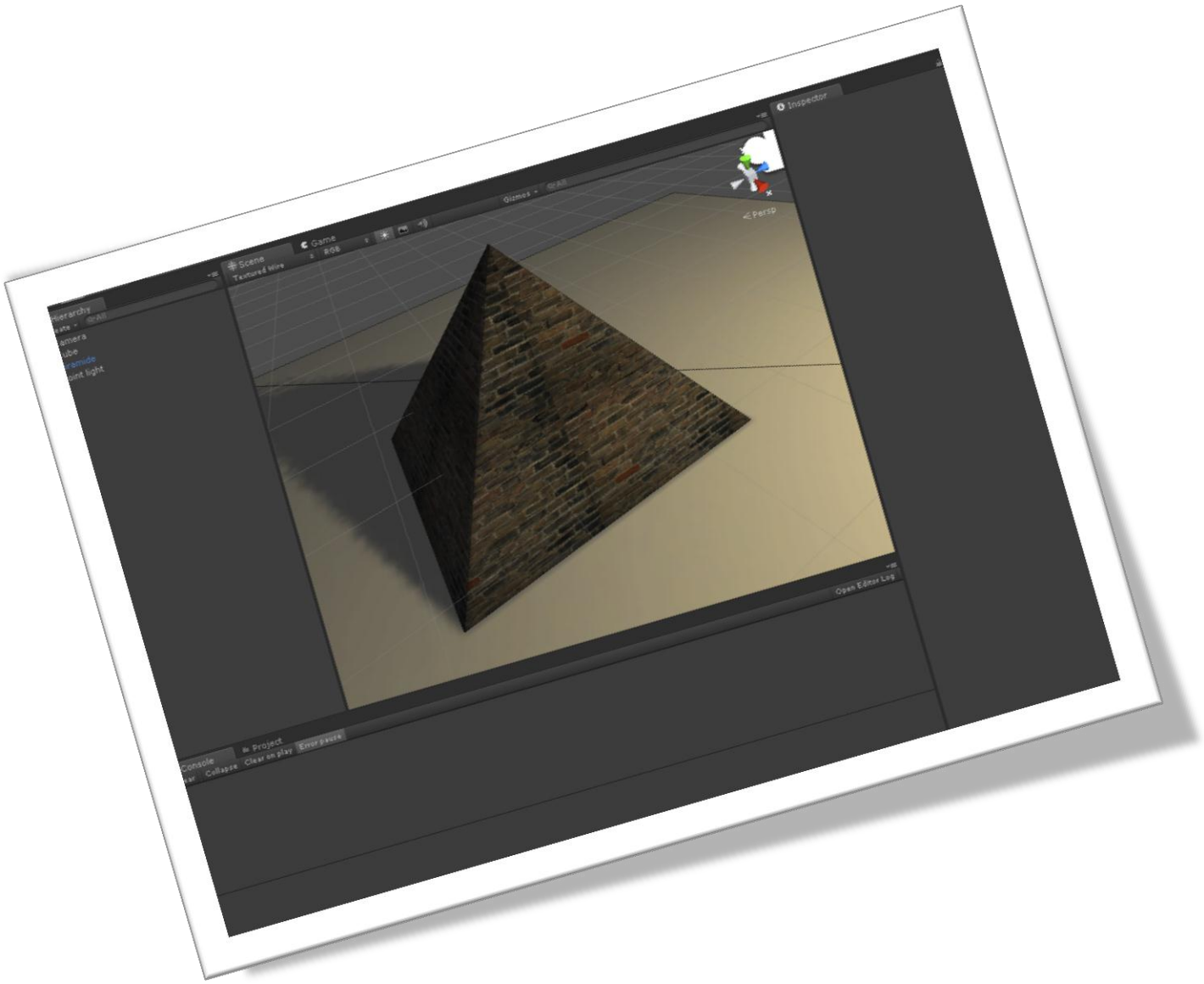
Unity 3d shader series

En este tutorial aprenderemos a usar los shaders de unity 3D

Nacho Cortizo Pol



http://thegameartist.wordpress.com/

Índice

Introducción

> Hoy en día, a la hora de crear un videojuego , una de las partes a la que los desarrolladores le dedican más tiempo es al resultado gráfico, ya que es una de las partes que los usuarios “criticarán” primero.

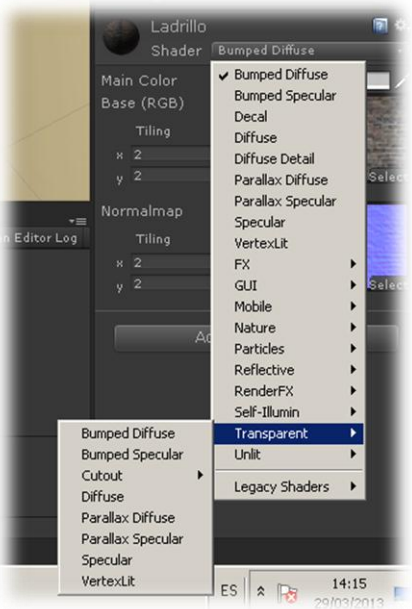
Los shaders, podríamos decir que son los intermediarios entre el motor gráfico y la GPU , o sea, que le dicen a nuestra tarjeta gráfica qué es lo que tiene que mostrar.

