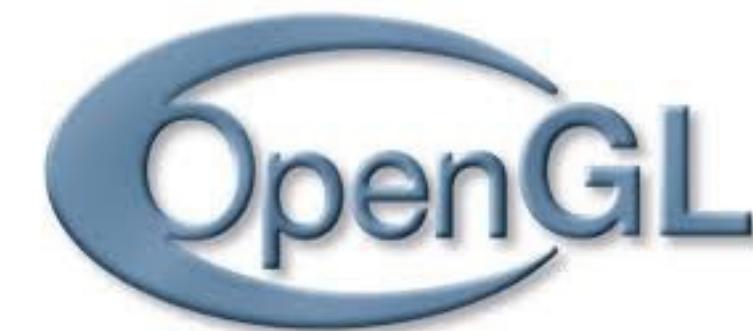


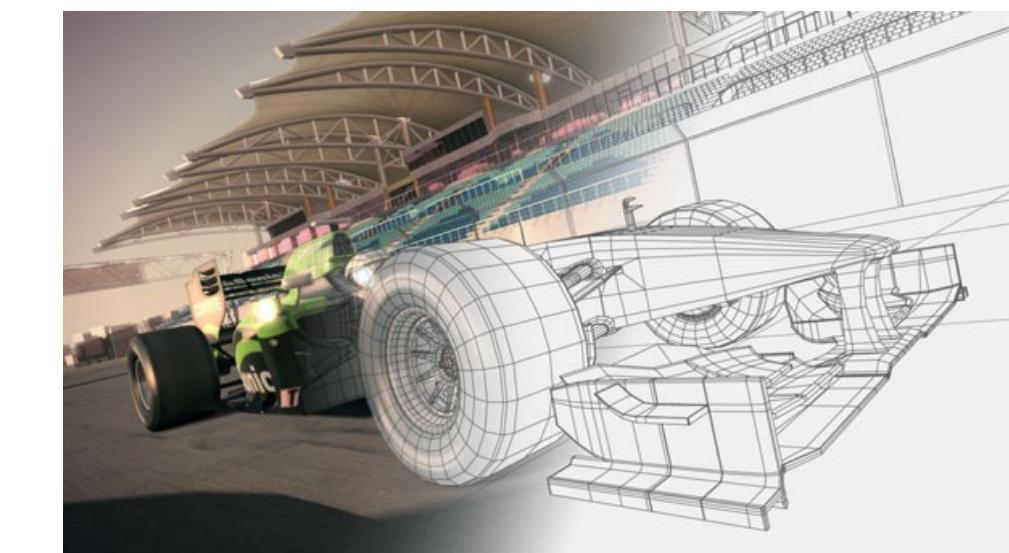
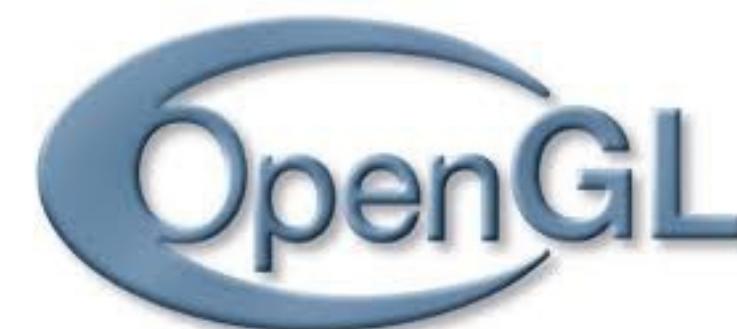
# Introducción a OpenGL/ES en Xamarin con UrhoSharp

Xamarin.Forms  
OpenGL/ES™

# ¿QUE ES OPENGL?



# ¿QUE ES OPENGL?



**OpenGL** (**O**pen **G**raphics **L**ibrary) es una especificación estándar que define una [API](#) multilenguaje y [multiplataforma](#) para escribir aplicaciones que produzcan gráficos [2D](#) y [3D](#). La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos. Fue desarrollada originalmente por [Silicon Graphics Inc. \(SGI\)](#) en [1992<sup>2</sup>](#)

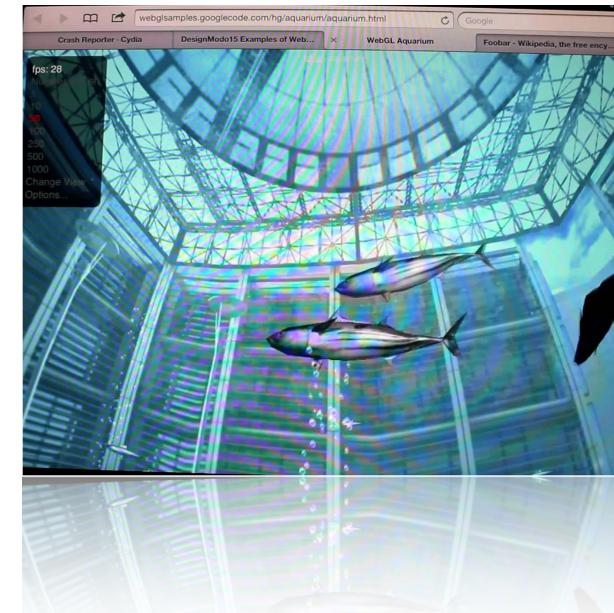
# ¿QUE ES OPENGL/ES?



**OpenGL ES** (**OpenGL** for **E**mbedded **S**ystems) es una variante simplificada de la **API** gráfica **OpenGL** diseñada para **dispositivos integrados** tales como **teléfonos móviles**, **PDA**s y **consolas de videojuegos**.

La define y promueve el **Grupo Khronos**, un **consorcio** de empresas dedicadas a hardware y software gráfico interesadas en APIs gráficas y multimedia.

# ¿QUE ES WEBGL?



**WebGL<sup>1</sup>** es una especificación estándar que está siendo desarrollada actualmente para mostrar [gráficos en 3D en navegadores web](#). El WebGL permite mostrar gráficos en 3D acelerados por hardware (GPU) en [páginas web](#), sin la necesidad de [plug-ins](#) en cualquier plataforma que soporte [OpenGL 2.0](#) u [OpenGL ES 2.0](#). Técnicamente es un [API](#) para [javascript](#) que permite usar la implementación nativa de OpenGL ES 2.0 que será incorporada en los navegadores. WebGL es gestionado por el consorcio de tecnología sin ánimo de lucro [Khronos Group](#).

# OPENGL/ES EN ANDROID

## OpenGL ES

Android incluye compatibilidad con gráficos 2D y 3D de alto rendimiento con Open Graphics Library (OpenGL®), específicamente, la API de OpenGL ES. OpenGL es una API multiplataforma de gráficos que especifica una interfaz de software estándar para hardware de procesamiento de gráficos 3D. OpenGL ES es una versión de la especificación OpenGL diseñada para dispositivos incorporados. Android admite varias versiones de la API de OpenGL ES:

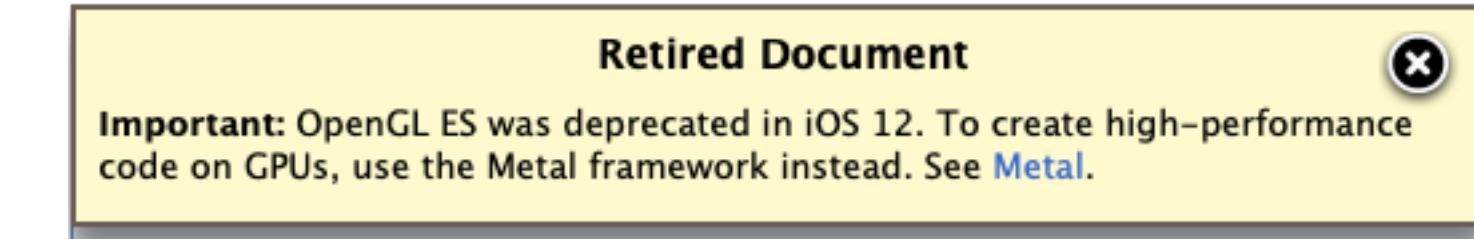
- OpenGL ES 1.0 y 1.1: Esta especificación de API es compatible con Android 1.0 y versiones posteriores.
- OpenGL ES 2.0: Esta especificación de API es compatible con Android 2.2 (API nivel 8) y versiones posteriores.
- OpenGL ES 3.0: Esta especificación de API es compatible con Android 4.3 (API nivel 18) y versiones posteriores.
- OpenGL ES 3.1: Esta especificación de API es compatible con Android 5.0 (API nivel 21) y versiones posteriores.

