



UNIVERSIDAD NACIONAL DEL LITORAL
FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS



**Tecnicatura en diseño
y programación de videojuegos**

UNL VIRTUAL



Manipulación de objetos en 3D

Unidad 2: Desarrollo de Video Juegos 3D optimizados
para mobile en Unity

Assets Workflow en Unity

Docentes
Jaime Fili
Nicolás Kreiff

Contenidos

Desarrollo de Video Juegos 3D optimizados para <i>mobile</i> en Unity.....	3
<i>Assets Workflow</i> en Unity	3
Manejo de <i>Assets</i> en Unity	3
Creación de un <i>asset</i>	4
Importación de un <i>asset</i>	4
Settings importados del <i>asset</i>	4
Agregar un <i>asset</i> a una escena	4
Combinando diferentes <i>assets</i> juntos	5
Prefabs.....	5
Instanciando los prefabs	6
Actualización de <i>assets</i>	7
Etiquetas	7

Desarrollo de Video Juegos 3D optimizados para *mobile* en Unity

Assets Workflow en Unity Manejo de Assets en Unity

En esta sección se verán los pasos a seguir para utilizar assets en Unity. Si bien estos pasos, son generales, servirán para revisar las funciones básicas. Además, a lo largo del apunte se explicará el uso de una malla 3D.

En todo proyecto de video juegos, los assets (gráficos, mallas 3D, animaciones, entre otros) son una parte muy importante del *pipeline* de trabajo. En este sentido, Unity provee un soporte muy amplio para herramientas de terceros, de modo de dar libertad con respecto a cómo desarrollar e importar dichos assets a los proyectos.

En las siguientes tablas se puede observar los formatos de archivo soportados por la versión actual de Unity:

3D Package Support				
	Meshes	Textures	Anims	Bones
Maya .mb & .ma ¹	✓	✓	✓	✓
3D Studio Max .max ¹	✓	✓	✓	✓
Cheetah 3D .jas ¹	✓	✓	✓	✓
Cinema 4D .c4d ^{1 3}	✓	✓	✓	✓
Blender .blend ¹	✓	✓	✓	✓
modo .bo ²	✓	✓	✓	
Autodesk FBX	✓	✓	✓	✓
COLLADA	✓	✓	✓	✓
Carrara ¹	✓	✓	✓	✓
Lightwave ¹	✓	✓	✓	✓
XSI 5.x ¹	✓	✓	✓	✓
SketchUp Pro ¹	✓	✓		
Wings 3D ¹	✓	✓		
3D Studio .3ds	✓			
Wavefront .obj	✓			
Drawing Interchange Files .dxf	✓			

¹ Import uses the application's FBX exporter. Unity then reads the FBX file.
² Import uses the application's COLLADA exporter. Unity then reads the COLLADA file.
³ Cinema4D 10 has a buggy FBX exporter. Please see [here](#) for workarounds.

Image Formats

- Photoshop .psd and .tiff are imported with layers automatically flattened.
- JPEG, PNG, GIF, BMP, TGA, IFF, PICT and many other image formats are supported.

Supported Audio & Video Formats

- MP3 and Ogg Vorbis .ogg audio files are natively supported, depending on platform. On mobile platforms, audio is converted to MP3 to take full use of hardware decompression.
- AIFF, WAV and most other audio format are supported, ideally suited for sound effects. Optional compression can be configured in the Unity Editor.
- MOD, IT, S3M, XM tracker files are fully supported.
- Ogg Theora video is natively supported.
- Video MOV, AVI, ASF, MPG, MPEG, MP4VIDEO files are transcoded by Unity with a configurable bitrate.

Other File Formats

- XML and text files with .xml and .txt extensions can be referenced at runtime.
- Any other file types, such as RTF and DOC, can be used for project notes and to-do lists.

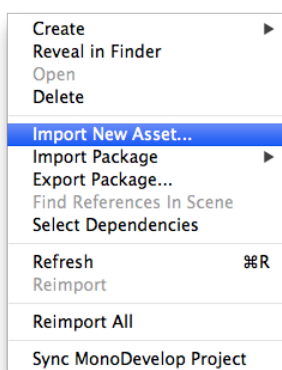
Como se puede observar, la cantidad de formatos es realmente importante y esto constituye una gran fortaleza para el motor.

Para más información con respecto al *asset pipeline* de Unity se puede visitar la siguiente dirección: <http://unity3d.com/unity/editor/importing>.

Creación de un *asset*

Quien lo desee puede utilizar cualquiera de los programas de modelación 3D soportado para crear un *asset*. A efectos de que puedan ser probados la cátedra proveerá también algunos ejemplos.

