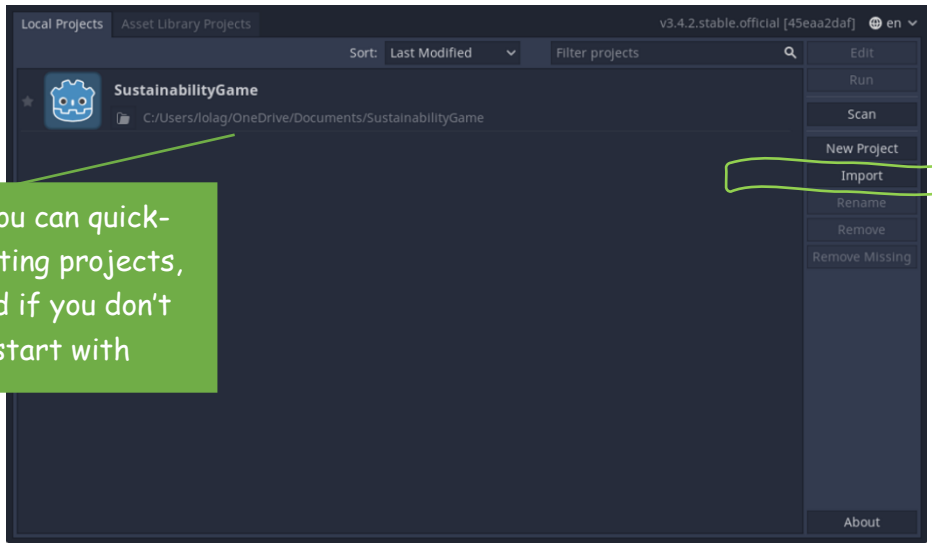
Games and Green Solutions

Table of Contents

Session 1

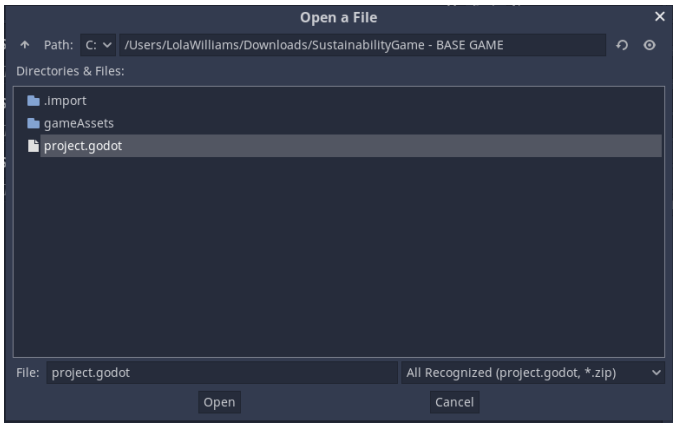
Getting Godot running

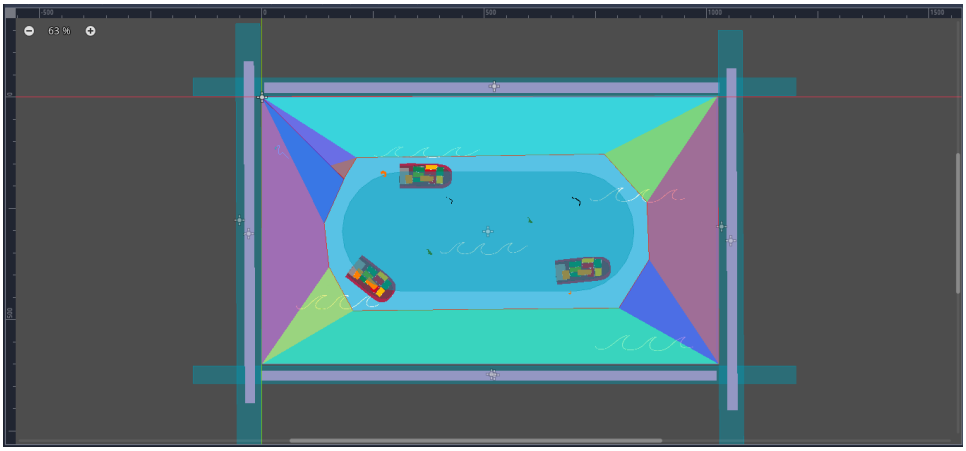
To start off launch the Godot engine, it should open a page that looks somewhat like this