**Precaución:** La compatibilidad de la API de OpenGL ES 3.0 en un dispositivo requiere una implementación de esta canalización de gráficos que proporciona el fabricante del dispositivo. Es posible que un dispositivo con Android 4.3 o versiones anteriores *no admita* la API de OpenGL ES 3.0. Para obtener información sobre cómo comprobar qué versión de OpenGL ES es compatible durante el tiempo de ejecución, consulta [Cómo averiguar la versión de OpenGL ES](#).

# OPENGL/ES EN IOS

## OpenGL ES

EN iOS encontraremos que la información sobre OpenGL dice lo siguiente. [link](#)



La razón por la que apple declara depreciado es porque OpenGL fue migrado a Metal

# METAL EN IOS



Metal  
Framework

## ¿Que es el Metal Framework?

Metal es un Api que te da acceso al GPU, lo que te permite maximizar los gráficos y potenciar tus aplicaciones de iOS 8.

Metal es una API simplificada, para shaders precompilados, y da apoyo para un uso eficiente de la Multitarea "multi-threading",

Metal puede llevar tus gráficos al siguiente nivel de rendimiento y capacidad.

# OPENGL EN XAMARIN

Xamarin tiene acceso a OpenGl desde la plataforma nativa

Por lo que en Android usara las librerías OpenGL/ES

Y

En iOS. Se tendrá que usar Metal Framework

# OPENGL EN XAMARIN FORMS

Xamarin Forms tiene acceso a OpenGL por la clase OpenGLView

<https://docs.microsoft.com/es-es/dotnet/api/xamarin.forms.openglview?view=xamarin-forms>



# URHOSHARP EN XAMARIN FORMS

## ¿Qué es UrhoSharp?

UrhoSharp es un motor 3D eficaz para los desarrolladores de Xamarin y .net. En la [se explica más acerca de la biblioteca UrhoSharp](#) y en [estas notas](#) se describe cómo programar las escenas y las acciones.

UrhoSharp se puede usar para representar gráficos en Xamarin.Forms aplicaciones.



# ¿QUE ES URHO ?

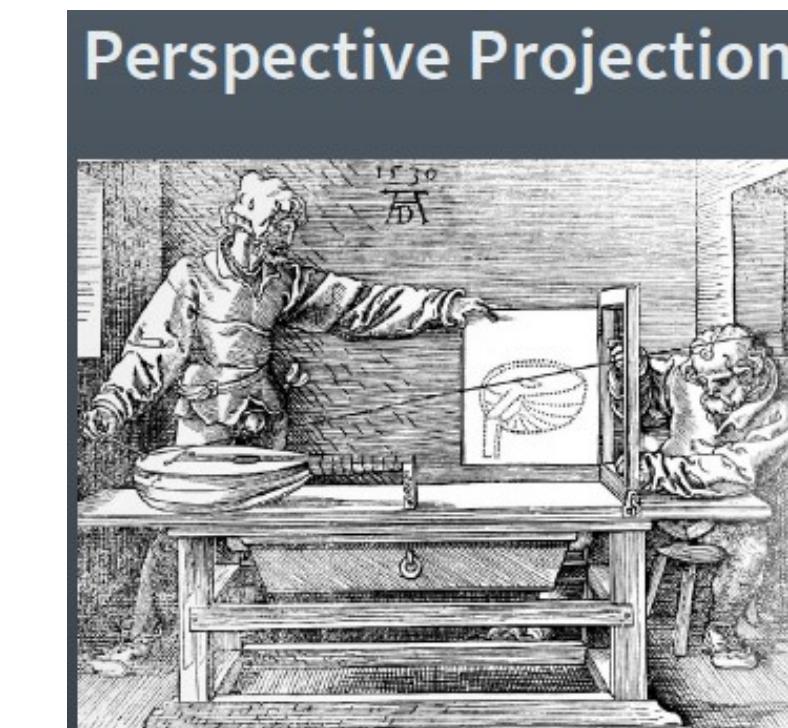
## ¿Qué es Urho3D?

Urho3D es un motor de juegos 2D y 3D multiplataforma ligero y gratuito implementado en C ++ y lanzado bajo la licencia MIT. Muy inspirado por OGRE y Horde3D.

