TRAYECTORIAS VECTORIALES EN UNITY 3D

CÁLCULO

Nombre: Daniela Beatriz Martínez Montes
Profesor: Félix Acosta
Grupo: 2.B

15/03/2017

UNITY

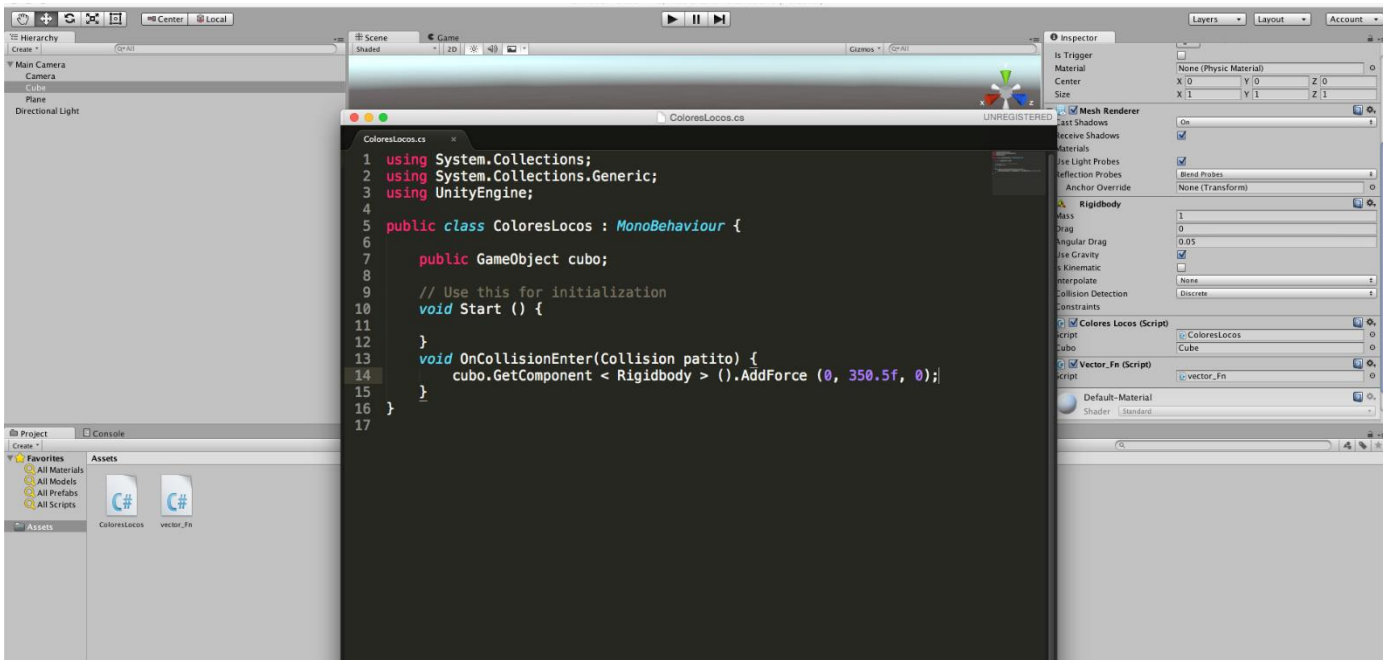
UNITY es un motor de desarrollo para la creación de juegos y contenidos 3D interactivos, con las características que es completamente integrado y que ofrece innumerables funcionalidades para facilitar el desarrollo de videojuegos. Unity está disponible como plataforma de desarrollo para Microsoft Windows, OS X. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas.

