



**OGRE 3D**

**Object-oriented Graphics Rendering Engine**

**<http://www.ogre3d.org>**

Ana Gil Luezas

Departamento de Sistemas Informáticos y Computación

Universidad Complutense de Madrid

# Object-oriented Graphics Rendering Engine

- ❑ Motor gráfico de código abierto (licencia MIT)
- ❑ Implementado en C++, con uso extensivo de mecanismos de O.O. y namespaces (**namespace Ogre**)
- ❑ Multiplataforma: Windows, Linux, Mac OSX, Android
- ❑ API gráfico seleccionable: OpenGL, Direct3D
- ❑ Diseñado para ser extendido: Clases abstractas, componentes, herencia (múltiple), polimorfismo y *plug-ins* (complementos).  
Permiten ampliar la funcionalidad sin modificar el código

- ❑ Un motor gráfico es un compendio de *gestores*:
  - ❑ Gestor del API gráfico (archivo **ogre.cfg**)
  - ❑ Gestor de recursos (archivo **resources.cfg**)
    - Mallas, Texturas, Materiales
    - Shaders (GPU programs), ...
  - ❑ Gestores de escena
  - ❑ Gestor de *log* (archivo **ogre.log**)
  - ❑ Gestor de plugins (archivo **plugins.cfg**)
  - ❑ Gestor de archivos
  - ❑ ...
- ❑ En general, los gestores son *singleton* (instancia única)

- ❑ Definición de materiales mediante scripts:
  - ❑ *Uso de shaders (GPU programs: Cg, GLSL, HLSL, ensamblador)*
  - ❑ Selección automática de la versión del material (*technique*)
  - ❑ Nivel de detalle (*LOD*)
- ❑ Comunicación mediante listas de observadores (*listeners*) para recibir notificaciones
- ❑ El objeto **Ogre::Root\* root** es el punto de entrada al sistema.

Debe ser el primero en crearse y el último en destruirse.

Permite inicializar el sistema e iniciar el bucle de renderizado.

## □ main

```
#include "IG2App.h"
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    IG2App app;
```

```
    app.initApp();
```

```
    app.getRoot()->startRendering();
```

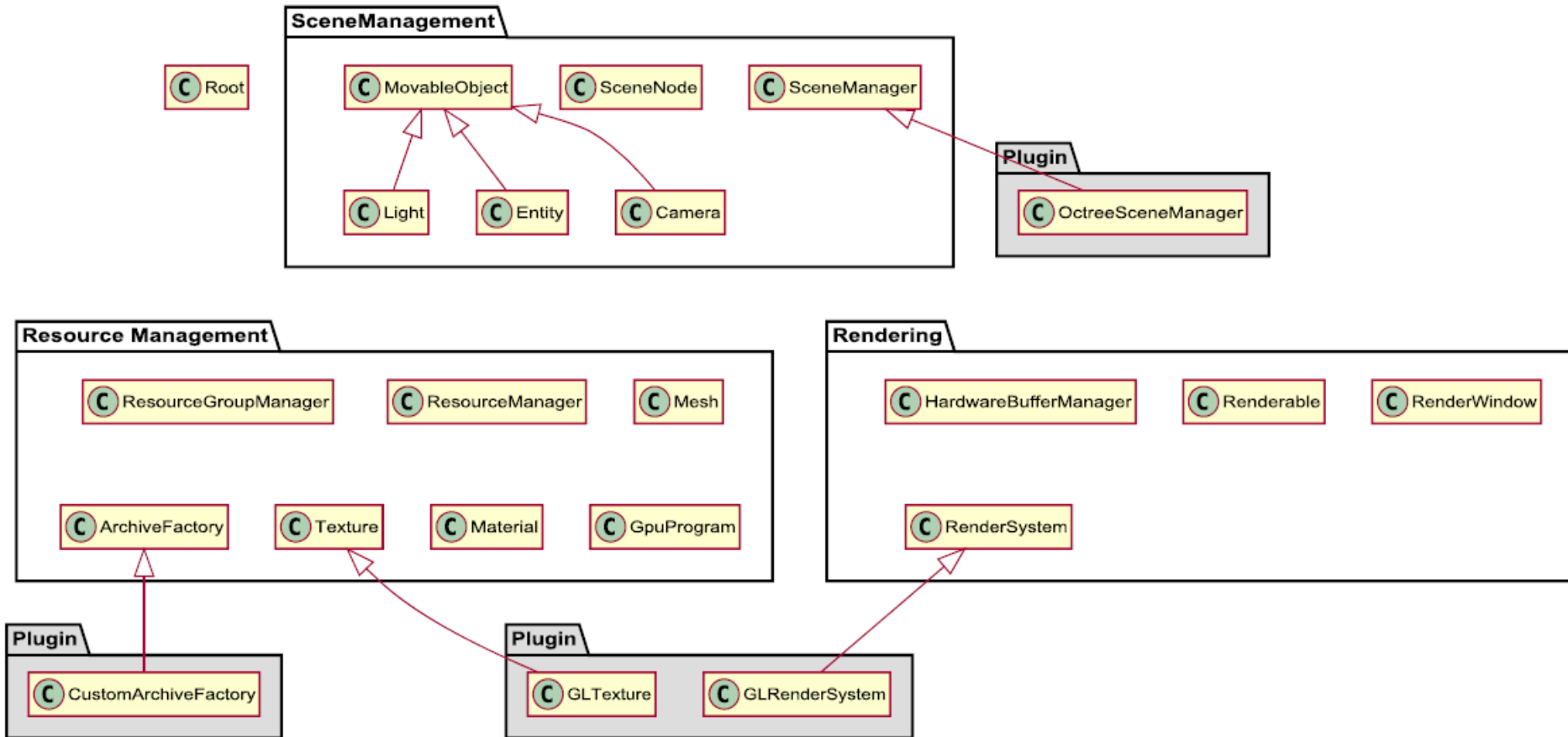
```
    app.closeApp();
```

```
    return 0;
```

```
}
```

```
IG2ApplicationContext::initApp()
    mRoot = new Root("...plugins.cfg",
                    "...ogre.cfg", "...ogre.log");
    mOverlaySystem = new OverlaySystem();
    setup() ->
        IG2ApplicationContext::setup()
        setupScene()
```

