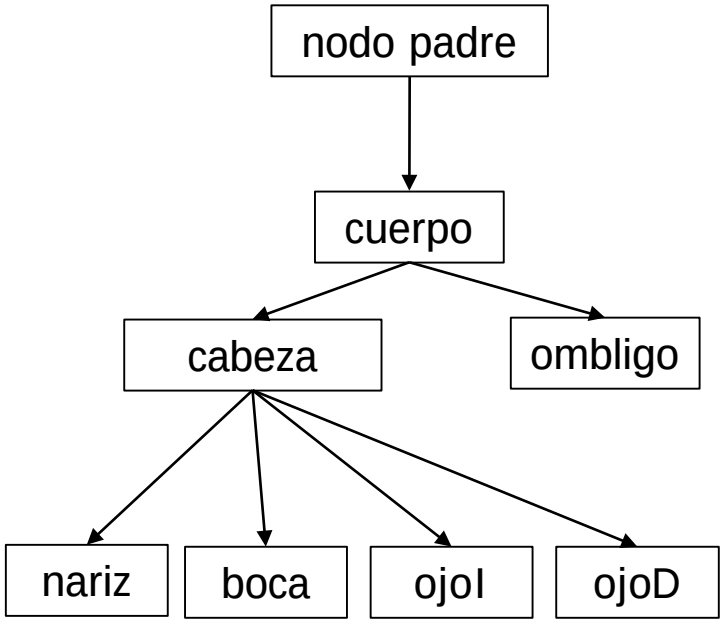
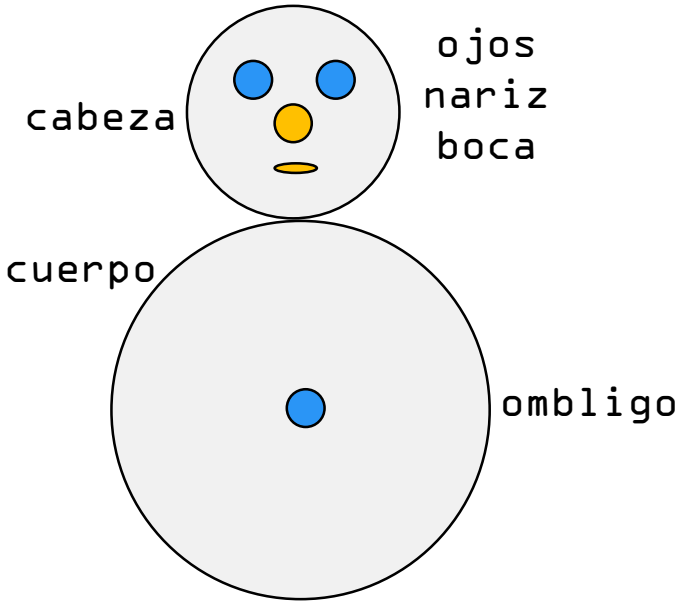
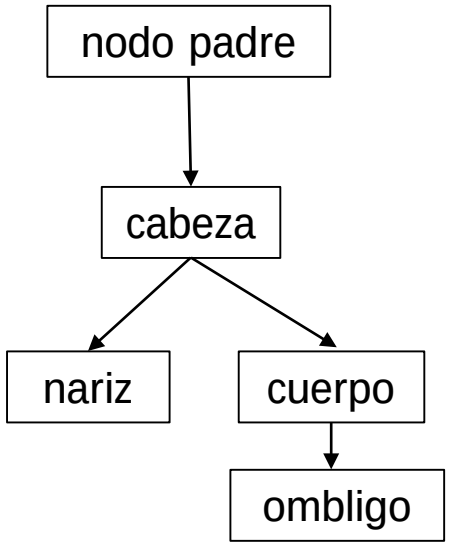
- Muñeco: la cabeza está debajo del cuerpo

