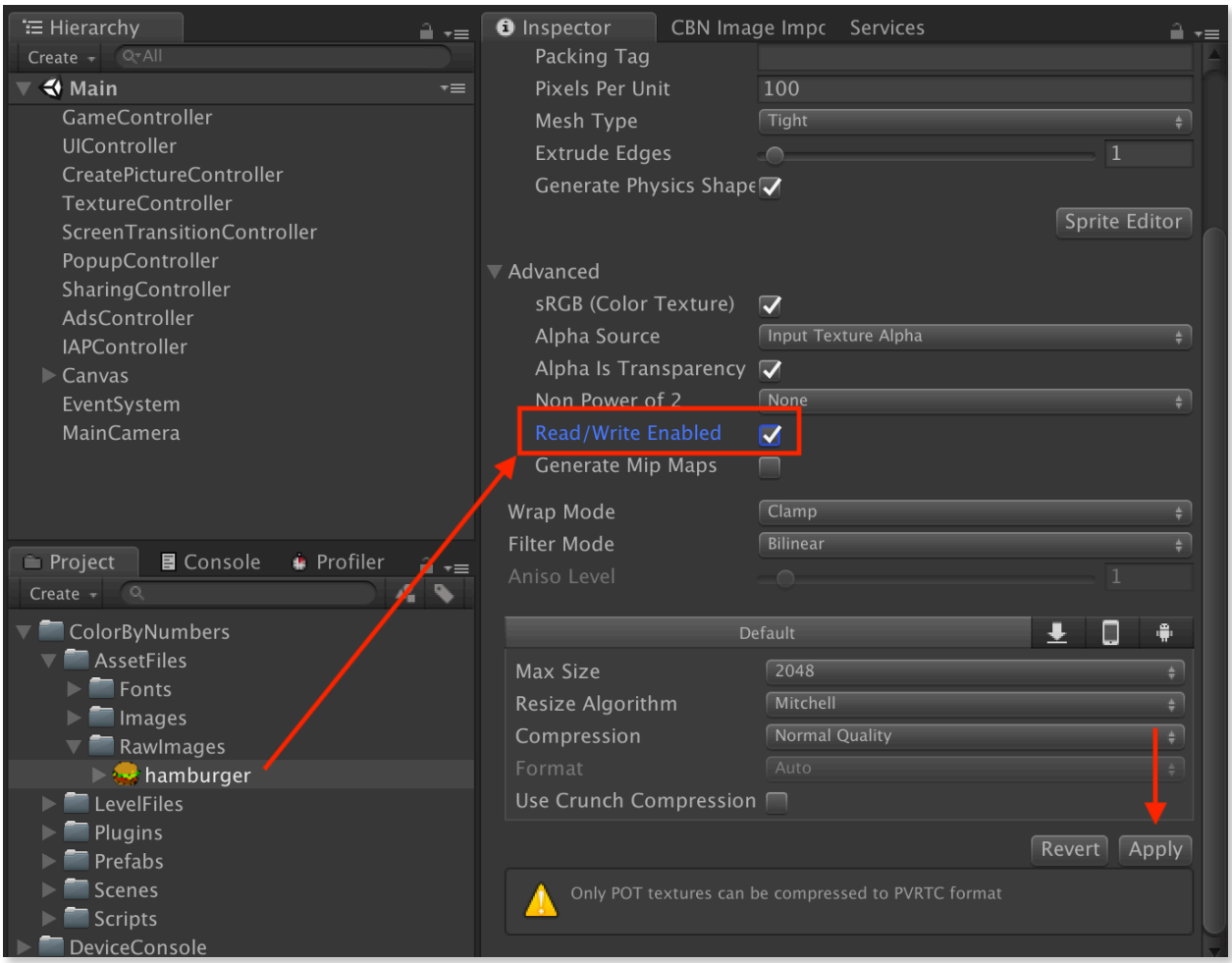
Color By Numbers Documentation

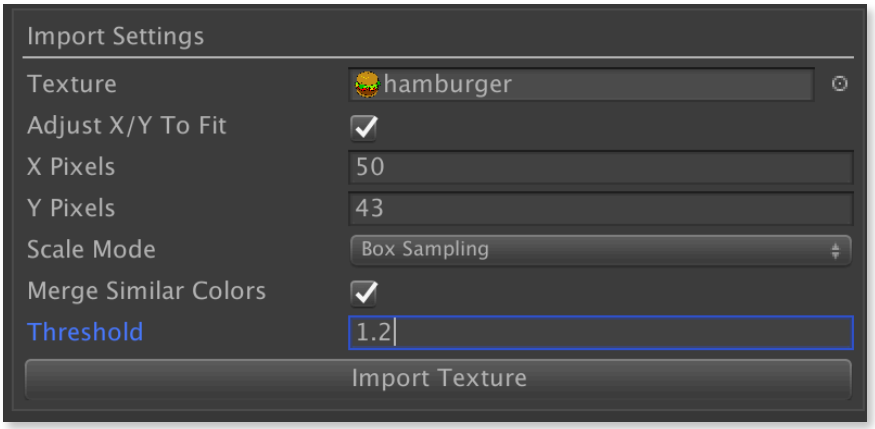
1 CBN Image Import Window

The CBN Image Import window is used to create the picture files that are loaded when the game runs. It provides a GUI interface for importing an image from the project folder, setting whether the picture is locked, setting the amount of currency to award on completion of the level, and exporting the image to a picture file that can be loaded at run time. To open the window, select the menu item **Tools -> Color By Numbers -> CBN Image Import**.

Note: Images must have the Read/Write flag set to true on the texture before importing them using the CBN Import Window. To do this, click on the image in the project then in the Inspector window check to **Read/Write Enabled** box, then click Apply at the bottom:



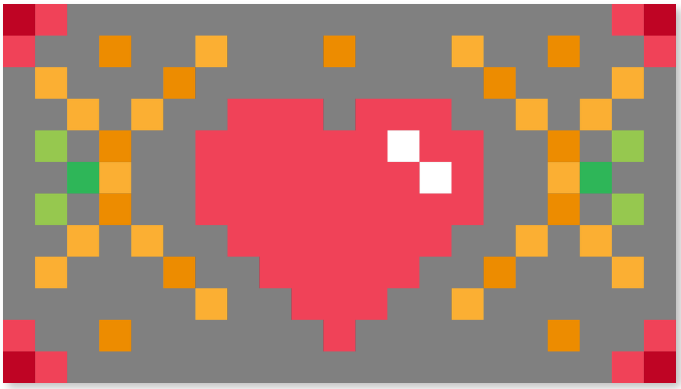
