Transformaciones sobre nodos

- Los nodos de la escena admiten tres tipos de transformaciones de modelado
  - Traslaciones para situar el nodo en un lugar determinado de la escena
  - Rotaciones para situar el nodo con una orientación determinada
  - Escalaciones para mostrar la entidad asociada a un nodo con un tamaño determinado
- Cada una de estas transformaciones admite, además, distintos comandos y muchos de ellos se pueden invocar con respecto a distintos espacios o sistemas de coordenadas

Traslaciones de nodos

- Los nodos se pueden trasladar de dos formas
  - **`node->setPosition(x, y, z)`**: traslada el nodo al punto de coordenadas **`(x, y, z)`**. Estas coordenadas son relativas a las coordenadas del padre del nodo. Por ejemplo, si **`node`** está en las coordenadas **`(1, 2, 3)`** y hacemos **`node->setPosition(4, 5, 6)`**, entonces pasa a estar en **`(4, 5, 6)`**
  - **`node->translate(x, y, z)`**: traslada el nodo a la posición resultante de sumarle a su posición, el vector **`(x,y,z)`**. Por ejemplo, si **`node`** está en las coordenadas **`(1, 2, 3)`** y hacemos **`node->translate(4, 5, 6)`**, entonces pasa a estar en **`(5, 7, 9)`**

Rotaciones de nodos

- Los nodos se pueden rotar de tres formas (en el parámetro es obligatorio poner los grados mediante **`Degree`** o **`Radian`**)
  - **`node->pitch(Radian(Math::HALF_PI))`**: gira **`node`** 90º, o lo que diga el parámetro, en sentido anti-horario con respecto al eje **`X`** local de **`node`**
  - **`node->yaw(Degree(90.0f))`**: gira **`node`** 90º en sentido anti-horario con respecto al eje **`Y`** local de **`node`**
  - **`node->roll(Ogre::Radian(Ogre::Math::HALF_PI))`**: gira **`node`** 90º en sentido anti-horario con respecto al eje **`Z`** local de **`node`**

  Por ejemplo, si **`node`** tiene un hijo **`node2`** y ambos nodos contienen a **`Sinbad.mesh`** y están situados en el mismo punto entonces:

```
node2->translate(10, 0, 0);
node2->pitch(Radian(Math::HALF_PI));
```

Escalaciones de nodos

- Los nodos se pueden escalar de dos formas
  - **`node->setScale(sx, sy, sz)`**: escala el nodo según los factores de escalación de **`sx`** para el eje **`X`**, **`sy`** para el eje **`Y`** y **`sz`** para el eje **`Z`**. Obviamente lo que se escala es la entidad adjunta al nodo, no el nodo mismo
  - **`node->scale(sx, sy, sz)`**: escala el nodo según los factores **`sx`**, **`sy`** y **`sz`**. La diferencia con el comando anterior son las mismas que las que hay entre **`setPosition()`** y **`translate()`**. Es decir, **`setScale()`** escala de forma absoluta y **`scale()`** lo hace de forma relativa al tamaño ya existente, que puede estar escalado por estarlo el padre, por ejemplo

Ejemplo

- //Se crea **Sinbad** y se sitúa, escalado, en la posición **(10, 10, 0)**

```
SceneNode* node = mSM->getRootSceneNode()
     ->createChildSceneNode();
Entity* ent = mSM->createEntity("Sinbad.mesh");
node->attachObject(ent);
node->setPosition(10, 10, 0);
node->setScale(50, 50, 50);
```

//Se crea **ninja** en un hijo de **node** y se sitúa, escalado, detrás de **Sinbad**