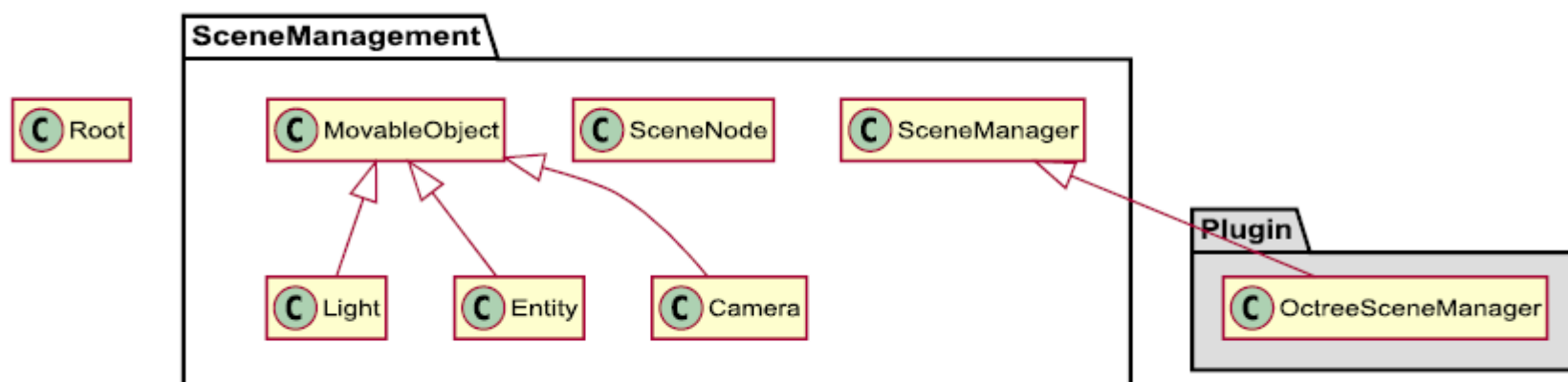
# The core objects



[ogrecave.github.io/ogre/api/latest/\\_the\\_core\\_objects.html](http://ogrecave.github.io/ogre/api/latest/_the_core_objects.html)

- ❑ **Ogre::SceneManager**: la clase más utilizada en la aplicación. Sirve para configurar el grafo de la escena: cámaras, luces, entidades, ... Actúa de factoría para varios tipos de objetos. Puede haber más de una instancia en la aplicación (no es singleton). Hay varias clases especializadas para distintos tipos de escena ().

```
Ogre:: SceneManager * mSM =  
    Ogre::Root::createSceneManager(...);
```