This is where you can quick-access your existing projects, don't be worried if you don't have any to start with

- From the Godot startup menu, click *Import*
- Navigate to the downloaded game folder, and double click on the *project.godot* file found inside

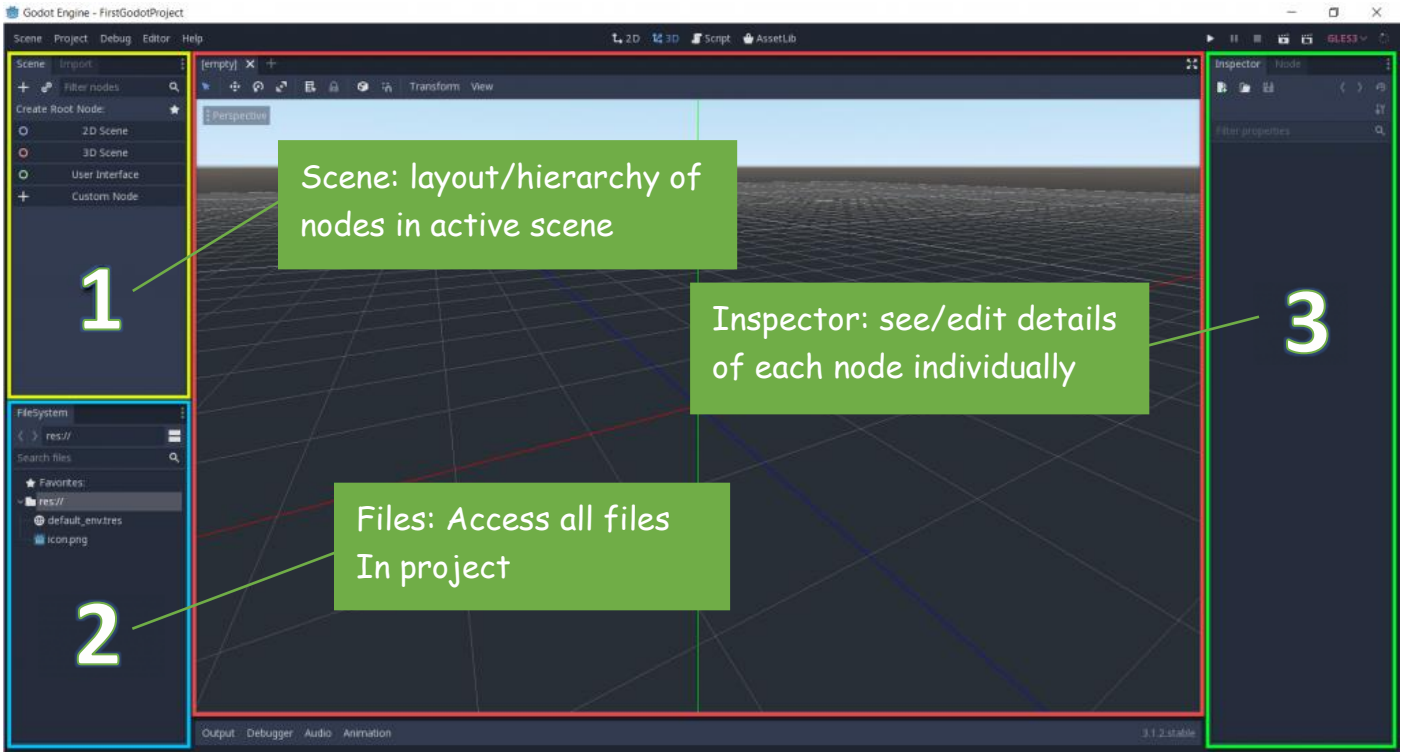




Now that we've opened the project we can see our main scene

A quick overview of the Godot editor

Now that we can view the editor, let's quickly look at the areas that will be most important to us



