TRUCK SIMULATION DOCUMENTATION

Contact me for any issue:

Gmail: stegameslive@gmail.com

Thanks for Purchasing this Asset!

Failure to follow this documentation step by step may result in errors.

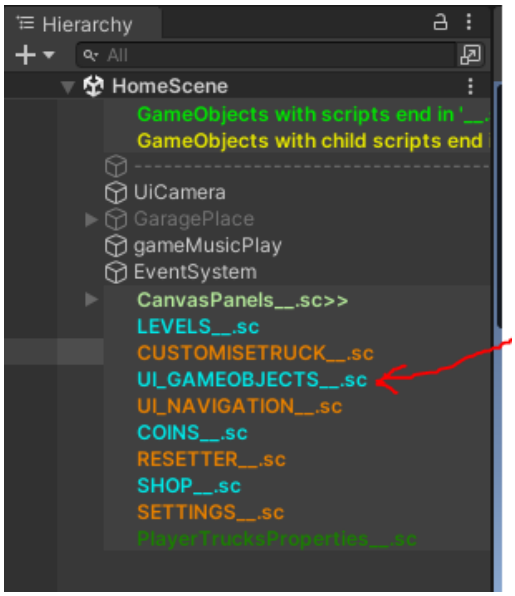
So PLEASE follow this documentation step by step numerically in setting up your game.

Note these:

How To:

1. GUI Game Objects

The first thing to notice is that in the editor hierarchy, there is a gameobject named *UI_GAMEOBJECTS__.sc*. Its purpose is to reference all (almost all) graphical user interfaces that will be changed during runtime in one way or another. The UIs are located in the *'CanvasPanels__.sc>>'* gameobject.



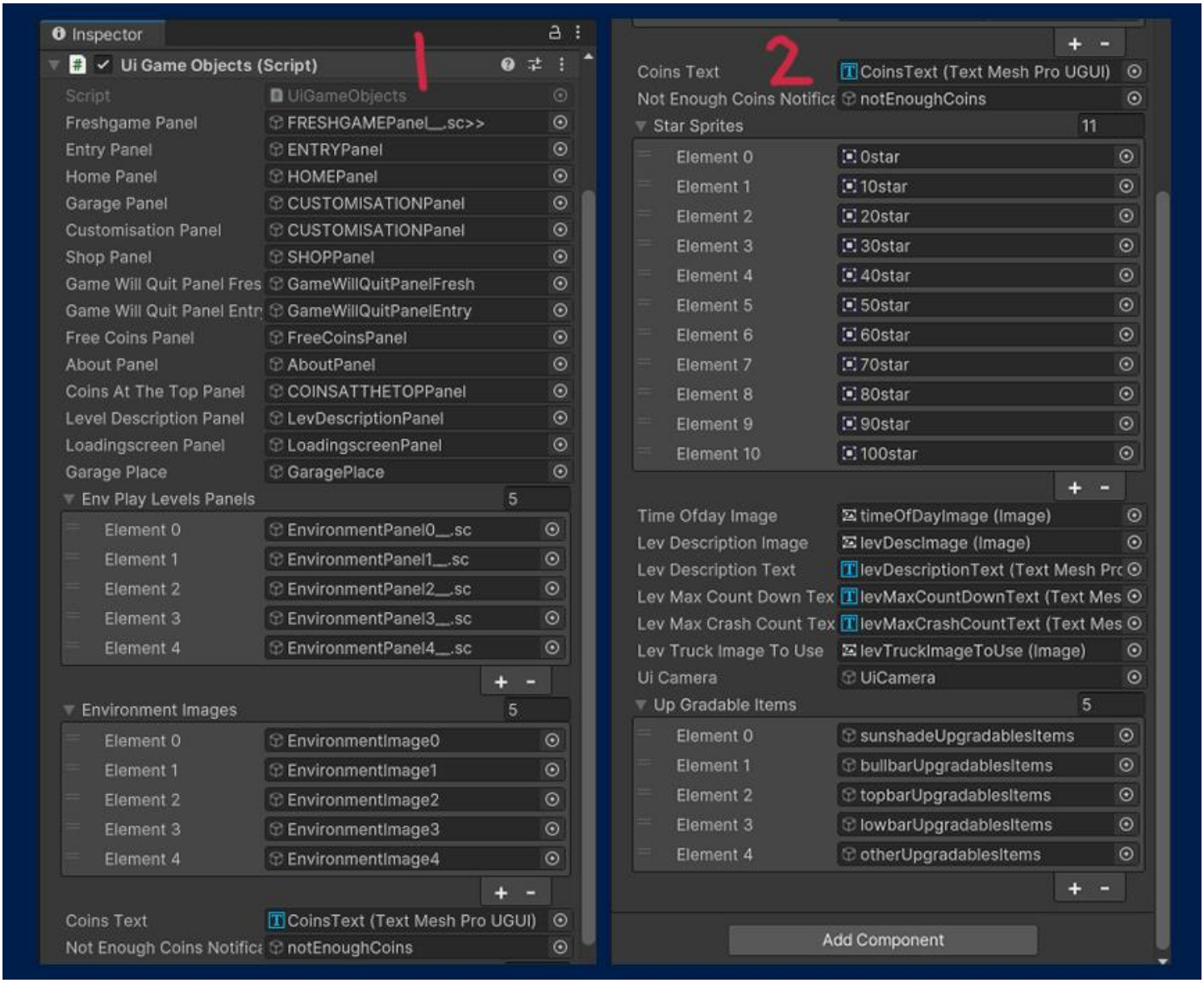
So in order to make this game **unique** to your preference, you will have to change the look of all these UIs. From buttons, to images, to sliders, to whatever.

In the individual environment scenes (in the Scenes folder named *Env0, Env1, Env2*, etc.), you will also have to change UIs there.

Make sure whatever UI you change/modify, you drag and drop it back into their appropriate slots in this *UI_GAMEOBJECTS__.sc* inspector in case it gets missing.

Here is what it looks like. So in the case you mess up, simply search for the name in the hierarchy and then drag and drop them back into their slots:

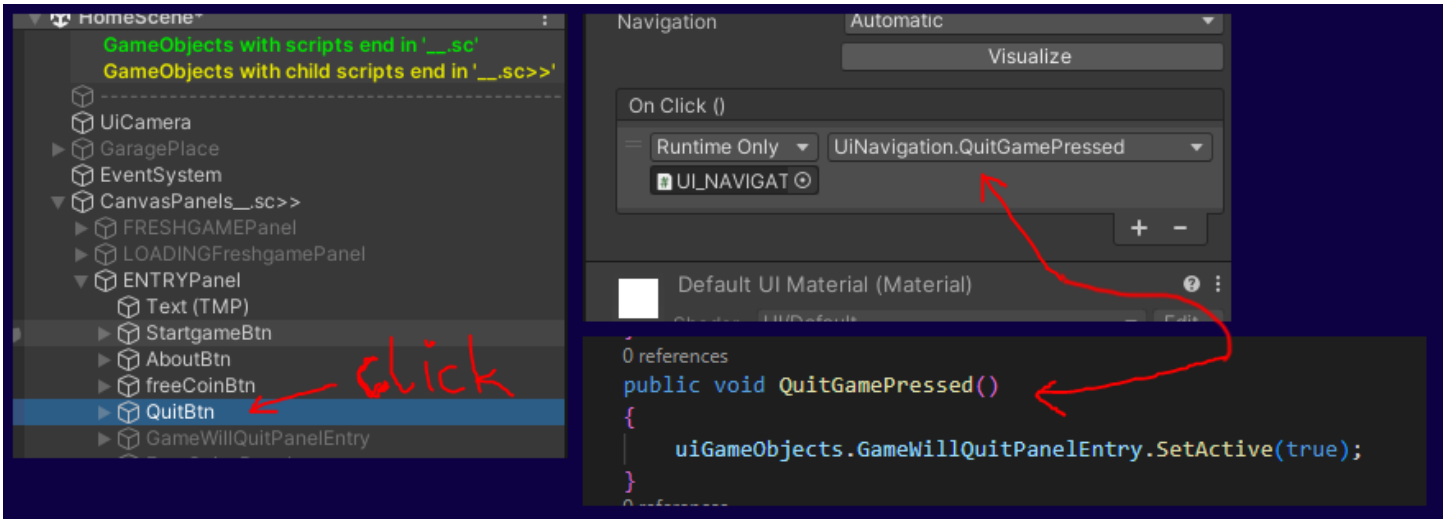
Zoom in to see clearly:


Inspector
1
Ui Game Objects (Script)
Script
UiGameObjects
Freshgame Panel
FRESHGAMEPanel__.sc>>
Entry Panel
ENTRYPanel
Home Panel
HOMEPanel
Garage Panel
CUSTOMISATIONPanel
Customisation Panel
CUSTOMISATIONPanel
Shop Panel
SHOPPanel
Game Will Quit Panel Fres
GameWillQuitPanelFresh
Game Will Quit Panel Entr
GameWillQuitPanelEntry
Free Coins Panel
FreeCoinsPanel
About Panel
AboutPanel
Coins At The Top Panel
COINSATTHETOPPanel
Level Description Panel
LevDescriptionPanel
Loadingscreen Panel
LoadingscreenPanel
Garage Place
GaragePlace
Env Play Levels Panels
5
Element 0
EnvironmentPanel0__.sc
Element 1
EnvironmentPanel1__.sc
Element 2
EnvironmentPanel2__.sc
Element 3
EnvironmentPanel3__.sc
Element 4
EnvironmentPanel4__.sc
Environment Images
5
Element 0
EnvironmentImage0
Element 1
EnvironmentImage1
Element 2
EnvironmentImage2
Element 3
EnvironmentImage3
Element 4
EnvironmentImage4
Coins Text
CoinsText (Text Mesh Pro UGUI)
Not Enough Coins Notifica
notEnoughCoins
2
Coins Text
CoinsText (Text Mesh Pro UGUI)
Not Enough Coins Notifica
notEnoughCoins
Star Sprites
11
Element 0
0star
Element 1
10star
Element 2
20star
Element 3
30star
Element 4
40star
Element 5
50star
Element 6
60star
Element 7
70star
Element 8
80star
Element 9
90star
Element 10
100star
Time Ofday Image
timeOfDayImage (Image)
Lev Description Image
levDescImage (Image)
Lev Description Text
levDescriptionText (Text Mesh Pro
Lev Max Count Down Tex
levMaxCountDownText (Text Mes
Lev Max Crash Count Tex
levMaxCrashCountText (Text Mes
Lev Truck Image To Use
levTruckImageToUse (Image)
Ui Camera
UiCamera
Up Gradable Items
5
Element 0
sunshadeUpgradablesItems
Element 1
bullbarUpgradablesItems
Element 2
topbarUpgradablesItems
Element 3
lowbarUpgradablesItems
Element 4
otherUpgradablesItems
Add Component