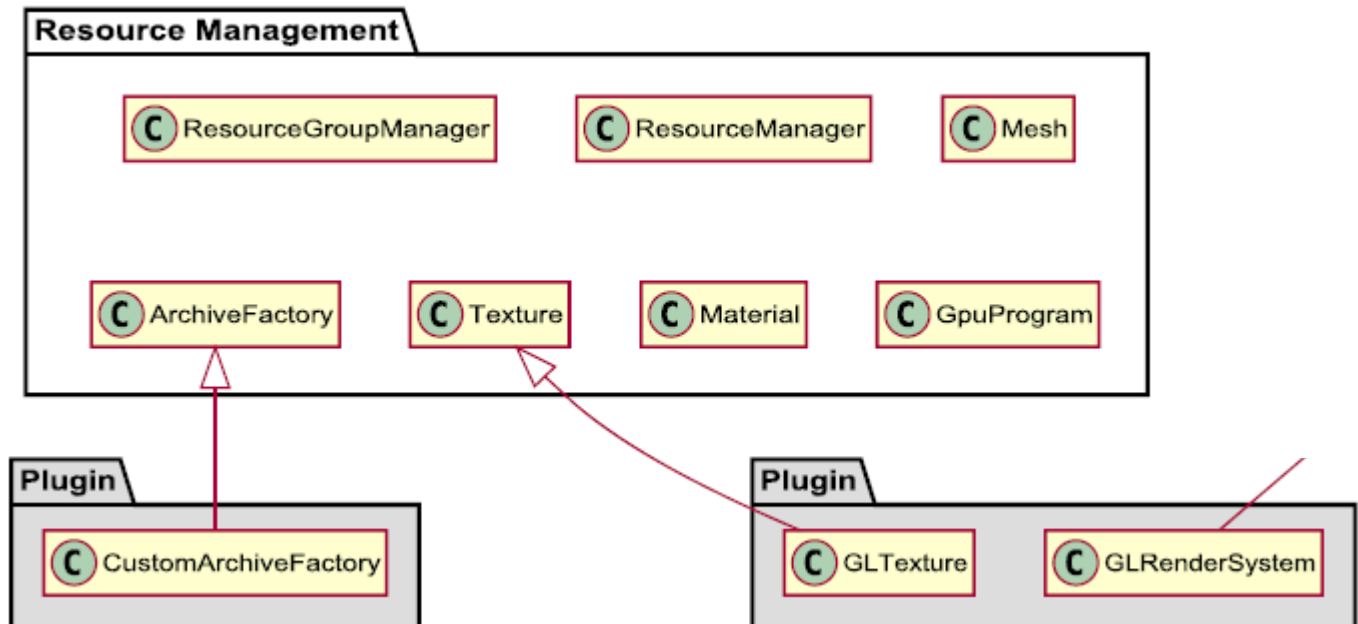


# Resource Management

- ❑ El objeto **Ogre::ResourceGroupManager** (es singleton):  
gestiona los grupos de recursos (p.e.: Essential, General, Tests).  
Cada uno agrupa distintas clases de recursos: Mesh, Texture,  
Material, GPUProgram, Compositor, Font. (archivo **resources.cfg**)  
Todos los recursos son compartidos.

