Object Lifetime

Objects have the concept of a lifetime. This is where you can specify a value in seconds, and the object will live only for that amount of time. This is perfect for spawned objects like bullets, or temporary objects.

But you can also specify one or more literals to create lifetime situations that are a little richer.

Before getting into that, let's create a blank project using the init script.

Numeric Lifetime

Let's begin will the most straight forward example, a numeric lifetime. Add a lifetime of 5.0 to the `Object` section:

```
[Object]
Graphic         = @
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = 5.0
```

Compile and run. The logo object will turn for 5 seconds and then disappear. It's dead. Completely gone.

FX Lifetime

The `fx` literal means that the object's life will only end after any fx that are attached end first.

Change the object's lifetime to `fx`:

```
[Object]
Graphic         = @
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = fx
```

The fx currently attached to the object is in a loop, so this condition will never be reached.

Change the loop property on the `ColorCycle` fx to false:

```
[ColorCycle]
Loop            = false
...
```

Then run that. The logo will cycle its colour once and then its life will end. Nice.

Sound Lifetime

In this case, if there is a sound attached to the object which plays on object creation, the object will die once the sound finishes playing. This is a nice enhancement over time based lifetimes because if you decide to change the sound your object uses, you no longer need to be mindful of how long or short the sound goes for.

```
[Object]
Graphic         = @
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = sound
```

Run that and as soon as the appearance sound is finished, the object's life is over.

Anim Lifetime

For an object, using the `anim` literal means that when all animations have finished playing on your object, then the object will be destroyed.

```
[Object]
Graphic         = @
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = anim
```

Check out the animation tutorials to learn about the animation system and give the `anim` literal a try.

Child Lifetime

For the last three lifetime literal types, I won't go into any code examples. I'll leave that as an exercise for the reader, as they simply follow the same principle.

```
[Object]
Graphic         = @
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = child
ChildList       = Child1 # Child2 # Child3
```

A `child` lifetime is one where the object's lifetime does not end unless the lifetime of its children ends first. For example, you might have a boss alien made up of many parts. Once all the parts are destroyed, the final parent object will finally be removed.

Spawn Lifetime

In a similar manner to the child lifetime literal, a `spawn` lifetime is one where the object's life does not end until it finishes spawning.

```
[Object]
Graphic         = @
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = spawn
Spawner         = MySpawner
```

This operates on the `TotalObjects` property of the spawner. Once the spawned objects are exhausted, the object will die. Clearly if the spawner is set to infinite spawning, the lifetime will not work.

Track Lifetime

Using this literal, the object's life will end when the attached track has finished it's job.

```
[Object]
Graphic         = @
```

```
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = track
Track           = MyTrack
```

Multiple Lifetime literals

You can combine two or more literals. If you do this, the object's lifetime will not end until all of the conditions are exhausted. For example:

```
[Object]
Graphic         = @
Texture         = logo.png
SoundList       = @
Sound           = appear.ogg
Pivot           = center
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
LifeTime        = sound fx
```

In this case, the sound will play and the fx will complete it's single cycle. Once both conditions are complete, it's curtains for the object.

Conclusion

There's a lot of combinations and scenarios that be covered with multiple lifetimes. I can see sound, fx and track being extremely useful.

See also

1. Having a Parent Object die after all children have died
2. orxOBJECT structure

From:
https://www.orx-project.org/wiki/ - **Orx Learning**

Permanent link:
https://www.orx-project.org/wiki/en/tutorials/objects/lifetime?rev=1598923107

Last update: **2025/09/30 17:26 (14 months ago)**