<https://urho3d.github.io>

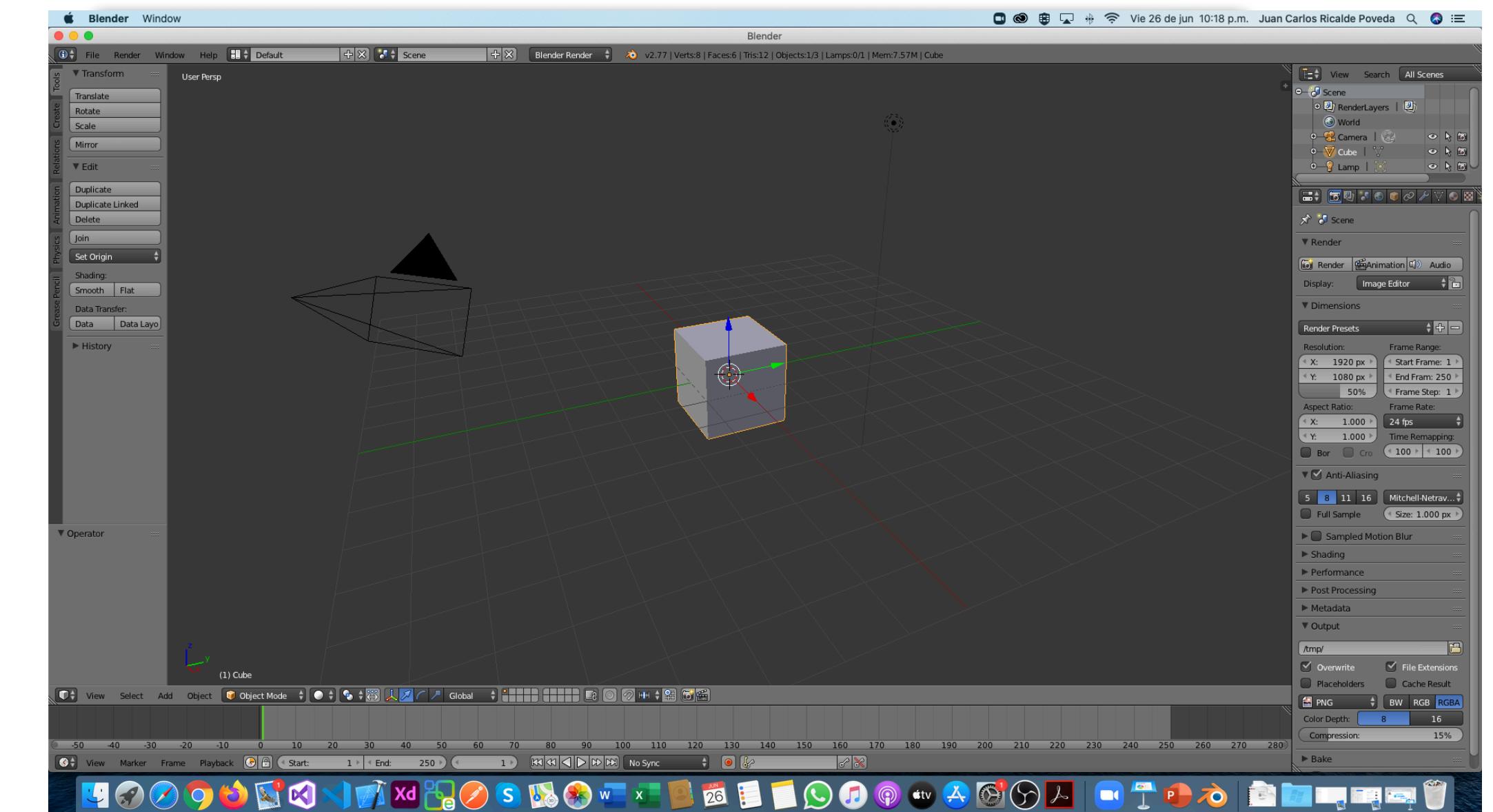
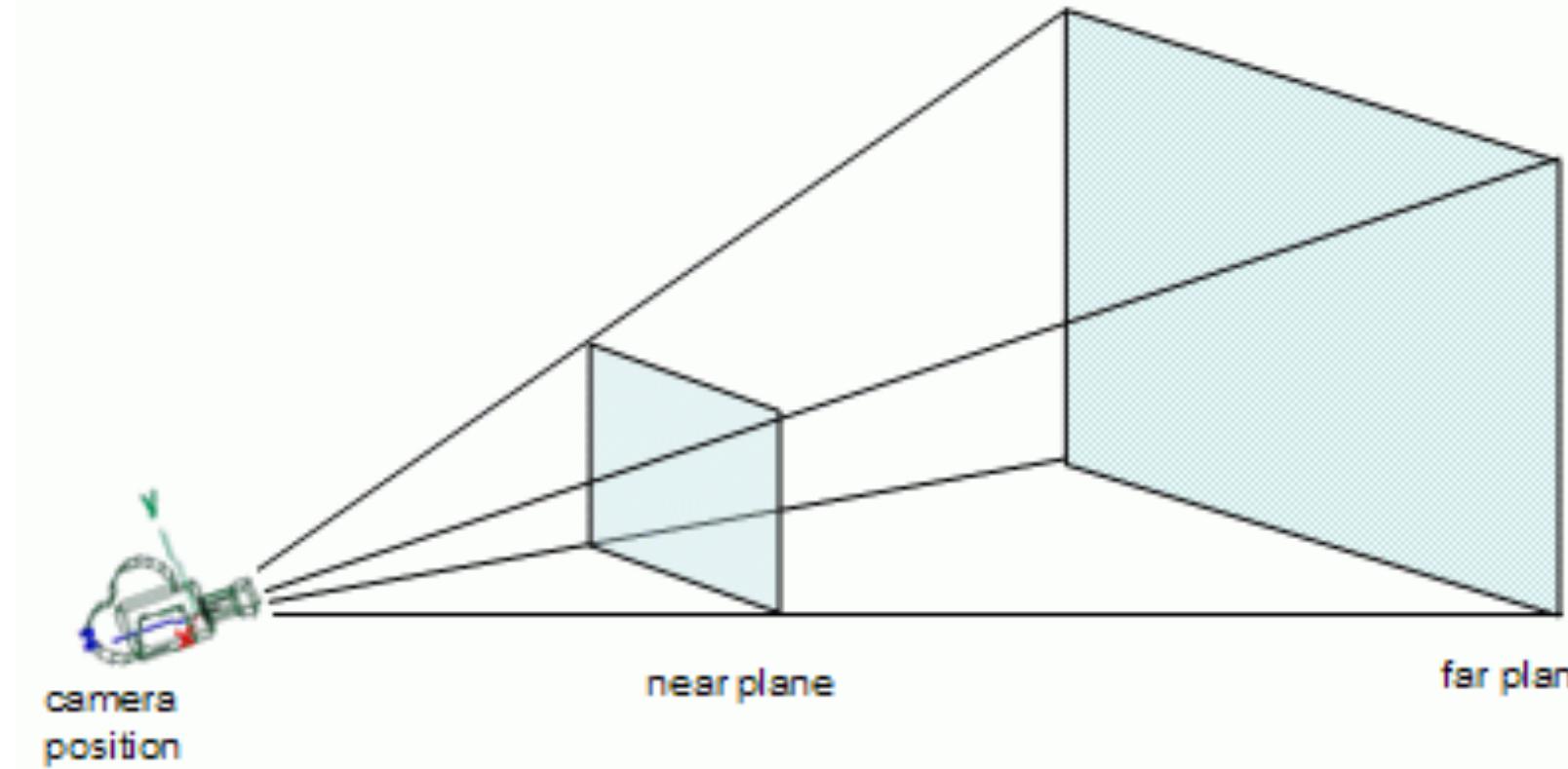
# ¿QUE NECESITO CONOCER PARA USAR OPENGL?

Empezamos por crear la vista de la escena y la escena  
Este es el punto inicial para generar una vista 3D en el GPU



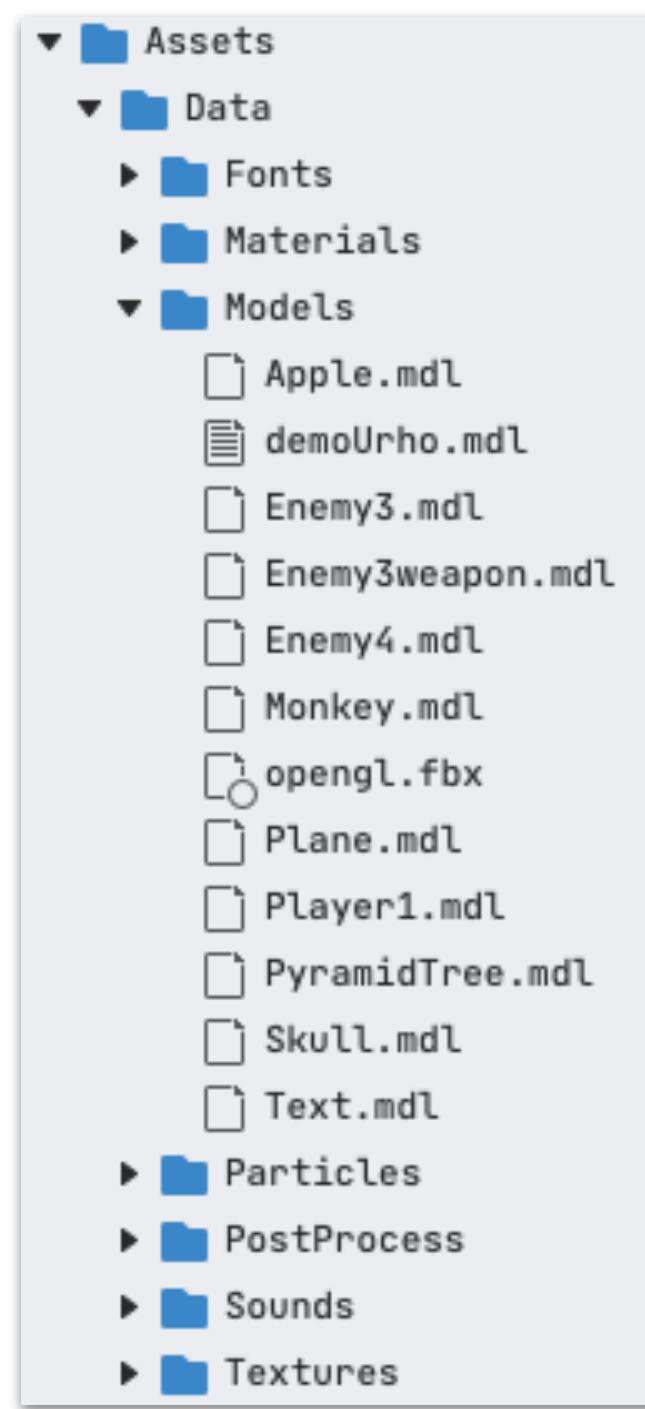
# CONCEPTOS 3D EXPLICADO CON BLENDER

Configuramos la app para que ponga la camara y las luces



# CONCEPTOS 3D EXPLICADO CON BLENDER

Traemos un modelo de los assets  
En Urho se maneja en Assets la carpeta Data:

