UseParentSpace for Relative Object Positioning and Scaling

If an object is a child of another object, the child can be positioned relatively within that parent's space. The property to use is `UseParentSpace`.

Setting that to `true` or `both` means that the object can be positioned using coordinates relative to the parent's size.

Setting up a new project

To help you work through this tutorial, first create a new blank project using the init script.

Start by removing the `[Object]` section from the config file, and everything below it.

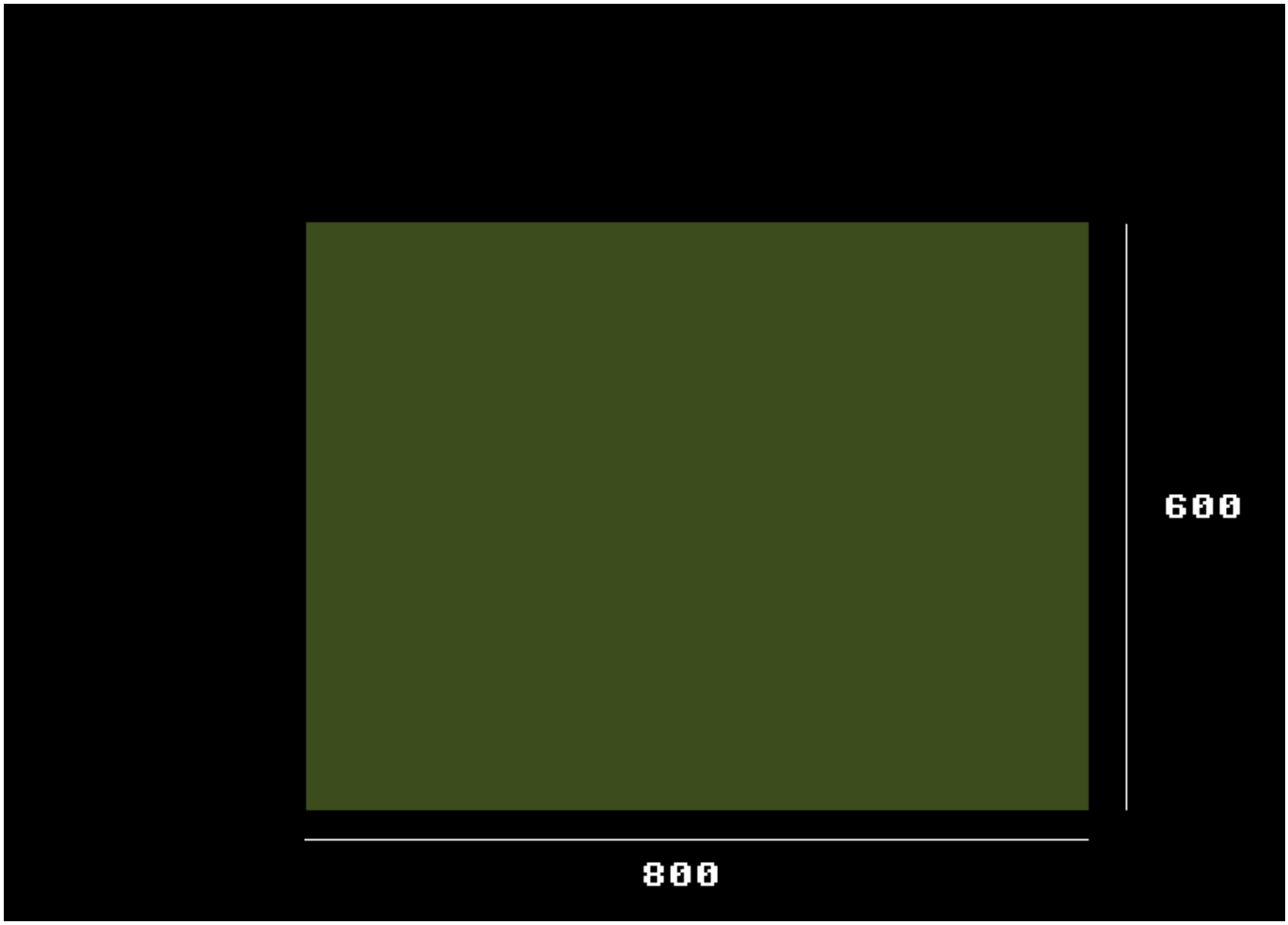
Define a `Container` object:

```
[Container]
Size      = (400, 300)
Graphic   = @
Texture   = pixel
Color     = (60.4, 76.1, 28.2)
Pivot     = center
Scale     = 1
Position  = (0, 0)
ChildList = PanelA
```

