Luces

- Las fuentes de luz en **IG2App:: setupScene()**

```
//Las luces se crean, con tipo y color
Ogre::Light* luz = mSM->createLight("Luz");
luz->setType(Ogre::Light::LT_DIRECTIONAL);
luz->setDiffuseColour(0.75, 0.75, 0.75);

//Las luces se asocian a nodos de la escena
Ogre::SceneNode* lightNode = mSM->createSceneNode();
mSM->getRootSceneNode()->addChild(lightNode);
lightNode->attachObject(luz);

//Las luces se orientan a través de su nodo
lightNode->setDirection(Ogre::Vector3(0, 0, -1));//Direccional, TS_LOCAL
lightNode->setPosition(0, 0, 1000);  //Posicional
```

Luces

- La clase **Light** hereda de **MovableObject**
- Las fuentes de luz las crea el gestor de la escena

```
Light* luz = mSM->createLight("Luz");
```

- Tenemos tres tipos de fuentes de luz (**LightTypes**):

LT_POINT, LT_DIRECTIONAL, LT_SPOTLIGHT

- Las fuentes de luz en Ogre tienen componente difusa y especular
- La componente ambiente es general de la escena y la fija el gestor de escena

```
mSM->setAmbientLight(ColourValue);
```

Luces

- Métodos heredados de **MovableObject**

```
light-> setVisible(bool)  //false->apagar la luz
```

- Métodos de configuración de la luz a través del nodo o de la luz

```
lightNode-> setDirection(Vec3); //luz direccional
light-> setPosition(Vec3); //luz posicional
```

- Métodos de configuración de la luz a través de la luz

```
light-> setType(LightTypes);
light-> setDiffuseColour(ColourValue);
light-> setSpecularColour(ColourValue);
light-> setAttenuation(…);
```

Focos

- Configuración de un foco

```
luzFoco = mSM->createLight("Luz Foco");
luzFoco->setType(Ogre::Light::LT_SPOTLIGHT);
luzFoco->
     setDiffuseColour(Ogre::ColourValue(1.0f,1.0f,1.0f));

luzFoco->setDirection(Ogre::Vector3(1, -1, 0));
luzFoco->setSpotlightInnerAngle(Ogre::Degree(5.0f));
luzFoco->setSpotlightOuterAngle(Ogre::Degree(45.0f));
luzFoco->setSpotlightFalloff(0.0f);

node->attachObject(luzFoco);
```

Spotlights

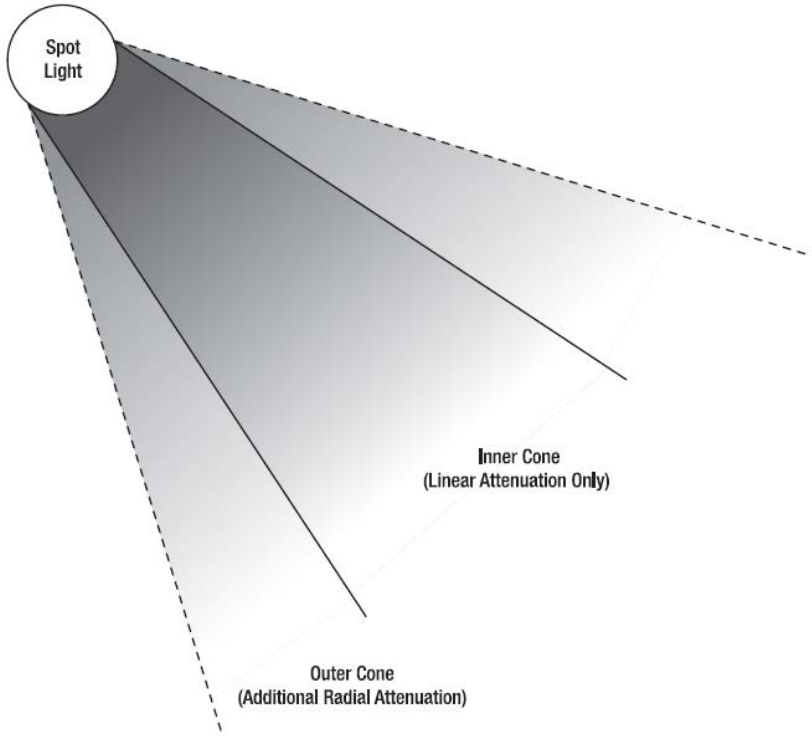
- Spot light cone effect:

  spotLight->setSpotlightRange(*innerAngle*, *outerAngle, falloff=1.0*);

  The inner cone applicable only to Direct3D, it'll always treat as zero in OpenGL.