2. GUI Navigation

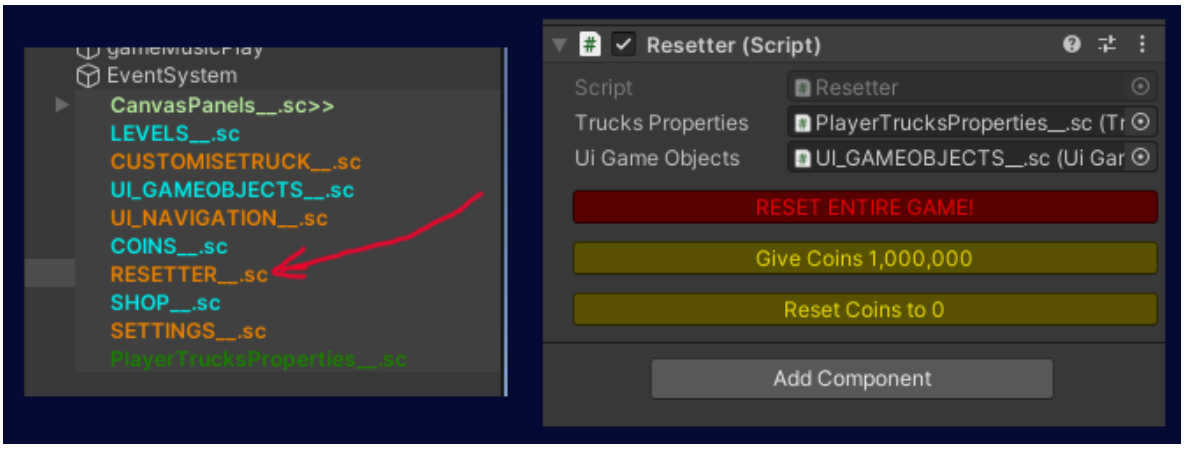
Locate *UI_NAVIGATION__.sc* in the hierarchy. Its script is responsible for BUTTON onclick events of the UI gameobjects. Open it and you see it has got public methods. These methods are exposed to their UI buttons (located in the *CanvasPanels__.sc>>*) click events.

E.g. for the Quit Game button:



3. Resetter

This is responsible for resetting your game to default values when ready to build and export it.



4. GameData

In this game, all player settings and data is saved in PlayerPrefs. The script *GameData* is therefore responsible for doing just that. It contains static methods that are called when necessary. It is pretty much explanatory when you look at it.

5. PlayerTruckProperties

Right there in the *GameData* script, are these functions:

```
public static void SetPlayerTruckProperties(int playerTruckID, int paintId, int sunshadeId, int bullbarId, int topbarId, int lowbarId, int otherId)
{
    int result = int.Parse(paintId.ToString() + sunshadeId.ToString() + bullbarId.ToString() + topbarId.ToString() + lowbarId.ToString() + otherId.ToString());
    PlayerPrefs.SetInt("PlayerTruckID_" + playerTruckID, result);
}

1 reference
public static int GetPlayerTruckProperties(int playerTruckID)
{
    return PlayerPrefs.GetInt("PlayerTruckID_" + playerTruckID);
}
```

What they do is that they are responsible for storing the properties of each playerTruck in the game. All playerTrucks have properties that determine what qualities they possess.

So each playerTruck has a

1. PlayerTruck identification integer (*int playerTruckID*): This tells the type of truck you set in the *Assets/TruckSimulator/Resources/PlayerTruck*s scriptable object.

2. Paint identification integer (*int paintId*): It assigns the paint material that has been bought for the playerTruck. Paint material index is read from the *Assets/TruckSimulator/Resources/TruckPaints* scriptable object.

3. Sunshade identification integer (*int sunshadeId)*: It tells the bought sunshade upgradable that the playerTruck has. Its index or Id is also read from the *Assets/TruckSimulator/Resources/Upgradables* scriptable object.

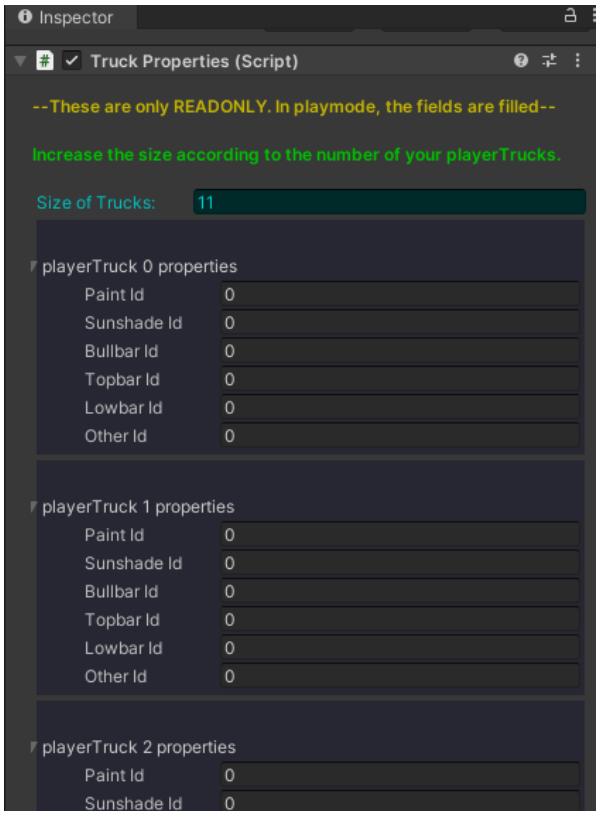
4. Bulbar identification integer (*int bullbarId)*: It tells the upgradable bulbar that has been bought for that playerTruck. Its index/Id is also read from the *Assets/TruckSimulator/Resources/Upgradables* scriptable object.

5. Same applies for lowbar and other.

Now what the *TruckProperties* script hosted in the *PlayerTruckProperties__.sc* gameobject in the hierarchy does is that, it reads from the GameData method

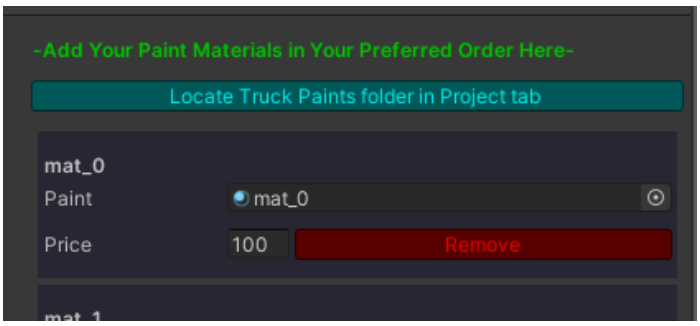
```
GetPlayerTruckProperties(int playerTruckID)
```

and assigns all playerTruck individual Ids to array sets as seen in this image immediately the game enters play mode:



This inspector is populated only when the game enters play mode. You do not have to assign any values here yourself.

Also note that 0 Id value is the default. So for an instance, for Paint Id = 0 (as seen from the image above) means that when you open the *Assets/TruckSimulator/Resources/TruckPaints* scriptable object, the first index paint is what all playerTrucks will have as default paint:



6. Prefabs

Look into ‘Assets*/TruckSimulator/Prefabs/Misc’* folder. The reason for creating these prefabs is so that you don’t have to individually modify their contents in different scenes.

For example, the ‘*Canvas*’ prefab contains the pause menu among other things. Editing the pause menu in this prefab mode will apply to all scenes. It makes your work easy. That is the purpose of prefabs in the first place.

7. How to modify and add your own PlayerTrucks

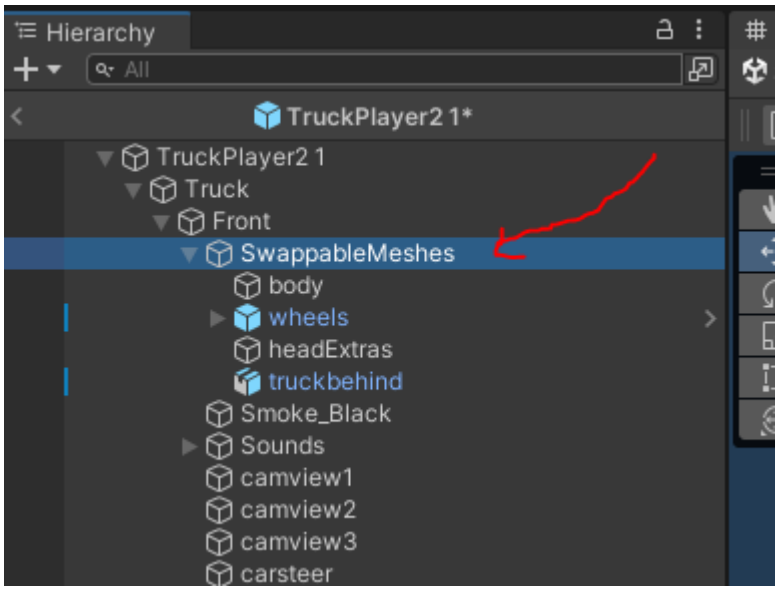
To modify a playerTruck or change its appearance, follow these steps closely:

1. Open the "*Assets/TruckSimulator/Prefabs/PlayerTrucks*" location.

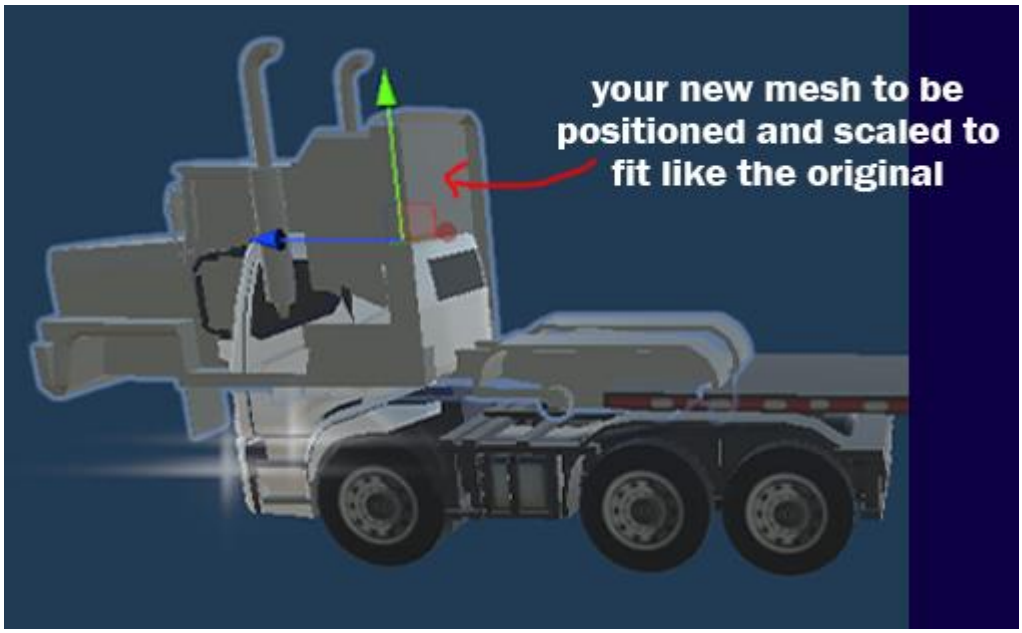
2. Pick the PlayerTruck you want to modify. Then duplicate by pressing *Ctrl + D* whilst it is selected in this projects folder. This is done so that in case you mess up the duplicated one, the original is intact.

3. Open the duplicated one in its prefab context mode.

4. Expand the *'Front'* child. And then expand the '*SwappableMeshes*' object. This contains the Truck head of the truck and the wheels.



5. Now as you have your own truck head model, drag and drop the truck model here and position, rotate and scale it appropriately fit.



6. When done, you can simply delete the original meshes **in** the '*SwappableMeshes'.*

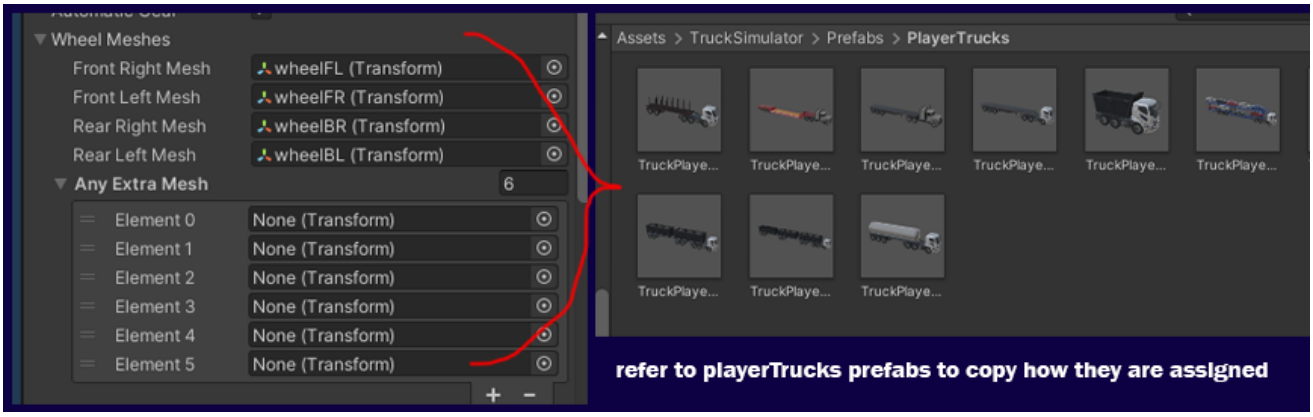
7. Please note that you MUST name your new mesh truck paintable body as "**body**" and place it in the '*SwappableMeshes'* object.

   That is because it is referenced from the 'PaintInstantiator' script in its exact place in the hierarchy. So not doing so is going to throw a null object reference error.

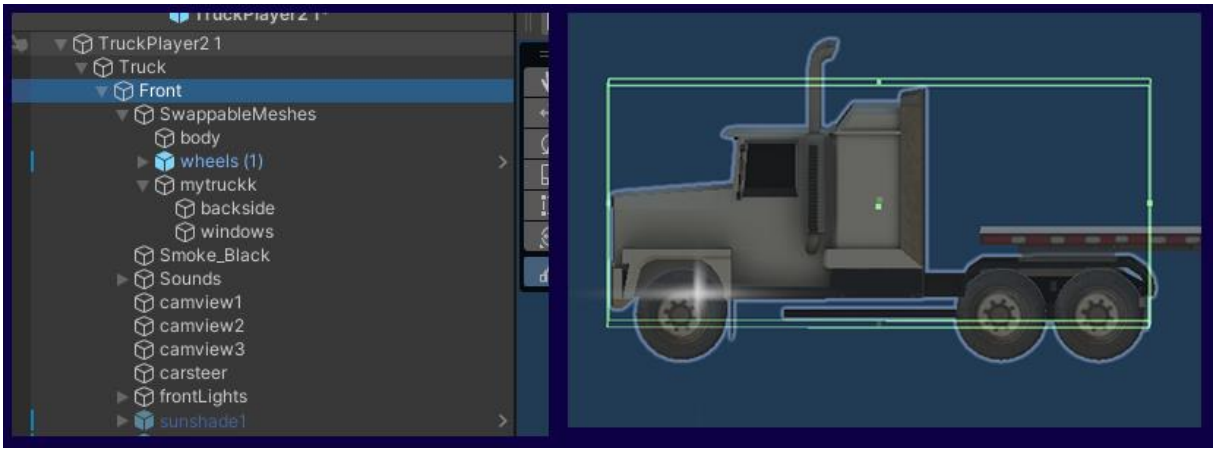
```
ck.transform.Find("Truck/Front/SwappableMeshes/body").transform.GetComponent<MeshRenderer>();
onReader.propertiesList.properties[GameData.GetSelectedTruck()].paintNumber].paintMaterial;
```

8. After positioning, scaling, and fitting your wheels too, go to the *'Front'* object. On its inspector is the *TruckController* script. Locate the WheelMeshes and drag and drop your new wheel meshes into appropriate slots.

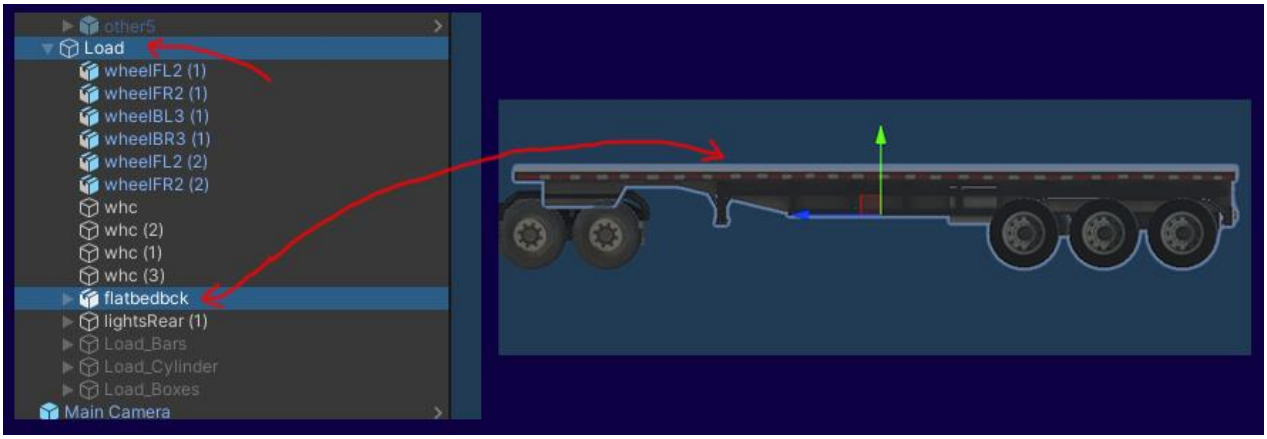
   For the '*AnyExtraMesh*' slots, please refer to other prefab playerTrucks to know what to drag and drop there. Load wheelmeshes go there by the way.