This object will be a large solid green block placed in the centre of the screen.

Also change the code to create a `Container` rather than an `Object`:

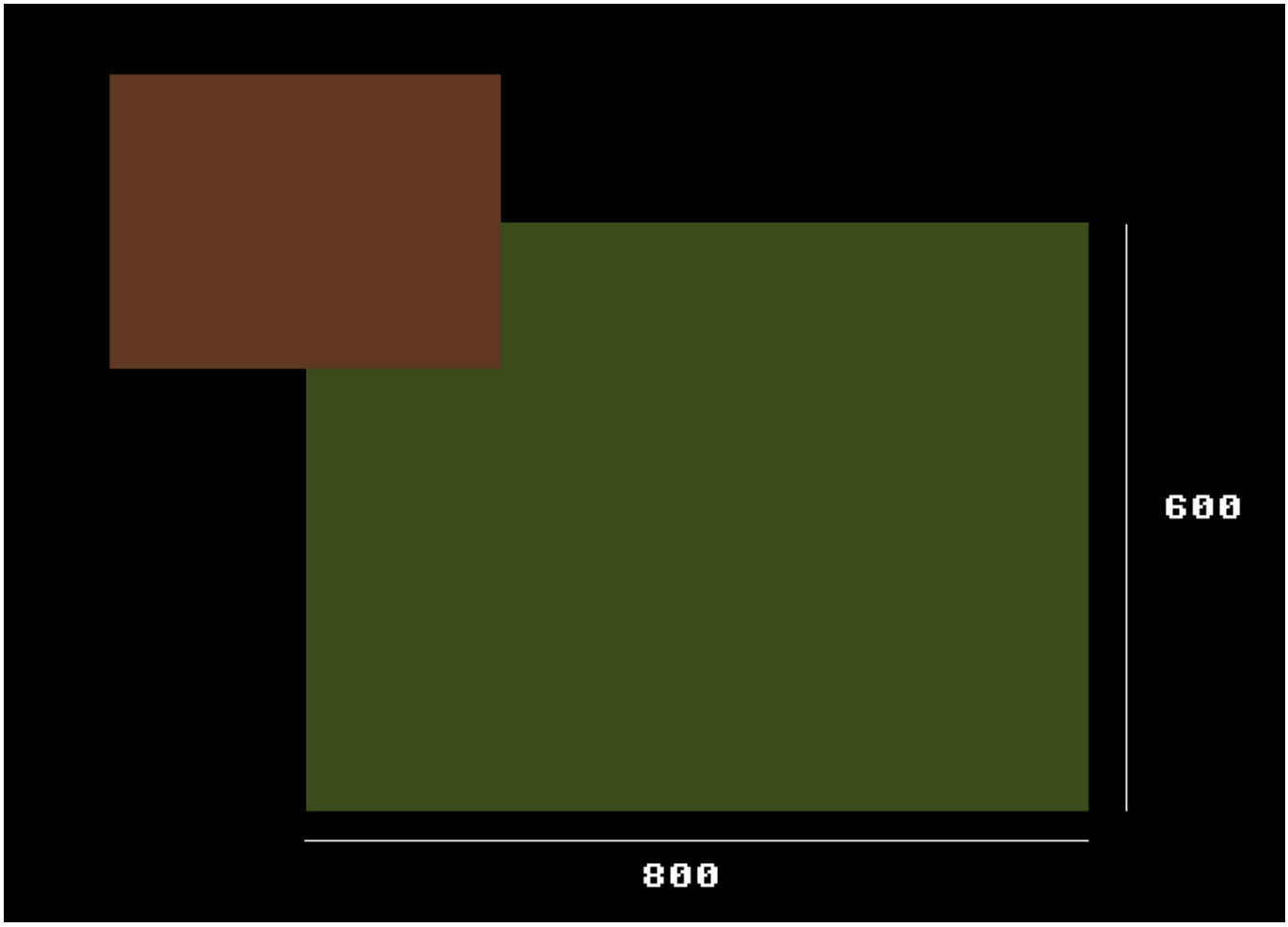
```
orxObject_CreateFromConfig("Container");
```