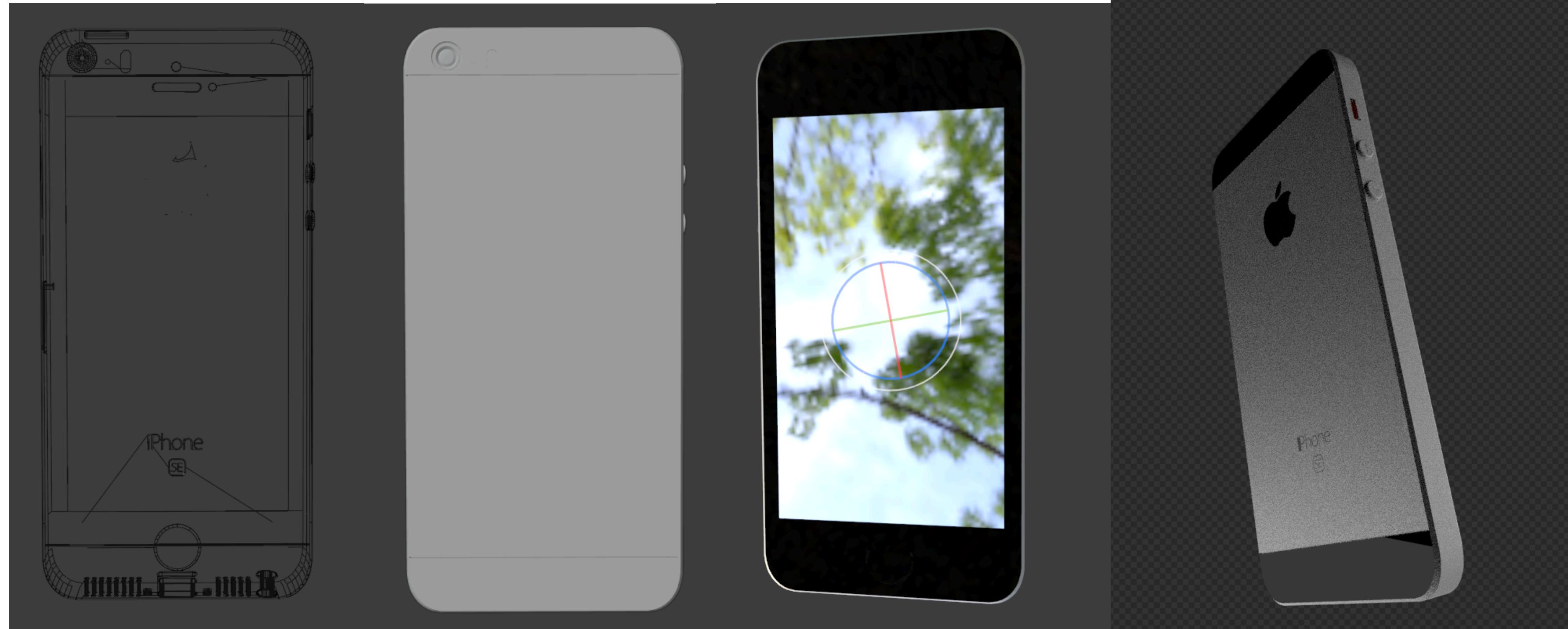


Las carpetas que usa Urho son:

- Fonts. Fuentes para su uso en la escena
  - Materials. Definiciones de materiales para los modelos
  - Particles. Definiciones de partículas ejemplo Fuego, Lluvia, etc.
  - PostPocess. Aqui van los codigos de shaders y filtros
  - Sounds. Carpeta de sonidos
  - Textures. Texturas usadas en las escenas
- 
- \* todos los assets deben ser para su uso dentro de Urho
  - \* En este demo trabajare con el Modelo Text.mdl que cree en Blender

# CONCEPTOS 3D EXPLICADO CON BLENDER

¿Que es un modelo 3D?