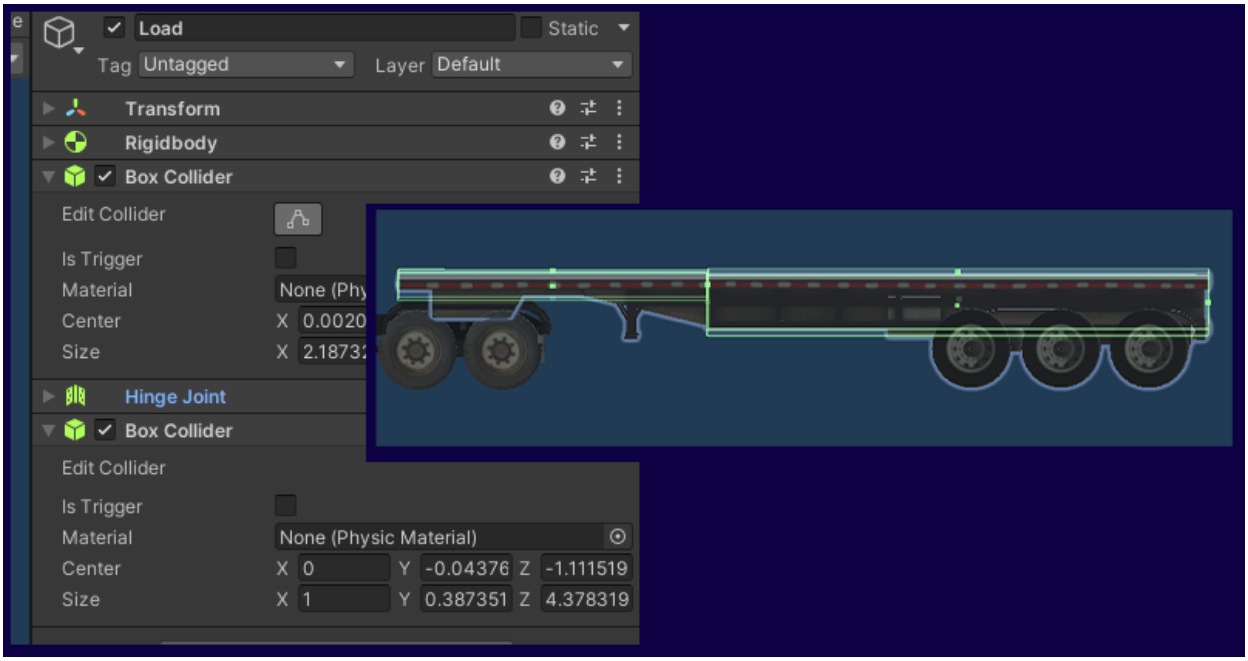
9. Now click the *'Front'* parent and resize its collider to fit your new truck size.



10. Next go to the *'Load'*. Then replace the mesh (flatbedbck in this case) with your own back load mesh.



11. Next click on this *'Load'* and you may need to resize its colliders to fit. Also swap the wheels if you want your own wheels. Make sure you drag and drop them in the *'AnyExtraMesh'* slot of the TruckController.



12. Now click on the parent gameobject named 'Truck__.sc'. You will see it contains 2 component scripts.
Go to the bottom one called *'Loads'*.
Assign your new loads, that if that truck can have different loads.

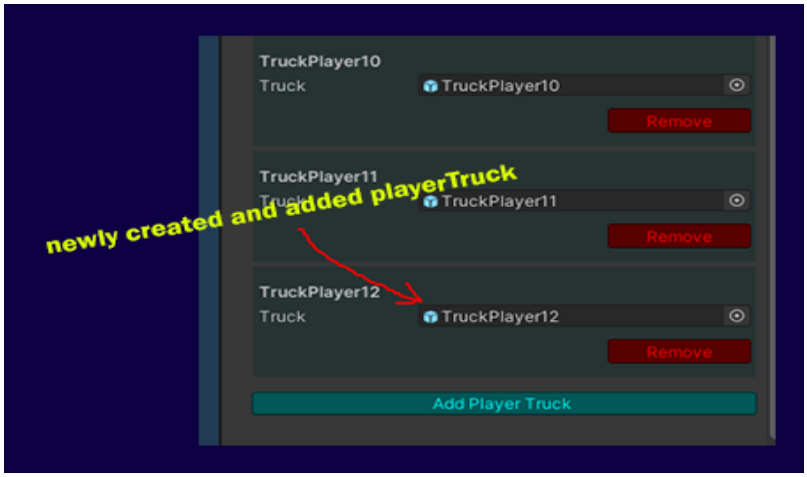
13. Now you are done with the truck. Now rename it from its duplicated name (such as TruckPlayer2 1) to **TruckPlayer12**. We use 12 because that is the last truck is 11.

Please make sure you follow this naming convention. That is 'TruckPlayer+number'. That is because this exact name is referenced in many scripts. If you rename it another way, you will get null reference errors.

14. Now, in the HomeScene, Env0, Env1, Env2, Env3 and Env4 scenes, go to their *"PlayerTrucksProperties__.sc"* gameobect in the hierarchy and increase the "size of Trucks" of the TruckProperties from 12 to 13.

15. Now to the Menu up there, click the '*Window/Truck Simulator Template'* and click Configure PlayerTrucks.

16. This opens the PlayerTrucks scriptable object where you add your newly created playerTruck your game.
    To do so, scroll to the bottom and add it.



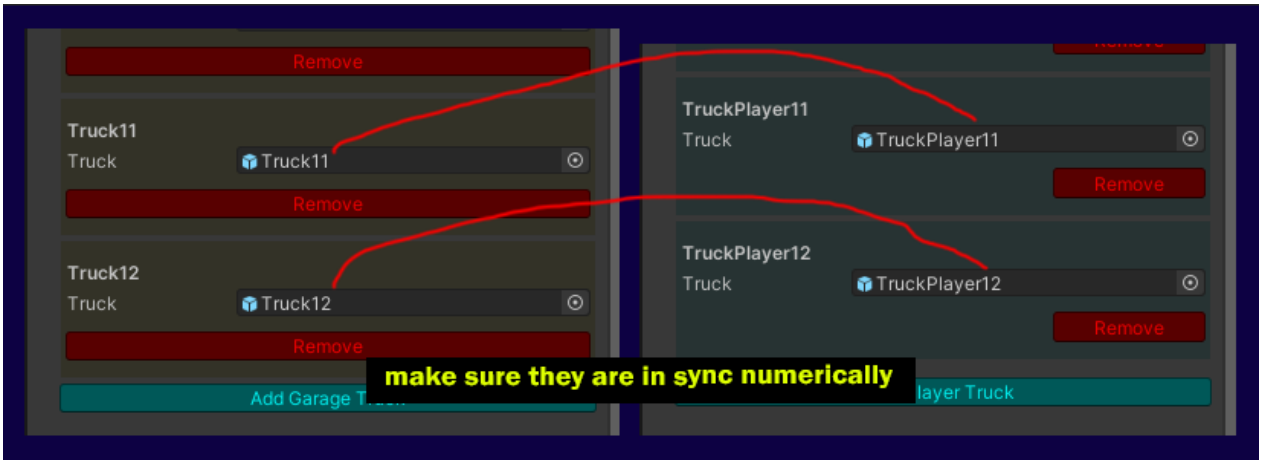
17. Next you have to also add this truck to the Garage too. To do so, open the GarageTrucks folder *'Assets/TruckSimulator/Prefabs/GarageTrucks'.*

18. And just like the PlayerTrucks, pick one and duplicate it. Then swap the duplicated one's meshes with yours.

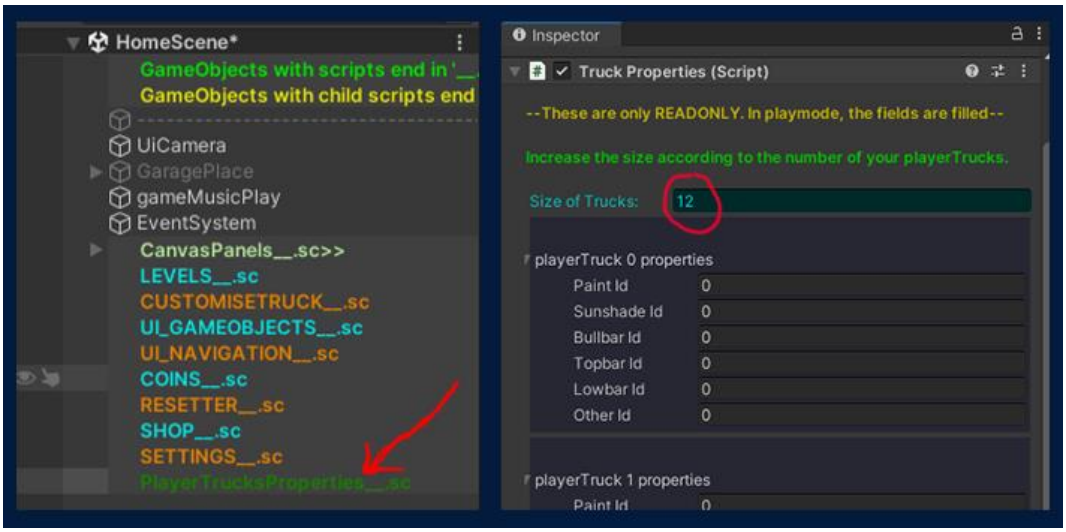
    Since garage trucks are just static objects with no scripts attached, it should be easy and straightforward. Just make sure you don't delete those *'sunshadePosition, bullbarPosition'* etc. or change their names. They are referenced in a script and changing their names will produce a null reference error.

19. When done, click the '*Window/Truck Simulator Template'* up there on windows. And click '*Configure Garage Trucks.'*

20. It opens the Garage Trucks scriptable object where you add your newly created garage truck. Please make sure its index order is in sync with the PlayerTrucks scriptable object.



