede contener cualquier carácter.

· boolean: Puede tomar valores True(Verdadero) o False (Falso)

Aquí tenéis ejemplos de declaración de los distintos tipos de variables:

```
var nivel : int = 1;

var distancia : float = 50.25;

var mensaje : String = "Bienvenido a Unity Spain";

var moviendose : boolean = false;
```

En nuestro primer ejemplo hemos declarado una variable entera, que puede ser usada para almacenar cualquier número como el nivel del jugador, o cuantas piezas de oro ha conseguido el mismo. En el segundo ejemplo hemos declarado una variable float que contiene números con punto decimal, que puede ser usado para almacenar, por ejemplo, la distancia entre dos objetos.

En el tercer ejemplo hemos declarado una variable de tipo cadena o string que contiene texto, fijate en como el texto, a diferencia de los otros valores, va entre comillas; esto es muy importante para las variables de tipo cadena.

Por último hemos definido una variable de tipo boolean que contiene un valor verdadero o falso (True o False) que puede ser usado para activar/desactivar algo, como por ejemplo si el jugador se esta moviendo o no.

Variables públicas y privadas

Puedes hacer que las variables sean públicas o privadas. Si declaras en un script una variable como esta:

```
var nivel : int = 1;
```

Esta variable será visible en el inspector.

Si la hacemos privada de la siguiente forma:

```
private var level : int = 1;
```

La variable no será accesible desde el inspector y solo podrá ser modificada desde código. Cuando una variable es pública, puedes sobreescribir el valor del script editando su valor en el inspector.

Pongamos esto en practica, en el script que tenemos abierto escribe:

```
var vidas : int = 10;

private var nivel : int = 1;

start()

{

}
```

Guarda el script y arrastralo a la cámara en la jerarquía para incluirlo en la escena. Entonces haz clic en la cámara y mira el inspector. Veras las vidas en el inspector y podrás editar su valor. La variable nivel no aparecerá porque es privada.

Seguramente obtengas un mensaje de error, esto ocurre porque el script no hace nada actualmente.

Variables globales

Existe otro tipo de variables, las variables globales. Se declaran de la siguiente forma:

```
static var vidas : int = 10;
```

Utilizando una variable global conseguimos que esta sea accesible desde cualquier otro script y además estará disponible desde el inicio de ejecución. Por esta misma razón no se debe abusar de estas variables ya que siempre estarán consumiendo memoria RAM.

Estructura básica de una función

Las funciones son partes reutilizables de código.

Empezare por explicar que Unity tiene funciones reservadas, que son funciones incluidas en Unity y pueden ser llamadas usando sus nombres. Dos funciones reservadas son muy importantes, "start()" y "update()".

La función start() se ejecuta una vez se llama al script, update() se ejecuta continuamente en cada frame.

Usare la función start() para demostrar cual es la estructura básica de una función:

```
function start()
{
}
```

Lo primero de todo debe ser siempre la palabra "function" a la hora de declararla. Entonces tenemos { y } al inicio y al final.Todo lo que está entre esos corchetes es la función, muy sencillo.

Casi siempre tendremos una función start() o update() en un script.

También puedes crear tus propias funciones de la misma forma, declarandolas igual. Cuando quieras usar una función que hayas declarado la llamamos en el script escribiendo su nombre seguido de "()".

Aquí tienes un ejemplo simple:

```
function imprimirMensaje()
{
        var mensaje : String = “esto es un mensaje”;
        Debug.Log(mensaje);
}
function Start()
{
imprimirMensaje();
}
```

Lo primero que hemos hecho es crear la función llamada imprimirMensaje() que imprime un mensaje al registro usando la función reservada Debug.Log(). Entonces llamamos a la función imprimirMensaje desde la función start.

Si adjuntas este script a la cámara y ejecutas el juego veras que aparece en el registro “esto es un mensaje”.

Pasando variables a las funciones

Podemos pasar variables a las funciones, tomemos el script anterior y movamos la declaración del mensaje a la función Start() y pasemos el mensaje a la función imprimirMensaje.

```
function imprimirMensaje(mensaje : String)
{
        Debug.Log(mensaje):
}
function Start()
{
        var mensaje : String = "esto es un mensaje";
        imprimirMensaje(mensaje);
}
```

Pasamos la variable definiendola en la declaración de la función y después incluimos el nombre de la variable al llamar a la función (entre los paréntesis).

Condicionales

Usamos los condicionales para seleccionar entre múltiples eventos dependiendo de una o más condiciones.

