Puzzled Wizard

11 January 2020

Overview

Coffeeshop Starter Pack is a beginner friendly project from a small team with ambitious dreams. All of our models are low-poly assets crafted by awesome 3D Art Wizard of our team. All the scripts included are thoughtfully written by a professional game developer. Although we tried to do our best, If you find something wrong in the project or you need help, don't hesitate to contact us at support@puzzledwizard.com

We suggest you start trying things with the ready-to-play GamePlayScene, or inspect all the prefabs in PropGrid or DecorativeScenes.

We hope you enjoy this package, you can always send us what you have created with the package from our social media accounts. In fact, please do it, cause we're looking forward to see what you've come up with it.

All the best!

Puzzled Wizard Team.

Editor Scripts And Their Uses

Overview Of Editor Scripts

You can find all editor scripts are inside “Assets/CoffeeShopStarterPack/Scripts/Editor/“ folder. This folder will be referred as “package editor folder” in through out the documentation.

“ScreenShotImportSettings.cs” is a AssetPostprocessor script that gets hooked to the process of importing textures as sprite whenever we generated an image file in any of our project’s “ScreenShots” folder. If you want to change the folder name to look, you can do that in the script by changing name in the line.

```
const string folderNamesToLook = “/ScreenShots/";
```

If you want to disable this functionality you can just delete the file.

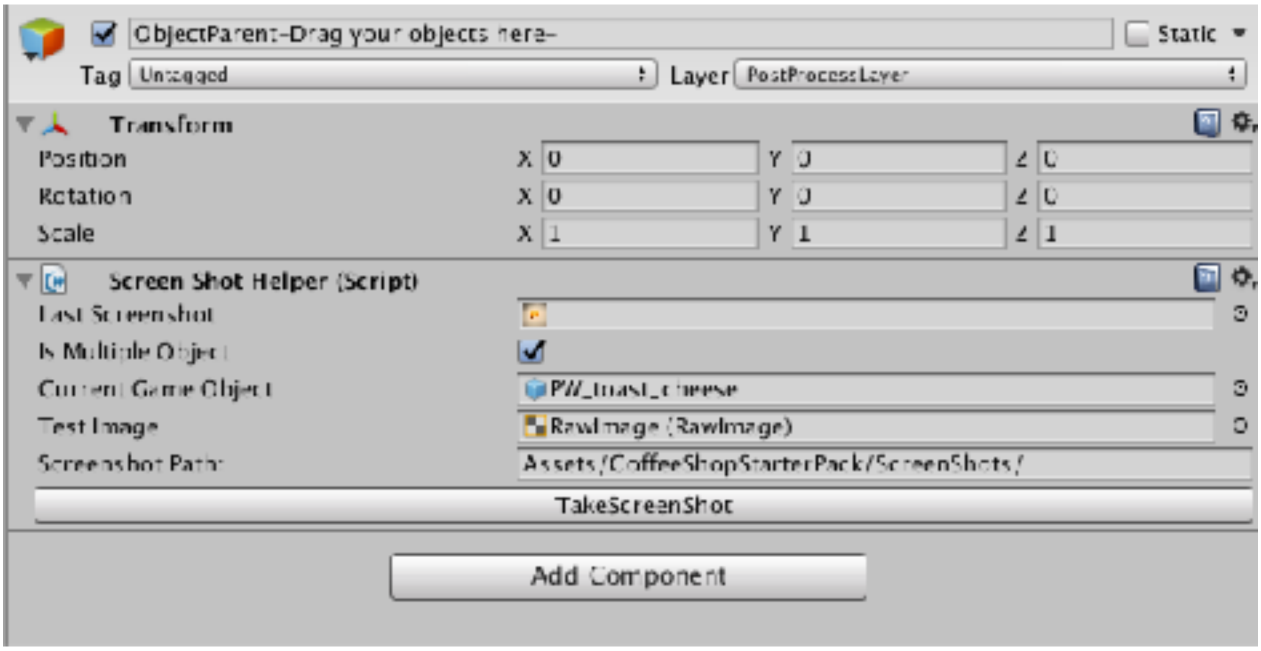
ScreenShotHelper

ScreenShotHelper script is a utility script to automatize recording of screenshots of objects. In our case it was the prefab of our products. But you can do the same with objects you added in editor or runtime.

Open the “CoffeShopStarterPack/Scenes/ScreenShotScene.unity” file in the editor.

To generate screenshot of an object drag it to this “ObjectParent-“ game object in the hierarchy as a child. You can drag multiple objects at once.

Select ObjectParent gameObject. You can see the script with its custom editor in the inspector window. The custom editor script is inside the package editor folder.