**Ogre::TextureManager::getSingleton().someMethod()**

**Ogre::MeshManager::getSingleton().someMethod()**

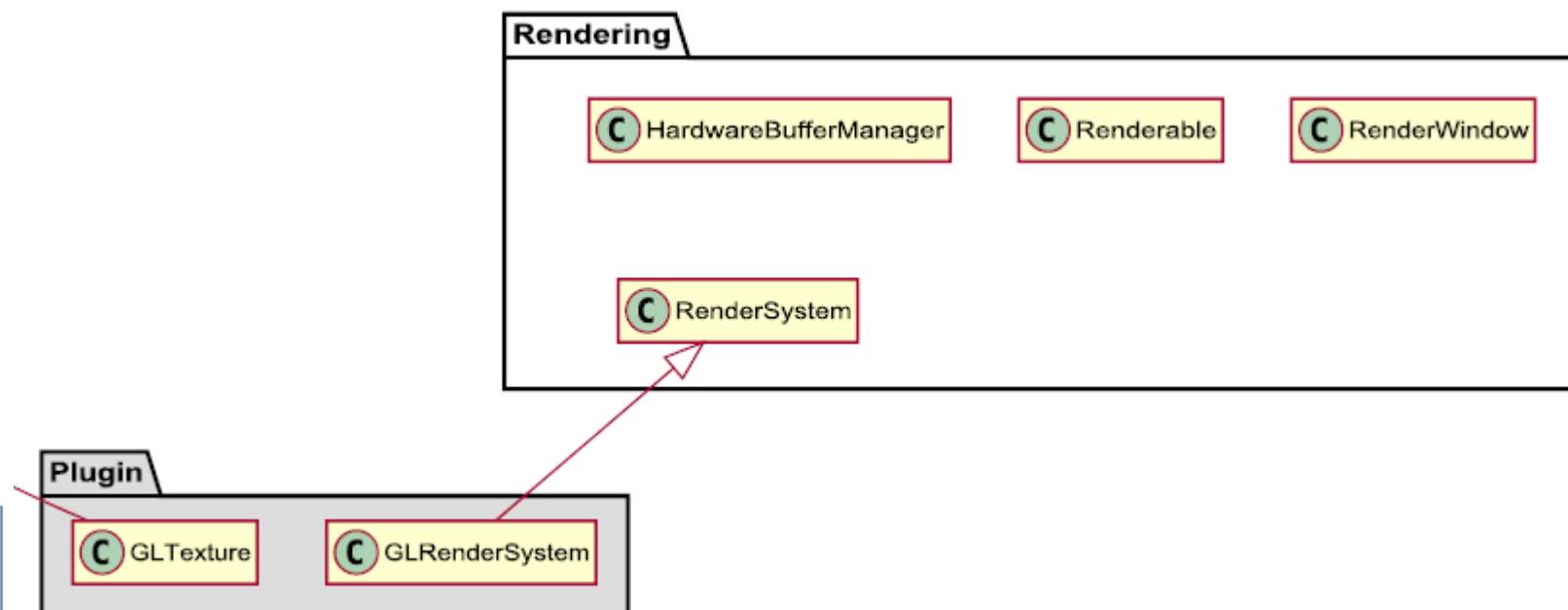




- ❑ **Ogre::RenderSystem** (clase abstracta): interfaz común para las diferentes APIs gráficas 3D que subyacen en sus implementaciones (OpenGL, Direct3D)

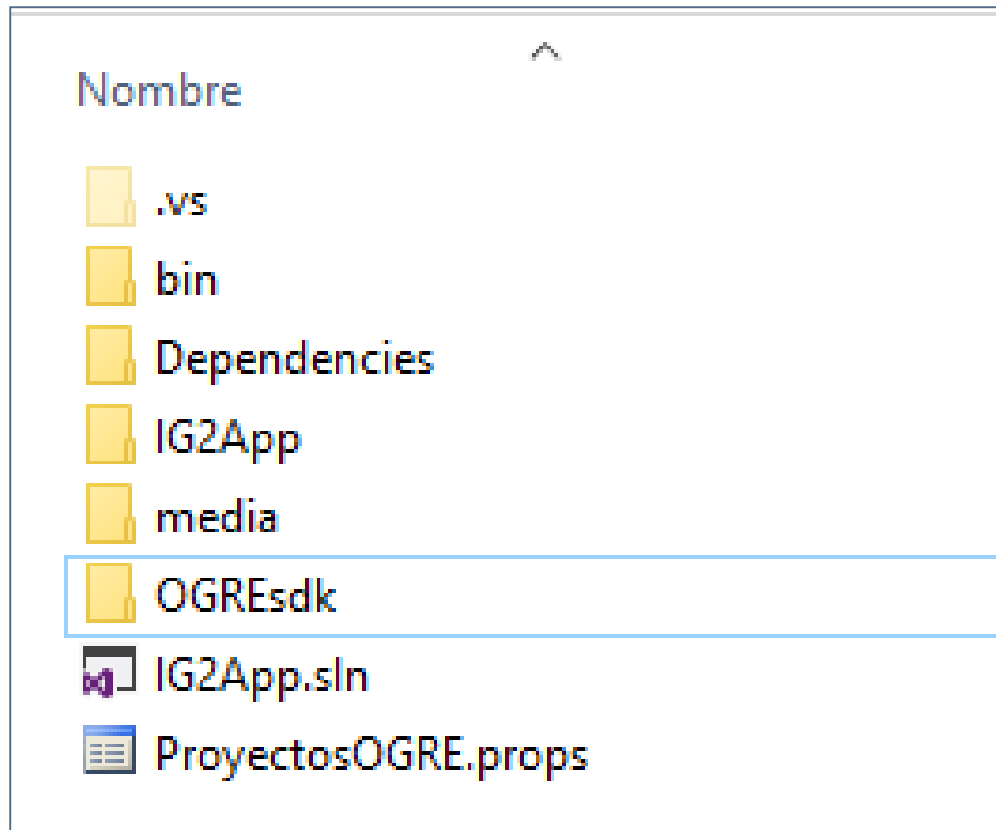
Habitualmente se utiliza a través del gestor de la escena, pero se puede acceder directamente a través de root:

**Ogre::RenderSystem\* pRS = root -> getRenderSystem();**

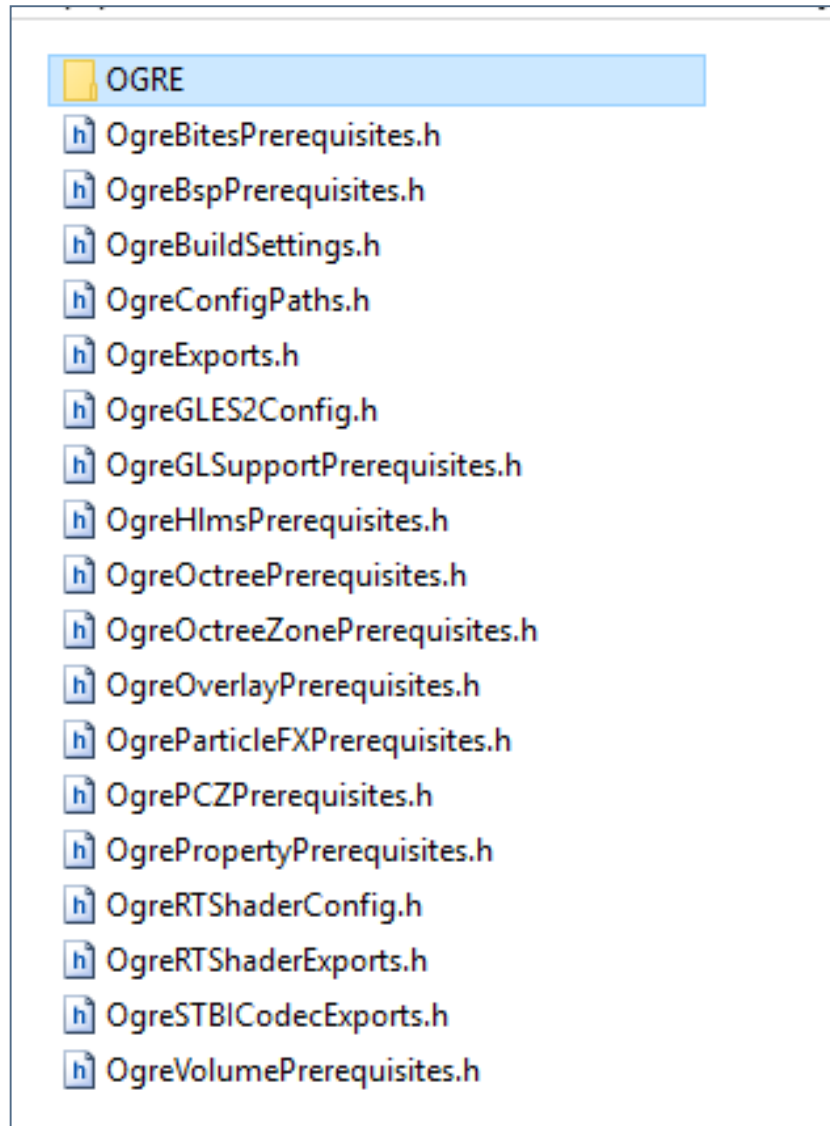


- ❑ **OgreSDK** es un grupo de herramientas para la programación de aplicaciones.

