Viewport Render to Texture

An amazing feature of Orx is the ability to have a separate Viewport, display a scene in it, and have that scene become rendered to a separate texture.

We can take this texture and use it for other purposes like for objects.

What could you use this for?

1. Complex effects and composition.
2. Long scrolling text routines.

In order to set up viewport rendering to a texture, you will need the following items as a minimum:

- A second small Viewport, the size of the texture you want to generate
- A camera for the viewport, the size of the texture you want to generate
- A name for your texture to render to
- At least one object in your new camera view (to provide something to see in the viewport)

In order for an object to consume this texture you need the following:

1. A game object that renders to the standard viewport, and has a graphic (as per normal)
2. A graphic that uses our rendered texture by name.

That's it!

Any game object can receive the texture, just like another texture. It's just a texture.

Make a project

Start by creating a project using Orx's init command.

Secondary Viewport and Camera

Once you have a project, start by creating a secondary viewport and camera:

```
[TextureViewport]
Camera           = TextureCamera
BackgroundColor  = (0, 32, 0)
Size             = (250, 250, 0)
RelativePosition = bottom left

[TextureCamera]
FrustumWidth  = 250
FrustumHeight = 250
GroupList     = texturegroup
```

We'll set the `BackgroundColor` to something that isn't transparent, just so we can see the boundaries of the viewport.

The position will be set to the bottom left for the moment so we can see it working.

This viewport camera will only see objects that are set to the `texturegroup` group. So if the camera sees it, it'll be rendered to the texture.

Now to create the `TextureViewport` in the Init() function.

```
orxViewport_CreateFromConfig("TextureViewport");
```

An Object to display in the Viewport Camera

Next we need an object to appear in the second viewport's camera so that there will be something written to the texture other than brown pixels! The default Logo Object from the project generator will do. It is currently defined as:

```
[Object]
Graphic         = @
SoundList       = @
Sound           = appear.ogg
Texture         = logo.png
Pivot           = center
Scale           = 0.5
Position        = (0, 0, 0)
AngularVelocity = 18
FXList          = FadeIn # ColorCycle
```

This will currently render to the regular game viewport. Change it so that is now renders to the secondary viewport:

```
Group = texturegroup
```

`TextureCamera` only sees objects in the group of `texturegroup`.

Run this and the Orx logo will be spinning in the secondary viewport, as a green box in the bottom left hand corner of the screen.

Viewport to Render to Texture instead

To make the viewport render to a texture instead of the screen, we supply the `TextureList` property to the `TextureViewport`:

```
TextureList = GeneratedTexture
```

So that it becomes:

```
[TextureViewport]
Camera          = TextureCamera
BackgroundColor  = (0, 32, 0)
Size            = (250, 250, 0)
RelativePosition = bottom left
TextureList      = GeneratedTexture  ; <- The size of the texture created
will match the viewport size.
```

If you run your code now, you will get a black screen. Believe it or not, things are working well, but there is no way to visually tell anymore.

Adding Game Objects

We're going to make three objects for our game area, and each object will use the generated texture.

We'll start with a overall `Scene` object that uses a track. The track will create our three objects for us, a second apart:

```
[Scene]
TrackList = CreateObjectsTrack

[CreateObjectsTrack]
0 = Object.Create ObjectUsingGeneratedTexture
1 = Object.Create ObjectUsingGeneratedTexture2
2 = Object.Create ObjectUsingGeneratedTexture3
```

Now to define our three game objects:

```
[CreateObjectsTrack]
0 = Object.Create ObjectUsingGeneratedTexture
1 = Object.Create ObjectUsingGeneratedTexture2
2 = Object.Create ObjectUsingGeneratedTexture3

[ObjectUsingGeneratedTexture]
Position  = (-300, -150, 0)
Graphic   = GraphicUsingGeneratedTexture
Scale     = 0.5

[ObjectUsingGeneratedTexture2@ObjectUsingGeneratedTexture]
Position  = (-300, -100, 0)

[ObjectUsingGeneratedTexture3@ObjectUsingGeneratedTexture]
Position  = (-300, -50, 0)
```

We need to define the `GraphicUsingGeneratedTexture` section. This is where it all comes together:

```
[GraphicUsingGeneratedTexture]
Texture = GeneratedTexture
```

Then in code, add the line to create the Scene itself:

```
orxObject_CreateFromConfig("Scene");
```