☑ ObjectParent-Drag your objects here- □ Static

Tag Untagged

Layer PostProcessLayer

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Screen Shot Helper (Script)

Last Screenshot

Is Multiple Object ☑

Current Game Object PW_toast_cheese

Test Image RawImage (RawImage)

Screenshot Path: Assets/CoffeeShopStarterPack/ScreenShots/

If you want to generate one screenshot only drag the child object to “Current Game Object” field and uncheck the “Is Multiple Object” toggle. Other than that you don’t need to change anything. Click the button TakeScreenShot. Your screenshots will be generated and imported as sprites in to the folder. “Assets/CoffeShopStarterPack/ScreenShots”. You can change this folder in the inspector if you wanted.

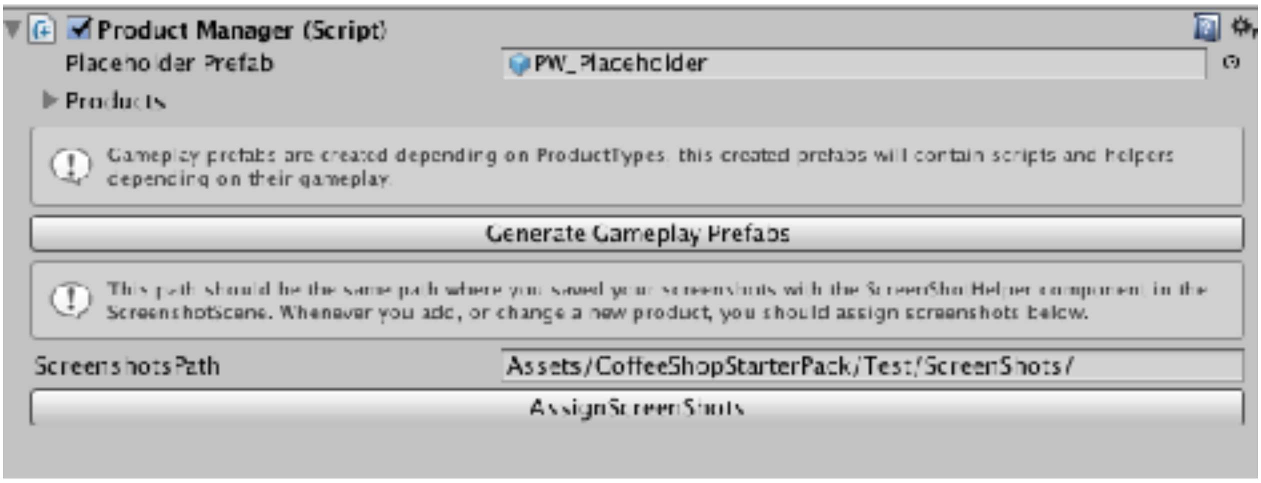
Test Image is the sprite in the canvas, it shows the output of every screenshot.You can disable the canvas if you don’t want or delete it. Our scripts does a null-check.

ProductManager

ProductManager script is where we generate the gameplay prefabs and set the product properties. You can see the ProductManager in the “GamePlayScene” inside the “<Manager and Controllers>” hierarchy object.

Select ProductManager and you will find an inspector editor created with custom editor script in the package editor folder. “ProductManagerEditor” is what draws this inspector view with the help of “ProductEditor” property drawers.

Product Manager works with “OrderGenerator” script to assign screenshots and product orders to the order generator. You can have different OrderGenerator and ProductManager on each scene. Orders you want to be generated can be different for each level so you can add a Product Manager prefab to your scene and edit the objects you want or not in this scene there. We didn’t want to use a ScriptableObject logic here to make it beginner friendly, but if our community request it we could add an optional scriptable object workflow later.



Placeholdar Prefab PW_Placeholder

Gameplay prefabs are created depending on ProductTypes. this created prefabs will contain scripts and helpers depending on their gameplay

This path should be the same path where you saved your screenshots with the ScreenShotHelper component in the ScreenshotScene. Whenever you add, or change a new product, you should assign screenshots below.

ScreenshotsPath Assets/CoffeeShopStarterPack/Test/ScreenShots/

Placeholder prefab is what is instantiated at the object's place after it's served or disposed. You can leave it as its or delete it. Our script does a null-check before instantiating. RegenerateProduct checkbox at ProductEditor defines if you want to use this placeholder object feature for each product.

Generate GamePlay Prefabs button creates gameplay prefabs from assigned values of products. Typically you want to do that when you want to change product properties. You don't need to change prefabs for each scene cause it will override the prefabs. If you want to generate different prefabs for each scene you need to modify this in editor script and maybe add sceneName too.