La primera versión de Unity se lanzó en la Conferencia Mundial de Desarrolladores de Apple en 2005. Fue construido exclusivamente para funcionar y generar proyectos en los equipos de la plataforma Mac y obtuvo el éxito suficiente como para continuar con el desarrollo del motor y herramientas. Unity 3 fue lanzado en septiembre de 2010 y se centró en empezar a introducir más herramientas que los estudios de alta gama por lo general tienen a su disposición, con el fin de captar el interés de los desarrolladores más grandes, mientras que proporciona herramientas para equipos independientes y más pequeñas que normalmente serían difíciles de conseguir en un paquete asequible

Gracias a UNITY, puedes acceder a Smartphones, navegadores web, Xbox 360, Wii U y PS3 entre otros, donde podrás desarrollar juegos que van desde los MMOG, shooters, hasta juegos de roles. El éxito de Unity ha llegado en parte debido al enfoque en las necesidades de los desarrolladores independientes que no pueden crear ni su propio motor del juego ni las herramientas necesarias o adquirir licencias para utilizar plenamente las opciones que aparecen disponibles. El enfoque de la compañía es "democratizar el desarrollo de juegos", y hacer el desarrollo de contenidos interactivos en 2D y 3D lo más accesible posible a tantas personas en todo el mundo como sea posible.

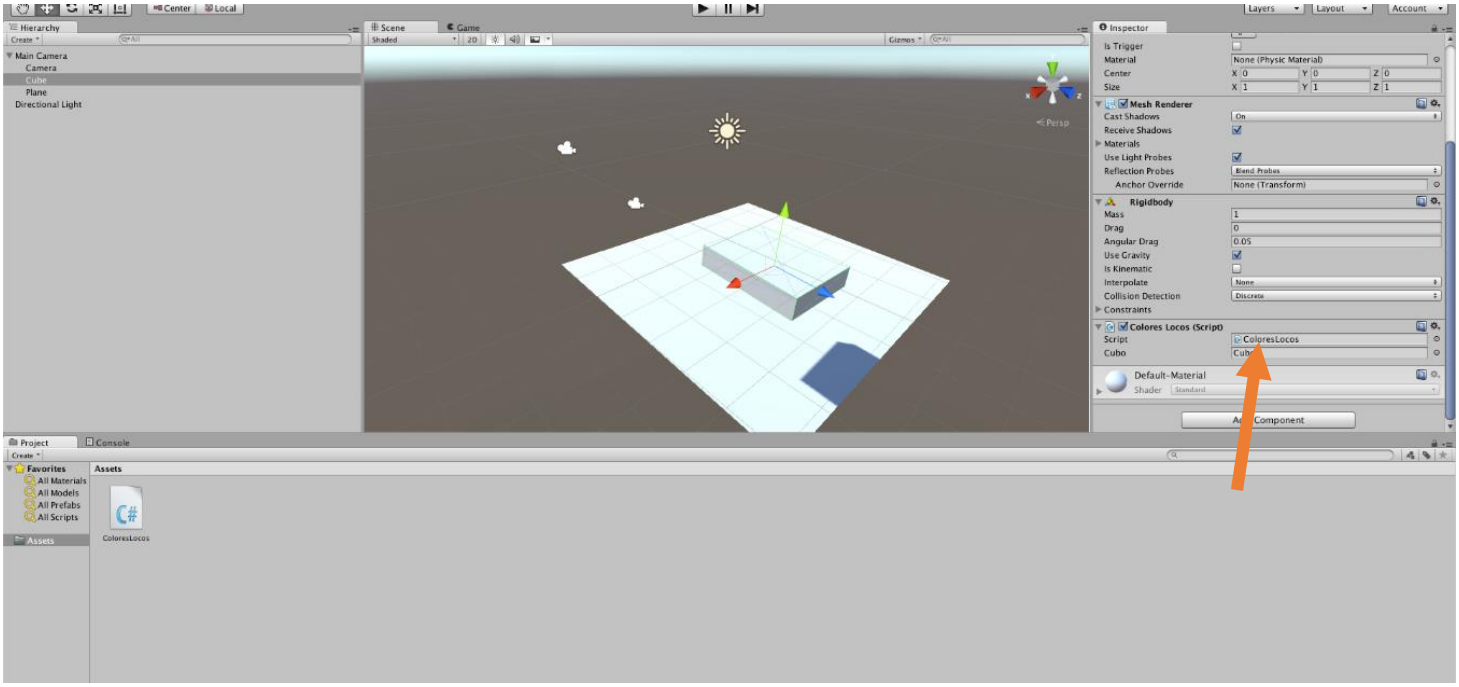
Para la creación de una figura en 3D, seleccionamos en Main camera-cube, para asignar una cámara, damos click derecho en camera, de esta forma podemos identificar y observar en vista previa nuestro cubo.