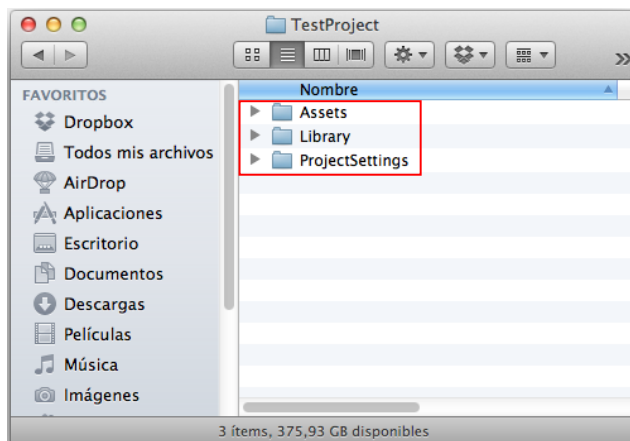
Importación de un *asset*



Existen dos formas de importar un *asset* en un proyecto de Unity. La primera es a través del menú Assets > Import New Asset, como se muestra en la imagen a continuación.

La segunda es importar un *asset* guardado inicialmente dentro de la carpeta Assets del proyecto.

Como se puede apreciar en la siguiente imagen, el proyecto posee tres carpetas principales: Assets, Library y ProjectSettings.



Cuando se abra Unity, el *asset* será detectado e importado dentro del proyecto. Al observar la vista del mismo, se podrá apreciar que el *asset* se encuentra exactamente en donde ha sido guardado.

Settings importados del *asset*

Si se selecciona el *asset* en la vista de proyecto, los seteos (configuraciones) que se han importado para dicho *asset* aparecerán en el *Inspector*. Las opciones que se muestren cambiarán, dependiendo del tipo de *asset* que se haya seleccionado.

Agregar un *asset* a una escena

Para agregar un *asset* a una escena simplemente se debe hacer click en éste y arrastrarlo desde la vista de proyecto hacia la vista de jerarquía o de escena. En el caso de una malla, cuando ésta es arrastrada a la escena se crea un *GameObject* que tiene un componente: *MeshRenderer*. En cambio, si se está trabajando con una

textura o un archivo de sonido se deberá agregar a un *GameObject* que ya exista en la escena o en el proyecto.

Combinando diferentes *assets* juntos

Los *assets* más comunes se relacionan de la siguiente manera:

- Una *Textura* se aplica a un *Material*
- Un *Material* es aplicado a un *GameObject* (con un componente *MeshRenderer*).
- Una *Animación* es aplicado a un *GameObject* (con un componente *Animation*).
- Un *archivo de sonido* es aplicado a un *GameObject* (con un componente *AudioSource*).

Prefabs

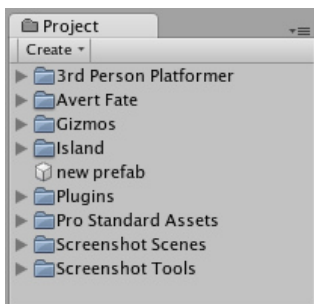
Los *Prefabs* son un tipo de *asset* utilizable en cualquier proyecto de Unity. Básicamente, los *prefabs* son *GameObjects* pre-configurados y almacenados en el proyecto, susceptibles de ser reutilizados cuando hace falta. Un *prefab* puede ser *instanciado* en cualquier escena del proyecto tantas veces como sea necesario.

Es posible pensar un *prefab* como un contenedor capaz de almacenar un *GameObject* con toda su configuración (su estructura completa de *GameObjects* hijos, componentes que posee, etcétera).

Imaginando un ejemplo, se supondrá que un usuario desea hacer un juego de carreras de autos. Seguramente el *asset* de su vehículo tendrá varias partes (ruedas, chasis, luces, etc) pero, además, es probable que se le adjunten varios componentes (scripts) que le ayudarán con tareas como el manejo de la física, el encendido de las luces, la toma de las entradas del usuario, entre otras.

Esta configuración podrá ser realizada en el *Editor* de Unity para la primera pista del juego, por ejemplo, pero si el juego tiene 40 pistas sería realmente muy tedioso replicar “a mano” toda la configuración del vehículo para las pistas del juego. Lo que se deberá hacer será crear un *prefab* con el vehículo e *instanciarlo* en cada pista.

A continuación se verá cómo se crea un *prefab*.



Para ello, primero se deberá crear un *prefab* vacío utilizando el menú *Assets > Create > Prefab*.

Un *prefab* vacío no contiene ningún *GameObject* y no se pueden crear instancias del mismo. Es como un contenedor vacío.

Para *almacenar* un *GameObject* en un *prefab* se deberá arrastrarlo desde el *Hierarchy View* hasta el *prefab* en el *Project View*. Hecho esto, la información del *GameObject* y la de todos sus hijos

habrán sido copiados al nuevo *prefab*.

Como casi todo en Unity, lo que puede hacerse mediante el *Editor* también se puede resolver por código pero, debido a que en la práctica es poco común su utilización, no se detallará aquí cómo hacerlo. Sin embargo, los interesados pueden consultar los métodos

`CreateEmptyPrefab`, `CreatePrefab` y `ReplacePrefab` de la clase `PrefabUtility` en el siguiente sitio:¹ <http://unity3d.com/support/documentation/ScriptReference/PrefabUtility.html>

Instanciando los prefabs

Hay dos formas de crear una *instancia* de un *prefab*. Utilizando el *Editor* de Unity o por código.