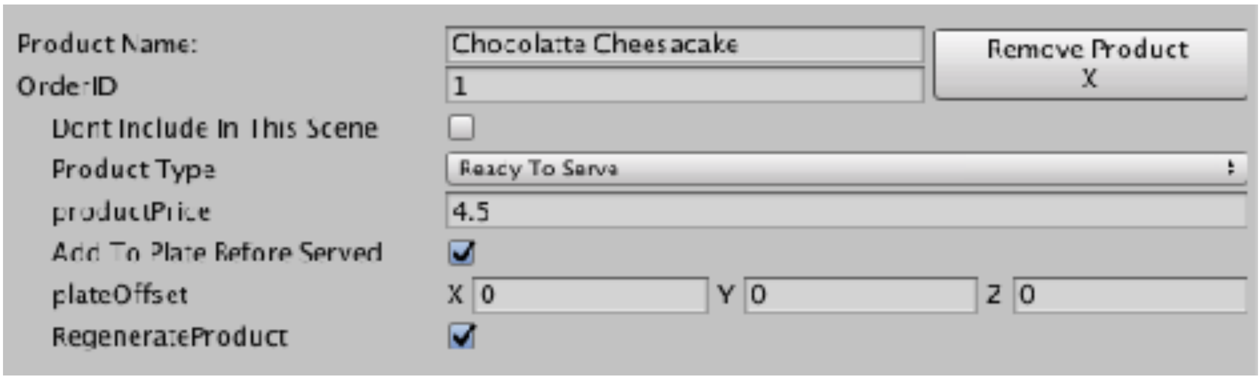
string longPrefabName = PrefabPath + prefab.name + "_type" + enumIndex + ".prefab";

Check the screenshotPath to match the one you created your screenshots in. The default path is "Assets/CoffeeShopStarterPack/Test/ScreenShots/". Whenever you change your orders, added or removed one, you should click on the "AssignScreenShots" button. This will communicate with the OrderGenerator script and edit the products to be generated on the scene.

If you fold out the products, you'll see the size of Products array and individual editors for products where you set the type, price and other properties depending on the type of the object.

Product Type can be Cookable, Heatable, Drinkable and Ready To Serve.

"Don't include in this scene" toggles the assigning the screenshot and orderID of this product to OrderGenerator script. If you uncheck it this product will not be generated in this scene.

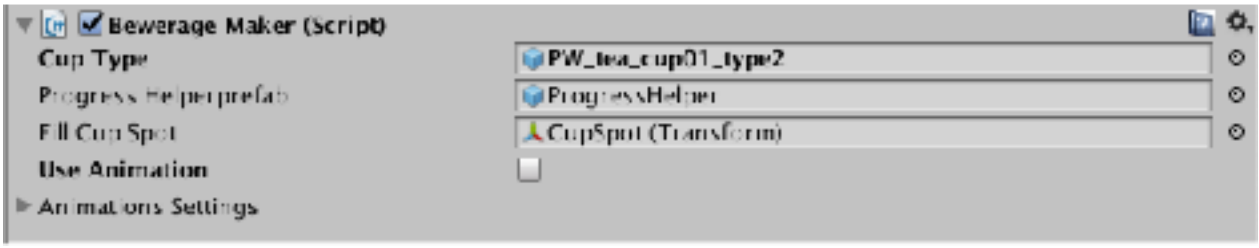


Product Name:	Chocolatta Cheesacake	Remove Product X
Order ID	1	
Dont Include In This Scene	□	
Product Type	Ready To Serve	
productPrice	4.5	
Add To Plate Before Served	☑	
plateOffset	X 0 Y 0 Z 0	
RegenerateProduct	☑	

BewerageMaker

BewerageMaker is a script we use on mocha pot, tea pot and Espresso machine. We imagined you can use the same script with any beverage makers you added. It has a custom editor script in the package editor folder named “BewerageMakerEditor.cs”. We suggest you open the Scene “Assets/CoffeeShopStarterPack/Scenes/InteractableObjects/“. This scene has every possible use of BewerageMaker where you can change and interact in play mode.

Cup type can be one of the prefabs you generated from product manager of types drinkable, or ready to serve product.



