Tutorial de Fotograma

Resumen

Para mas información vea tutoriales básicos, creación básica de objetos y trabajo con relojes.

Las propiedades de los objetos como: posiciones, escalas y rotación son almacenadas en las estructura orxFrame.

Estos marcos estan juntos en la jerarquia de gráficos, lo que significa que las propiedades del marco padre afectarán las propiedades de sus hijos.

En este tutorial, tenemos 4 objetos que serán vinculados a un padre en común ¹⁾ y un quinto sin padre.

Los dos primeros hijos son creados usando las propiedades de objetos de la configuración ChildList mientras que los otros dos son creados y vínculados desde el código (por propositos didácticos). El objeto padre invisible seguira el puntero del ratón. Las teclas "SHIFT IZQUIERDA" y "CONTROL IZQUIERDA" cambiarán la escala arriba y abajo del objeto padre, mientras que el click "IZQUIERDO" y "DERECHO" lo harán rotar.

Todas estas transformaciones afectaran a los 4 hijos.

Esto nos muestra una forma fácil de crear grupos de objetos complejos y transformar sus propiedades (posición, escala, rotación, vélocidad...) fácilmente.

Detalles

Como en el tutorial anterior, comenzamos cargando nuestro fichero de configuración y creando la vista.

```
orxConfig_Load("../03_Frame.ini");
orxViewport_CreateFromConfig("Viewport");
```

Luego creamos nuestro objeto padre.

```
pstParentObject = orxObject_CreateFromConfig("ParentObject");
```

Definimos en el fichero de configuración ParentObject para nuestro objeto padre

```
[ParentObject]
ChildList = Object3 # Object4
```

Así cuando creamos nuestro objeto padre, los dos hijos ²⁾ son además *automagicamente* creados y vínculados.

Esto podiamos haberlo hecho en el fichero de configuración, pero para propositos de aprendizaje, crearemos y víncularemos los otros dos objetos en el código.

```
orxOBJECT *pstObject;
orxObject_CreateFromConfig("Object0");
pstObject = orxObject_CreateFromConfig("Object1");
orxObject_SetParent(pstObject, pstParentObject);
pstObject = orxObject_CreateFromConfig("Object2");
orxObject_SetParent(pstObject, pstParentObject);
```

El Object0 es nuestro objeto estático: el único que no estará vinculado a nuestro ParentObject (Objeto padre).

Note que cuando creamos y vínculamos manualmente los objetos en el código, es nuestro deber borrarlos luego. En el caso del Object3 y Object4 serán automáticamente eliminados cuando ParentObject sea eliminado.

A continuación, creamos un reloj de 100 Hz y registramos nuestra función Update. En esta función es donde vamos a gestionar las entradas para escalar / rotar el ParentObject y asegurarnos de que va a seguir a nuestro cursor del ratón.

```
pstClock = orxClock_Create(orx2F(0.01f), orxCLOCK_TYPE_USER);
orxClock_Register(pstClock, Update, orxNULL, orxMODULE_ID_MAIN, orxCLOCK_PRIORITY_NORMAL);
```

Ahora vamos a echar un vistazo a nuestra función Update.

En primer lugar, nos aseguramos de que podemos encontrar la posición en nuestro espacio del mundo que corresponde a nuestro cursor del ratón en el espacio de la pantalla.

A continuación, copiamos nuestra coordenada Z de ParentObject (es decir, mantenemos la misma profundidad que antes) sobre el mismo y finalmente la establecemos en nuestro ParentObject.

```
if(orxRender_GetWorldPosition(orxMouse_GetPosition(&vPosition), &vPosition))
{
   orxVECTOR vParentPosition;
   orxObject_GetWorldPosition(pstParentObject, &vParentPosition);
   vPosition.fZ = vParentPosition.fZ;
   orxObject_SetPosition(pstParentObject, &vPosition);
}
```

Lo único que queda por hacer es aplicar la escala y la rotación de acuerdo a nuestras entradas. En nuestro caso, definimos las siguientes entradas en 03_Frame.ini: RotateLeft, RotateRight, ScaleUp and ScaleDown.

Veamos como las manejamos. Primero, las rotaciones.

```
if(orxInput_IsActive("RotateLeft"))
{
  orxObject_SetRotation(pstParentObject,
  orxObject_GetRotation(pstParentObject) + orx2F(-4.0f) * _pstClockInfo->fDT);
}
if(orxInput_IsActive("RotateRight"))
{
  orxObject_SetRotation(pstParentObject,
```

```
orxObject_GetRotation(pstParentObject) + orx2F(4.0f) * _pstClockInfo->fDT);
}
```

Y ahora, las escalas.

```
if(orxInput_IsActive("ScaleUp"))
{
   orxObject_SetScale(pstParentObject, orxVector_Mulf(&vScale,
   orxObject_GetScale(pstParentObject, &vScale), orx2F(1.02f)));
}
if(orxInput_IsActive("ScaleDown"))
{
   orxObject_SetScale(pstParentObject, orxVector_Mulf(&vScale,
   orxObject_GetScale(pstParentObject, &vScale), orx2F(0.98f)));
}
```

Es todo!



Nuestro ParentObject será mejorado junto a todos sus hijos.

PD:

- Podríamos haber utilizado los valores de la configuración en lugar de las constantes para los valores de rotación y escala. De esta manera, podríamos cambiarlos sin tener que recompilar e incluso ponerlos al día en tiempo real pulsando BackSpace. 3)
- Como usamos el DT del reloj para las rotaciones, se beneficiarán de la consistencia del tiempo
 4). Por desgracia, no será el caso para las escalas. (Que suele ser algo que realmente no queremos!)

Recursos

```
Código fuente: 03 Frame.c
```

Fichero de configuración: 03 Frame.ini

un objeto vacío, sin contenido visual
Object3 y Object4

Clave por defecto en el lanzador orx para recargar la configuración

no dependerán de la velocidad de los fotogramas y será el momento-elástico

From:

https://orx-project.org/wiki/ - Orx Learning

Permanent link:

https://orx-project.org/wiki/es/orx/tutorials/frame?rev=1330523480

Last update: 2025/09/30 17:26 (4 weeks ago)