1.1 Import Settings



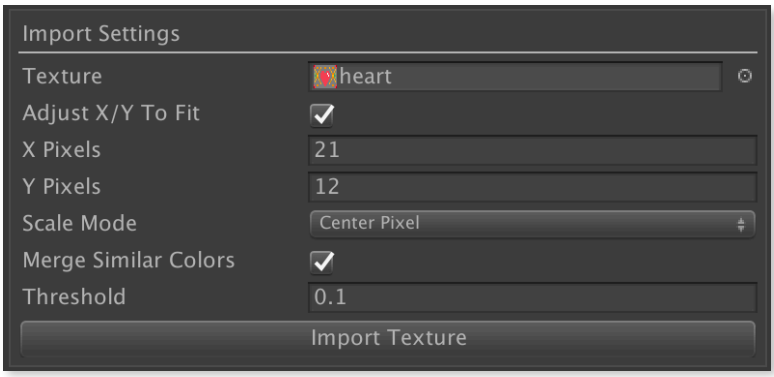
Fields	
Texture	The image to import into the window, drag a texture from the project window to import it.
Adjust X/Y To Fit	If this is checked then any time the X Pixels or Y Pixels changes the other will automatically change to keep the aspect ratio of the imported texture.
X Pixels	The number of x pixels in the final picture.
Y Pixels	The number of y pixels in the final picture.
Scale Mode	Specifies how the texture should be scaled to when imported. The given texture will be scale from it's current size to a width/height of X Pixels / Y Pixels. Box Sampling will take the average of all the pixels when scaling. Center pixel will simply take the middle pixel.
Merge Similar Colors	If checked then colors that are close together will be merged together to create one color.
Threshold	The minimum difference allowed between two colors when merging.