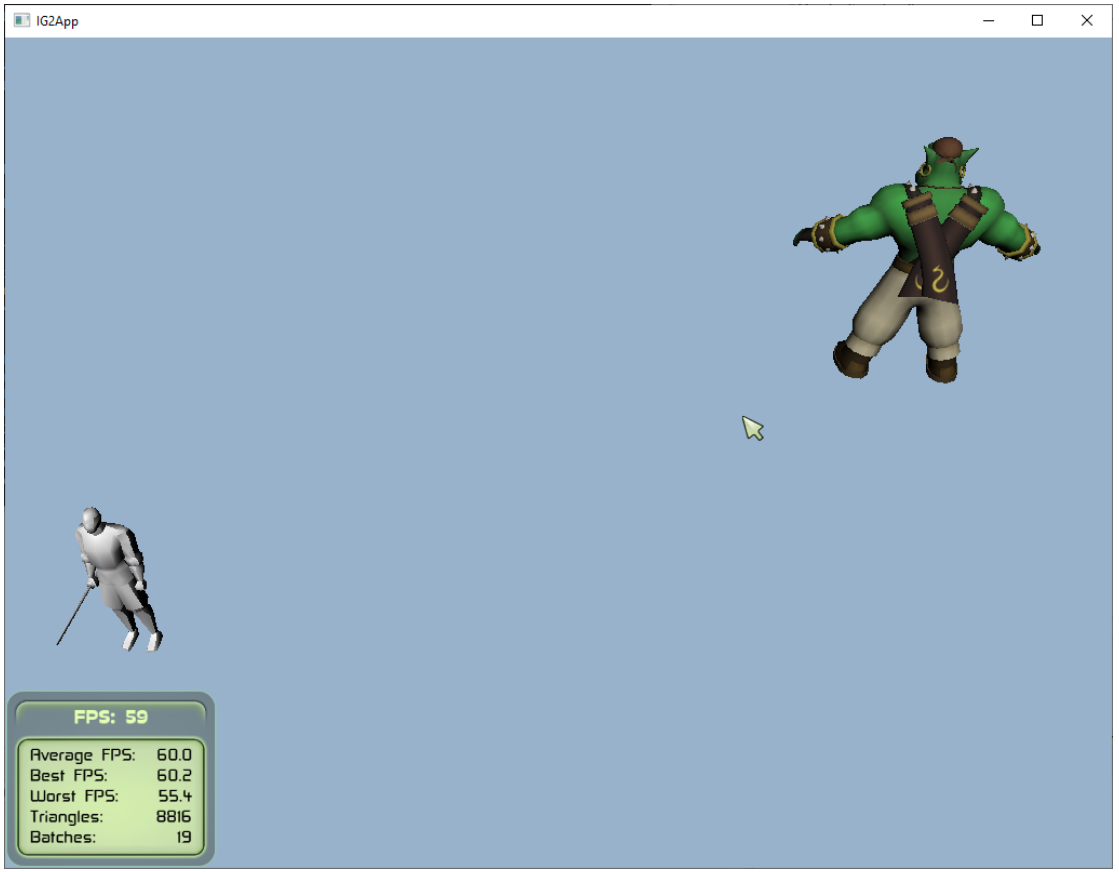
//ninja se crea siempre de espaldas

```
SceneNode* node2 = node->createChildSceneNode();
ent = mSM->createEntity("ninja.mesh");
node2->attachObject(ent);
node2->translate(20, 0, -20);
node2->scale(0.02, 0.02, 0.02);
```

Ejemplo

- Ahora hacemos:

```
node->yaw(Radian(Math::PI));
```



Espacios de coordenadas

En Ogre se puede especificar con respecto a qué espacio de coordenadas se realizan las transformaciones de modelado de un nodo:

- **World space (TS_WORLD):** con respecto al espacio de coordenadas global (el del nodo raíz, si no se ha modificado).
- **Parent space (TS_PARENT):** con respecto al espacio de coordenadas del padre del nodo. Este sistema incluye todas las rotaciones desde la raíz del árbol al padre del nodo.
- **Local space (TS_LOCAL):** con respecto al espacio de coordenadas del nodo. Este incluye todas las rotaciones desde la raíz del árbol al propio nodo.

Lo más habitual es realizar las **traslaciones con respecto al padre** (por defecto en Ogre), **rotaciones con respecto al local** (por defecto en Ogre) y **escalaciones siempre con respecto al local**

Las coordenadas de los vértices nunca cambian, siempre están en el **espacio del objeto (local)**

Transformaciones

- Traslaciones

```
sceneNode->translate(100.0, 10.0, 0.0);  //Por defecto TS_PARENT
sceneNode->translate(100.0, 10.0, 0.0, Ogre::Node::TS_WORLD);
sceneNode->translate(0.0, 0.0, 100.0, Ogre::Node::TS_LOCAL);
```

- Escalaciones

```
sceneNode->scale(2.0, 1.0, 1.0); //Escala en el eje X por un factor de 2
```