Configuring our own boat

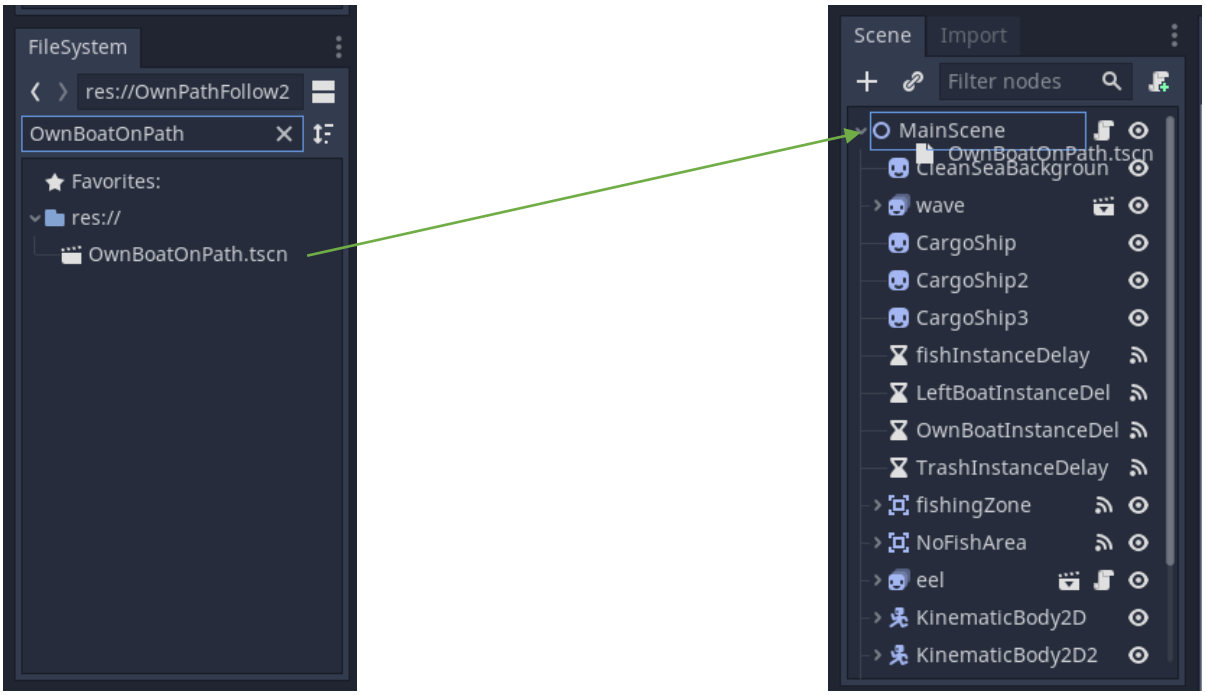
We already have some boats in our game, but they are only decorative. Let's add some interactive fishing boats and start making the game playable.

The logic of our boat will be as follows:

- The boat instances (spawns) randomly and travels across the screen on pre-set path
- If the player clicks the boat its speed increases
    - If the player clicks before the boat enters the *fishingZone*, the fish count is not changed
    - Else if the player does not click fast enough, a fish is deducted from the *fishingZone*