Note: I've added dimensions and relative position markers. These won't show up on your screen.

`Container` has a single child called `PanelA`. We'll define this so that it gets created in the top left position of the Container:

```
[PanelA]
Graphic        = @
Texture        = pixel
UseParentSpace = both
Pivot          = center
Scale          = 0.5
Color          = (96.1, 56.9, 35.7)
Position       = (-0.5, -0.5, -0.1)
```



PanelA is a brownish block and is placed centred on the top left corner of the Container. PanelA is using UseParentSpace so that it uses the position and scale of the parent. The Scale is set to 0.5 so that the panel becomes half the size of the parent Container.

To emphasise the range of positions, define a number of *TestPoints*:

```
[TestPoint1]
Graphic        = @
Texture        = pixel
UseParentSpace = both
Pivot          = center
Scale          = 0.25
Color          = (50, 50, 50)
Position       = (-0.5, -0.5, -0.1)

[TestPoint2]
Graphic        = @
Texture        = pixel
UseParentSpace = both
Pivot          = center
Scale          = 0.25
Color          = (100, 100, 100)
Position       = (-0.25, -0.25, -0.1)

[TestPoint3]
```

```
Graphic        = @
Texture        = pixel
UseParentSpace = both
Pivot          = center
Scale          = 0.25
Color          = (150, 150, 150)
Position       = (0.0, 0.0, -0.1)

[TestPoint4]
Graphic        = @
Texture        = pixel
UseParentSpace = both
Pivot          = center
Scale          = 0.25
Color          = (200, 200, 200)
Position       = (0.25, 0.25, -0.1)

[TestPoint5]
Graphic        = @
Texture        = pixel
UseParentSpace = both
Pivot          = center
Scale          = 0.25
Color          = (250, 250, 250)
Position       = (0.5, 0.5, -0.1)
```