Para comenzar a asignar movimiento a nuestro cubo, en este caso movimiento de caída, creamos un script, en este caso, llamado colores locos, el cual usa el lenguaje de programación C#



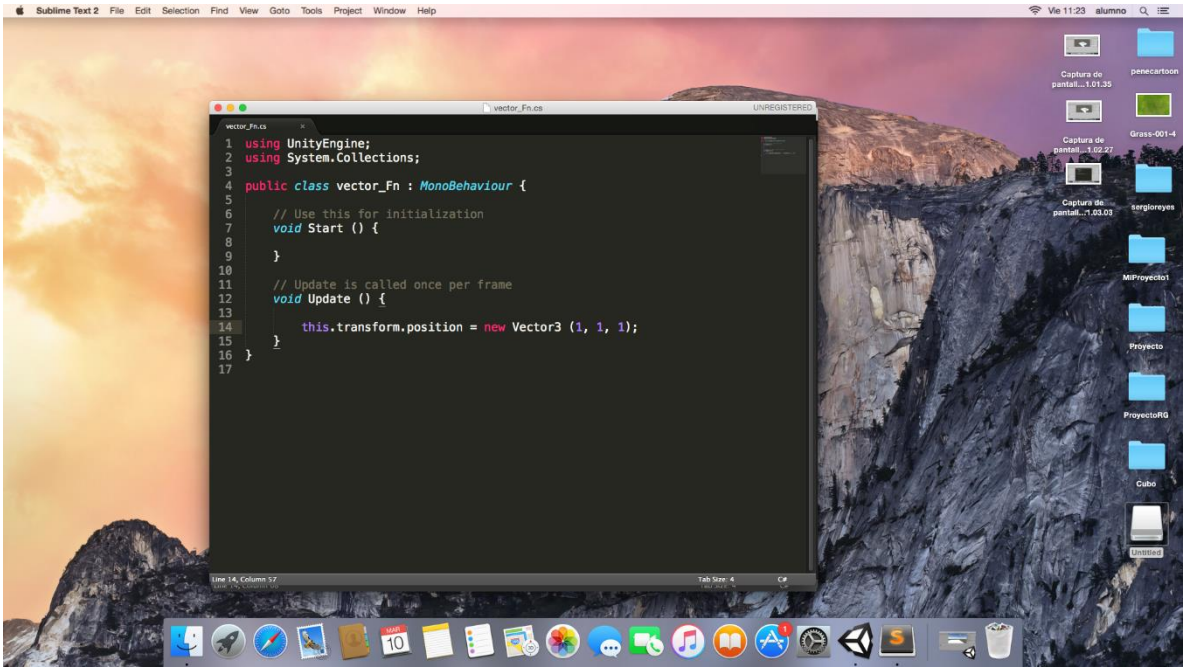
OnCollisionEnter se llama cuando el Collider o Rigidbody entra en contacto con otro Collider o Rigidbody. OnCollisionEnter se le pasa la clase Collision, no un Collider. La clase Collision contiene información sobre puntos de contacto y/o velocidad de impacto

Una vez que tenemos finalizado nuestro script, se lo asignaremos al objeto de nuestra escena (en este caso, representado por un cubo) el cual queremos que siga un patrón de movimientos, la forma de asignarlo es muy sencilla, solo arrastramos “cube” el cual se encuentra en el lado izquierdo a “colores locos (script)” el cual se encuentra en lado inferior derecho, se muestra con la flecha naranja.