When importing a texture that is already "pixelized" it's best to use Center Pixel for the scale mode and either un-check Merge Similar Colors or set the threshold really low to something like 0.1. Then just set the X Pixels and Y Pixels to the number or X/Y pixels in the image.

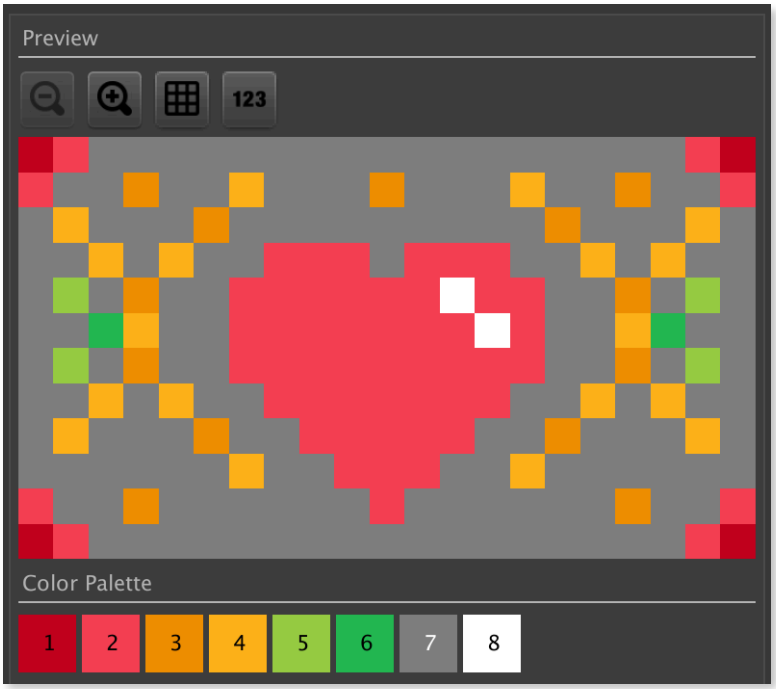
Example, say you want to import this image:



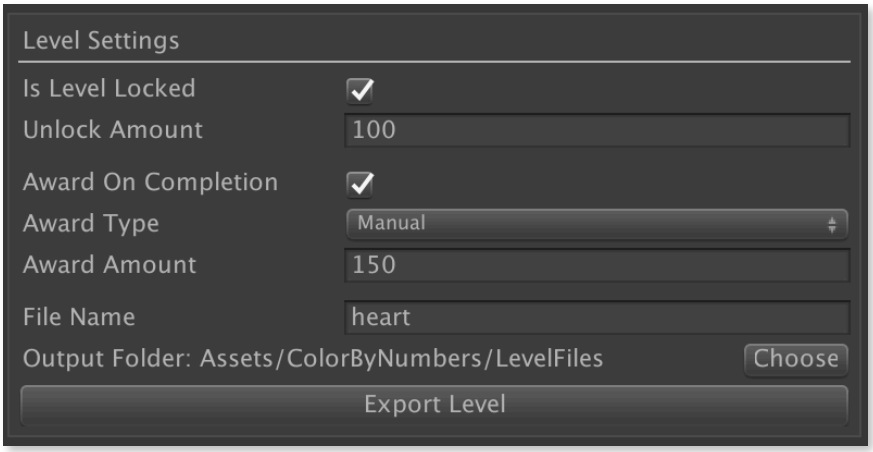
Then set the Import Settings to the following and click Import:



The imported image will be the same image with the same color palette and can now be exported to the picture file used by the game:



1.2 Level Settings



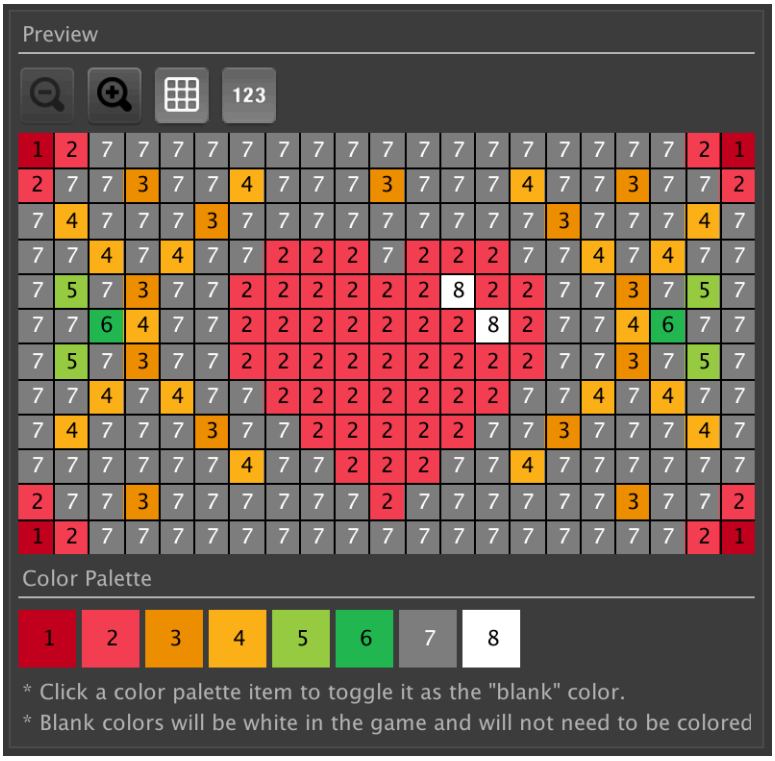
Fields	
Is Level Locked	Set to true if the level is locked and the player needs to spend coins to unlock it.
Unlock Amount	The amount of coins required to unlock the level.
Award On Completion	Set to true if this level awards coins when the player completes it the first time.
Award Type	Specifies how to assign the amount of coins to award below.
Award Amount	This field is shown if Award Type is Manual. Sets the amount of coins to award when the level is completed.
Amount Per Pixel	This field is shown if Award Type is Per Pixel. Sets the amount of coins per pixel that is awarded when the level is complete. Award Amount will appear and specify the amount that will be awarded.
File Name	The name of the picture file when exported. If left blank then the unique id of the picture file will be used.
Output Folder	The output folder for the picture file. Can be set by clicking the Choose button. The folder must be in the projects Asset folder.

1.3 Preview

The preview section is used to see what the picture will look like in the game and the color palette that is used. There are control buttons to zoom in/out of the picture to get a closer look at larger pictures. There are also two toggles that show / hide the grid lines and color numbers.

Note: The color numbers will not appear on large images until the preview is zoomed in enough so the individual cells are large enough to show the number.

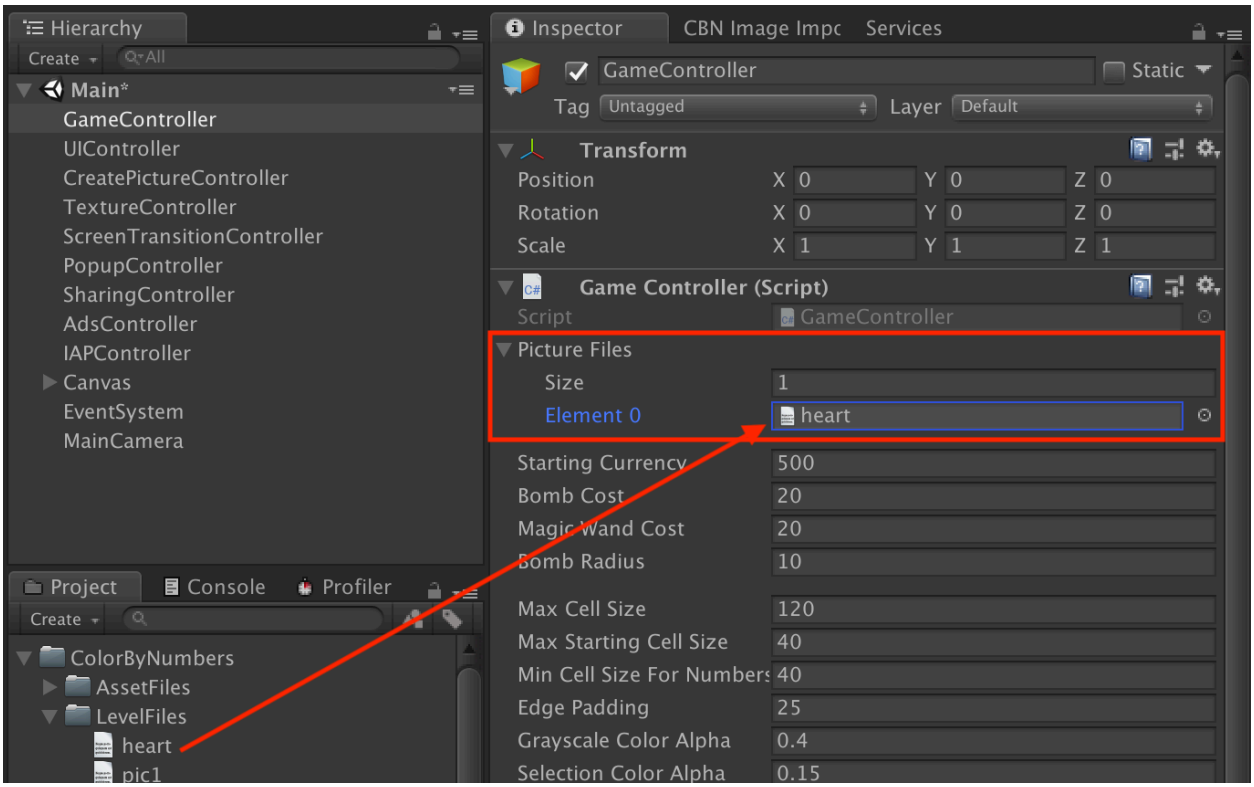
Clicking a color in the color palette will set it as the **blank** color. The blank color be set to completely transparent and will not need to be colored in in the game.