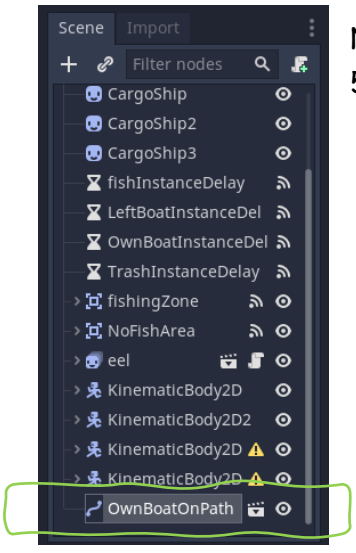
Creating a path for the boat to follow

On the left side of the editor, search for *OwnBoatOnPath* in files to find the pre-made boat scene

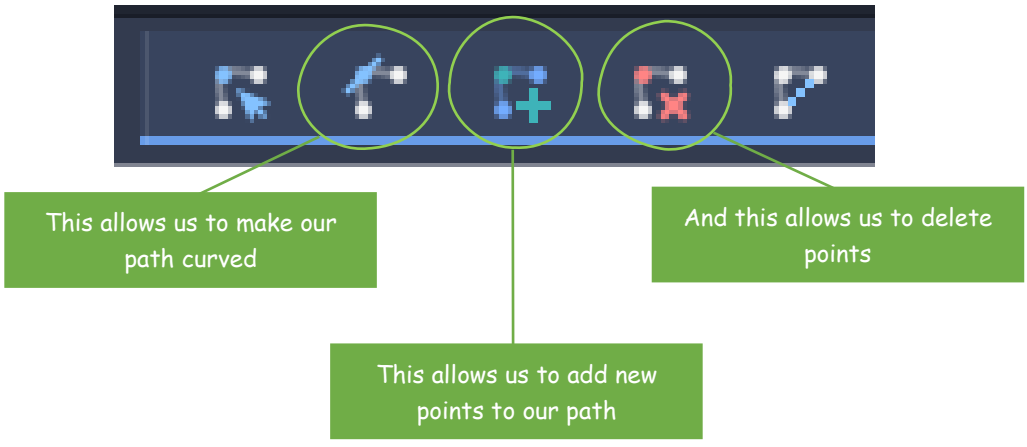


- Drag the scene up toward the scene viewer, and make sure to drag it to the *MainScene*
    - This will add it as a child of our main scene (which will add it to the scene)

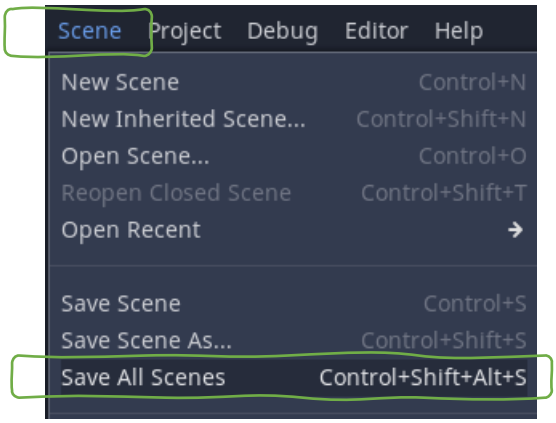
To begin creating the path, first make sure you have clicked on the newly added *OwnBoatOnPath* node, to let the editor know that this is the node we would like to make changes to



Now we can add new points to our path. At the top of the editor there will be 5 path related options. Let's look at the most important ones to us:



If at any point you're confused on the function of each option, you can always hover over them, and the editor will tell you what they do



Once you have created the new path make sure to save all scenes, and now we will look at instancing the new boat scene we've created.

Instancing the new boat through code

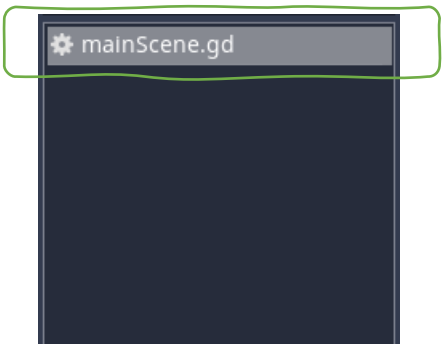
Once you have saved all scenes, delete the *OwnBoatOnPath* node from the *MainScene*. We had only instanced the boat through the editor so we could see how our new path would look with the other assets in the main scene.