Alambre

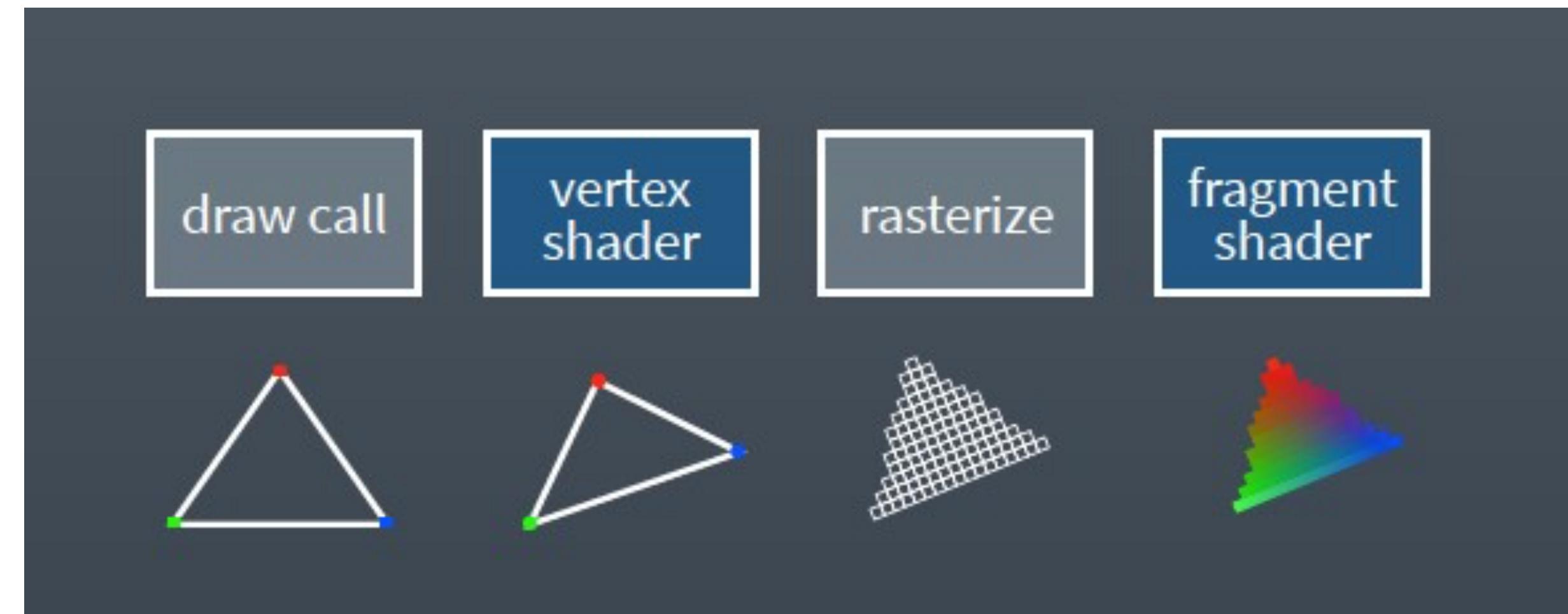
Solido

OpenGL

Render

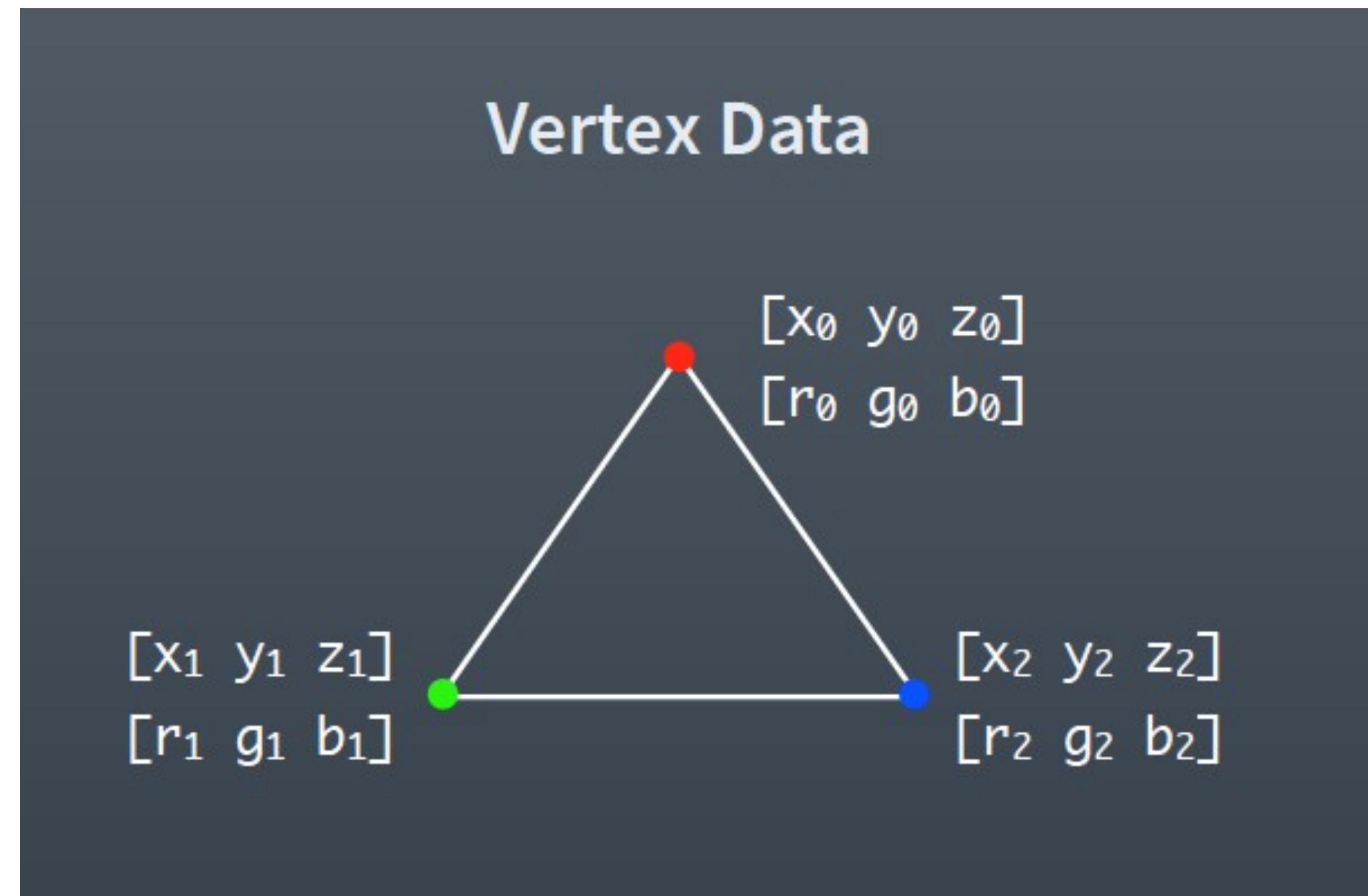
# CONCEPTOS 3D EXPLICADO CON BLENDER

## Que es un modelo 3D



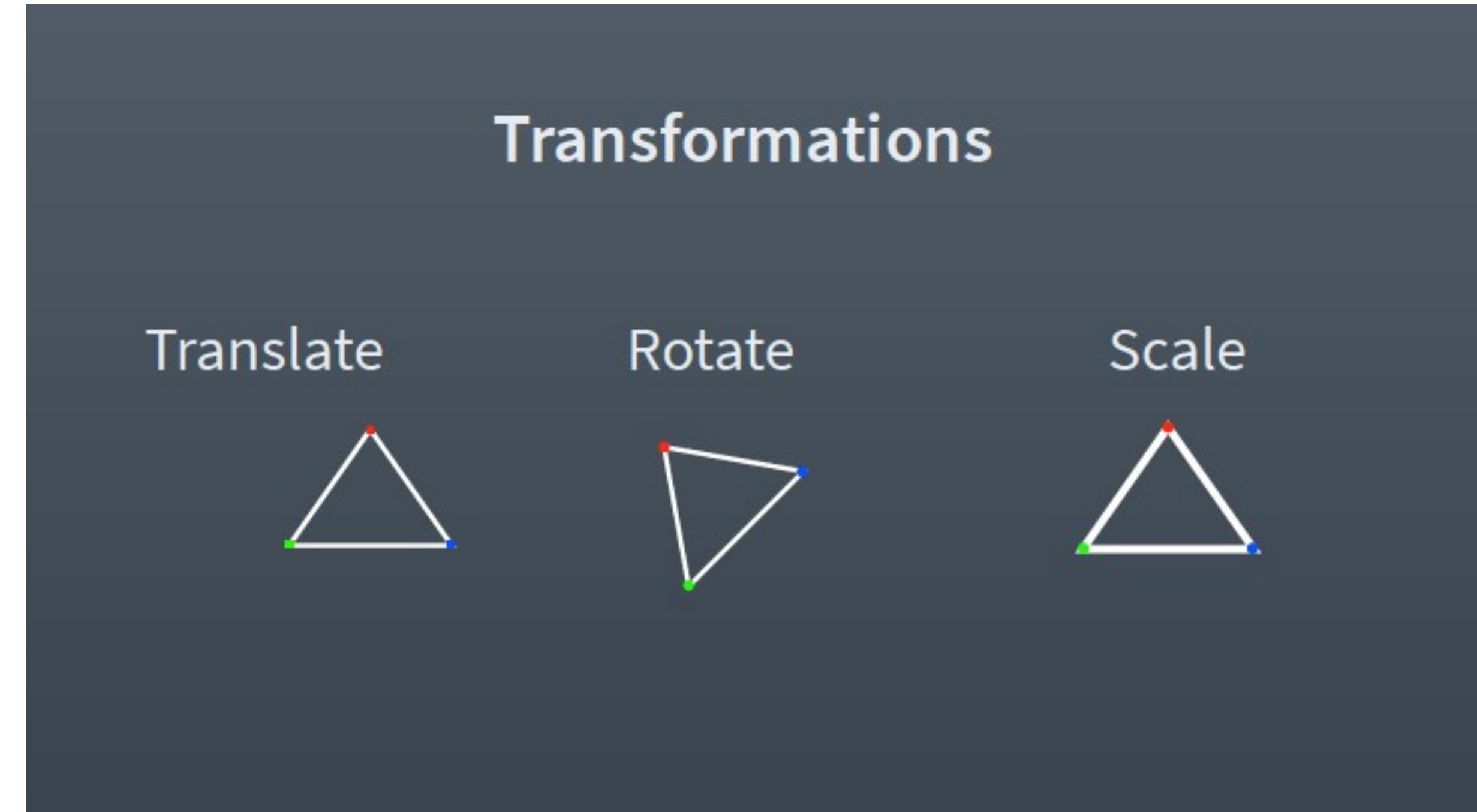
# CONCEPTOS 3D EXPLICADO CON BLENDER

Como se definan los vertices



# CONCEPTOS 3D EXPLICADO CON BLENDER

## Transformaciones en vertices



# DE LA TEORIA A LA PRACTICA

El repositorio del ejemplo es:

<https://github.com/jucaripo/Xamarin-UrhoSharp>



Android

jucaripo committed bbee5db 3 days ago ...