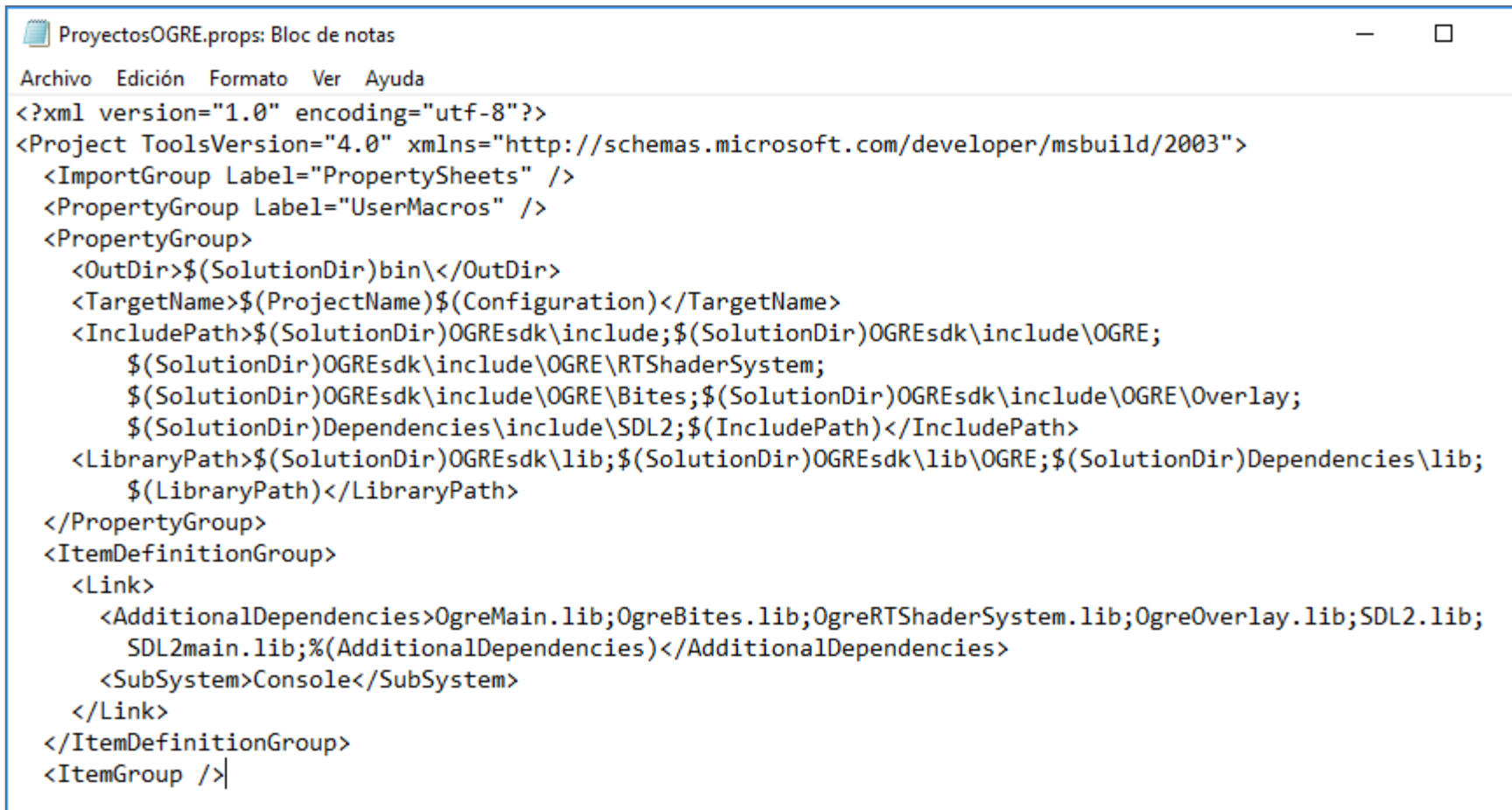
Recomiendan **CMake**



## ❑ OGREsdk\include



## ❑ ProyectosOGRE.props



```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ImportGroup Label="PropertySheets" />
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <OutDir>$(SolutionDir)bin\</OutDir>
    <TargetName>$(ProjectName)$(Configuration)</TargetName>
    <IncludePath>$(SolutionDir)OGREsdk\include;$(SolutionDir)OGREsdk\include\OGRE;
      $(SolutionDir)OGREsdk\include\OGRE\RTShaderSystem;
      $(SolutionDir)OGREsdk\include\OGRE\Bites;$(SolutionDir)OGREsdk\include\OGRE\Overlay;
      $(SolutionDir)Dependencies\include\SDL2;$(IncludePath)</IncludePath>
    <LibraryPath>$(SolutionDir)OGREsdk\lib;$(SolutionDir)OGREsdk\lib\OGRE;$(SolutionDir)Dependencies\lib;
      $(LibraryPath)</LibraryPath>
  </PropertyGroup>
  <ItemDefinitionGroup>
    <Link>
      <AdditionalDependencies>OgreMain.lib;OgreBites.lib;OgreRTShaderSystem.lib;OgreOverlay.lib;SDL2.lib;
        SDL2main.lib;%(<AdditionalDependencies>)</AdditionalDependencies>
      <SubSystem>Console</SubSystem>
    </Link>
  </ItemDefinitionGroup>
</ItemGroup> />
```

## ❑ Utiliza utilidades de OgreBites:

OgreApplicationContext

OgreBitesConfigDialog

OgreInput

OgreTrays

OgreCameraMan

## ❑ **IG2App** contiene las siguientes clases:

❑ **IG2ApplicationContext** (adaptación de **OgreApplicationContext**):  
crea **Root**, crea la ventana de renderizado, inicia los gestores,...  
Implementa **FrameListener** e informa de eventos de entrada a los observadores (objetos del tipo **InputListener\***) suscritos.