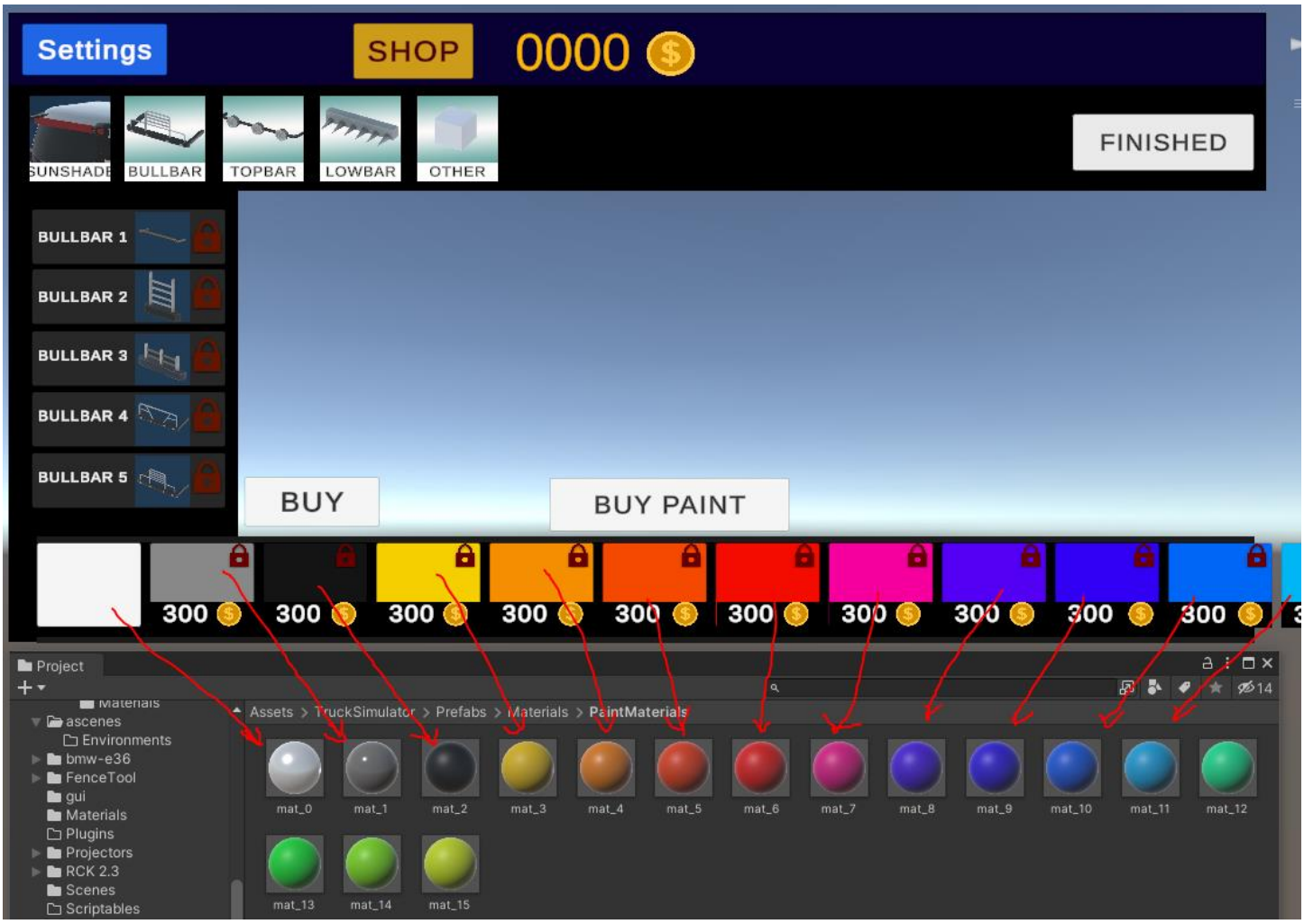
21. Then go to the *'PlayerTruckProperties__.sc'* in the hierarchy. Click on it. On its inspector you see its size is 11. Simply increase it depending on the number of playerTrucks you have added. So for instance after following all these procedure, the trucks increase from 11 to 12. So make the size now 12.



And that is all for modifying the PlayerTrucks.

8. How to modify Paint for PlayerTrucks (and Garage Trucks)

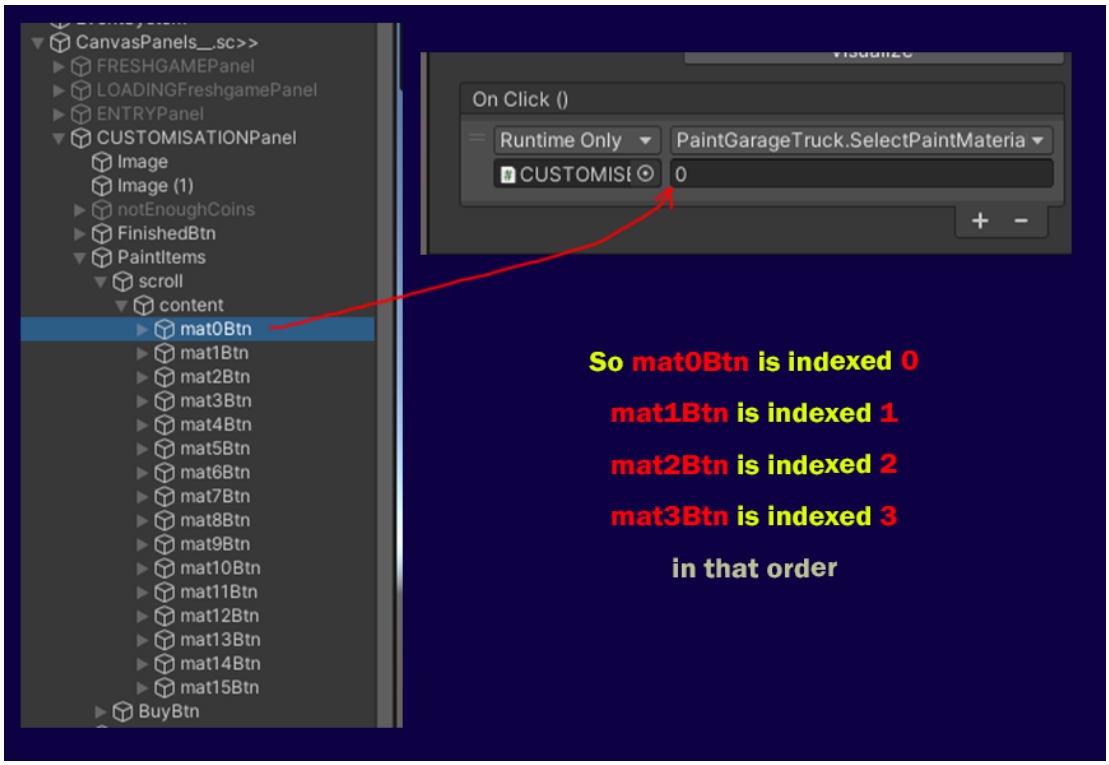
The paint for the playerTrucks are normal materials. They are located in *'Assets/TruckSimulator/Materials/PaintMaterials'.* As you can see below, they are mapped onto their UI buttons.



1. So first of all, create your materials or edit these ones. Or add more as you fit.

2. Then on the paint buttons, color the buttons to match the materials color.

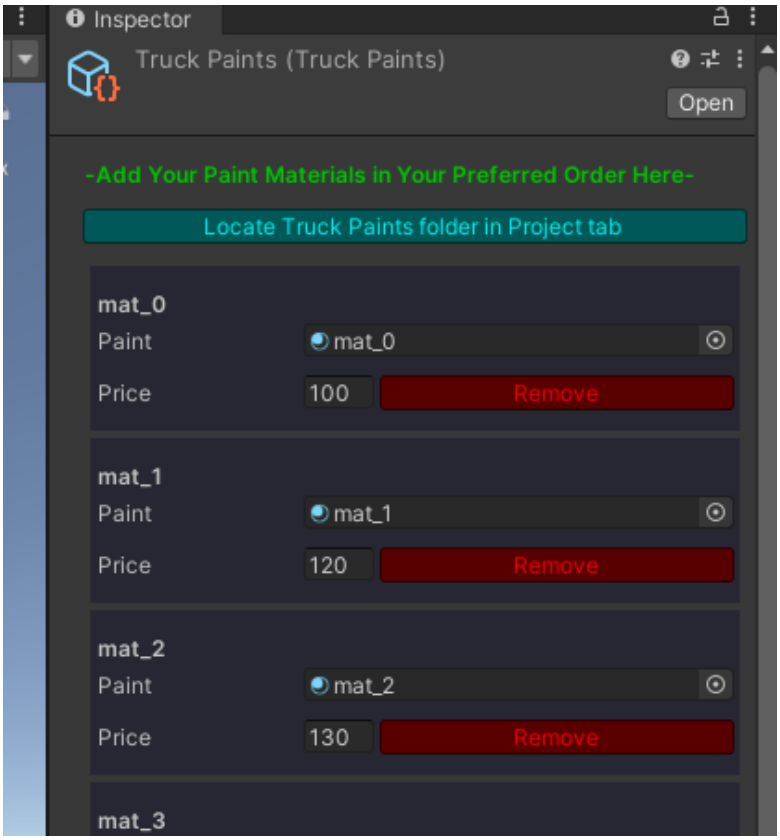
3. Also on the buttons (see image below for location on hierarchy) , on their Onclick events in the inspector, make sure they are indexed orderly:



You can therefore add more buttons as you please. Just make sure they are indexed as in the above image.

4. Then open go to the windows at the top and click the '*Window/Truck Simulator Template/TruckPaints*'

It opens the PaintMaterials scriptable object where you add new paint materials (from the paint materials folder) and set their prices as you please.



And that is all for setting Paint Materials.

9. How to Modify Upgradables for PlayerTrucks

There are currently only 5 upgradables that can be on each playerTruck. I will add others in future releases. Especially wheels upgradables. But for now, we only have sunshades, bullbars, top and lowbars. And then 'other'.

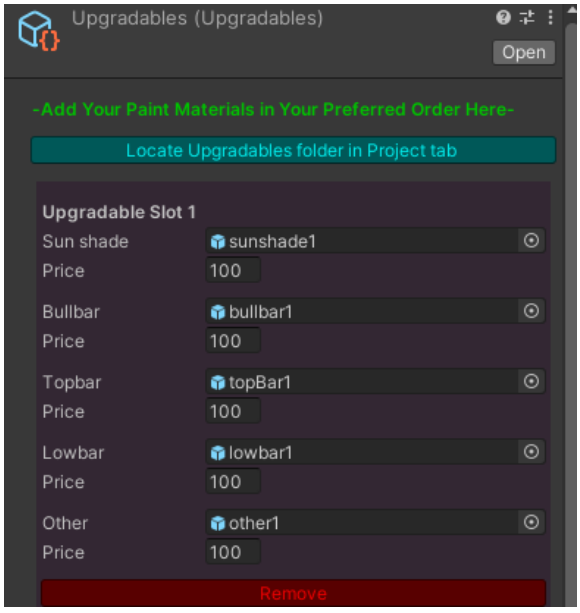
For the 'other', you must replace it with any extra upgradable you wish to. To do so, just change the prefabs in the *Assets/TruckSimulator/Prefabs/Upgradables* folder.