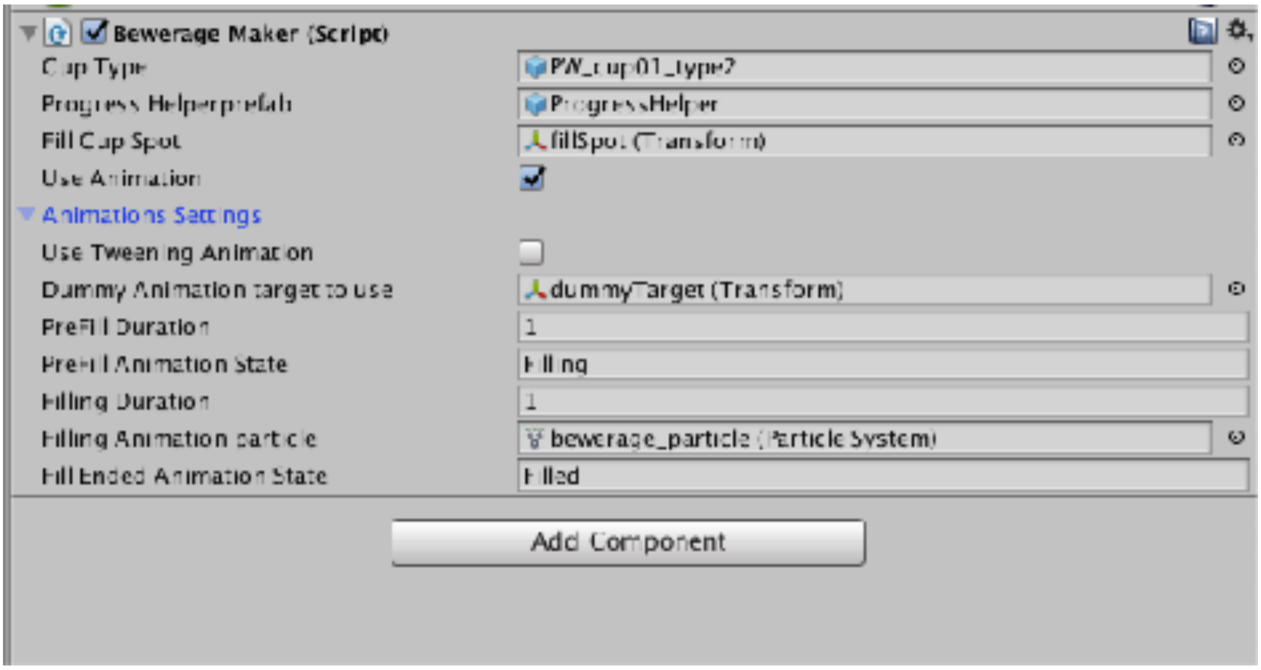
☑ Bewerage Maker (Script)

Cup Type	PW_tea_cup01_type2
Progress Helperprefab	ProgressHelper
Fill Cup Spot	CupSpot (Transform)
Use Animation	□
▸ Animations Settings	

Fill cup spot is a transform you can assign for where you want the cup to be generated. If you don’t use animation or use animation with tweeting, you can choose any place. If you use the animation with the dummy animation target you need to place it where the animation is targeted exactly. Refer to the scene to understand dummy target animation. There are examples for both use cases in the “InteractableObjectScene”.

Here you can see the dummy animated one. For dummy animation states to trigger you need to set the state trigger name in PreFill and Fill Ended Animation states.

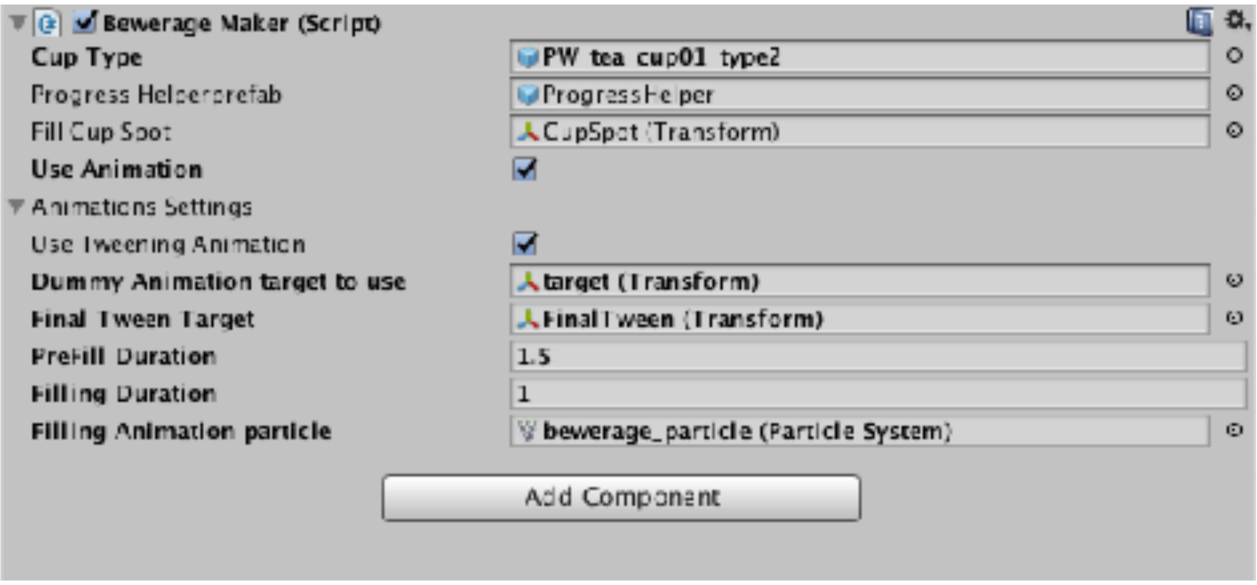
Don’t change the profile duration and filling duration on dummy target animations as they are suitable for the animation’s duration.