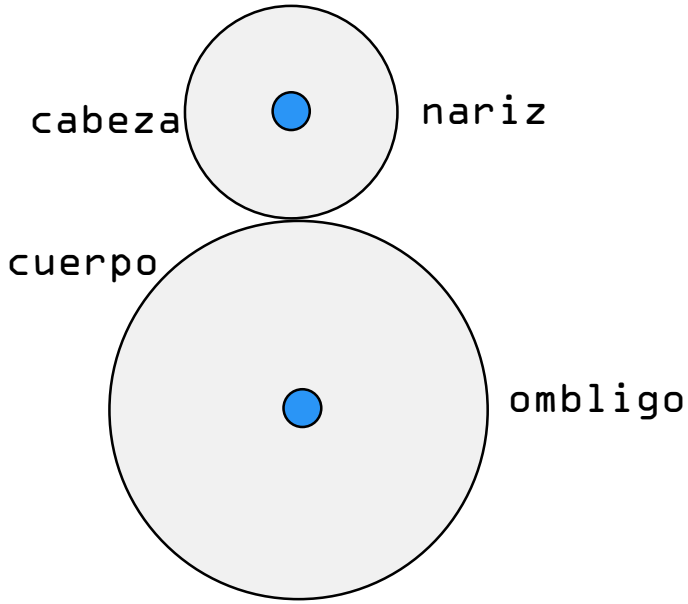




La cabeza puede girar hacia los lados, incluyendo los ojos, nariz y boca, independientemente del cuerpo
El cuerpo puede girar para desplazarse, sin modificar la cabeza que se desplaza con el cuerpo.
Todas las entidades son esferas (sphere.mesh)

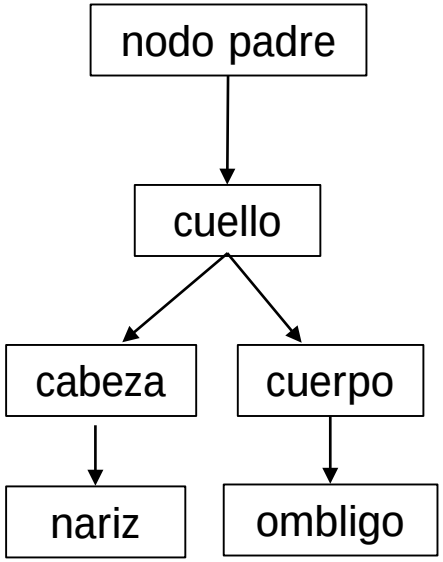
- Muñeco: el cuerpo está debajo de la cabeza

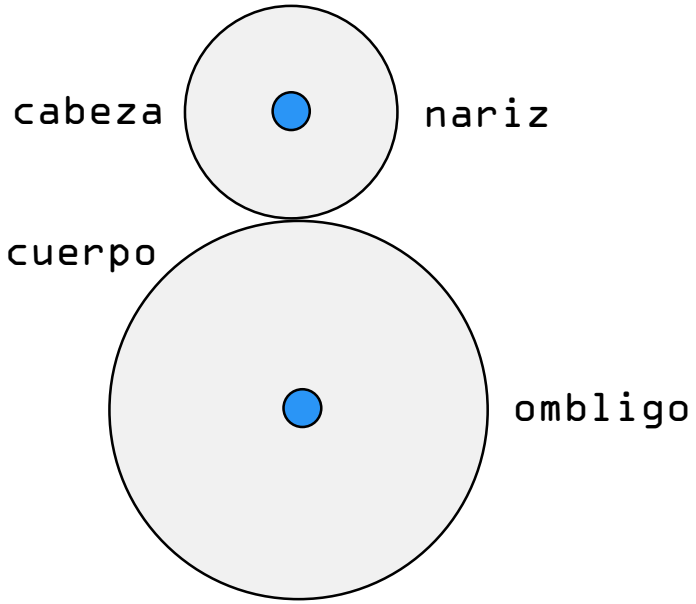




La cabeza puede girar hacia los lados, incluyendo la nariz y el cuerpo con el ombligo
El cuerpo puede girar de manera independiente