- Otros métodos

```
rotate(…); //Por defecto TS_LOCAL
setPosition (…); //Por defecto TS_PARENT
setOrientation(quaternion);
setRotation(…);
```

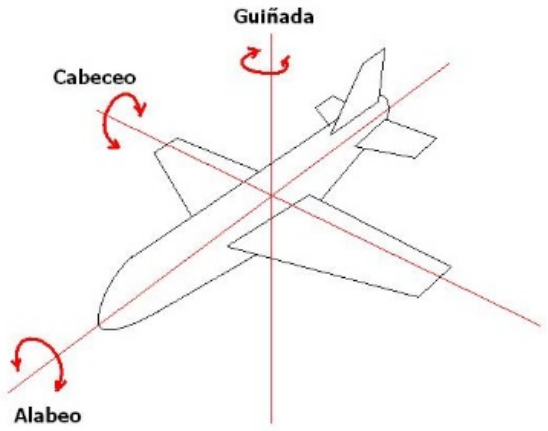
Transformaciones

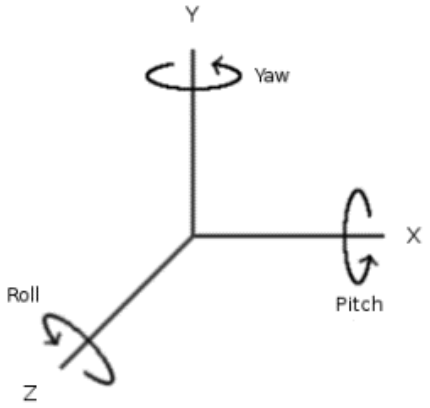
- Rotaciones

```
sceneNode->yaw(Ogre::Radian(1.0));
//Por defecto TS_LOCAL: rota alrededor del eje Y del objeto

sceneNode->pitch(Ogre::Radian(1.0), Ogre::Node::TS_PARENT);
//Rota alrededor del eje X del padre

sceneNode->roll(Ogre::Degree(-45), Ogre::Node::TS_WORLD);
//Rota alrededor del eje Z del mundo
```





Transformaciones

- Se puede especificar que un nodo no se vea afectado por algunas transformaciones del nodo padre.

```
sceneNode-> setInheritOrientation (bool inherit);
// default true: this node's orientation will be affected by its parent's orientation

sceneNode-> setInheritScale (bool inherit);
// default true: this node's scale will be affected by its parent's scale
```

- Podemos reiniciar las transformaciones de un nodo:

```
sceneNode->resetToInitialState();
sceneNode->setInitialState();  // Útil para animaciones
sceneNode->resetOrientation()
```

Ejemplo

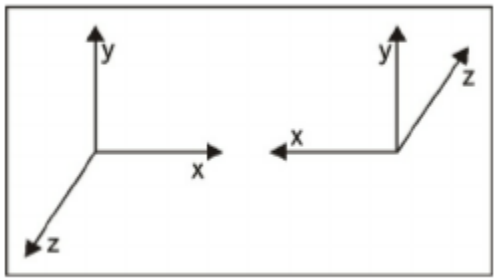
- **Traslaciones con respecto al espacio global**

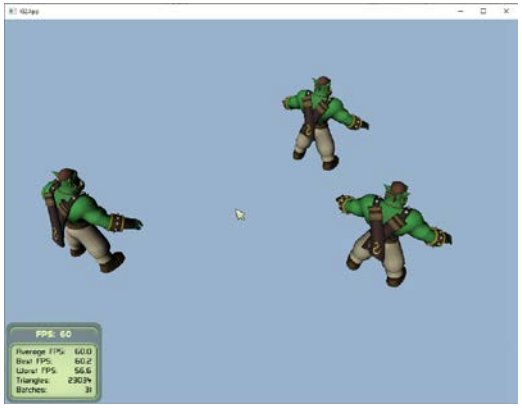
```
SceneNode* node = mSM->getRootSceneNode()
        ->createChildSceneNode();
Entity* ent = mSM->createEntity("Sinbad.mesh");
node->attachObject(ent);
node->setPosition(0, 0, 100);
//Las escalaciones (node) siempre son con respecto al espacio local (node)
node->scale(50, 50, 50);
//Las rotaciones (node) son, por defecto, con respecto al espacio local (node)
node->yaw(Degree(180.0));
SceneNode* node2 = node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node2->attachObject(ent);
node2->setPosition(10, 0, 0);
//Las traslaciones (node2) son, por defecto, con respecto al padre (node)
node2->translate(0, 0, 10);
SceneNode* node3 = node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node3->attachObject(ent);
node3->setPosition(20, 0, 0);
node3->translate(0, 0, 10);
```