1.4 Create Picture File

When you are happy with the image that has been imported and finished setting the Level Settings, click the **Export Level** button. If there are no errors then a picture file will be generated and placed in the specified **Output Folder**.

In order to use this picture file in the game, simply add it to a **Picture Files** list in the **GameControllers** inspector.



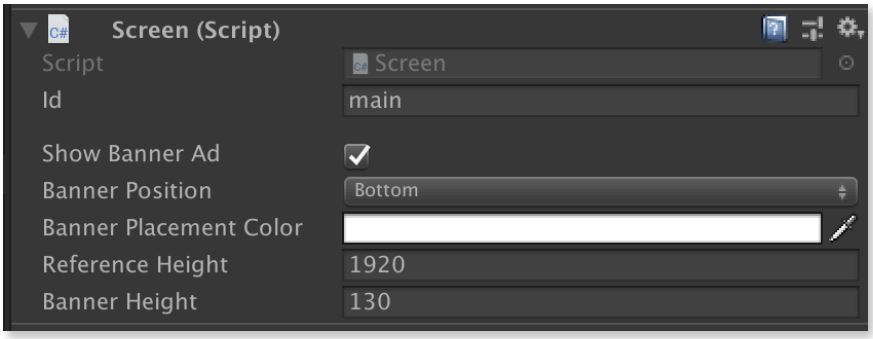
2 Ads

You can enable / disable banner and interstitial ads in the **AdsController** inspector. You can use either Unity Ads or AdMob for interstitial ads. AdMob ads will only appear on device, Unity Ads can be enabled to appear in the Unity Editor for testing purposes.

Interstitial ads will display right after the player selects a picture level to play. You can set the number of levels that must be started before a new interstitial ad appears by changing the **Num Levels Till Ad Shown** field on the **UIController**.

The AdMob Unit IDs that come with the asset are Googles test ids, you will need to replace them with your own if you would like to use AdMob.