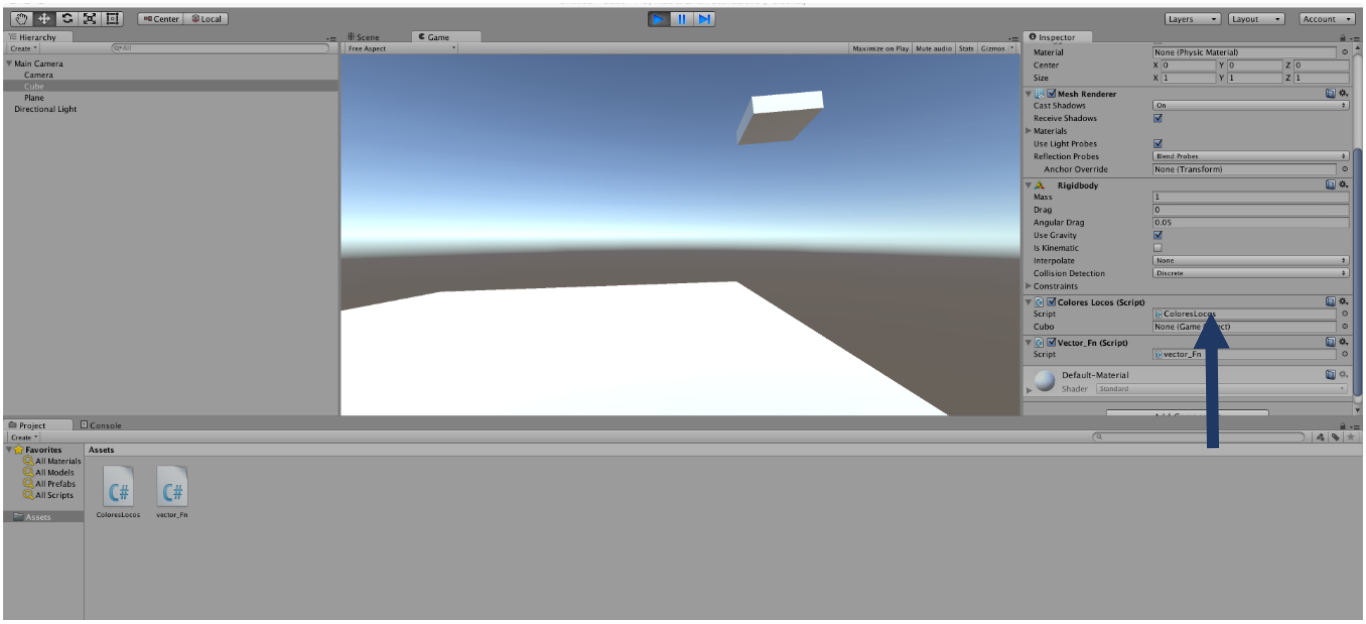
El único requerimiento que debe cumplir este objeto para que el script funcione correctamente es que contenga el componente Rigidbody. En el script “Colores locos” se puede apreciar como llama a la función Rigidbody.

Para modificar la posición creamos un nuevo script, en este caso llamado vector_Fn, en este estamos explicando que la queremos modificar la posición de nuestro vector actual, por esto mismo se expresa this.transform.position, ya que le estamos diciendo al sistema que el vector existente es el que recibirá los cambios. La posición se cambio a 1,1,1