- Muñeco: añadimos una articulación en el cuello





La cabeza puede girar hacia los lados, incluyendo la nariz, independientemente del cuerpo
El cuerpo puede girar, incluyendo el ombligo, independientemente de la cabeza

IG2App:: setupScene()

```
//Añadir elementos a la escena
Ogre::Entity* sinbad = mSM->createEntity("Sinbad.mesh");
Ogre::SceneNode* sinbadNode = mSM -> createSceneNode("nSinbad");
mSM->getRootSceneNode()->addChild(sinbadNode);
sinbadNode->attachObject(sinbad);

sinbadNode->scale(5, 5, 5);    //TS_LOCAL (defecto)
sinbadNode->translate(5, 0, 25);  //TS_PARENT, TS_WORLD, TS_LOCAL
sinbadNode->yaw(Ogre::Degree(-45));  //TS_LOCAL, TS_PARENT, TS_WORLD
sinbadNode->showBoundingBox(true);
```

- Cada tipo de transformación se guarda por separado

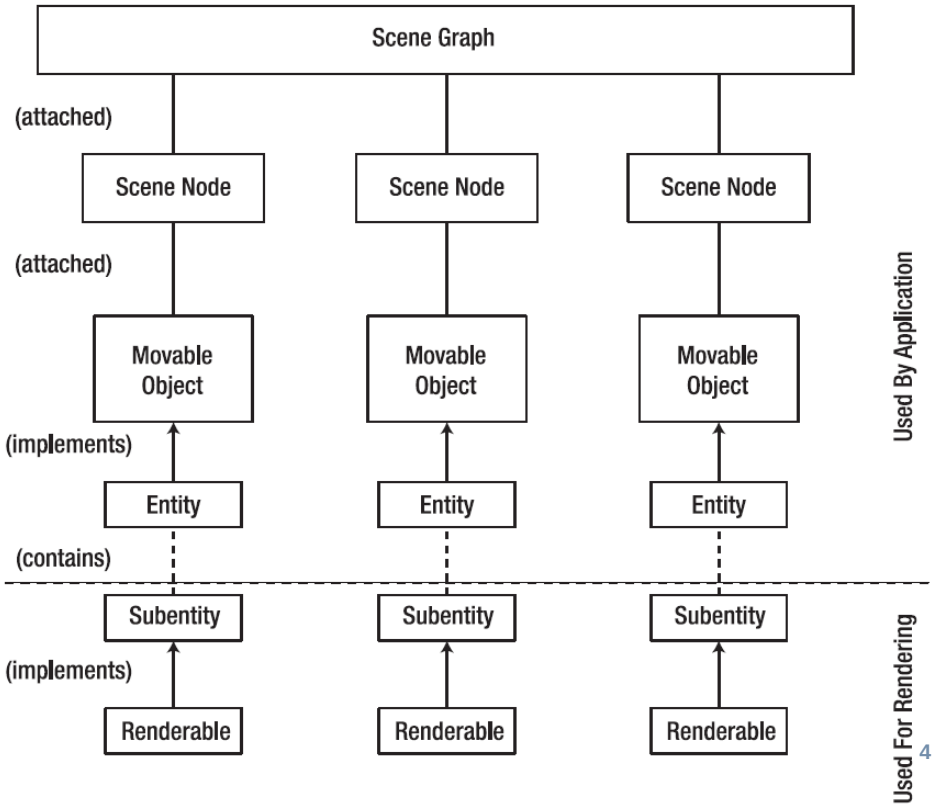
  posición, orientación y escala

  A partir de las tres partes y el árbol, se aplican las transformaciones y se genera la matriz de cada nodo según el orden