Los shaders se utilizan para muchas cosas, para partículas como humo o fuego, para materiales, efectos especiales…

En Unity tenemos shaders básicamente en todo, los modelos, los terrenos, partículas, skyboxes… En fin, todo lo que tiene un material está controlado por shaders.

Aquí viene un tema de confusión, un material , en Unity, es lo que agregamos a nuestros modelos para que tengan ciertas texturas pero debemos saber que , materiales y shaders trabajan conjuntamente.

El shader ,sería el tipo de material y dentro de cada shader tenemos distintas características.



Unity viene con más de 80 shaders “de fábrica”…

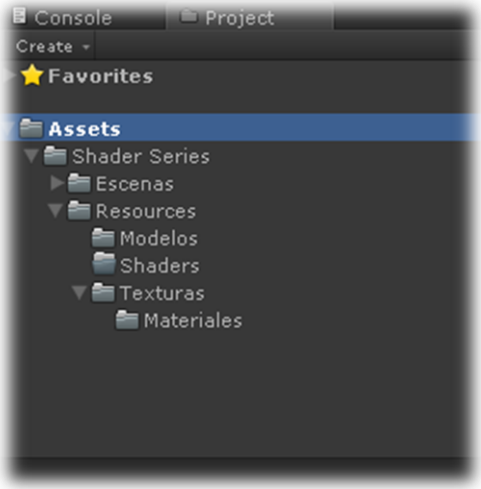
Queda decir , que los shaders, son pequeñas unidades de código precompiladas en el motor gráfico.

Inspeccionando los shaders de Unity

Con este PDF, viene un paquete de Unity (“ShaderSeries.unitypackage”) ,importarlo dentro de un proyecto vacio…

Como veis en la imagen, tenemos una carpeta principal Shader Series , y luego otras dos : Escenas (contiene la escena) y Resources (en ella están los diferentes elementos que usaremos durante el tutorial).

En las distintas carpetas hay varias texturas y materiales creados.

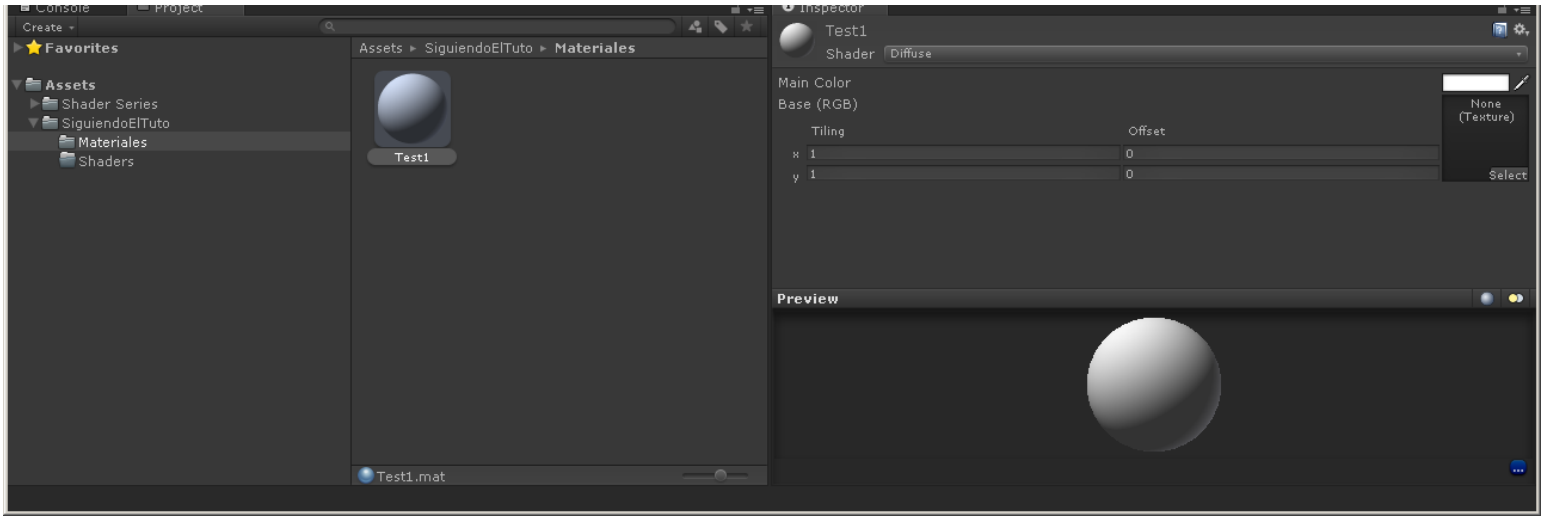


Para comenzar crear una carpeta para vosotros.