To change or replace upgradables, follow these steps:

1. First create prefabs of the upgradables. Locate the upgradables folder *'Assets/TruckSimulator/Prefabs/Upgradables'*. You can simply open each prefab and just replace the meshes with yours like we did with playerTruck above.

2. When done click on the menu bar at the top *'Window/Truck Simulator Template'* up there and click on the Configure Upgradables.

3. It opens its scriptable object where you replace them with yours and prices.



4. Next is setting up the Upgradables GUI. I mean changing the button images to look like yours in the *'CanvasPanels__.sc>>'*. That is photoshop stuff you can figure out yourself.

Remember that *'UI_GAMEOBECTS__.sc'* references all these UIs. So make sure you drag and drop them there in case they get lost.

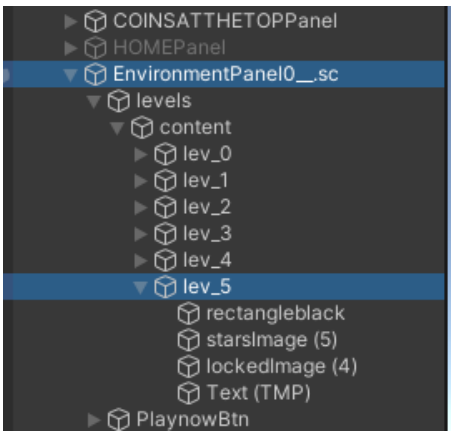
But you don’t even have to touch anything in the *‘UI_GAMEOBECTS__.sc’.* That is because all what you are doing is just changing images in the button images and minor stuff like that.

10. How to Setup Levels (and environment tracks)

In this game, there are **5 environments**. The first one, environment 1 looks like a barren forest, second looks like a desert, 3rd looks like a City and so on. You can modify them to look like whatever you want. Such as you can make the 3rd environment a snowy landscape.

Each of this environment contains Levels. Currently, they all contain only 6 levels each. Of course you are to add more levels as you please. To do so:

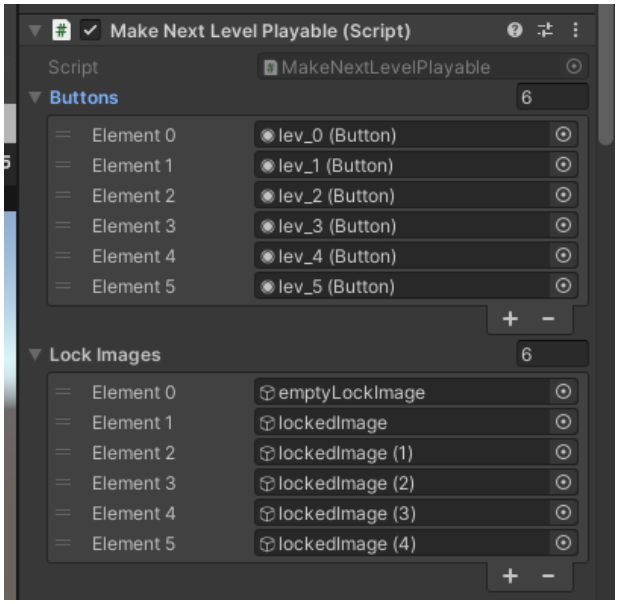
1. Simply Expand any EnvironmentPanel. So for environment 0, it is under *'CanvasPanels__.sc>>/EnvironmentPanel0__.sc'/levels/content/.* You can see the 6 levels there. To add a level, simply duplicate the last one.



2. So duplicate the *'lev_5'* and rename it to *'lev_6'.* You see that it automatically repositions itself in line with the other levels perfectly. That is because it has a scroll rect parent.

3. So after duplication and renaming, expand your new *'lev_6'* and edit its contents.
   That is rename the *'level 5'* to *'level 6'* or whatever you wish. Then the button image, use your own image, etc.

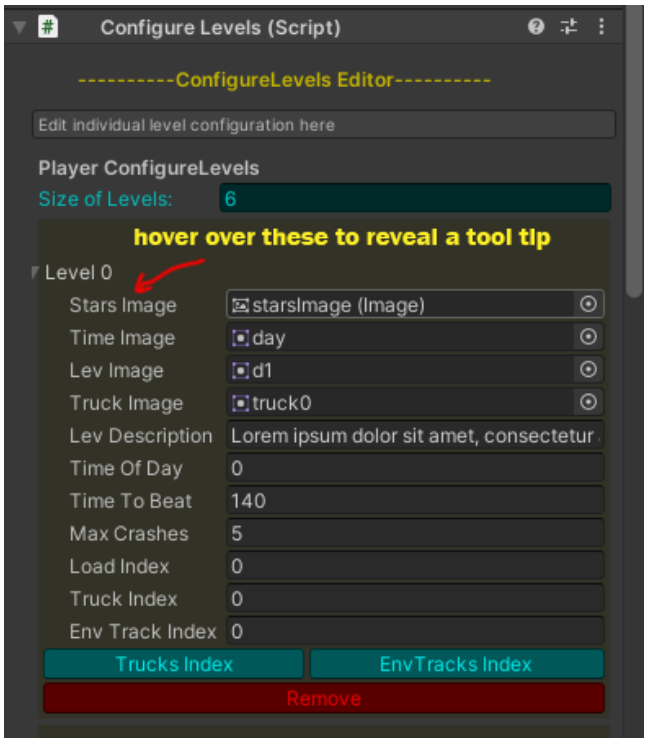
4. When done, click the *'EnvironmentPanel0__.sc'* panel. You will see in its inspector that it's got some attached scripts.
   Look at the *'MakeNextLevelPlayable'* script. It's got a slot for buttons and 'lockImages'. Add your new 'lev_6' button and child lockImage to it.



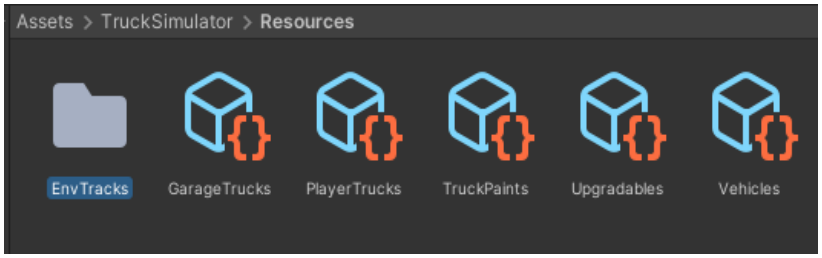
Then now look at the *'ConfigureLevels'* script. You see that each level has its entry in there.

Scroll to the last one (level 5), and click the Add button to add your new level 6.

Then edit the *'StarsImage, TimeImage, LevImage, TimeToBeat,'* and etc. values to setup your game. **Hover over these names and it reveals a tool tip to guide you**.



5. Next we are to create the actual track for the level. To do so, go to '*Assets/TruckSimulator/Resources/EnvTracks/*'

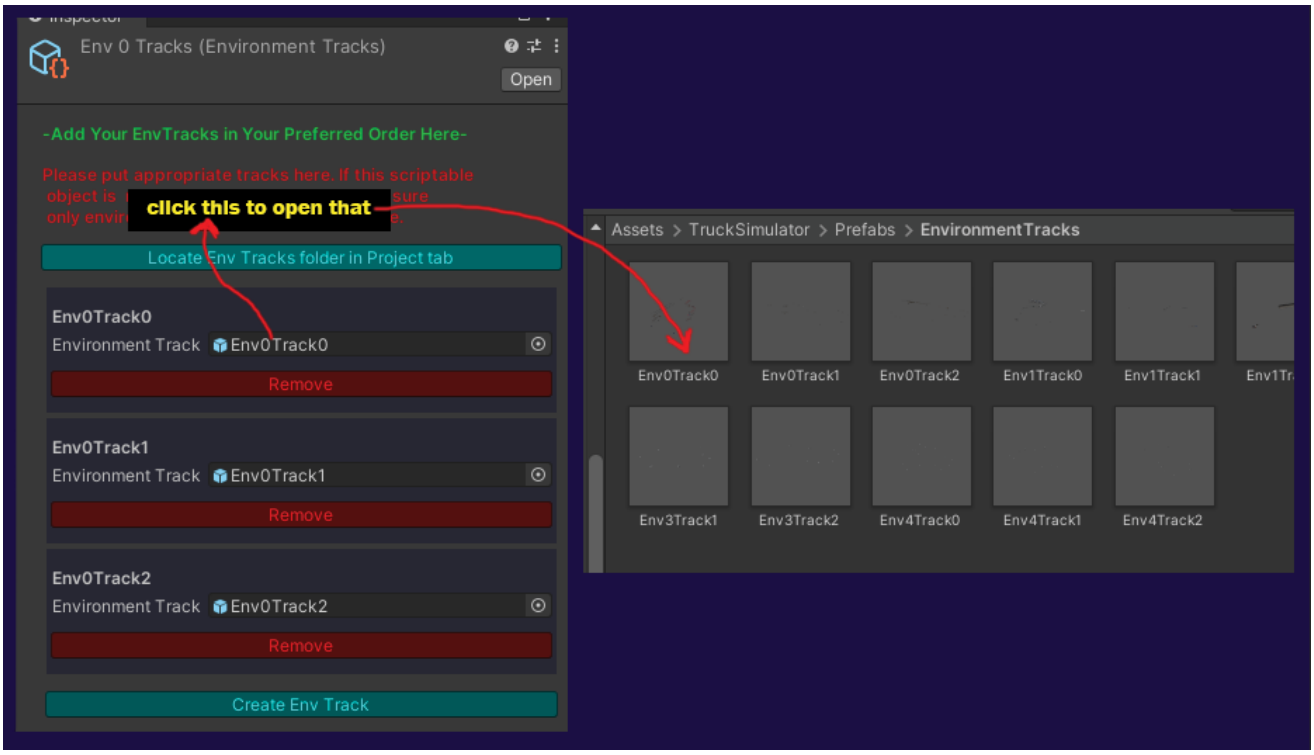


   Open the folder.