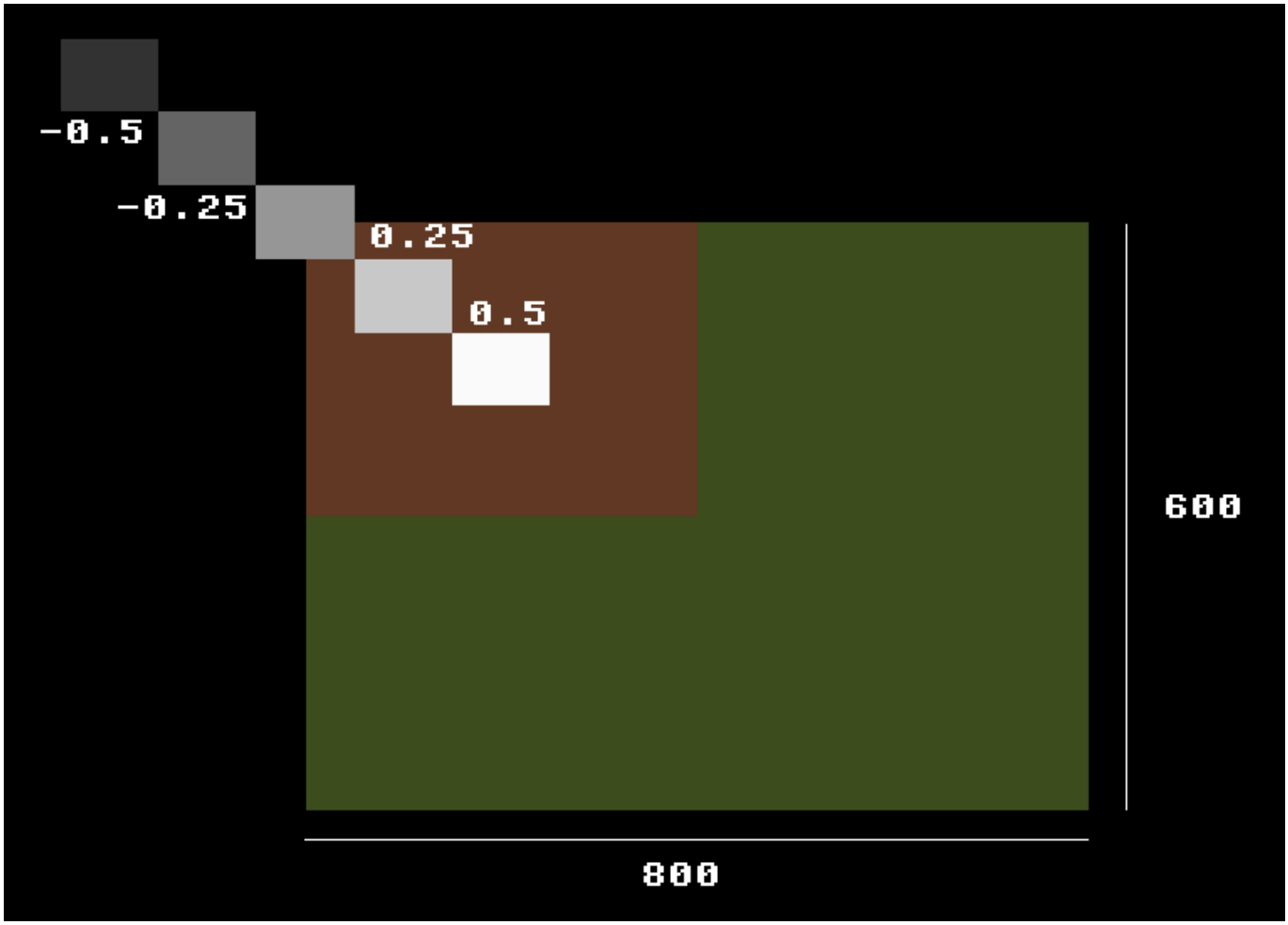
Then add these testpoints to the `ChildList` of `PanelA`:

```
[PanelA]
Graphic        = @
Texture        = pixel
UseParentSpace = both
Pivot          = center
Scale          = 0.5
Color          = (96.1, 56.9, 35.7)
Position       = (-0.5, -0.5, -0.1)
ChildList      = TestPoint1 # TestPoint2 # TestPoint3 # TestPoint4 #
TestPoint5
```

Again, note the scale on the `TestPoint` children. They are all set to 0.25 or a quarter of the size of `PanelA`.

Finally, to demonstrate the effect of using a different pivot on a child (and how it affects grandchildren), change the `Pivot` of `PanelA` to `top left`:

```
[PanelA]
...
Pivot           = top left
...
```



Notice that the `TestPoint` children don't shift their position with the change of pivot on the `PanelA` parent. This is because `PanelA` is just being rendered from a different pivot, but the position space that the TestPoint children operate in is still the same.

So for a parent whose Pivot is centered, the relative top/left and bottom right corners are at `(-0.5, -0.5)` and `(0.5, 0.5)`, which maps to `(-ParentSizeX/2, -ParentSizeY/2)` and `(ParentSizeX/2, ParentSizeY/2)`

Respectively if the parent has a `top left` Pivot, then the relative corners are at `(0, 0)` and `(1, 1)`, which maps to `(0, 0)` and `(ParentSizeX, ParentSizeY)`.

Relative co-ordinates are not limited to the ranges above. They can extend beyond this range to move relatively further away from their parent.

There are lots of applications for relative positioning. Relative screen sizes are handy for simple UI, screen savers, and games where the screen size and scale will vary.