UIScreen / Banner Ads

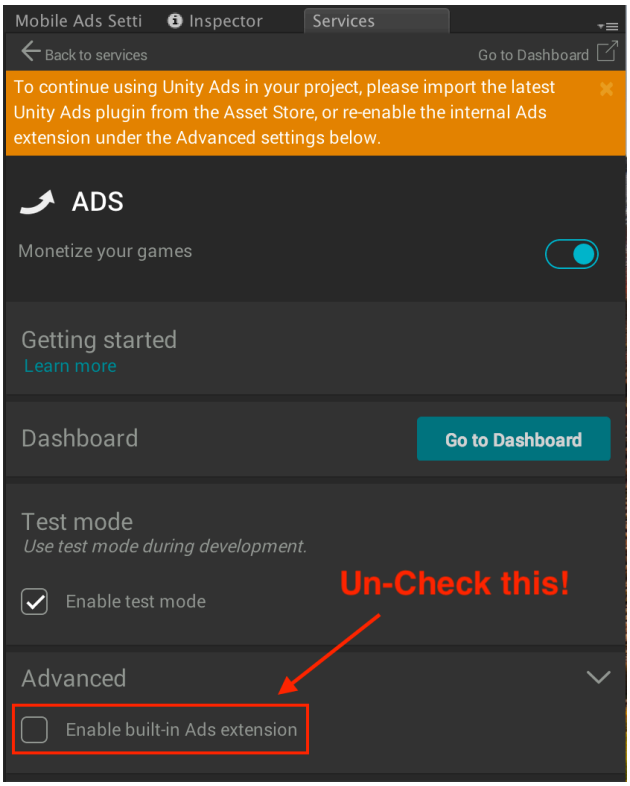


The UIScreen inspector can be used to enable / disable banner ads and also set the position of the banner ads on a screen by screen bases. There are only two screens currently in the asset (MenuScreen and GameScreen). If banner ads are enable on a UIScreen then at run time the UIScreen will automatically adjust it's layout to make room for the banner ad so that it does not block any UI.

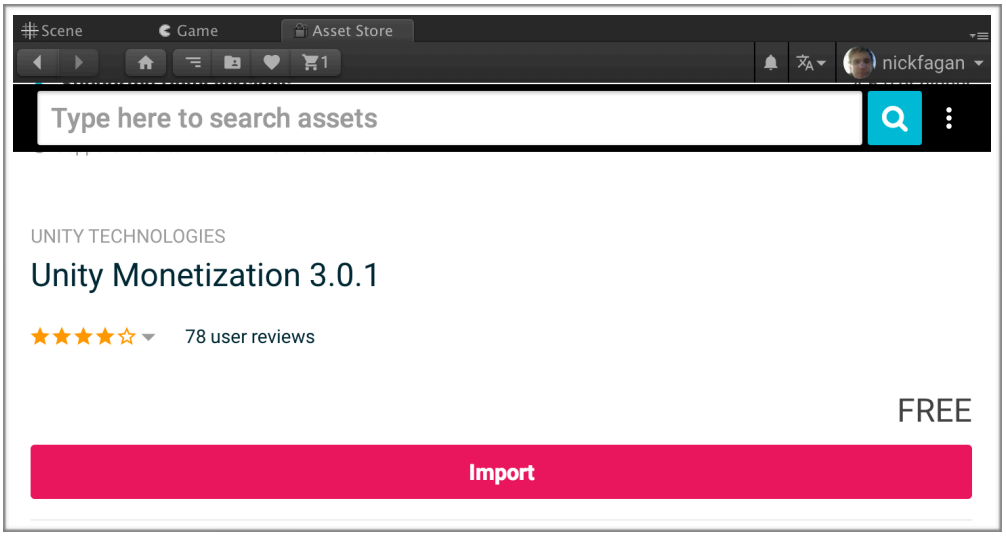
2.1 Unity Ads Setup

Step1. Enable Ads in the Services window (Navigate to **Window -> Services** and enable **Ads**.)

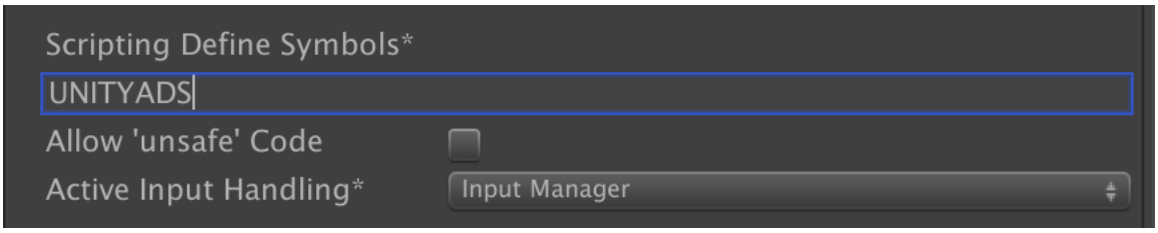
***** IMPORTANT ***** Make sure “Enable built-in Ads Extension” is disabled or it will collide with the Monetization plugin you will import in the next step. To do so expand the **Advanced** section and un-check the field if it is checked