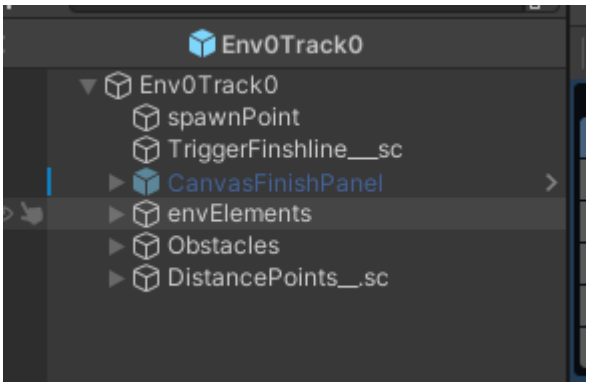
6. It's got 5 scriptable objects. Now each of these scriptable objects contain slots for 'environment tracks'. What is an environment track?

   An envTrack is the path where your playerTruck will drive on. It can be a loop or not. It is instantiated when your environment scene opens in play mode.

   So clicking any of the environment track like in the image below opens the prefabs of each environment track.



Now open any of the prefabs and see how it looks like.



As you can see, it's got a

1. **SpawnPoint**: This is where your playerTruck is instantiated when the scene opens in play mode.

2. **TriggerFinishline__.sc**: This is a box collider with **disabled meshrenderer** that detects when playerTrucks has entered or touched it. So put this gameobject at the finish end of your track. It has got a script. Look into it. This script enables the CanvasFinishPanel.

3. **CanvasFinishPanel**: This panel is disabled on default and is enabled by the TriggerFinishline__.sc script.

4. **EnvElements**: These are all roadblocks, directional signs, etc.

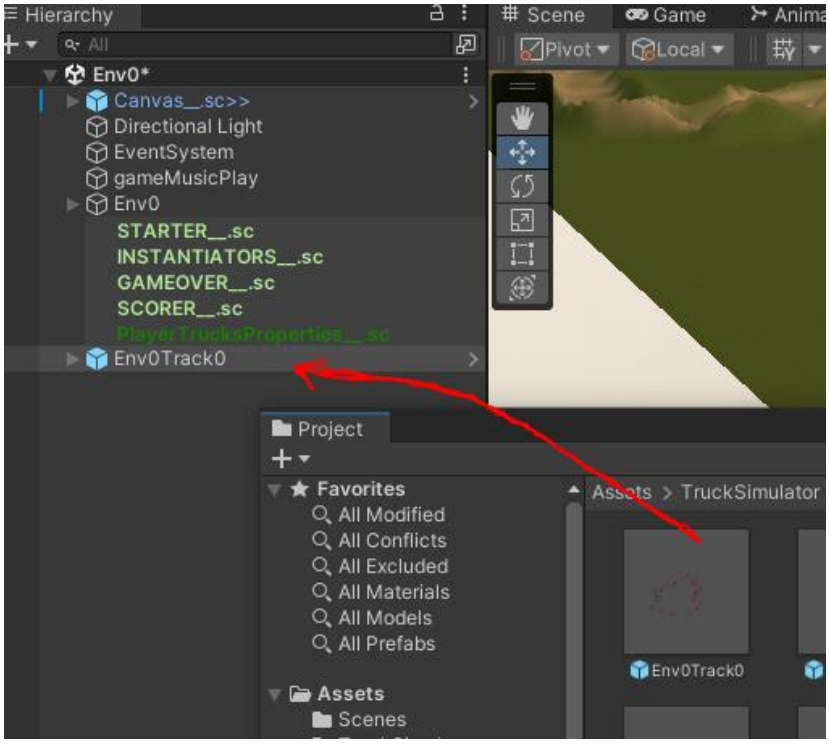
5. **Obstacles**: These are objects which adds to your crashcount when you hit them with your playerTruck. Note that they have their tags set to 'Obstacle' or else they will not work.

6. **DistancePoints__.sc**: This contains box colliders **with disabled meshrenderers**. What they do is that they help in calculating how far PlayerTruck has gotten to in a level track in the *'GameOverCaller'* script.

So basically, each environment contains Levels, and each level is actually an environment track.

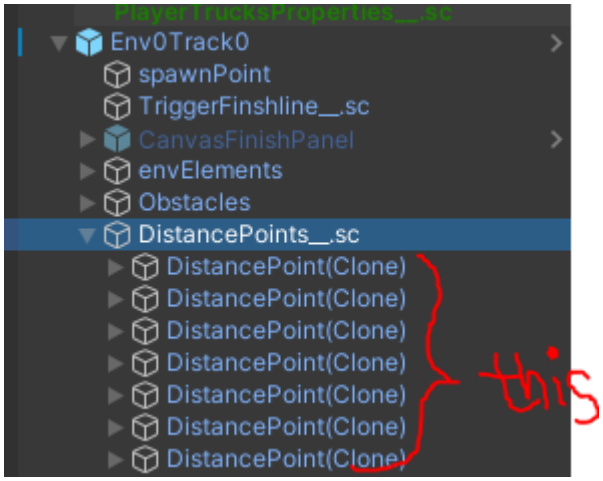
So how do you create an environment track?

1. First of all, drag and drop an environment track prefab on to the hierarchy as below:



2. Then right click it and Prefab > Unpack.

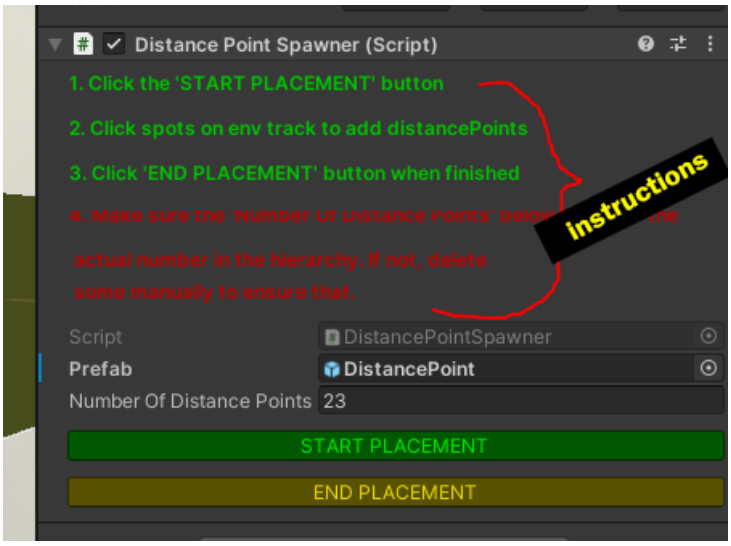
3. Then click the *"DistancePoints__.sc"* gameobject. You will see it's got some child objects.



4. These child objects called "DistancePoint" are actually sphere colliders with their triggers enabled. They help in calculating the distance the playerTruck has travelled in the *GameOverCaller* script (in line 72).

5. Delete all these DistancePoints clones. All of them.

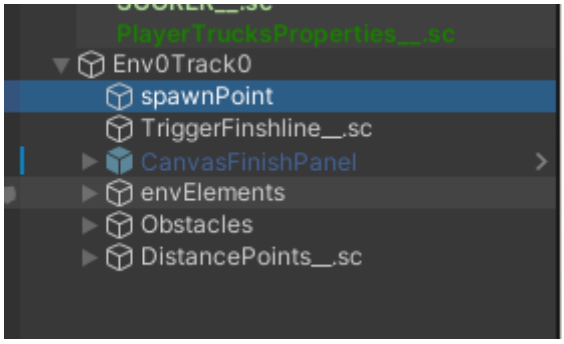
6. Then click the *DistancePoints__.sc* gameobject. On its inspector you see this:



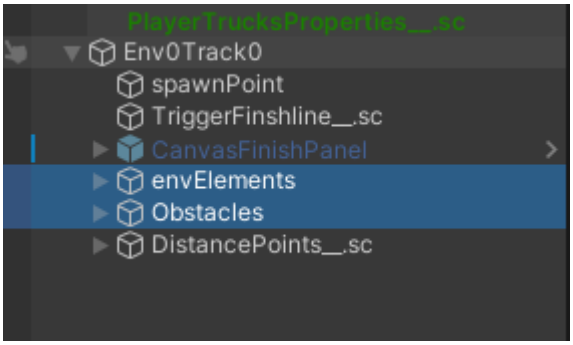
7. Follow the instructions.