☑ Bewerage Maker (Script)

Cup Type	PW_cup01_type2
Progress Helperprefab	ProgressHelper
Fill Cup Spot	fillSpot (Transform)
Use Animation	☑
Animations Settings	
Use Tweening Animation	□
Dummy Animation target to use	dummyTarget (Transform)
PreFill Duration	1
PreFill Animation State	Filling
Filling Duration	1
Filling Animation particle	bewerage_particle (Particle System)
Fill Ended Animation State	Filled

If you use tweening animations you don’t need animation state names.



☑ Bewerage Maker (Script)

Cup Type	PW_tea_cup01_type2
Progress Helperprefab	ProgressHelper
Fill Cup Spot	CupSpot (Transform)
Use Animation	☑
Animations Settings	
Use Tweening Animation	☑
Dummy Animation target to use	target (Transform)
Final Tween Target	FinalTween (Transform)
PreFill Duration	1.5
Filling Duration	1
Filling Animation particle	bewerage_particle (Particle System)

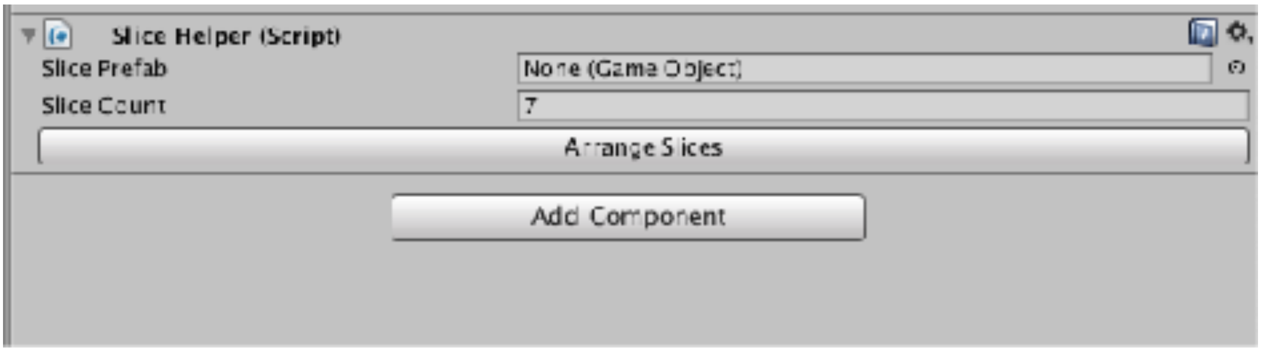
Here you can configure the duration and finalTween transform. You can duplicate the prefab object and set as the final tween target transform. You should delete any scripts on the final tween object, as it will only be used to tween as a transform for the final position and rotation of the object. If you use a Tweening Package like DOTween you can assign the tweeting yourself.

Animation particle can be used on both types of bewerage makers. Its basically representing the fluid of coffee or tea. You can change the material or particle however you like as long as you parent it to the object and position accordingly.

SliceHelper

SliceHelper is a basic utility script to use with cheesecakes. Basically we placed the cheesecake slice objects in the center of the display object. So that you can select one of them generate how many slices you wanted.

Add "ProductsWithCover" prefab from "Assets/CoffeShopStarterPack/Prefabs/Gameplay/" in to the scene. Select the object "CheeseCakeSlices" from children of the object.


Slice Helper (Script)
Slice Prefab: None (Game Object)
Slice Count: 7

Select a prefab from children of this object. Set slice count and press Arrange Slices. Slices will be generated and positioned. You can delete the component Slice-

Helper after you are done. If you want to you can create another version of this script to use for runtime purposes.