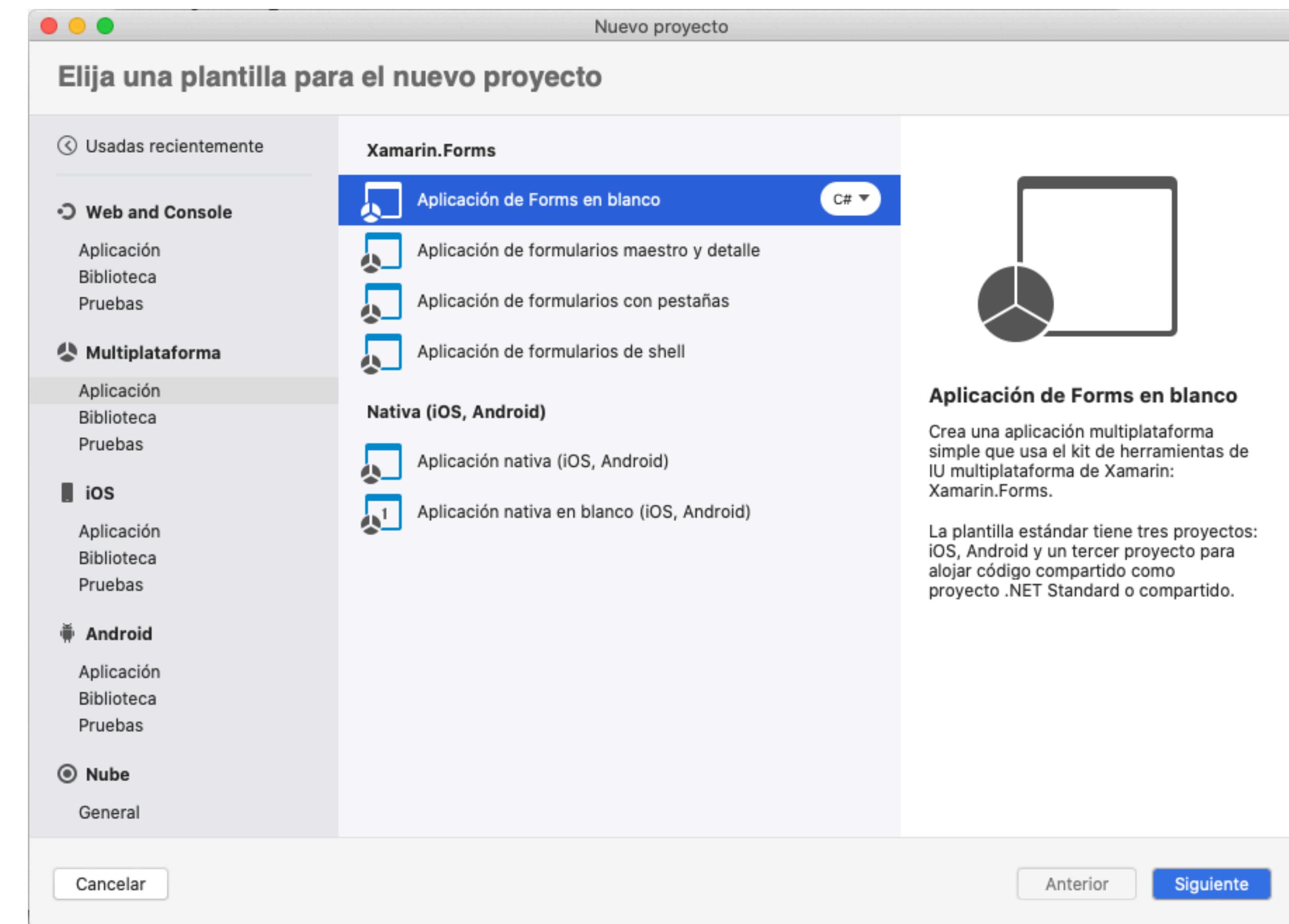
4 commits 1 branch 0 tags

UrhoDemo.Android	* .gitignore: Demo de OpenGL con Xamarin Forms y UrhoSharp	3 days ago
UrhoDemo.iOS	* .gitignore: Demo de OpenGL con Xamarin Forms y UrhoSharp	3 days ago
UrhoDemo	* .gitignore: Demo de OpenGL con Xamarin Forms y UrhoSharp	3 days ago
.gitignore	* .gitignore: Demo de OpenGL con Xamarin Forms y UrhoSharp	3 days ago
README.md	Imagenes	3 days ago
UrhoDemo.sln	* README.md:	3 days ago



iOS

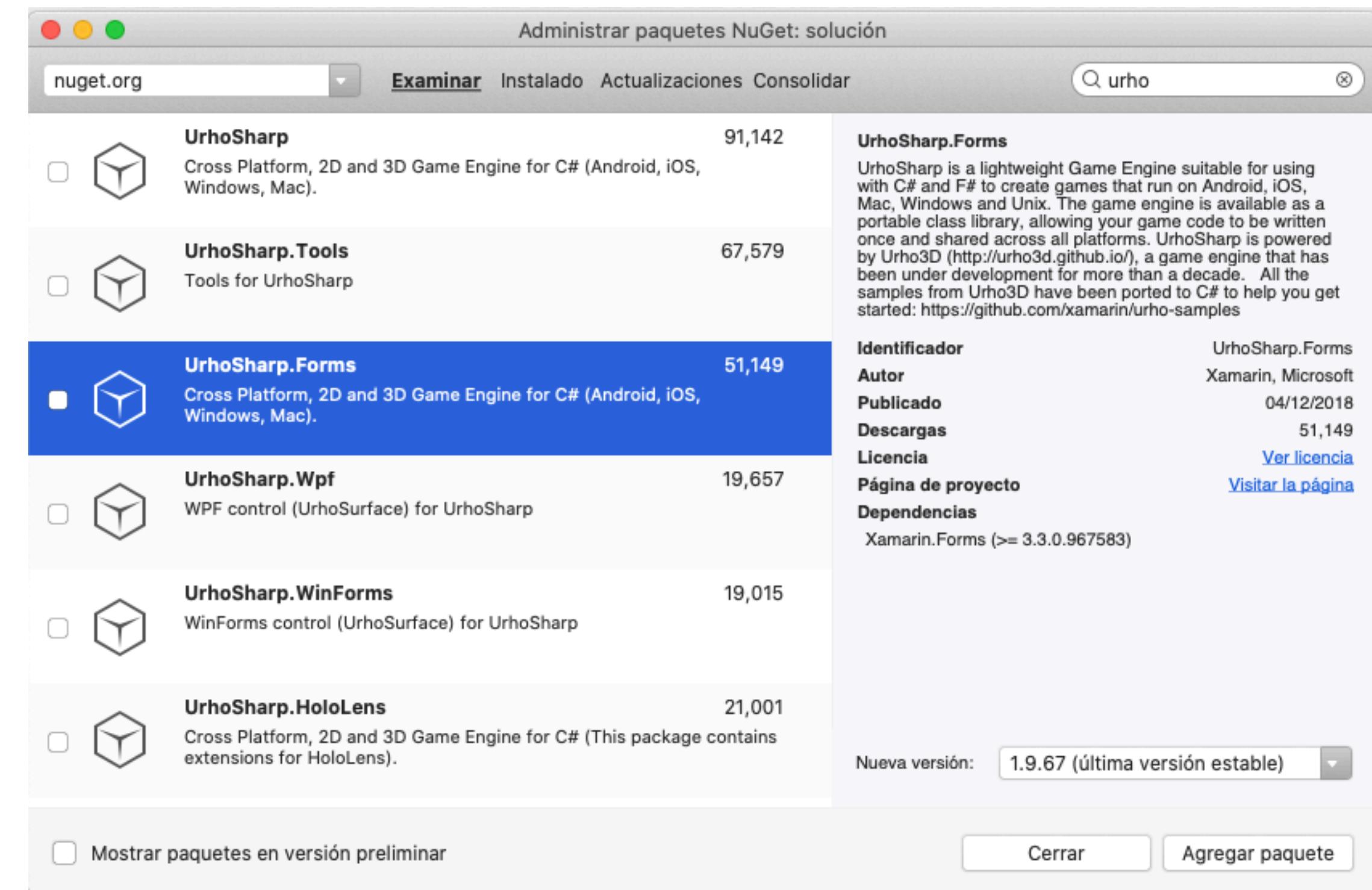
# Crear un proyecto de Xamarin Forms en blanco