Ejemplo. Traslación con respecto al espacio global

Traslaciones con respecto al espacio global

```
SceneNode* node = mSM->getRootSceneNode()
        ->createChildSceneNode();
Entity* ent = mSM->createEntity("Sinbad.mesh");
node->attachObject(ent);
node->setPosition(0, 0, 100);
node->scale(50, 50, 50);
node->yaw(Degree(180.0));
SceneNode* node2 = node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node2->attachObject(ent);
node2->setPosition(10, 0, 0);
node2->translate(0, 0, 10);
SceneNode* node3 = node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node3->attachObject(ent);
node3->setPosition(20, 0, 0);
node3->translate(0, 0, 500, Ogre::Node::TS_WORLD);
```





Ejemplo. Traslación con respecto al espacio local

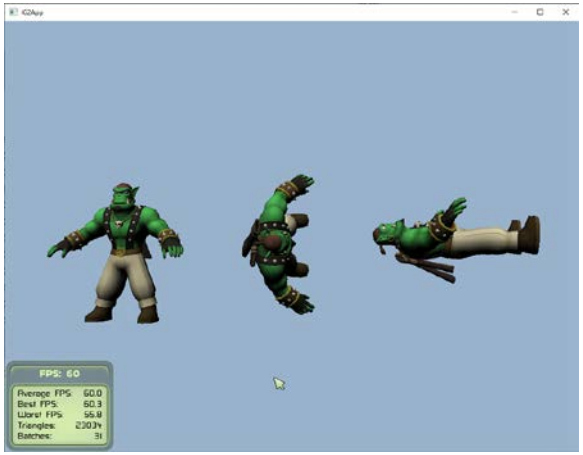
Traslaciones con respecto al espacio local

```
SceneNode* node = mSM->getRootSceneNode()->createChildSceneNode();
Entity* ent = mSM->createEntity("Sinbad.mesh");
node->attachObject(ent);
node->setPosition(0, 0, 400);
node->scale(50, 50, 50);
node->yaw(Degree(180.0));
SceneNode* node2 =
        node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node2->attachObject(ent);
node2->yaw(Ogre::Degree(45));
node2->translate(0, 0, 20);
SceneNode* node3 =
        node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node3->attachObject(ent);
node3->yaw(Ogre::Degree(45));
node3->translate(0, 0, 20, Ogre::Node::TS_LOCAL);
```

Ejemplo. Rotación con respecto al espacio global

Rotaciones con respecto al espacio global

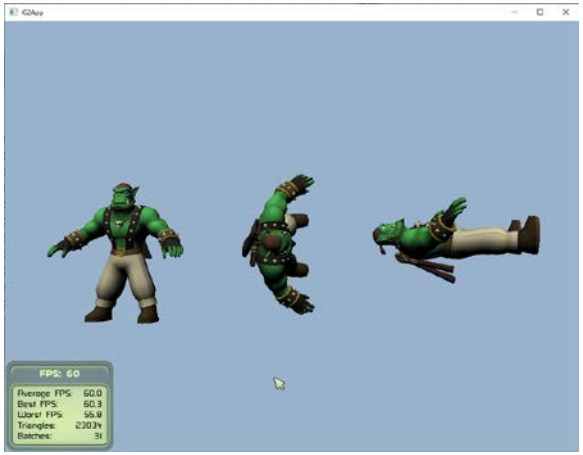
```
SceneNode* node = mSM->getRootSceneNode()->createChildSceneNode();
Entity* ent = mSM->createEntity("Sinbad.mesh");
node->attachObject(ent);
SceneNode* node2 = mSM->getRootSceneNode()->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node2->attachObject(ent);
node2->setPosition(10, 0, 0);
node2->yaw(Ogre::Degree(90));
node2->roll(Ogre::Degree(90));
SceneNode* node3 = node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node3->setPosition(20, 0, 0);
node3->attachObject(ent);
node3->yaw(Ogre::Degree(90),
        Ogre::Node::TS_WORLD);
node3->roll(Ogre::Degree(90),
        Ogre::Node::TS_WORLD);
```