❑ **IG2App**: hereda de **IG2ApplicationContext** e **InputListener**

❑ **main**: lanza el bucle de renderizado

- ❑ **media** contiene directorios para materials (texturas, scripts y shaders) y para models (mallas)
- ❑ **media** contiene un directorio **IG2App** que reúne los elementos de materials y models que usa nuestra aplicación
- ❑ **bin** contiene los archivos de configuración **ogre.cfg**, **resources.cfg** y **plugins.cfg**

## ❑ OgreBitesConfigDialog



## ❑ bin\logre.cfg

```
Render System=OpenGL 3+ Rendering Subsystem
```

```
[OpenGL Rendering Subsystem]
```

```
Colour Depth=32
```

```
Display Frequency=60
```

```
FSAA=0
```

```
Full Screen=No
```

```
RTT Preferred Mode=FBO
```

```
VSync=Yes
```

```
VSync Interval=1
```

```
Video Mode=1024 x 768
```

```
sRGB Gamma Conversion=No
```

```
[OpenGL 3+ Rendering Subsystem]
```

```
Colour Depth=32
```

```
Display Frequency=60
```

```
FSAA=0
```

```
Full Screen=No
```

```
RTT Preferred Mode=FBO
```

```
VSync=Yes
```

```
VSync Interval=1
```

```
Video Mode=1024 x 768
```

```
sRGB Gamma Conversion=No
```



## ❏ bin/resources.cfg

```
# Resources required by the sample browser and most samples.
[Essential]
Zip=../media/packs/SdkTrays.zip
Zip=../media/packs/profiler.zip
FileSystem=../media/thumbnails
# Common sample resources needed by many of the samples.
# Rarely used resources should be separately loaded by the samples which require them
[General]
# Bites uses the next entry to discover the platform shaders
FileSystem=../media
...
FileSystem=../media/IG2App
Zip=../media/packs/Sinbad.zip
# FileSystem=../media/HLMS
# Zip=../media/packs/DamagedHelmet.zip
...
```

## bin\ogre.log

Informe sobre puglins  
Informe sobre CPU  
Informe sobre GPU  
Informe sobre recursos