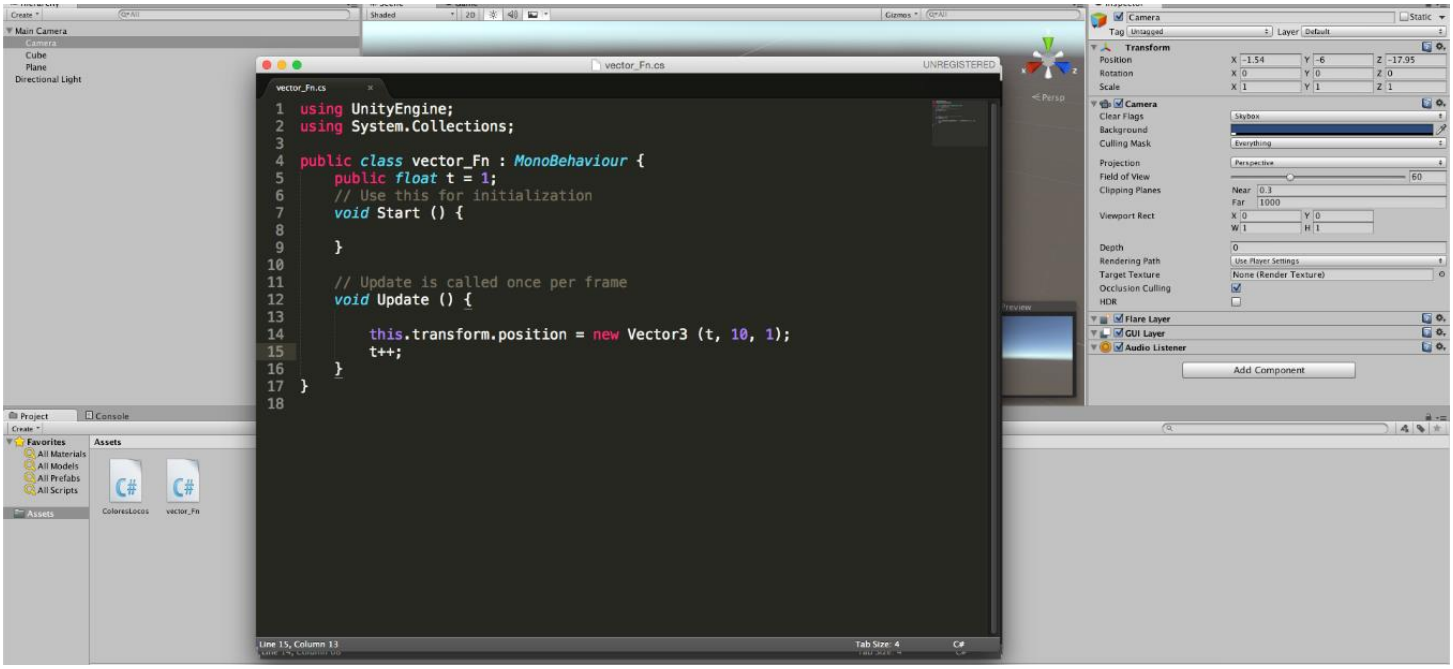


Es importante mencionar que “Transform” se usa para almacenar y manipular la posición, rotación y escala del objeto

Al momento de crear el script, tenemos que asignarlo, se realiza de la misma forma que “Colores locos” solo arrastramos el script de Vector_Fn a “Vector_Fn(script)” el cual se encuentra en lado inferior derecho, se muestra con la flecha azul