En ella crear dos carpetas una con el nombre de materiales y la otra con el nombre de shaders.

Procederemos ahora a crear un material dentro de la carpeta “Materiales” con el nombre de Test1.

Los Materiales vienen predefinidos con el Shader “Diffuse” , que es el más sencillo de todos los shaders, en el podremos modifica el color principal así como la textura…



En la imagen tenemos el material creado.

Analicemos las distintas partes en el Inspector:

Shader:este es el shader que está aplicado al material, si pulsamos podremos ver los distintos shaders que tenemos disponibles.

Main Color: es el color principal del material, si no tenemos una textura el material será de dicho color.

Base(RGB):esta es la textura “base” del material , podremos controlar dos factores el tilling(repetición) y el offset.

Probar con distintos colores para ver el resultado aplicándoselo a un cubo por ejemplo o a la pirámide que viene en ***ShaderSeries->Resources->Modelos->Piramide.fbx***

Echemos un vistazo a otros shaders básicos.

En el material que acabamos de crear seleccionemos “Bumped Diffuse”.

Nos aparecerá una textura más con el nombre de “Normal Map”.

Este tipo de shaders se utiliza cuando queremos darle a las texturas un toque más realista, lo que conseguimos con esto es que, parezca que las texturas están en 3D.

¿Cómo conseguir un normal map?, es muy sencillo, una vez importada tu textura, procedes a duplicarla , le das click y donde pone texture type le dás a “Normal Map”, si le das a “Apply” verás como la textura cambia y aparece de un color azulado con puntitos.

Podemos ajustar el nivel de “normalizado” de nuestra textura, para ello variamos los valores de “Bumpiness” los cuales es recomendable dejarlos por debajo de 0.15 para conseguir un mayor realismo…

Veamos la diferencia entre un modelo con un shader Diffuse y otro con Bumped Diffuse:

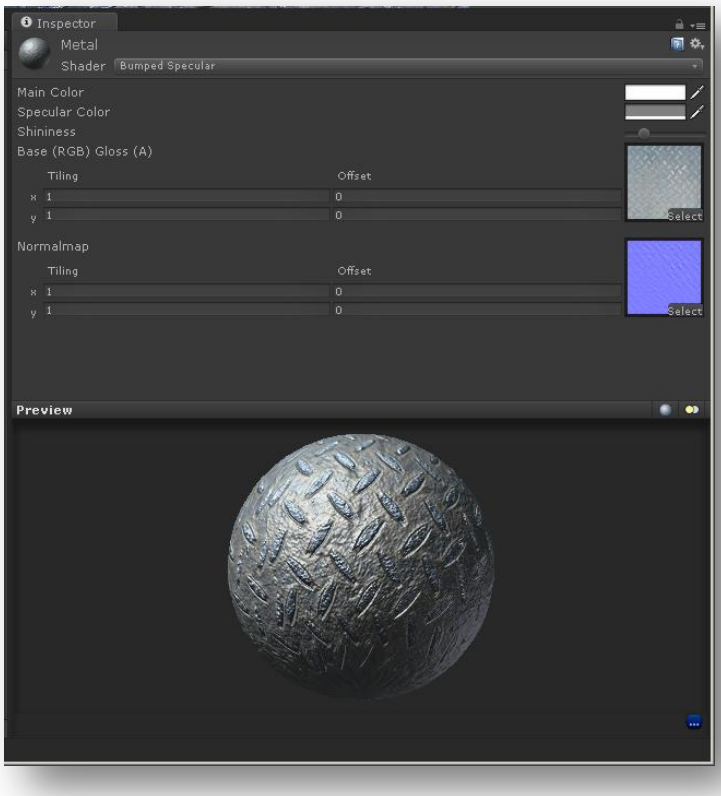
En la imagen , a la izquierda con el normal map y en la derecha con el diffuse, apreciamos que la que tiene normal map parece estar en 3D…

Sigamos con otro shader, en la pestaña de shader tenemos uno que se llama “Bumped Specular” , este , como su nombre nos indica se puede utilizar a la hora de aplicar materiales a objetos reflectantes, pinturas de coches,metal,agua etc…

Procedemos a crear un material ahora en nuestra carpeta de materiales, le aplicamos el shader “Bumped Specular” en el folder ***ShaderSeries->Resources->Texturas*** , hay dos texturas metálicas, se las aplicamos a nuestro material…

Obtendremos algo semejante a lo que aparece en la foto.

