**Making a Shrink/Grow Mailbag in Godot 4 with GDScript**

What is GDScript?

GDScript is a built in scripting language in Godot. It is a bit like Python for users as it uses indentation instead of semicolons, functions begin with func and no semicolons are used. It also has a nice editor integrated so you can write a script and hit play to see immediately what has occured. There is no compile step. Godot is free too as nothing is charged.

I'm making: a magic mailbag from my game, Last Post. The magic mailbag can shrink so it can pass through small openings and can expand to hold mail that is too large. Therefore the script must keep track of the current state of the mailbag and allow for it to be resized by pressing a key. This tutorial is for Godot 4 so the tween code will not be the same for previous versions.

Step 1: Create the scene

1. Create a new project in Godot 4
2. Create a new scene, make root a Node2D and name it MailBag
3. Add a child to the MailBag node of type Sprite2D
4. Assign a texture to that node by dragging in an image. For the purpose of this tutorial the Godot logo will do.
5. Save this scene as mail_bag.tscn

Step 2: Setup the input keys

1. Navigate to Project > Project Settings > Input Map
2. Add two actions: shrink and grow
3. Click the + next to each action and add a key to each. I used Q for shrink and E for grow.

Step 3: Attach script

Right click on the MailBag in the scene and select Attach Script. In the box that appears

paste in the following code

gdscript

```
extends Node2D

enum BagState { NORMAL, SMALL, LARGE }

const SCALES = {
	BagState.NORMAL: Vector2(1, 1),
	BagState.SMALL: Vector2(0.5, 0.5),
	BagState.LARGE: Vector2(1.5, 1.5),
}

var state: BagState = BagState.NORMAL

func _unhandled_input(event):
	if event.is_action_pressed("shrink"):
		toggle_state(BagState.SMALL)
	elif event.is_action_pressed("grow"):
		toggle_state(BagState.LARGE)

func toggle_state(target: BagState):
	# if we toggle the same state twice it returns to NORMAL
	if state == target:
		set_state(BagState.NORMAL)
	else:
		set_state(target)
```

```
func set_state(new_state: BagState):
    state = new_state
    var tween = create_tween()
    tween.tween_property(self, "scale", SCALES[state], 0.15)
```

So what is this script actually doing?

- Using an enum allows each state to be named rather than just assigned a number. This makes the script much more readable.
- The SCALES dictionary maps to each state the desired scale. If a new state is added, it can be done with just one extra line of code.
- The _unhandled_input function will be called automatically each time an event is triggered by the user.
- Using create_tween causes a smooth animation of the scale instead of a abrupt change. With out the tween the mailbag would teleport between states and with the tween it has a very magical feel to it.

Step 4: Test the scene

1. Create a new scene and add the mail_bag.tscn scene into this new scene
2. Press F6 to run this new scene
3. Press Q and E and observe the changes in the mailbag size

- Keys do nothing. Make sure the names used for actions in the input map match exactly to the actions used in the script (shrink and grow)
- Indentation errors. GDScript is picky about tabs and spaces, so just use whatever the default is in the editor
- Collision shapes acting up. If a collision shape is added later make sure it is scaled uniformly. Otherwise issues can occur in the physics of the scene.

What next?

Once this is working a collision shape can be added so that only when the bag is in a small state can it pass through narrow spaces. Also the set_state function can be tied to a signal

so bridges can react to the state of the bag (i.e. heavy or not). This is what I plan to do next for my game, Last Post.