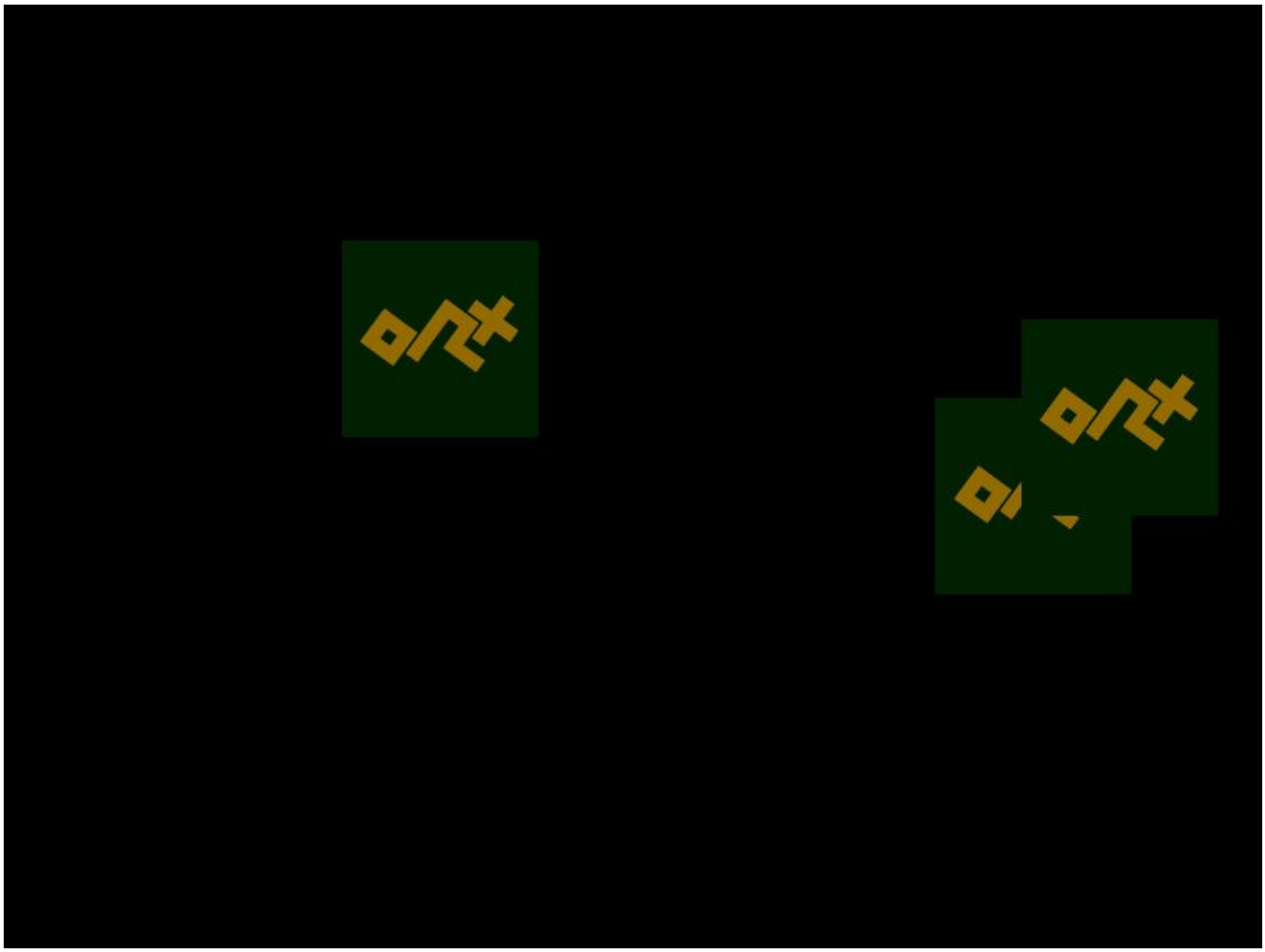
Simple right? The graphic that our three game objects are using is the one that our secondary viewport is generating (`GeneratedTexture`). As the logo turns, the texture is being updated on our game objects. Very neat! Also worth noting that our objects are scaled to 0.5. Like any texture, it will scale with our object.

Now to make things just a little nicer, let's give the objects a nice fade in and some movement:

```
[ObjectUsingGeneratedTexture]
Position  = (-300, -150, 0)
Graphic   = GraphicUsingGeneratedTexture
Scale     = 0.5
FXList    = MovementFX # FadeIn

[MovementFX]
SlotList        = MovementSlotFX
Loop            = true

[MovementSlotFX]
Type            = position
Curve           = sine
StartTime       = 0.0
EndTime         = 4.0
Absolute        = false
StartValue      = (0, 0, 0)
EndValue        = (600, 0, 0)
```



That looks much nicer now. And we're done. You can do amazingly complicated effects with this technique. Enjoy.

References

- orxVIEWPORT structure