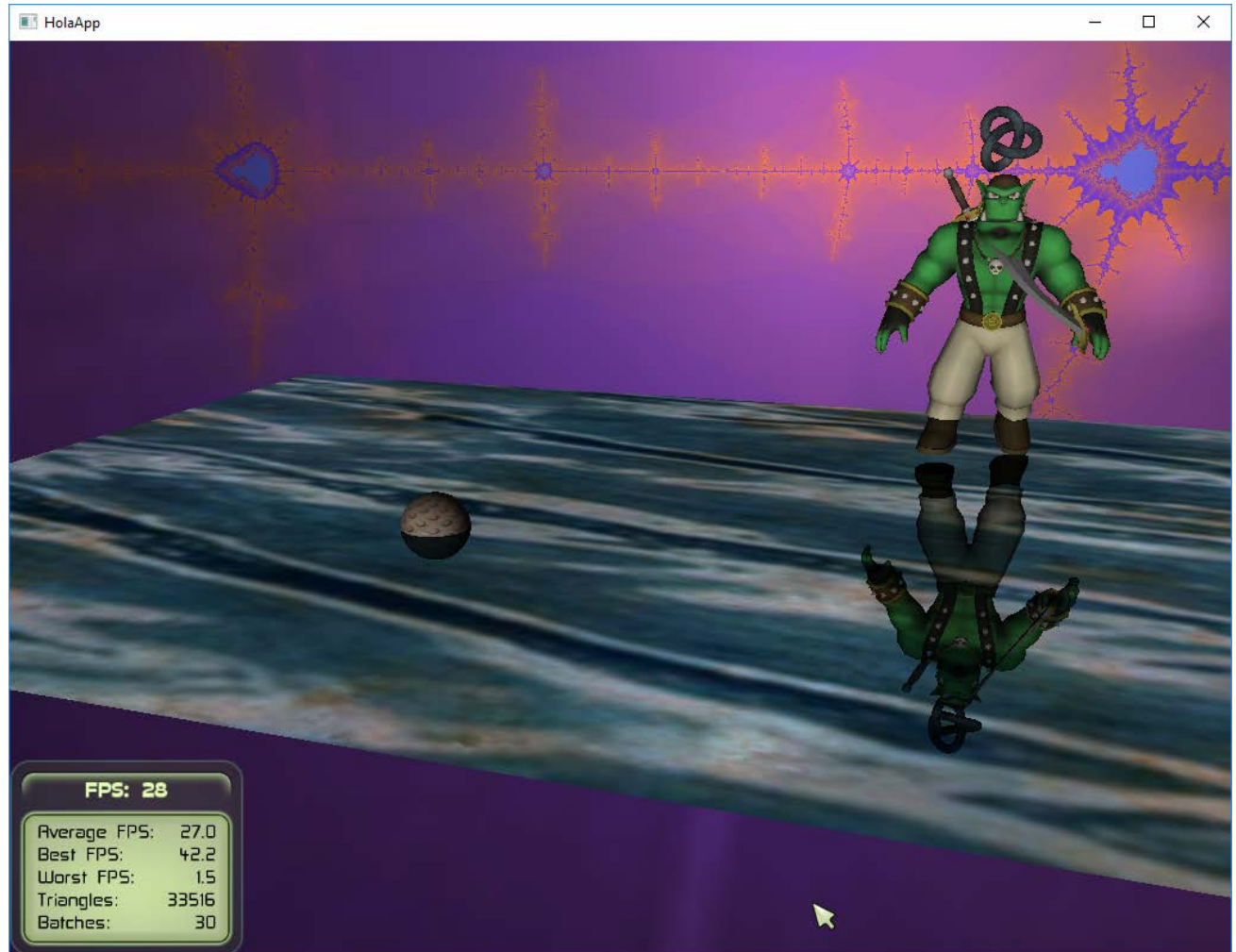
## ❑ Plugins (bin\plugins.cfg)

```
# Defines plugins to load

# Define plugin folder
PluginFolder=.

# Define plugins
# Plugin=RenderSystem_Direct3D9
# Plugin=RenderSystem_Direct3D11
Plugin=RenderSystem_GL
Plugin=RenderSystem_GL3Plus
# Plugin=RenderSystem_GLES2
Plugin=Plugin_ParticleFX
Plugin=Plugin_BSPSceneManager
# Plugin=Plugin_CgProgramManager
# Plugin=Codec_EXR
Plugin=Codec_STBI
# Plugin=Codec_FreeImage
Plugin=Plugin_PCZSceneManager
Plugin=Plugin_OctreeZone
Plugin=Plugin_OctreeSceneManager
```

## ❑ OgreTrays (Overlay system)



**OGRE** utiliza archivos para:

- ❑ Indicar qué plugins se deben cargar: ProyectosOgre/bin/**plugins.cfg**

- ❑ Indicar dónde están los recursos que se van a utilizar: mallas, texturas, shaders, materiales (scripts):

ProyectosOgre/bin/**resources.cfg**

Al inicio se analizan los recursos, pero se cargan al ser utilizados. Todos son compartidos.

- ❑ Guardar información sobre la configuración. Se crea durante la primera ejecución. ProyectosOgre/bin/**ogre.cfg** (\*)

- ❑ Guardar información sobre la inicialización, carga de recursos, posibles errores durante la ejecución, .... Se crea con cada ejecución (la información también se muestra en la consola).

ProyectosOgre/bin/**ogre.log** (\*)

(\*) Modificado: MisDocumentos/Ogre/nombreProyecto/ogre.X

# Setting up an OGRE project

- ❑ [www.ogre3d.org](http://www.ogre3d.org)

Download OGRE (versión  $\geq 1.11.2$ )

- ❑ [github.com/OGRECave/ogre](https://github.com/OGRECave/ogre) -> BuildingOgre.md

Utiliza **CMake** (cross platform make -> [cmake.org/download](http://cmake.org/download)), herramienta multiplataforma de generación de código, de más alto nivel que el sistema Make de Unix.

Compilador C++: VS2017, ...

Dependencias: SDL2 (ventana y eventos), FreeType, ...

Plugins: Octree, BSP, ParticleFX (archivo **plugins.cfg**)

APIs gráficos: GL y GL3Plus (archivo **ogre.cfg**)

- ❑ **OgreSDK** (Software Development Kit)