In order to start coding we will have to find the code editor on Godot, there are multiple ways to do this.

1. Browsing through all of the .gd files

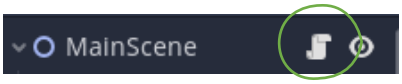
First change to *Script* mode at the top of the editor screen

And now you will see a list of available code files on the left side of the editor. Click the one you would like to see



2. Directly selecting the nodes .gd file

Click the scroll icon on the node that contains the code you would like to see



Choosing which ever way works best for you, make find your way to the *mainScene.gd* code file

In the *mainScene.gd* code file, there are 2 methods named *instanceLeftBoat* and *instanceOwnBoat*

```
func instanceLeftBoat():
    var LeftBoat = preload("res://LeftBoatOnPath.tscn").instance()
    add_child(LeftBoat)
    $LeftBoatInstanceDelay.start(rand_range(10,25))


func instanceOwnBoat():
    pass
```

instanceLeftBoat contains the code that allows the boat on the left side of the screen to be instanced (brought into the new scene)

In order to do the same thing with the new boat we've created, we can reuse the code already written, but make the following edits:

- Load our *OwnBoatOnPath.tscn* scene
- Start the *OwnBoatInstanceDelay* timer

Once we have made these changes, the method should now look something like this:

```
func instanceOwnBoat():
    var BottomBoat = preload("res://OwnBoatOnPath.tscn").instance()
    add_child(BottomBoat)
    $OwnBoatInstanceDelay.start(rand_range(10,25))
```

We are now using *OwnBoatInstanceDelay* which is a new node that is very important to our game, as it is a *Timer* node. A timer is simply a timer, much like a stopwatch, and we are using these to create a random time period before we instance a new scene.

Once we've setup the instanceOwnBoat function, we will need to call it. As we want the logic of this boat to be:

- Instance a new boat at a random time
- Once a new boat has been instanced, start a new timer
- Once the timer times-out, instance a new boat

We will be calling the new function in one of Godot's built-in methods that detects when our timer is complete.

```
func _on_OwnBoatInstanceDelay_timeout():
    pass
```

```
func _on_OwnBoatInstanceDelay_timeout():
    instanceOwnBoat()
```

We have just created a new method to instance the new boat by setting up a timer, and calling a function when the timer is complete 😊

However, to complete this and actually allow our boat to instance we will also need to start our timer when the game is first run.

Godot has another built-in method named *_ready()*. The logic of the ready method is simple – Godot runs the code inside of this method as soon as the scene is run.

```
func _ready():
    $LeftBoatInstanceDelay.start(rand_range(0,15))
```

```
func _ready():
    $LeftBoatInstanceDelay.start(rand_range(0,15))
    $OwnBoatInstanceDelay.start(rand_range(0,15))
```

So just as *LeftBoatInstanceDelay* is started in the ready method, we can call *OwnBoatInstanceDelay*

Make sure to save all scenes. Now let's click the play button in the top right corner of the editor to run the game and see our new boat!

Session 2

Programming random fish occurrences

The fish in our game will be used to represent the effects that illegal fishing and irresponsible waste dumping have on sea life.

```
func instanceFish():
    randomize()

    var packedFishScene = [
    preload("res://EelFishInstance.tscn"),
    preload("res://OrangeFishInstance.tscn"),
    preload("res://greenFishInstance.tscn")
    ]

    var size = get_viewport().get_visible_rect().size
    location.x = rand_range(0,size.x)
    location.y = rand_range(0,size.y)

    packedFishScene.shuffle()
    var fish = packedFishScene[0].instance()
    fish.position = location

    add_child(fish)
    fishInScene.append(fish)
```

Fixing the fish instancing method

The pre-written method that instances our new fish contains working logic, however when we save and run our game we do not see any fish being instanced?