Los condicionales se usan con frecuencia en cualquier juego, ya que siempre necesitaremos seleccionar entre diferentes opciones basándonos en condiciones. Aunque las posibilidades y los usos de los condicionales son muchas, aquí tienes algunos ejemplos de donde se pueden usar:

- Para detectar si un botón ha sido pulsado

- Para detectar la vida del jugador y continuar si el jugador esta vivo o morir si la vida alcanza o.

- Para cambiar interruptores

- Para realizar distintas acciones dependiendo de la acción del usuario

- Para detectar si una partida guardada existe y se ha cargado correctamente.

- Para validar formularios

Lo más común es usar el condicional IF, que será el primero que veamos.

Condicionales IF

Los condicionales If son probablemente los más comunes. Son sencillos de programar, aceptan parámetros sencillos o múltiples y también aceptan los condicionales else y else if para selecciones múltiples.

Veamos algunos condicionales If sencillos para mostrar como es la sintaxis y como se usan:

Parámetros sencillos

```
if(Input.GetButtonDown("Jump"))
{
        código para hacer que el personaje salte
}
```

Este código es la formas más sencilla de usar un condicional IF. Solo se ha pasado un parámetro (en este caso, un opción de entrada(el botón ha sido etiquetado como "Jump")). Si el botón Jump ha sido pulsado, activara este condicional con un valor true y entonces se ejecutará el código entre {}.

Cuando pasamos un valor sencillo a un condicional IF, comprobará un resultado true/false. En el caso anterior, comprueba si el parámetro es "true", si queremos comprobar si el parámetro es falso usamos el operador "not", así:

```
if(!Input.GetButtonDown("Jump"))
{
        el código que se ejecutará cuando el personaje no este saltando.
}
```

El ! antes del operador indica a Unity que quieres saber si el parámetro que pasamos esta retornando un valor nulo o esta vacío.

Múltiples parámetros

Ahora vamos a comprar múltiples parámetros.

Para empezar, declaramos dos variables al inicio del script (antes del Start() o la función Update())

```
var variable1 : int =1;
var variable2 : int = 0;
```

Entonces creamos la función start:

```
function start()
{
}
```

Y dentro de la función start() colocamos el siguiente código:

```
if(variable 1 > variable2)
{
        print("La variable 1 es mayor que la variable 2");
}
```

Este ejemplo compara dos variables para ver si la primera es mayor que la segunda, si esto se cumple el código entre corchetes se ejecuta. Cuando comparamos variables podemos usar varios operadores:

'>' A es mayor que B

'<' A es menor que B

'==' A es igual a B

'!=' A no es igual a B

'>=' A es mayor o igual a B

'<='A es menor o igual a B

¿Que ocurre si queremos comparar más de dos variables? Veamos un ejemplo:

```
if(variable1 > variable2  && variable1 != 0)
{
        print("La comparación devuelve true");
}
```

En este ejemplo comparamos dos condiciones diferentes, que la primera variable es mayor que la segunda Y que la primera no es igual a 0. El operador '&&' se usa para comparar dos operadores, con && las dos condiciones deben retornar true para que el código entre corchetes se ejecute. Veamos otro ejemplo:

```
if(variable1 > variable2  || variable1 != 0)
{
        print("La comparación devuelve true");
}
```

En este ejemplo cambiamos '&&' por '||' que significa "O", por lo que el código se ejecutara si la primera variable es mayor que la segunda, pero también se ejecutará si la primera variable no es 0. Puedes crear condiciones muy complejas añadiendo más comparaciones al

condicional como estas; aunque parezcan muy complejas reducirán el número de condicionales y harán todo más eficiente.

Else y Else If

En los IF anteriores todo era simple, hacíamos una condición simple, pero que ocurre si queremos hacer cosas diferentes según los valores, por ejemplo, si la primera variable contiene un 5 queremos hacer una cosa, si contiene un 3 queremos hacer otra y si hay otro valor, otra.

Haremos esto usando ELSE y ELSE IF, veamos un ejemplo:

```
if(variable1 == 5)
{
        print("var1 es 5");
}
else if(variable1 == 3)
{
        print("var1 es 3");
}
else
{
        print("var 1 es otro valor");
}
```

En este ejemplo, el primer bloque de código se ejecutará si la primera variable es igual a 5, el segundo bloque si es igual a 3 y el último bloque si la variable contiene otra cosa. Puedes tener tantos if como quieras, para selecciones más complejas.

Anidando

```
if(variable1 == 5)
{
        print("var1 es 5");
}
else
{
        if(variable1 == 3)
        {
                print("var1 es 3");
        }
        else
        {
                print("var 1 es otro valor");
        }
}
```

En este ejemplo hemos anidado dos condicionales IF. Anidar nos permite segmentar los condicionales.