  T*R*S*vertex

Scene graph

- El grafo de la escena tiene todos los nodos del mismo tipo **SceneNode**. Es un árbol general con un nodo raíz (**RootSceneNode**)
- Un nodo contiene objetos (de la clase abstracta **MovableObject**) y una transformación de modelado común para todos:
  Translate (**position**), **Rotate** (**orientation**) y **Scale**.
- **MovableObject** es
  Implementada por **Light**,
  Camera, **Entity** (subentities)
- **SubEntity** (malla, material) implementa **Renderable**



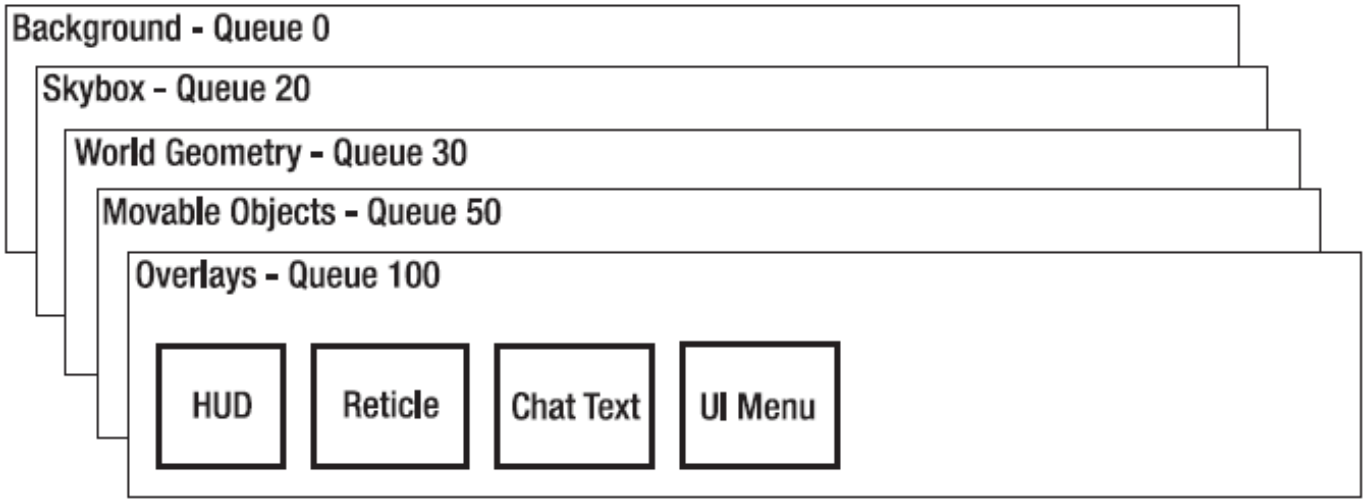
Rendering

- Una entidad está formada por varias subentidades.

  Cada subentidad contiene una referencia a una submalla y una referencia a un material.

- Las subentidades son elementos susceptibles de ser renderizados (**Renderable**).

  El gestor de la escena forma colas (Z-order) con estos elementos en el proceso de renderizado (transparentes, opacos, estáticos)



Ogre initialization

```
#include "IG2App.h"
int main(int argc, char *argv[])
{
        IG2App app;
        app.initApp();

        app.getRoot()->startRendering();

        app.closeApp();
        return 0;
}
```

```
...
while ( !mQueuedEnd )
{
   mQueuedEnd = renderOneFrame();
}
```

Ogre initialization

- **Render Loop:** root->startRendering();

```
mActiveRenderer->_initRenderTargets();
clearEventTimes();
mQueuedEnd = false;
// Infinite loop, until broken out of by frame listeners
// or break out by calling queueEndRendering()
while( !mQueuedEnd )
{
   mQueuedEnd = renderOneFrame();
}
```

```
_fireFrameStarted();  // avisa a los FrameListener registrados  -> pollEvents();
_updateAllRenderTargets();  // el renderizado -> frameRenderingQueued();
_fireFrameEnded();  // avisa a los FrameListener registrados
```