The logic of our fish instancing should be:

- Instance a new fish at a random time
- Once a new fish has been instanced, start a new timer
- Once the timer times-out, instance a new fish

The method right now does not work, but this can be fixed using the same skills that we have used when instancing our boat scenes.

Can you find the errors in the code stopping our fish from being instanced by looking at:

- *fishInstanceDelay* timer node
- *instanceFish* method
- *_ready* method

The solution:

```
func _ready():
    $LeftBoatInstanceDelay.start(rand_range(0,15))
    $OwnBoatInstanceDelay.start(rand_range(0,15))
    $fishInstanceDelay.start(rand_range(0,2))
```

```
fishInScene.append(fish)
$fishInstanceDelay.start(rand_range(0,2))
```

```
func _on_fishInstanceDelay_timeout():
    instanceFish()
```

In order to get the fish instancing we have:

- Started a *fishInstanceDelay* timer in the _ready method
- Started a *fishInstanceDelay* timer in the *instanceFish* method
- And called the *instanceFish* method when our timer times-out

Now save all scenes and run your game. After a while you should be able to see small fish spawning on our screen.

However, even though we have loaded 3 fish, only the black eel and green fish are being spawned.

```
preload("res://EelFishInstance.tscn"),
preload("res://OrangeFishInstance.tscn"),
preload("res://greenFishInstance.tscn")
```

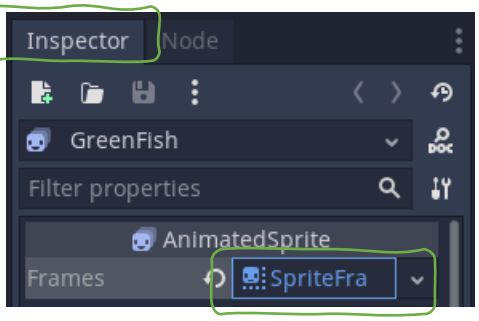
This is because the orange fish sprite and animation have not been setup. Let's do this now 😊

Animating our own fish

To start animating the orange fish, let's find our way to Godot's built in animation editor.

1. From the file section of the editor, find and double click on *OrangeFishInstance.tscn*

2. In the *orangeFishInstance* scene, click on the scene icon to select the *orangeFish* scene, where the sprite animation is stored



3. In the Inspector on the right side of the editor, make sure you are on *Inspector* view, then click on Frames to load the animation editor

And a new section of the editor should display, this is the animation editor.

There is already an animation named *default*, so we can easily add frames to this instead of creating a new one

This button allows us to add new frames from singular images. Since our fish don't have many animations, we will be using this option

This button allows us to add new frames from a sprite sheet. Sprite sheets are usually used to animate characters as they have a lot of animations which can be setup a lot faster with a sprite sheet

Once you select a button, Godot will prompt you to select an image. As we have fish.PNG images pre-made in our game assets file we can use these as our frames

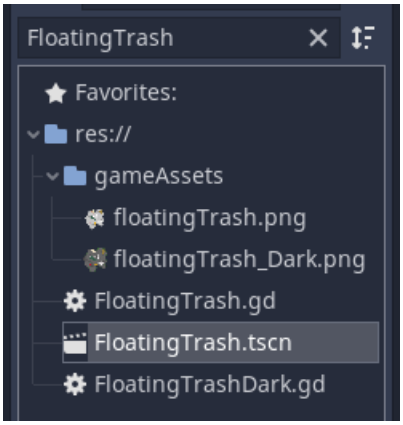
Now when we save all scenes and run our game, after a while we will be able to see a little orange fish instance on our screen 🐟🐟🐟🐟

Session 3

Adding interactive trash in our game

Now it's time to add a new feature to our game, interactive trash. Through this new feature we will be representing the effects of illegal waste dumping.

