Tutorial de Reloj

Resumen

Vea el Tutorial de Objeto para más información para la creación básica de un objeto.

En este tutorial vamos a registrar el proceso de llamada de retorno en dos relojes diferentes solamente para propósitos didácticos. Todos los objetos serán actualizados desde el mismo reloj. [1)]

El primer reloj corre a 0.01s por tictac (100 Hz) y el segundo corre a 0.2s por tictac (5 Hz).

Sí presionas las teclas ARRIBA, ABAJO y DERECHA, podrás alterar el tiempo del primer reloj. Este será actualizado al mismo tiempo, pero el tiempo para la llamada de retorno del reloj será modificado.

Esto permite de forma fácil hacer distorsión al tiempo y tener varias partes de la lógica actualizándose en diferentes frecuencias. Un reloj puede tener tantas llamadas de retornos registradas como quieras, cada una con un parámetro de contexto opcional.

Por ejemplo, el contador FPS mostrado en la esquina arriba izquierda es calculado con un reloj no-alterado que corre a 1Hz.

Detalles

Cuando usamos orx, no necesitamos escribir un ciclo

```
while(1){}
```

global para actualizar nuestra lógica. En su lugar lo que haremos es crear un reloj [2)], especificando su frecuencia de actualización.

Podemos crear cuantos relojes queramos, debemos asegurarnos que la parte más importante de nuestra lógica (jugadores, enemigos...) serán actualizados muy frecuentemente, mientras que el código de baja prioridad (objetos no interactivos, fondos...) será llamado una vez en el ciclo (while(1){}). Por ejemplo, la física y la representación de gráficos usan dos relojes diferentes que tiene frecuencias distintas.

Existe otra gran ventaja al usar varios relojes pues podemos fácilmente obtener distorsión del tiempo en algunos elementos mientras que otros se actualizan en otra frecuencia.

En este tutorial, crearemos dos relojes, uno que corre a 100Hz (período = 0.01s) y otro a 5Hz (período = 0.2s).

```
orxCLOCK *pstClock1, *pstClock2;

pstClock1 = orxClock_Create(orx2F(0.01f), orxCLOCK_TYPE_USER);

pstClock2 = orxClock_Create(orx2F(0.2f), orxCLOCK_TYPE_USER);
```

Note que pasamos el tipo orxCLOCK_TYPE_USER para poder obtener el reloj desde cualquier lugar de nuestro código (si no queremos almacenarlo). Cualquier valor de frecuencia por encima de este, es válido. Los más bajos son reservados para uso interno del motor.

Ahora usaremos el mismo actualizador de llamada de retorno para los dos relojes. Sin embargo, vamos a definir diferentes contextos, por lo tanto el primer reloj modificará al primer objeto y el segundo reloj al otro objeto:

```
orxClock_Register(pstClock1, Update, pstObject1, orxMODULE_ID_MAIN,
orxCLOCK_PRIORITY_NORMAL);

orxClock_Register(pstClock2, Update, pstObject2, orxMODULE_ID_MAIN,
orxCLOCK_PRIORITY_NORMAL);
```

Esto significa que nuestra llamada de retorno será ejecutada 100 veces por segundo con pstObject1 y el segundo será ejecutado 5 veces por segundo con el objeto pstObject2.

Como nuestro actualizador de llamada de retorno solamente rota el objeto que se obtiene del parámetro “contexto”, tendremos como resultado dos objetos que rotan a la misma velocidad. Sin embargo, la rotación del segundo objeto se hace en mayor tiempo (5 Hz) que el primero (100 Hz) por lo que obtenemos diferentes velocidades de rotación.

Ahora veamos el código de la llamada de retorno.

Lo primero: necesitamos obtener nuestro objeto desde el parámetro “contexto”.
Como orx usa OOP en C, necesitamos castearlo usando un ayudante de casteo que verificará el objeto.

```
pstObject = orxOBJECT(_pstContext);
```

Sí retorna NULL, el parámetro es incorrecto o no es un orxOBJECT.

Nuestro próximo paso será aplicar rotación al objeto.

```
orxObject_SetRotation(pstObject, orxMATH_KF_PI * _pstClockInfo->fTime)
```

Veamos que aquí usaremos el valor tomado de nuestro reloj.
Esto es porque toda nuestra lógica está relacionada con los relojes.

Por supuesto, existe una mejor forma de hacer rotar a un objeto [3)].
Pero volvamos a lo que nos interesa: reloj y distorsión del tiempo!

En nuestro actualizador de llamada de retorno, además encuestaremos las entradas activas. Las entradas son cadenas de caracteres que están atadas, en cada fichero de configuración o por código, a teclas presionadas, botones del ratón o incluso botones de joystick.

En nuestro caso, si las teclas “arriba” o “abajo” son presionadas, cambiaremos el valor del tiempo para el primer reloj que habíamos creado.
Si “derecha” o “izquierda” son presionadas, pondremos el reloj a su frecuencia original.

Como no guardamos el primer reloj creado [4)], necesitamos saber cómo obtenerlo devuelta!

```
pstClock = orxClock_FindFirst(orx2F(-1.0f), orxCLOCK_TYPE_USER);
```

Esto retornará el primer reloj creado con el tipo `orxCLOCK_TYPE_USER` y el período `1.0`, el cual actualiza nuestro primer objeto.

Ahora, si presionamos las teclas explicadas, cambiaremos la velocidad del reloj en un factor de 4x.

```
if(orxInput_IsActive("Faster"))
{
  /* "Arriba" hace ir al reloj 4 veces más rápido */
  orxClock_SetModifier(pstClock, orxCLOCK_MOD_TYPE_MULTIPLY, orx2F(4.0f));
}
```

De la misma manera hacemos esto para definir un factor de 4x pero más lento.

```
else if(orxInput_IsActive("Slower"))
{
  /* "Abajo" hace ir al reloj 4 veces más lento */
  orxClock_SetModifier(pstClock, orxCLOCK_MOD_TYPE_MULTIPLY, orx2F(0.25f));
}
```

Por último, queremos poner el tiempo normal cuando presionemos la tecla asignada.

```
else if(orxInput_IsActive("Normal"))
{
  /* Remueve los modificadores del reloj */
  orxClock_SetModifier(pstClock, orxCLOCK_MOD_TYPE_NONE, orxFLOAT_0);
}
```

Como puedes ver, la distorsión del tiempo puede hacerse con solo una línea de código. Como nuestra parte lógica se encarga de rotar el objeto modificado por el tiempo, vemos la rotación del primer objeto cambiar basándose en el valor modificado del reloj.

Un ejemplo práctico es usar un reloj distinto para los enemigos y otro para el jugador, permitiendo enlentecer a los enemigos mientras el jugador se mueve a la misma velocidad.

Existen otros tipos de modificadores de reloj pero los veremos más adelante.

Recursos

Código fuente: 02_Clock.c

Fichero de configuración: 02_Clock.ini

1)
El contexto del reloj es además usado aquí solo para demostración
2)
o usamos uno existente, como el del núcleo o el reloj de la física
3)
asignándole una velocidad angular por ejemplo o usando un `orxFX`

4)

solo para mostrar cómo recuperarlo