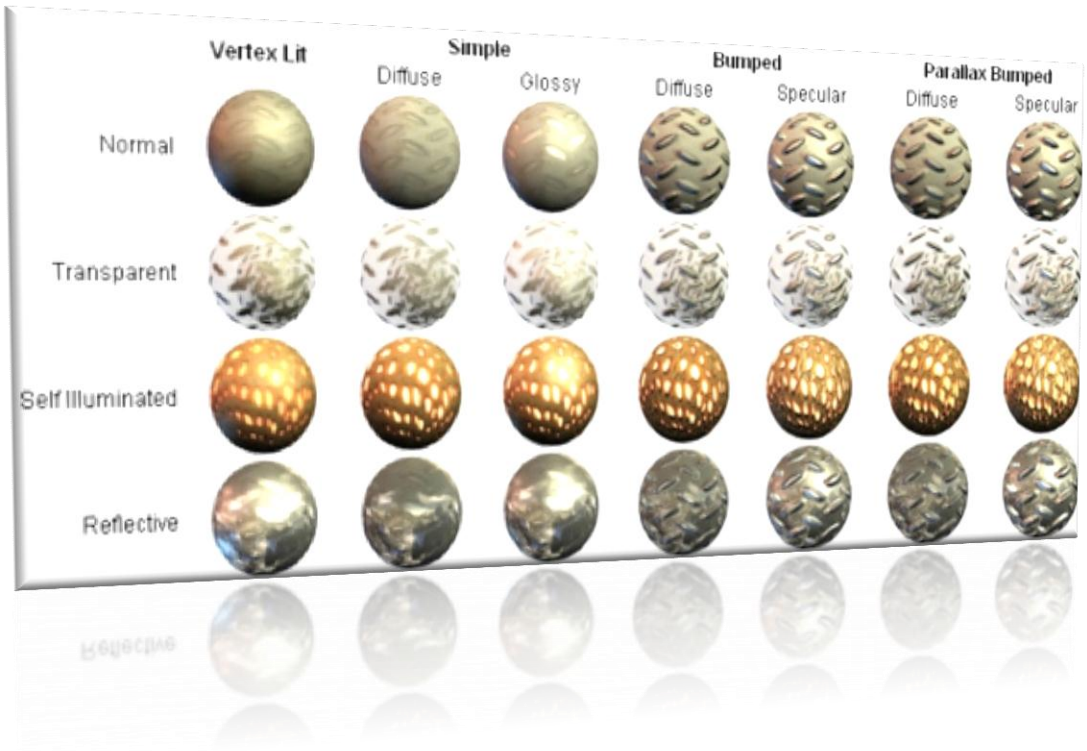
Con este shader, nos aparecen dos nuevas variables, Specular Color (color reflejado) y Shininess (la cantidad del reflejo).



Podéis ir probando para conseguir distintos resultados…

Estos shaders son los más básicos y utilizados , podéis ir experimentado con otros y comprobando su utilidad…

Aquí os dejo una imagen con diferentes shaders aplicados a texturas:



Creando nuestro primer shader

Hay tres tipos distintos de shaders:

Surface Shaders: estos shaders están afectados por la iluminación y las sombras.

Vertex and Fragment Shaders: estos los utilizaremos cuando no tengamos que tener en cuenta sombras e iluminación para nuestros shaders.

Fixed Function Shaders: utilizaremos estos shaders cuando tengamos que tartar con hardwares viejos.

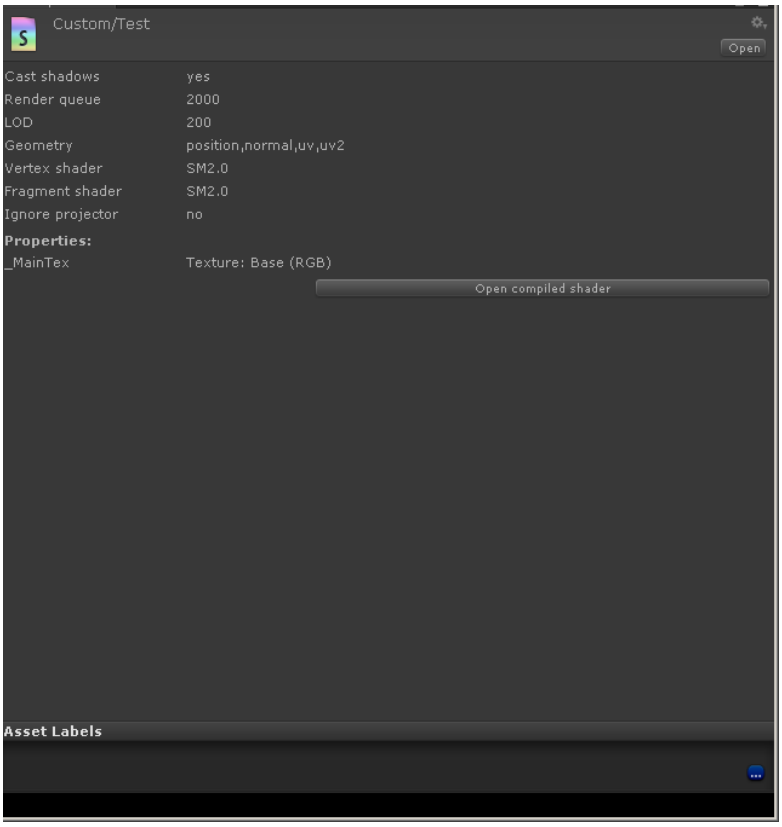
En este tutorial, trabajaremos sobre Vertex and Fragment Shaders, en otras gruías podremos analizar más a fondo los otros dos tipos.