Ejemplo. Rotación con respecto al espacio del padre

- **Rotaciones con respecto al espacio del padre**

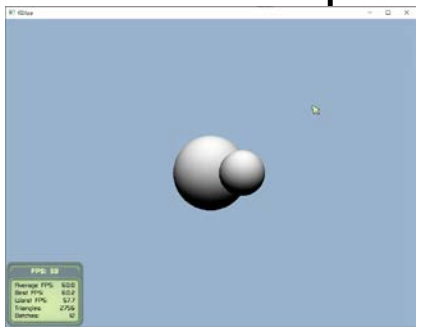
```
SceneNode* node = mSM->getRootSceneNode()->createChildSceneNode();
Entity* ent = mSM->createEntity("Sinbad.mesh");
node->attachObject(ent);
SceneNode* node2 = mSM->getRootSceneNode()->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node2->attachObject(ent);
node2->setPosition(10, 0, 0);
node2->yaw(Ogre::Degree(90));
node2->roll(Ogre::Degree(90));
SceneNode* node3 = node->createChildSceneNode();
ent = mSM->createEntity("Sinbad.mesh");
node3->setPosition(20, 0, 0);
node3->attachObject(ent);
node3->yaw(Ogre::Degree(90),
        Ogre::Node::TS_PARENT);
node3->roll(Ogre::Degree(90),
        Ogre::Node::TS_PARENT);
```



Ejemplo. Rotación de un nodo alrededor de otro

Mover la tierra alrededor del sol. Soluciones:

- Hacer que el nodo de la tierra sea hijo del nodo del sol, rotando el nodo del sol como tratamiento del evento
- El nodo de la tierra **no** es hijo del nodo del sol
  - Reposicionamiento: Reposicionando la tierra (cos, 0, sin) como tratamiento del evento
  - Nodo ficticio: Creando un nodo ficticio que sea hijo del nodo del sol y padre del nodo de la tierra y rotando el nodo ficticio como tratamiento del evento
  - Truco: Haciendo la rotación del nodo de la tierra en el origen, trasladándolo allí antes y volviéndolo a llevar a su órbita después, como tratamiento del evento



Situación inicial

La tierra está en un nodo que **NO** es hijo del nodo del sol

```
solNode = mSM->getRootSceneNode()
    ->createChildSceneNode("sol");
Entity* sol = mSM->createEntity("sphere.mesh");
solNode->attachObject(sol);

tierraNode = mSM->getRootSceneNode()
    ->createChildSceneNode("tierra");
Entity* tierra = mSM->createEntity("sphere.mesh");
tierraNode->attachObject(tierra);
```

Reposicionamiento

En la escena:

```
tierraNode
    ->setPosition(200*Ogre::Math::Cos(Ogre::Degree(phi)),
          0, 200*Ogre::Math::Sin(Ogre::Degree(phi)));
```

En el tratamiento del evento:

```
phi -= 5;
tierraNode
    ->setPosition(200*Ogre::Math::Cos(Ogre::Degree(phi)),
          0, 200*Ogre::Math::Sin(Ogre::Degree(phi)));
```

Nodo ficticio

En la escena:

```
tierraNode2 = mSM->getRootSceneNode()
    ->createChildSceneNode("tierra fake");
tierraNode = tierraNode2->createChildSceneNode("tierra");
Entity* tierra = mSM->createEntity("sphere.mesh");
tierraNode->translate(200, 0, 0);
tierraNode->scale(0.5, 0.5, 0.5);
tierraNode->attachObject(tierra);
```

En el tratamiento del evento:

```
tierraNode2->yaw(Ogre::Degree(-5));
```

El (gran) truco

En la escena:

```
tierraNode->translate(200, 0, 0);
tierraNode->scale(0.5, 0.5, 0.5);
```

En el tratamiento del evento:

```
tierraNode->translate(-200, 0, 0, SceneNode::TS_LOCAL);
tierraNode->yaw(Ogre::Degree(-5));
tierraNode->translate(200, 0, 0, SceneNode::TS_LOCAL);
```