Condicionales SWITCH / CASE

Si has programado alguna vez usando otros lenguajes ya conocerás los condicionales switch/case. Estos condicionales nos permiten tomar una sola variable y realizar múltiples operaciones dependiendo de lo que contiene la variable.

Aquí tenéis un ejemplo simple:

```
switch(una variable)
{
        case "A":
                funcioncualquiera();
                break;
        case "B":
                otrafuncion();
                break;
        default:
                break;
}
```

El condicional switch toma la variable y realiza uno de los tres eventos dependiendo del valor de la variable. Si la variable contiene A entonces se ejecutara funcioncualquiera(). Si la variable contiene B, entonces se ejecutara otrafuncion(), si no es ninguna de las dos se utilizara la operación default. "Break" detrás de cada case detiene el script en ese punto, por lo que los condicionales case son recomendables en escenarios específicos.

Bucles

Los bucles nos permiten repetir bloques de código hasta que se cumplan ciertas condiciones. Veamos los tipos básicos de bucles y como implementarlos.

Antes de empezar, crea un nuevo script y adjuntalo a un objeto vacío en la escena. Después asegurate de que tiene una función Start() y no Update. Obviamente puedes usar blucles en la función Update() de la misma forma, pero para el propósito de este tutorial usaremos la función start().

```
function Start()
{
}
```

A través de este apartado iremos comentado código, simplemente incluyelo entre los corchetes a no se que diga otra cosa.

Bucles While

Los bucles while nos permiten repetir código. Esto se usa normalmente para leer arrays de datos (como cuando lees un archivo que has cargado). Veamos un bucle while sencillo que incrementará un contador en cada ciclo hasta que el contador llegue a 10.

Antes de la función Start(), declara la siguiente variable:

```
var contador : int = 0;
```

Ahora, dentro de la función start(), colocamos el siguiente código:

```
while(contador <= 10)
{
        print("Contando" + contador);
        contador++;
}
```

Después de esto, ejecuta el juego. Probablemente veras "contando 10" en la pantalla ya que se repetirá tan rápido que no veras el incremento. Para el juego y abre la consola, verás la cuenta desde 0 hasta 10. El bucle while ha ejecutado el print hasta que el contador ha llegado a 10.

Te habrás fijado en que he utilizado “contador++;” en lugar de “contador = contador + 1;”, contador++ es una forma más simple de hacer lo mismo.

Como puedes ver, los bucles while son muy simples de crear, solo recuerda asegurarte de que tu bucle alcance la condición que lo detiene o el código se repetirá en un bucle infinito llegando incluso al punto de que el ordenador detenga la ejecución.

Bucles FOR

Otro tipo de bucle es el bucle for, a diferencia del bucle while, en el que definimos unas variables y una condición que detiene el bucle dentro del bucle, con los bucles for podemos definir las condiciones y el incremento sin salir de la inicialización del bucle.

Estos bucles se usan especialmente para contadores o otros bucles de tipo incremental.

Remplaza el código en la función Start() con el siguiente código:

```
for(var contador = 0; contador <= 10; contador++)
{
print(“Contando” + contador);
}
```

Con los bucles for definimos más parámetros en la inicialización del bucle, estos son:

for (valor_de_inicio; condición_para_detener ; incremento)

El código hace exactamente lo mismo que nuestro bucle while.

Con esto terminamos con los bucles.

CONCLUSIÓN

Con los bucles terminamos este tutorial que ha costado muchas horas de trabajo, y que pese a que se ha hecho en apenas unos días, de lo que podríamos deducir que no lleva tanto me gustaría comentar que me he centrado en el tutorial al 100% durante estos días llegando a trabajar en el 7 horas diarias hasta terminarlo.

En el tutorial he hablado creo que de todo a modo introductorio, desde que es unity hasta como programar en unity, pasando por terrenos, luces, escenas, proyectos, interfaz...

Espero que como comprador estés contento con tu compra y que recuerdes el trabajo que desde el equipo de unity Spain hacemos cada día para ofrecer los mejores contenidos en castellano para este increíble motor gráfico llamado Unity3D.

Un saludo y Muchas Gracias,

David Collado,

Webmaster,

Unity Spain

Para este manual me he basado en diversas fuentes a parte de en mis propios conocimientos como son los documentos oficiales de unity y unitylabs. Del primero he obtenido datos fundamentales y del segundo la estructura de este tutorial. Cuento con el permiso de ambos, por supuesto.