Desde el *Editor* de Unity: Para crear una *instancia* de un *prefab* en la escena actual, se arrastra el *prefab* desde el *Project View* hacia la escena o el *Hierarchy View*.

En tiempo de ejecución: Los prefabs son especialmente útiles cuando deseamos instanciar `GameObjects` complejos en tiempo de ejecución. La alternativa a instanciar un *prefab* sería crear el `GameObject` por código, desde cero, cada vez que fuese necesario. Instanciar un *prefab* tiene varias ventajas ante este escenario:

- Es posible instanciar un *prefab* con una sola línea de código. Crear un `GameObject` equivalente desde cero por código involucraría varias líneas de código entre creación e inicialización.
- Se puede crear y/o modificar un *prefab* rápida y fácilmente desde la escena o el inspector sin requerir modificación de código alguna.
- Se puede cambiar el *prefab* que será instanciado sin necesidad de cambiar el código que lo instancia (se verá un ejemplo de esto en breve).

La función que permite instanciar un *prefab* es: ***Instantiate***.

```
static Object Instantiate (Object original, Vector3 position, Quaternion rotation)
```

```
static Object Instantiate (Object original)
```

En la primera, es posible especificar la posición y rotación del objeto instanciado. En cambio, en la segunda se mantendrá la posición y rotación original.

A continuación se mostrará un ejemplo, que luego se analizará en detalle:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class MyShip : MonoBehaviour {
05
06     public GameObject bullet;
07
08     void Update() {
09         MoveShip();
10
11         DoSomeOtherStuff();
12
13         if (Input.GetButtonDown( "MainFire" )) {
14             GameObject b =
15                 Instantiate(bullet, transform.position, transform.rotation);
16             MyBulletScript bulletScript = b.GetComponent<MyBulletScript>();
17             bulletScript.direction = transform.forward;
18         }
19     }
20 }
```

¹ Cabe aclarar a aquellos que quieran hacer alguna prueba al respecto, que la clase `PrefabUtility` sólo puede utilizarse en plugins del *Editor* de Unity.

En este ejemplo se declara el componente *MyShip* que representará una nave capaz de disparar un proyectil.

El proyectil que disparará la nave dependerá del objeto que se vincule a la variable pública *bullet* declarada en la línea 06. Se podrá hacer que la nave dispare un tipo distinto de proyectil sin cambiar ninguna línea de código. Todo lo que habrá que hacer será asociar un objeto distinto a esta variable desde el *Editor* de Unity (arrastrándolo desde el *Project View* hacia la variable en el *Inspector*).

En las líneas 08 a 18 se declara el método *Update* de la supuesta nave.

Las líneas 09 y 11 figuran sólo a modo de ejemplo para representar cosas extra que deberá hacer la nave, como moverse.

Lo interesante está entre las líneas 14 y 16. En la línea 14 ocurrirá la instanciación del proyectil en la posición actual de la nave y con su rotación también actual. Las dos líneas siguientes obtendrán un supuesto componente del proyectil llamado *MyBulletScript* e indicarán la dirección en la cual deberá avanzar.

El método *Instantiate* es muy utilizado para instanciar sistemas de partículas de explosiones, proyectiles, entidades enemigas o cualquier otro objeto complejo.

Actualización de *assets*

Habiendo importado, instanciado y linkeado un *asset* a un *prefab*, se supondrá ahora que se desea editar el *asset* original. Para esto, se deberá hacer doble click en éste, en la vista de proyecto. La aplicación apropiada (desde la cual se generó el *asset*) se iniciará y se podrán hacer los cambios que se deseen. Cuando haya finalizado con la actualización, simplemente se guardará el archivo. Luego, cuando vuelva a Unity, el cambio será detectado y el *asset* re-importado.


Los links del *asset* con un *prefab* serán mantenidos. De esta manera, se verá que el *prefab* ha sido actualizado.

Etiquetas

Siempre es bueno agregar etiquetas a los *assets* si se desea mantenerlos organizados pero esto es opcional, desde ya.

De esta forma, por las etiquetas asociadas a los *assets*, éstos se pueden organizar en el campo de búsqueda, en la vista de proyecto o en el selector de objetos.

Los pasos que se deben seguir para agregar una etiqueta a un *asset* son los siguientes:

- Seleccione el *asset* al que desea agregarle una etiqueta desde la vista de proyecto.
- En el inspector, hacer click en el ícono *Add Label* () si no se tiene ninguna etiqueta asociada al *asset*.
- Si ya se tiene una etiqueta asociada al *asset* entonces simplemente hacer click en el listado de etiquetas.

Como comentario se puede añadir que un *asset* puede tener más de una etiqueta. Para separar o crear etiquetas simplemente se debe presionar espacio o la tecla Enter cuando se estén escribiendo las etiquetas para el *asset*.