8. You should have something like this:

9. Now that you are done with the DistancePoints, locate the *"spawnPoint"* and position it at the exact point and rotation that your playerTruck will be spawned when the game starts:

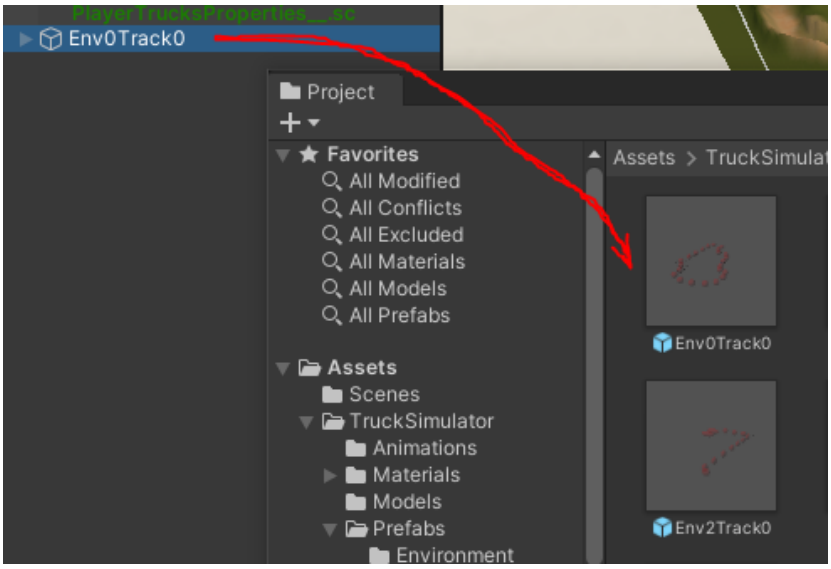


10. Then expand the "*envElements*" and "*Obstacles*" gameobjects and edit them according to how you want your track to look.



11. Then rename your "Env0Track0" to the appropriate name, so to Env0Track3 (since Env0Track1, Env0Track2 are already there).

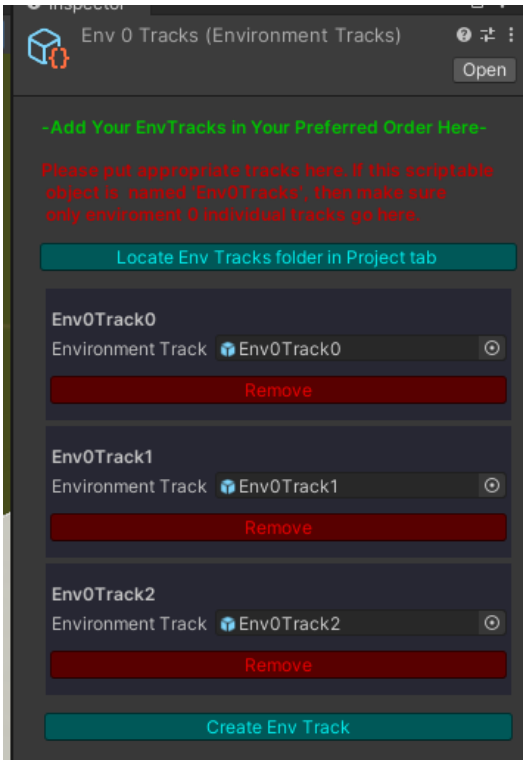
12. Then drag and drop your envTrack to the environment Tracks folder as a new prefab.



13. Now go the menu window above and click on the '*Window/Truck Simulator Template/Locate Environment Tracks folder'.*

Open the folder in the projects tab. In it you see 5 different scriptable objects:

Click the one corresponding to the environment you are editing. So if you are in the “Forest” environment, click Env0Tracks. Then look at the inspector.



14. Click the *“Create Env Track”* to make a slot. Now drag and drop your created environment track you put in the “EnvironmentTracks” into the new slot.

    And that is pretty much it.

11. Adding IAP / Ads

Unfortunately no IAP or ads setup is implemented in this template. But this video will help:

https://www.youtube.com/watch?v=PthFk5je350&t=1s&pp=ygUKdW5pdHkgaWFwIA%3D%3D

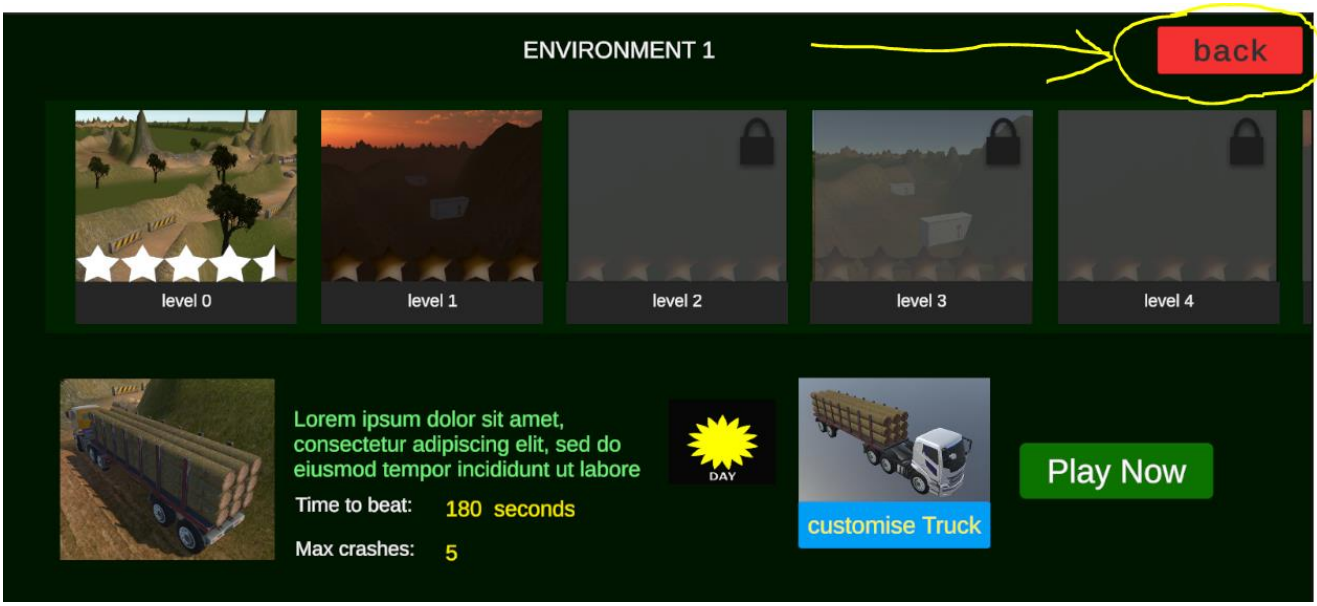
Click the *'SHOP__.sc'*. On its inspector, open the *'ShopCoins'* script.

There you see places in the code you can use to setup your IAP and Ads.

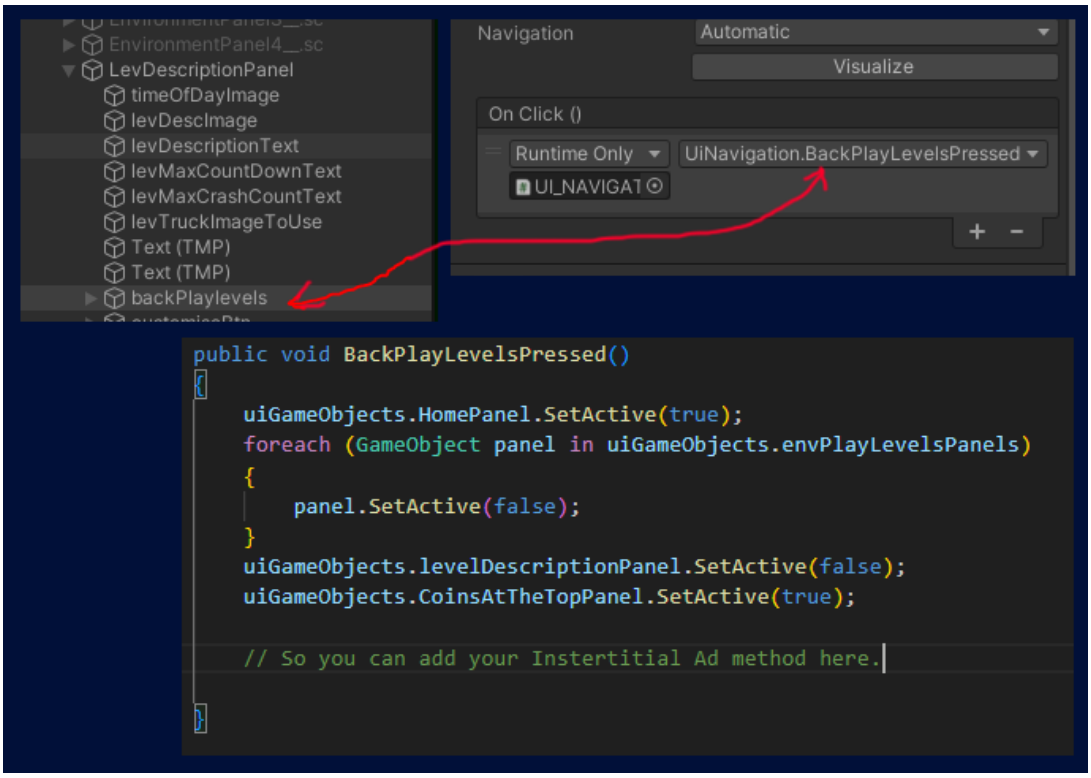
Also locate *'CanvasPanels__.sc>>/SHOPPanel'*. You see the buttons beneath. On each button's onclick event in the inspector, notice that there are int coins amount values. They are called in the ShopCoins script. Pretty much explainable in the script.

Now there are places in the game GUI navigation that you would like to fit in some interstitial Ads.

For example, when the player presses this "back" button as shown below, you want to show an interstitial ad.



All you have to do is locate this "back" button in the hierarchy and find its assigned OnClick Method as below:


EnvironmentPanel4__sc
LevDescriptionPanel
timeOfDayImage
levDesImage
levDescriptionText
levMaxCountDownText
levMaxCrashCountText
levTruckImageToUse
Text (TMP)
Text (TMP)
backPlaylevels
Navigation
Automatic
Visualize
On Click ()
Runtime Only
UiNavigation.BackPlayLevelsPressed
UI_NAVIGAT
public void BackPlayLevelsPressed()
{
uiGameObjects.HomePanel.SetActive(true);
foreach (GameObject panel in uiGameObjects.envPlayLevelsPanels)
{
panel.SetActive(false);
}
uiGameObjects.levelDescriptionPanel.SetActive(false);
uiGameObjects.CoinsAtTheTopPanel.SetActive(true);
// So you can add your Instertitial Ad method here.
}