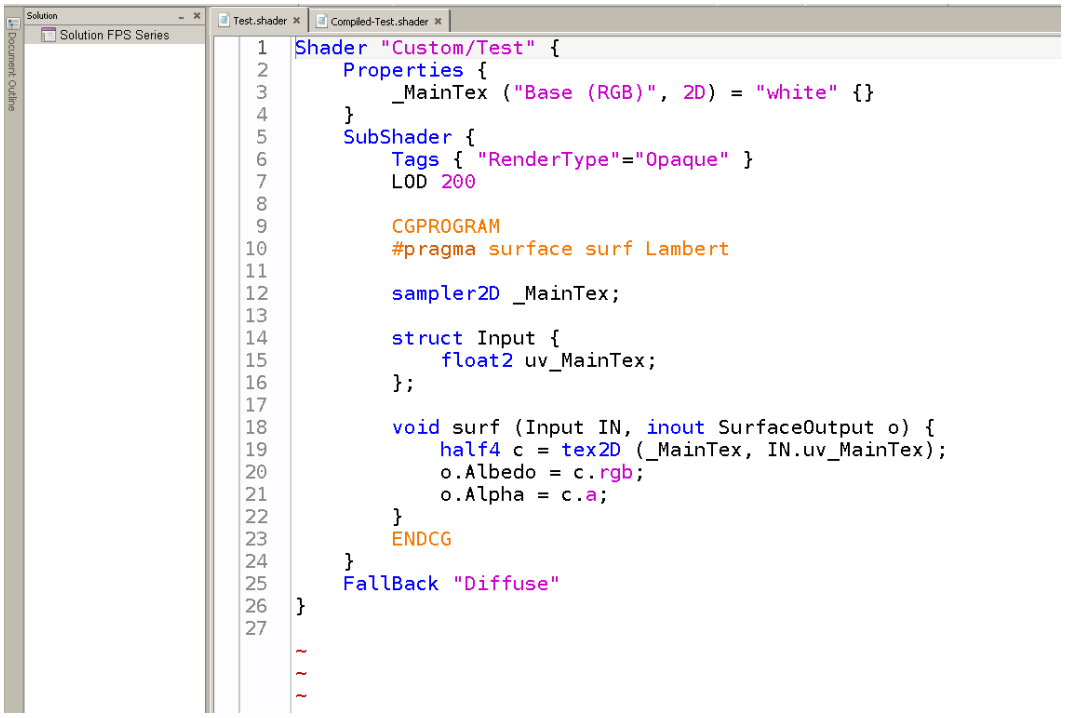
En la carpeta que hemos creado antes (“Shaders”), le damos click derecho , create , Shader… Le ponemos el nombre que queramos ya que el nombre del archivo no se relaciona con el nombre que veremos en el explorador (en la pestañita).

Le damos doble click para abrir el shader en MonoDevelop o el programa que tengamos ligado a Unity.

En Unity si pulsáis en Open compied Shader , os abrirá un archivo en el MonoDevelop muy largo, este es el resultado de la compilación de nuestro shader y no debemos modificarlo.

Aquí tenemos el shader en el Inspector.





Aquí el código del shader en el MonoDevelop.

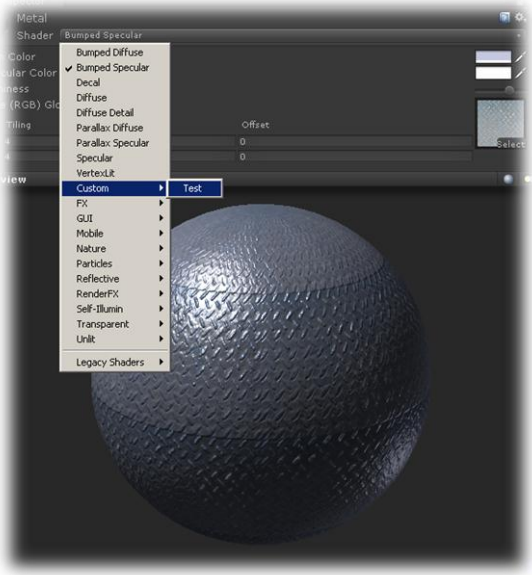
Si tenéis algún conocimiento en programación, veréis que la estructura del shader está englobada en:

```
Shader "Custom/Test" {
}
```

“Custom/Test” sería la “carpeta” donde estaría el shader y el nombre del mismo, como dije antes el nombre no tiene nada que ver con el que aparece en el explorador ya que lo podemos cambiar…

Procederemos a eliminar todo lo que está dentro del bloque principal (“{}”).