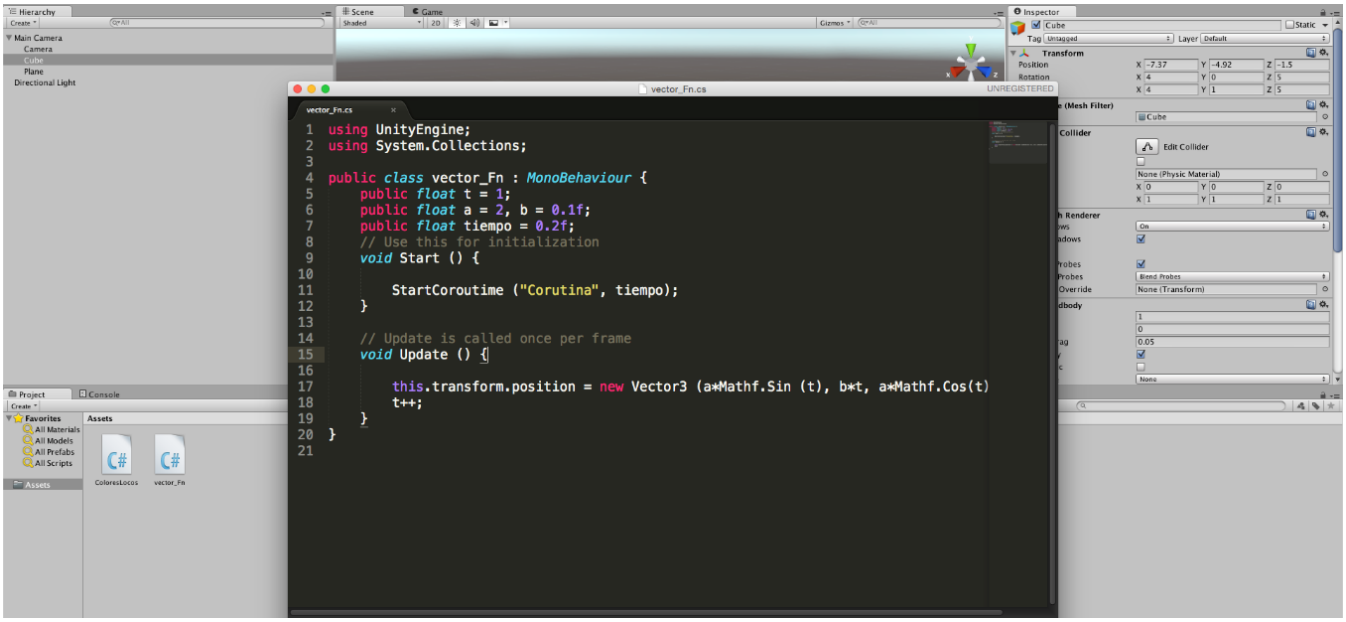
Al igual que en la imagen anterior, en este script estamos modificando la posición, con la diferencia de que asignamos el tiempo, en este caso uno, todo esto provocara que el cubo obtenga más movimiento, un poco más rápido, además creamos un nuevo vector3 que representa vectores en 3D y sus puntos



Es importante mencionar que MonoBehaviour es la clase base de todo script que se deriva de unity, en caso de que se esté haciendo con c#

Una coroutine es una función que se ejecuta parcialmente, presuponiendo que se cumplen las condiciones adecuadas, y se continuará en el futuro hasta que esta finalice.Hay que tener en cuenta que una coroutine no se ejecuta de forma asíncrona. Unity procesa las coroutine en cada frame del juego o animacion.

Con la función StartCoroutime, lanzamos esta función. La sintaxis es StartCoroutime(“Nombre de función”); Cuando en una coroutime se llega a un yield, la ejecución de dicha función se detiene hasta el siguiente frame en la mayoría de los casos. Esto podría usarse para mostrar un efecto a lo largo del tiempo. Esto quiere decir que el script que se muestra abajo provoca que nuestro cubo vaya a una velocidad menor



Conclusiones

Unity 3D es una herramienta muy útil para principiantes una herramienta muy útil, podremos crear cualquier tipo de videojuego para prácticamente cualquier plataforma que existe actualmente. Es un motor indispensable para todos los desarrolladores de videojuegos que no cuentan con muchos recursos, una de las contras que le puedo encontrar es que no nos permite empezar desde unas bases o unas plantillas, por decirlo asi, e ir luego implementando detalles. Si no que debes empezar de cero con cada juego o animación

Bibliografía:

https://docs.unity3d.com/ScriptReference/Transform-position.html

https://docs.unity3d.com/Manual/index.html

http://www.gamedev.es/unity-3d-startcoroutine/