  The rate of falloff between the inner and outer cones.

  falloff=1.0 means a linear falloff, less means slower falloff, higher means faster falloff.



Sombras

- Cambiar el sistema de renderización a **`OpenGL Rendering Subsystem`**
  - Mostrar la caja de diálogo
- Añadir la instrucción

```
mSM->setShadowTechnique(
        Ogre::SHADOWTYPE_STENCIL_ADDITIVE);
```

- Todas las luces producen sombra por defecto. Se desactiva la producción de sombras por **`light`** con el comando

```
light-> setCastShadows(false);
```

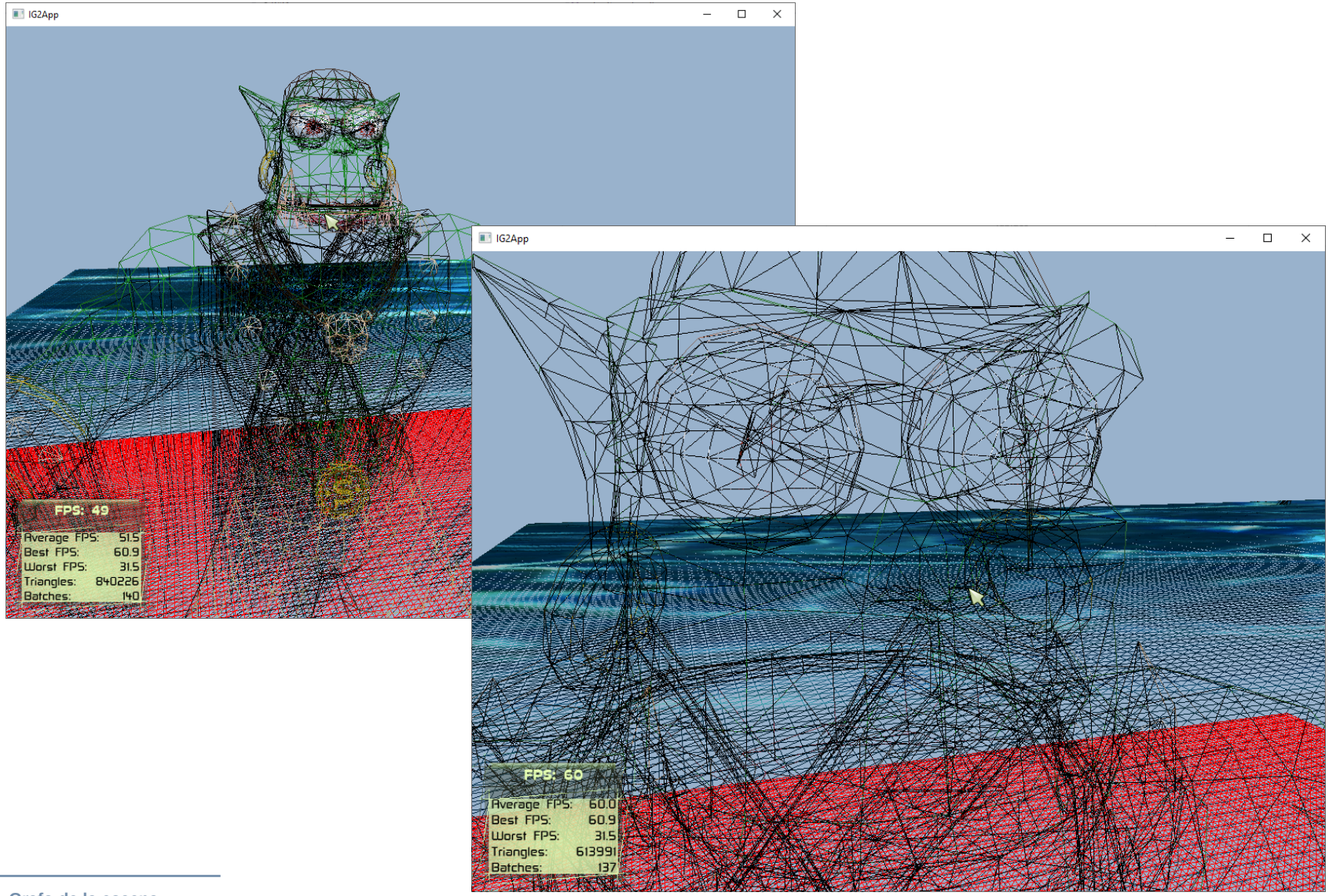
Cámaras

- La cámara en **IG2App:: setupScene()**

```
//Las cámaras se crean
Ogre::Camera* cam = mSM->createCamera("Cam");
cam->setNearClipDistance(1);
cam->setFarClipDistance(10000);
cam->setAutoAspectRatio(true);
//cam->setPolygonMode(Ogre::PM_WIREFRAME);

//Las cámaras se asocian a nodos de la escena
Ogre::SceneNode* camNode = mSM->
     getRootSceneNode()->createChildSceneNode();
camNode->setPosition(0, 0, 1000);
     //La posición se puede expresar relativa a un espacio de
coordenadas local, del padre o mundial
camNode->lookAt(Ogre::Vector3(0, 0, -1),
     Ogre::Node::TransformSpace::TS_WORLD);
     //Espacio de coordenadas en que se expresa el punto
camNode->attachObject(cam);
//camNode->setDirection(Ogre::Vector3(0, 0, -1));
```

Wireframe


IG2App
FPS: 49
Average FPS: 51.5
Best FPS: 60.9
Worst FPS: 31.5
Triangles: 840226
Batches: 140
IG2App
FPS: 60
Average FPS: 60.0
Best FPS: 60.9
Worst FPS: 31.5
Triangles: 613991
Batches: 137

Puerto de vista

- El puerto de vista en **IG2App:: setupScene()**
  - Recuerda que el punto (0, 0) es la esquina superior izquierda de la pantalla
  - Cada puerto de vista solo puede renderizar lo que ve una sola cámara

```
Viewport *vp = getRenderWindow()->addViewport(cam);
```

  - Se puede definir el color de fondo del puerto de vista

```
vp->setBackgroundColour(ColourValue(0.6, 0.7, 0.8));
```

  - Se pueden fijar sus dimensiones con

```
vp->setDimensions(Real left, Real top,
        Real width, Real height);
```

  Se expresan como valores de [0,1]. Es decir (0, 0, 1, 1) es todo el área

Puerto de vista

- El puerto de vista en **IG2App:: setupScene()**