## Darle un nombre al proyecto



# Agregar el paquete Nuget de UrhoForms



## Abrir el Xaml de nuestra vista



The screenshot shows a code editor window with several tabs at the top: MainPage.xaml.cs, Main.cs, README.md, and MainPage.xaml. The MainPage.xaml tab is active, displaying the following XAML code:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4   xmlns:urho="clr-namespace:Urho.Forms;assembly=UrhoSharp.Forms"
5   xmlns:d="http://xamarin.com/schemas/2014/forms/design" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6   <StackLayout>
7     ←— urho surface —→
8     <urho:UrhoSurface x:Name="HelloWorldUrhoSurface" VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand" BackgroundColor="White" />
9
10    </StackLayout>
11 </ContentPage>
```

## Agregar una Clase nueva HelloWorld.cs

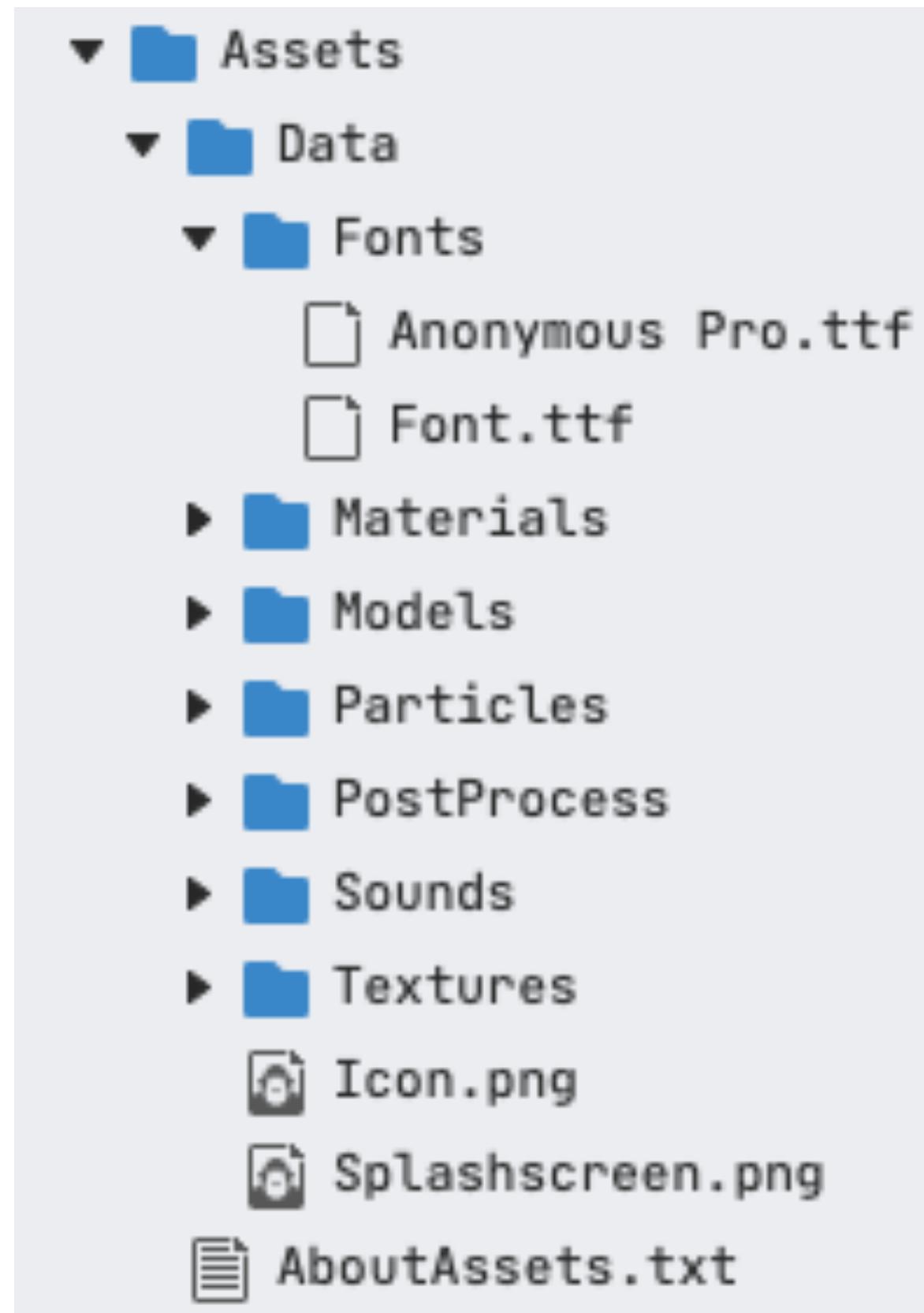
```
1  using System;
2
3
4  namespace UrhoDemo
5  {
6      public class HelloWorld
7      {
8          public HelloWorld()
9          {
10         }
11     }
12 }
13 }
```

## Heredamos como clase de Urho.Application

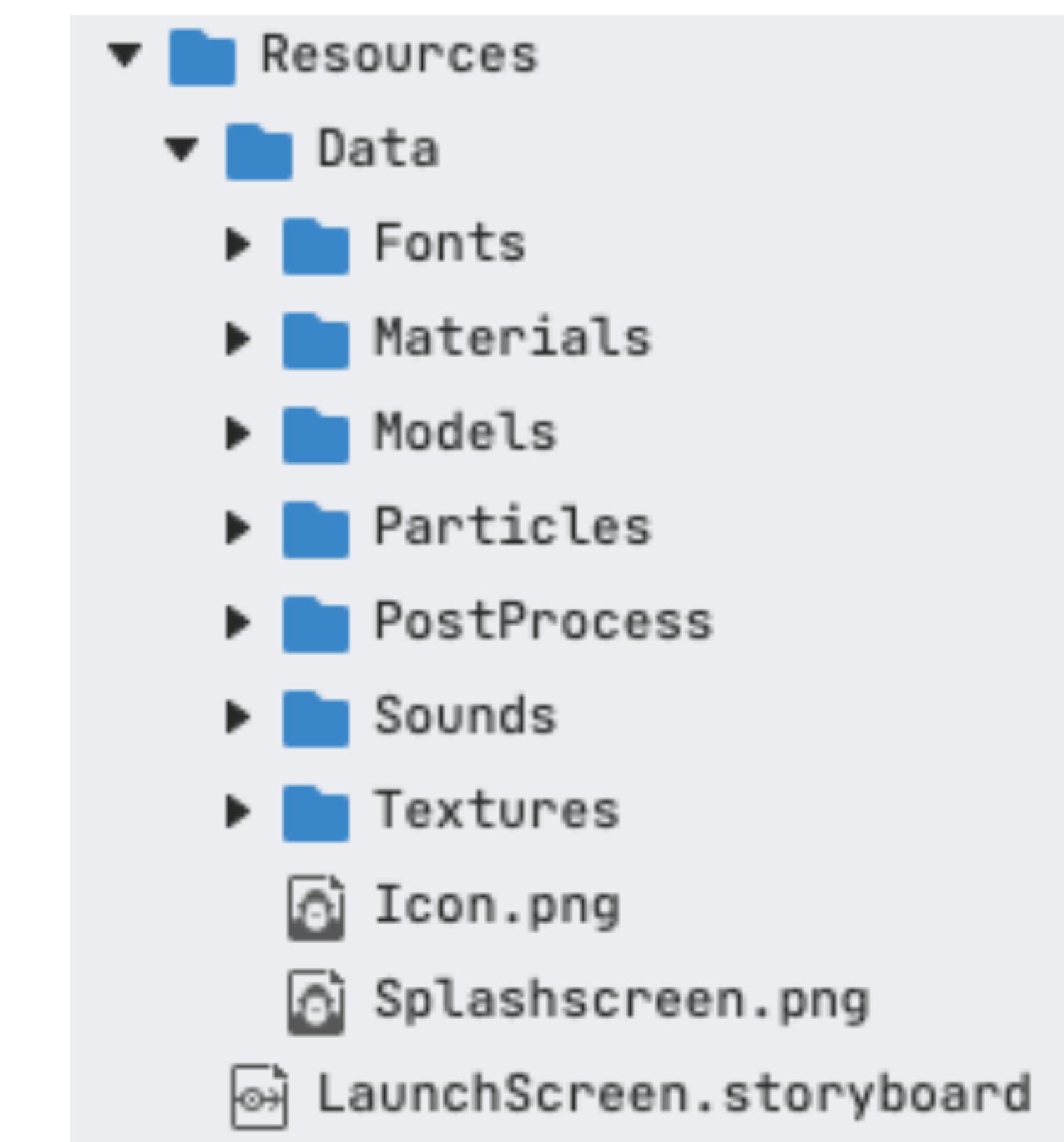
```
1  using System;
2  using Urho;
3  using Urho.Actions;
4  using Urho.Gui;
5
6  namespace UrhoDemo
7  {
8      public class HolaUrho : Urho.Application
9      {
10         public HolaUrho(ApplicationOptions options) : base(new ApplicationOptions(assetsFolder: "Data"))
11         {
12         }
13     }
14 }
15 |
```

En el constructor necesitamos tener el ApplicationsOptions  
Y en Base creamos una nueva AplicactionOptions para indicar  
Que usaremos el assetsFolder la carpeta Data.  
Esto le dira que en cada plataforma tendremos en assets los recursos

## Creamos nuestro árbol de recursos data en los assets



Android



iOS

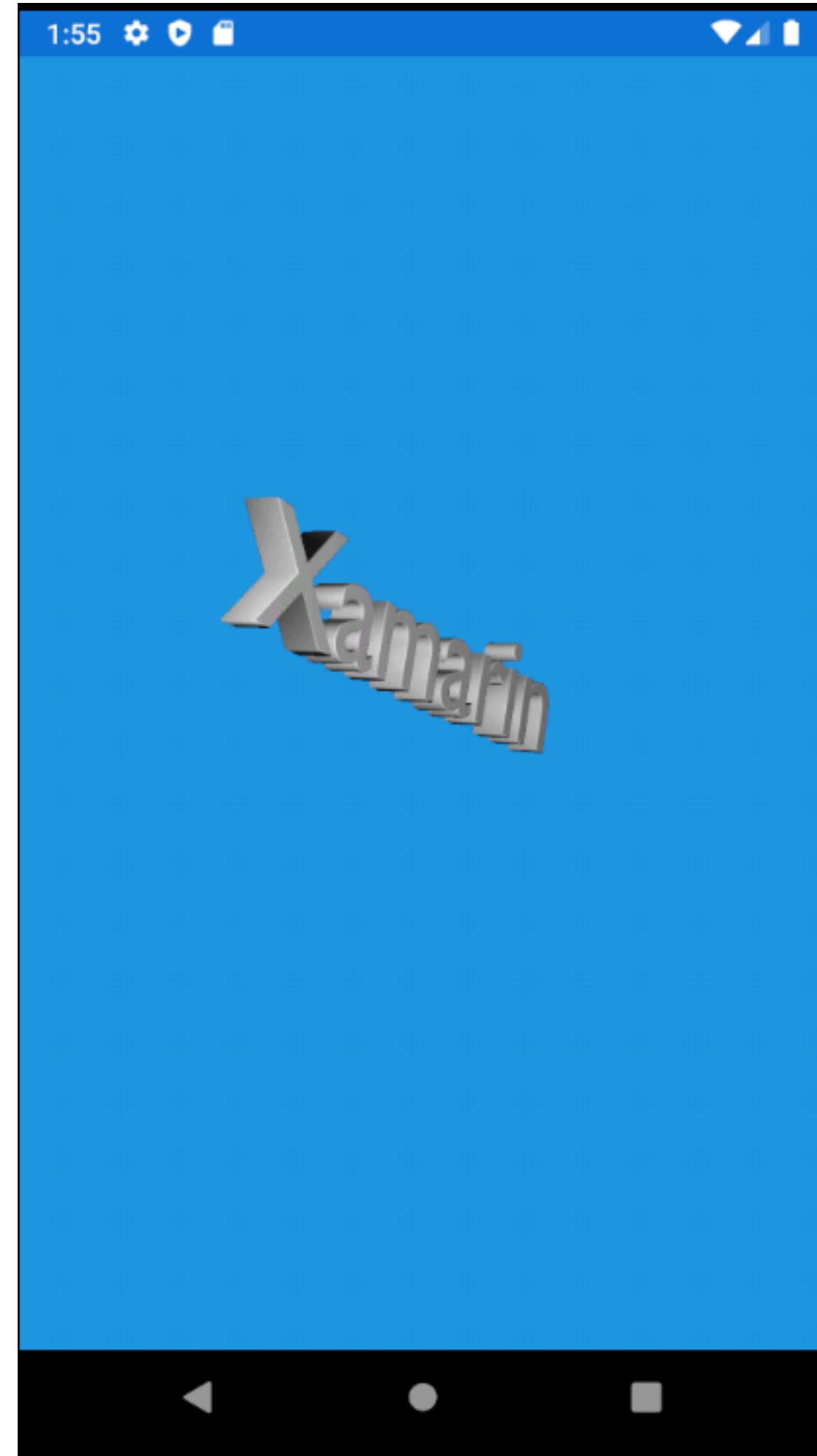
## Código para escribir un texto en la pantalla