Comenzaremos ahora con un shader sencillo.




```
Shader "Custom/Test" {
 Properties {
      _Color ("Color", Color) = (1,1,1)
   }
   SubShader {
      Pass {
         Material {
            Diffuse [_Color]
         }
         Lighting On
      }
   }
}
```

Analicemos este shader, en primer lugar tenemos Properties{..} , aquí dentro es donde irán las variables que veremos en el inspector.

```
_Color ("Color", Color) = (1,1,1)
```

_Color(…):es el nombre de la variable.

(“Color”,…):es el nombre que se mostrará en el inspector.

(…,Color):es la propiedad o el tipo de variable

=(1,1,1): es el valor por defecto , recordemos que está en la escala de porcentajes de RGB, en este caso 100% de todos por lo tanto sería negro.

Despues tenemos el SubShader{} y dentro del SS el Pass{}… Material{}, dentro de el Diffuse[_Color],esto significa que el material será la variables “_Color”.

Finalmente tenemos un nuevo factor Lighting on.

Esto suena muy embrolloso viendo así pero con el código alado se entiende bien.

Como he dicho, este es un shader muy sencillo, veamos ahora uno más complejo:

```
Shader "Custom/Test2" {
   Properties {
      _Color ("Main Color", Color) = (1,1,1,0.5)
      _SpecColor ("Spec Color", Color) = (1,1,1,1)
      _Emission ("Emmisive Color", Color) = (0,0,0,0)
      _Shininess ("Shininess", Range (0.01, 1)) = 0.7
      _MainTex ("Base (RGB)", 2D) = "white" { }
   }

   SubShader {
      Pass {
         Material {
            Diffuse [_Color]
            Ambient [_Color]
            Shininess [_Shininess]
            Specular [_SpecColor]
```

```
            Emission [_Emission]
        }
        Lighting On
        SeparateSpecular On
        SetTexture [_MainTex] {
            constantColor [_Color]
            Combine texture * primary DOUBLE, texture * constant
        }
    }
  }
}
```

Ahora veamos las subestructuras dentro de los shaders

```
Shader "Custom/Test3" {
   Properties { /* ...propiedades... }
   SubShader {
         // ...PS3 y PC..
   }
   SubShader {
         // ...todo, desde el PC del abuelo hasta la Nintendo Color…
    }
  }
```

Dentro de los shaders hay subunidades (“SubShader”), los cuales tienen distintas definiciones del shader en global.

Tenemos que recordar que lo que hace el shader es decir a la GPU como mostrar tal cosa, no es lo mismo la GPU de una PS3 , que una de un PC o que la de un IPod…

Por esto mismo hay diversas unidades y dento de cada una se trata el shader de manera distinta adecuándose a las distintas GPU´s .

Más info:
http://docs.unity3d.com/Documentation/Components/SL-SubShader.html

Dentro de los SubShader tenemos otra unidad , los Passes , es necesario siempre dentro de cada SubShader tener al menos un Pass , ya que estos controlan el renderizado de los objetos.

```
Pass {
   Material {
      Diffuse [_Color]
      Ambient [_Color]
      Shininess [_Shininess]
      Specular [_SpecColor]
      Emission [_Emission]
   }
   Lighting On
   SeparateSpecular On
   SetTexture [_MainTex] {
      constantColor [_Color]
      Combine texture * primary DOUBLE, texture * constant
   }
}
```

Dentro del Pass{} lo primero que hacemos es definir el material, para ello, asignamos a los distintos valores nuestras variables.