  - El tamaño del puerto de vista determina el ”aspect ratio” de la cámara

```
cam->setAspectRatio(Real(vp->getActualWidth()) /
                    Real(vp->getActualHeight()));
```

  - Si se cambian las dimensiones de la ventana del puerto de vista, con el siguiente comando se ajustan automáticamente las proporciones del frustum

```
cam->setAutoAspectRatio(true);
```

Camera

- La clase **Camera** hereda de **Frustum** y esta de **MovableObject**.

  Las cámaras las crea el gestor de la escena.

  Camera* cam = mSM->createCamera("Cam");

  mCamNode->attachObject(cam);

- Tenemos dos tipos de proyección (**ProjectionType**): **PT_ORTHOGRAPHIC, PT_PERSPECTIVE**

  setProjectionType(ProjectionType);

- Configuración a nivel de frustrum

  setAspectRatio, setAutoAspectRatio, setFOVy, setNearClipDistance, setfarClipDistance, setOrthoWindow

Cameras

- Configuración a nivel de nodo

```
mCamNode->setPosition(0, 0, 1000);
mCamNode->lookAt(Vector3(0, 0, -1),
                 Ogre::Node::TransformSpace::TS_WORLD);
yaw(), pitch(…), roll(…)
```

- Otros

```
enableReflection(Plane)
enableCustomNearClipPlane(Plane)
setAutoTracking(SceneNode*, …)
```

Gestor de cámara

- Otra pequeña utilidad de OGRE Bites -> OgreCameraMan

  La clase **CameraMan** hereda de **InputListener** para gestionar la respuesta a los eventos de entrada siguiendo dos estilos (**CameraStyle**): **CS_FREELOOK, CS_ORBIT**

- La constructora **CameraMan(Ogre::SceneNode* cam);**

  recibe el nodo del grafo que contiene la cámara que queremos gestionar

  OgreBites::CameraMan * mCamMgr = new CameraMan(camNode);

  mCamMgr->setStyle(CameraStyle);

  addInputListener(mCamMgr);

  Recuerda eliminarla en shutdow(): **delete mCamMgr;**

- En modo CS_ORBIT: mCamMgr->setTarget(SceneNode*);

  Por defecto sigue al nodo raíz del grafo de la escena.

IG2App:: setupScene()

```
// camera manager
mCamMgr = new OgreBites::CameraMan(camNode);
// -> Recuerda liberarla en shutdow():  delete mCamMan;

addInputListener(mCamMgr);
mCamMgr->setStyle(OgreBites::CS_ORBIT);  //  CS_FREELOOK
// mCamMgr ->setYawPitchDist(Degree(0), Degree(0), 100);
// mCamMgr -> setTarget(node);
```