To get started, find the *FloatingTrash.tscn* file to view the scene that we will be instancing



The code file for this scene is at first a bit more confusing to look at as we are introducing new features in it:

- A timer is being instanced and started through code
- There is a method that allows the trash sprite to move along the x and y axis'
- Detecting when the trash is clicked by a mouse, and deleting it from the scene

Since the movement is set up for the trash scene, let's look at instancing it into our main scene.

We can instance this new trash scene in the same way we have done with our fish. The logic will be very similar:

- Instance a new trash at a random time
- Once a new trash has been instanced, start a new timer
- Once the timer times-out, instance a new trash

Using the methods

- InstanceTrash():
- _on_TrashInstanceDelay_timeout():
- _ready():

Write the code to instance the trash in your scene

The solution:

1. Firstly we can preload and instance our trash scene at a random location

```
func instanceTrash():
    var Trash = preload("res://FloatingTrash.tscn").instance()
    var size = get_viewport().get_visible_rect().size
    location.x = rand_range(0,size.x)
    location.y = rand_range(0,size.y)

    Trash.position = location
    add_child(Trash)
    $TrashInstanceDelay.start(rand_range(0,10))
```

2. Now we can call this method when the timer times out, as we have seen before

```
func _on_TrashInstanceDelay_timeout():
    instanceTrash()
```

```
func _ready():
    $LeftBoatInstanceDelay.start(rand_range(0,15))
    $OwnBoatInstanceDelay.start(rand_range(0,15))
    $fishInstanceDelay.start(rand_range(0,2))
    $TrashInstanceDelay.start(rand_range(0,2))
```

3. And lastly we must make sure we start the *TrashInstanceDelay* timer in our ready method, so that our *instanceTrash* method is called

Now when you save all and run the game you will have interactive trash. Feel free to experiment with the time you set the *TrashInstanceDelay* timer, or even the speed at which the trash travels in the *FloatingTrash.gd* file

Session 4

Change background bounds depending on fish amount

Now to add a fun feature, allowing us to change the background of the game to let the player know how well they are doing

```
onready var backgroundColour = get_node("CleanSeaBackground")
var cleanSeaBackGround = preload("res://gameAssets/cleanSeaBackground.png")
var dirtySeaBackground = preload("res://gameAssets/dirtySeaBackground.png")
```

This can be quickly setup using the three mysterious global variables towards the top of the code, (these store background images we can use) and the currently empty _process method.

```
func _process(delta):
    pass
```

This built-in method is perfect for this purpose as it runs every frame. This means we can check a condition every frame from within this method.

We can change the background colour using simple if conditions in the _process method like so:

```
func _process(delta):
    if len(fishInScene) > 2:
        backgroundColour.set_texture(cleanSeaBackGround)
    if len(fishInScene) < 2:
        backgroundColour.set_texture(dirtySeaBackground)
```

Here we are targeting our *cleanSeaBackground* node, which is a sprite (node that displays an image), and changing the image

Using our fish count we can define the boundaries for good or bad progress in our game. You do not have to keep these as less than or greater than 2, you can change these to fit your view as the developer.

Now to save all scenes and run the game, after some time of fish instancing our background colour will change!

Recap on programming skills learnt

After adding the background colour switcher we now have a fully working game 🥳

We have used nodes and methods in the Godot engine that are very common and useful in 2D game design.

- Fish
    - AnimatedSprites – With this knowledge you can animate your own characters
- Illegal fishing boats
    - Path2D – These are often used to setup enemy movement in stealth games
- Trash
    - Instancing at random locations
    - Timer – This is a very useful node, many developers use these in combat games to set the time a round will be
    - _physics_process – This method is very valuable for input controlled player movement, very similar to the movement of our trash
- Background
    - _process – This method is perfect for keeping track of a score in a game, as it is updated constantly
    - Background change – This was a small non-required feature that has added 'playability' to our game

Discuss how we are representing sustainable development goals

The theme of this game is centred around SDG (Sustainable Development Goal) 14: Life below water

What other SDGs could this game relate to?

What new features could you add to your game that relate to either SDG 14, or another of your choosing?