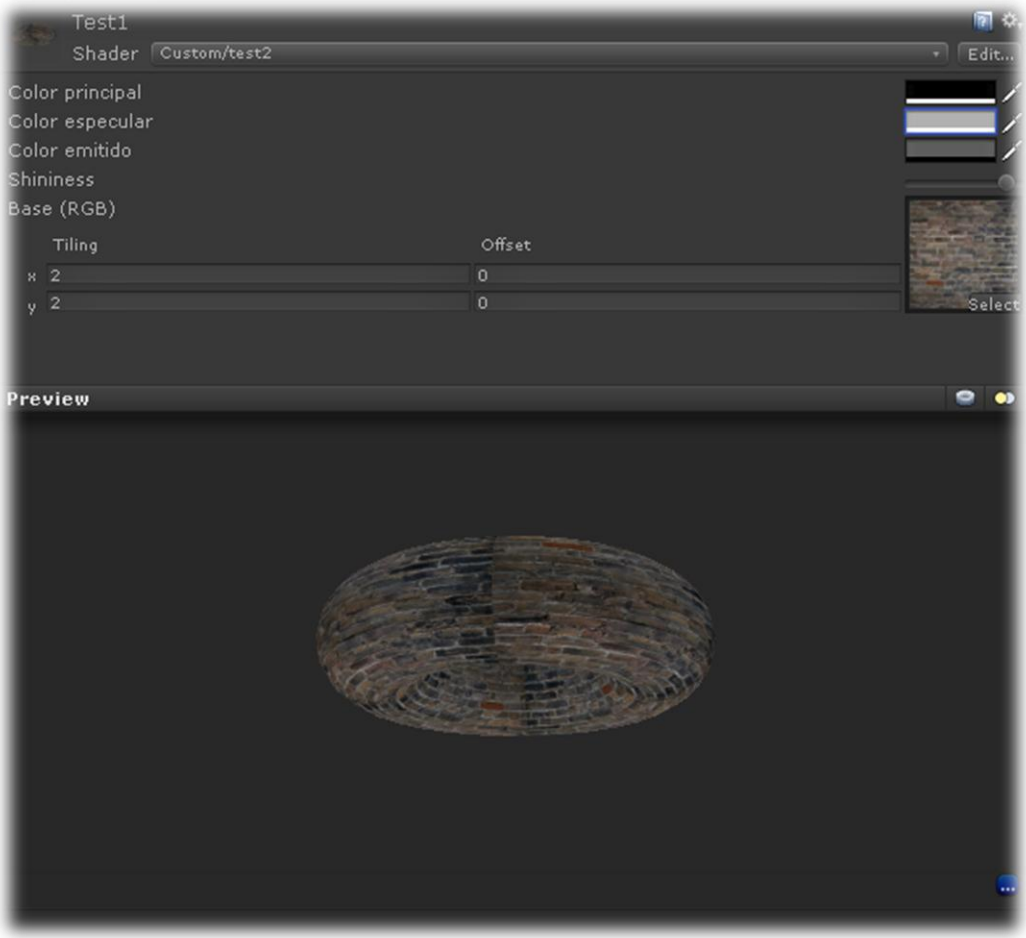
Diffuse[_Color] -> quiere decir que le aplicamos al Diffuse la variable _Color(“Main Color”,Color) = (1,1,1)

También aplicamos distintos factores de la iluminación así como el ajuste especular (“SeparateSpecualr”).

En la última parte se añade la textura y se modifica con los valores del color y la refracción.

Si guardamos el shader,y nos vamos a Unity, podemos seleccionar el material (test1) que creamos antes y aplicarle nuestro shader.(“Custom/Test2”).

Aquí vemos el resultado con nuestro shader.



Implementación de cristal

Como última parte del tutorial, veremos como implementar un shader para crear cristales.

El shader os lo paso en la carpeta Shaders, dentro de los resources.

El código es el siguiente:

```
Shader "Transparent/Glass" {
  Properties {
    _Color ("Color", Color) = (1,1,1,1)
    _SpecColor ("Specular Color", Color) = (1,1,1,1)
    _Shininess ("Specular Falloff", Range (0.01, 1)) = 0.7
    _ReflectColor ("Reflection Color", Color) = (1,1,1,0.5)
    _MainTex ("Main Texture", 2D) = "white" {}
    _Cube ("Reflection Cubemap", Cube) = "_Skybox" { TexGen CubeReflect }
  }

  Category {
    Tags {"Queue"="Transparent"}

    SubShader {

      Pass {
        Cull Off
        ZWrite Off
```

```
ZTest Less
Lighting On
SeparateSpecular On
Blend SrcAlpha OneMinusSrcAlpha
AlphaTest Greater 0.01



Material {
   Diffuse [_Color]
   Ambient [_Color]
   Shininess [_Shininess]
   Specular [_SpecColor]
}



//Reflection
SetTexture [_Cube] {
   ConstantColor [_ReflectColor]
   combine texture * constant alpha, texture
   Matrix [_Cube]
}

//Reflection illumination
SetTexture [_Cube] {
   ConstantColor [_ReflectColor]
```

```
            combine constant * constant alpha - previous, previous
            Matrix [_Cube]
        }

        //Texture
        SetTexture [_MainTex] {
            ConstantColor [_Color]
            combine texture +- previous, constant
        }

        //Texture illumination
        SetTexture [_MainTex] {
            ConstantColor (1,1,1,0.5)
            combine previous * primary double , previous
        }

    }


    }//End of Subshader

    Fallback "Diffuse"

}//End of Category

}//End of Shade
```

En la carpeta de Modelos , hay un marco, si queréis añadirlo a la escena y aplicarle el material madera.

Ahora, crear un plano y ajustarlo para que quede en el interior del marco.