```
/// <summary>
/// función demo de como crear un texto en la vista de urhosharp
/// </summary>
private void CreateText()
{
    // Create Text Element
    var text = new Text()
    {
        Value = "UrhoSharp \n Comunidad Xamarin en español",
        HorizontalAlignment = HorizontalAlignment.Left,
        VerticalAlignment = VerticalAlignment.Top
    };

    text.SetColor(Color.Green);
    textSetFont(font: ResourceCache.GetFont("Fonts/Anonymous Pro.ttf"), size: 30);
    // Add to UI Root
    UI.Root.AddChild(text);
}
```

## Código para dibujar la escena

```
...  
private async Task Create3DObject()  
{  
    // Crear la escena  
    var scene = new Scene();  
    scene.CreateComponent<Octree>();  
  
    // Node (Rotation and Position) de la escena  
    Node node = scene.CreateChild();  
    node.Position = new Vector3(0, 0, 5);  
    node.Rotation = new Quaternion(10, 60, 10);  
    node.SetScale(.5f);  
  
    // Modelo traer la geometria de un objeto para su uso en el motor 3d  
    StaticModel modelObject = node.CreateComponent<StaticModel>();  
    modelObject.Model = ResourceCache.GetModel("Models/Text.mdl");  
  
    // Luces crear una nueva luz  
    Node light = scene.CreateChild(name: "light");  
    light.SetDirection(new Vector3(0.4f, -0.5f, 0.3f));  
    light.CreateComponent<Light>();  
  
    // Camara crear la camara de la escena  
    Node cameraNode = scene.CreateChild(name: "camera");  
    Camera camera = cameraNode.CreateComponent<Camera>();  
  
    // Viewport unir los elementos a la vista  
    var viewport = new Viewport(scene, camera, null);  
    Renderer.SetViewport(0, viewport);  
    viewport.SetClearColor(Color.FromHex("#2d96da"));  
  
    // Action  
    await node.RunActionsAsync(  
        new RepeatForever(new RotateBy(duration: 1,  
            deltaAngleX: 0, deltaAngleY: 190, deltaAngleZ: 0)));  
}
```



## Código para iniciar urho start

```
/// <summary>
/// Start clase para iniciar nuestra urho aplicación
/// </summary>
protected override async void Start()
{
    base.Start();
    CreateText(); // crear un texto en la vista ver la función
    await Create3DObject();
}
```

## Código para iniciar en la vista la app de urho

```
protected override async void OnAppearing()
{
    base.OnAppearing();

    await HelloWorldUrhoSurface.Show<HelloWorld>(new Urho.ApplicationOptions(assetsFolder: "Data"));
}
```



Twitter: @jucaripo

Facebook: <https://www.facebook.com/jucaripoBlog/>  
[youtube.com https://www.youtube.com/user/jucaripo](https://www.youtube.com/user/jucaripo)  
<https://www.linkedin.com/in/jucaripo/>