Procederemos ahora a crear nuestro material en la carpeta materiales, le asignamos nuestro shader (“Transparent/Glass”)…

Como podéis ver nos aparece un nuevo elemento:

Un cubemap, en Unity los cubemap son elementos imaginarios que se utilizan para reproducir reflexiones.

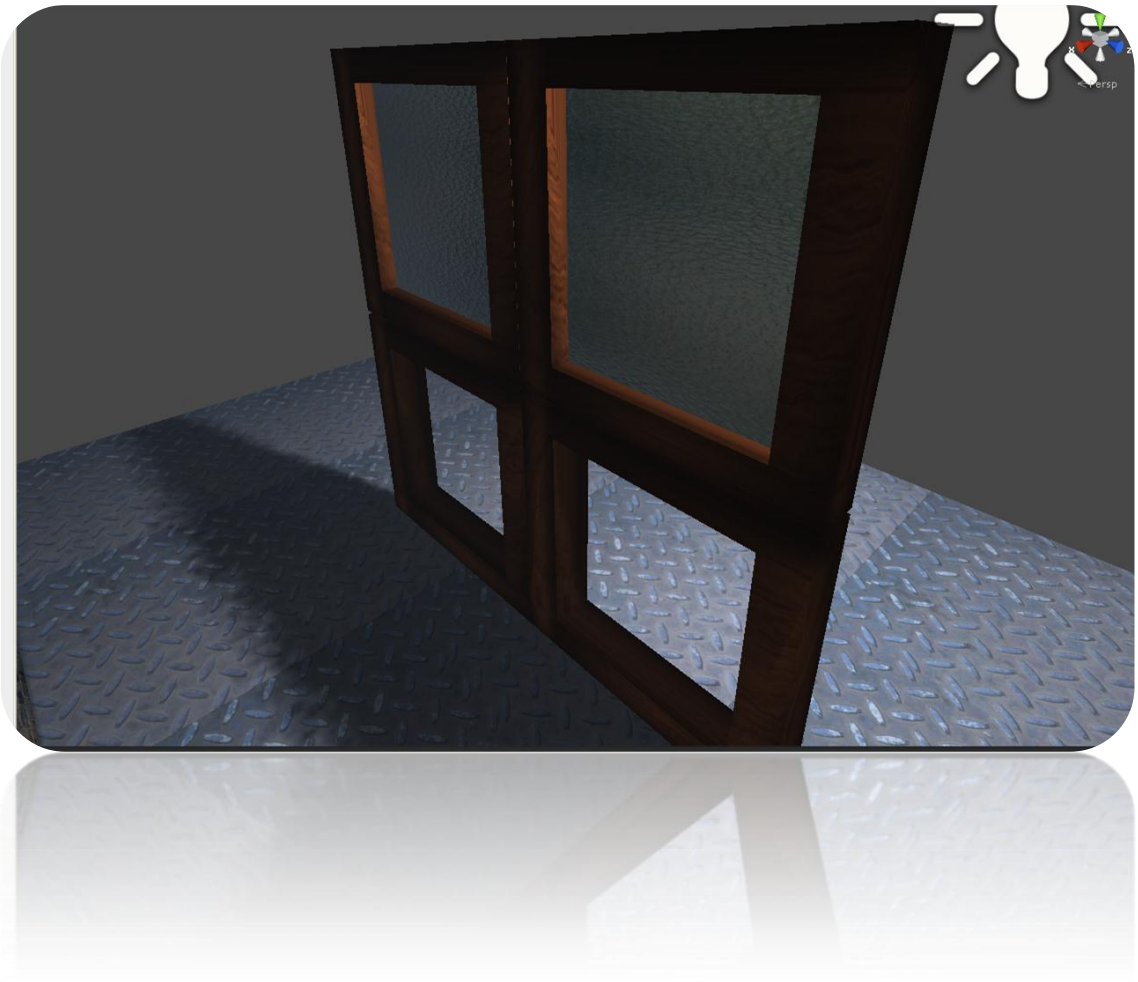
Nos los tenemos que imaginar como un pequeño cubo en el que hay texturas que se reflejarán, podemos añadir desde un cielo o un paisaje…

Para crearlo simplemente dar clik derecho y create-> cubemap.

Podéis dejarlo como está o buscar unas texturas (las de un skybox valen) para que quede mejor, añadimos entonces nuestro cubemap al material glass.

He añadido una textura de cristal para probar, añadirla también.

Finalmente asignamos nuestro material al plano que creamos antes…



Esto seria lo que nos quedaría.

Bueno hasta aquí ha llegado esta guía, espero que os haya servido.

Cualquier duda o problema me podéis encontrar en la comunidad de Unity Spain (piluve)o en nacocpol@gmail.com

Si distribuís este tutorial recordar en nombrarme , y NO lo modifiquéis.