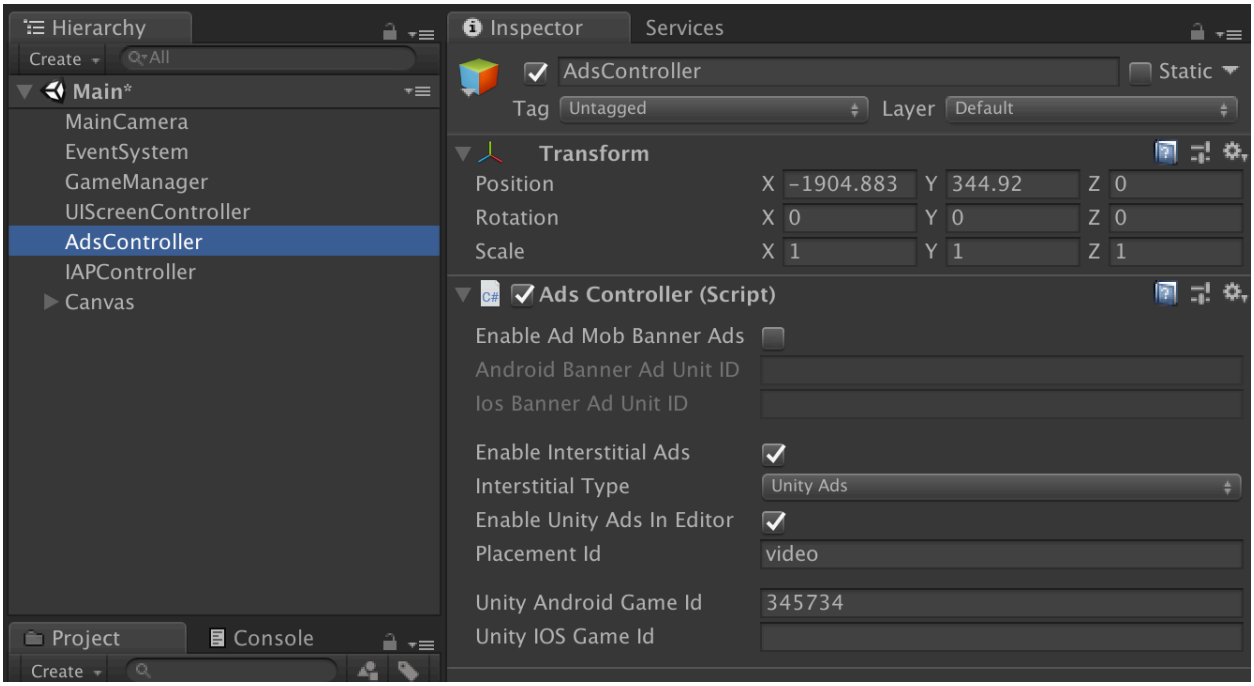
Step2. Open the Asset Store window and Download/Import the Unity Monetization asset:



Step3. Open Player Settings and add **UNITYADS** to the Scripting Define Symbols. (Make sure you hit the Enter key on your keyboard after typing it in so the changes are applied)



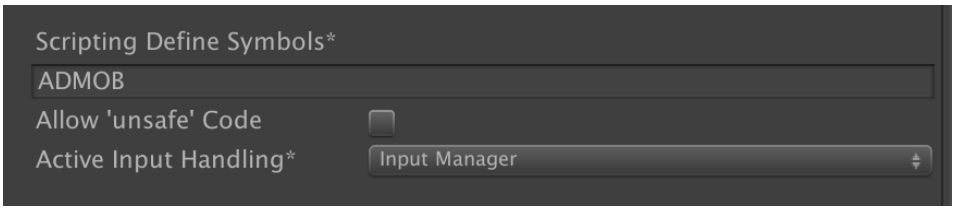
Step4. Select the AdsController in the Main scenes hierarchy and fill in your Game Ids and Placement Id for interstitial ads



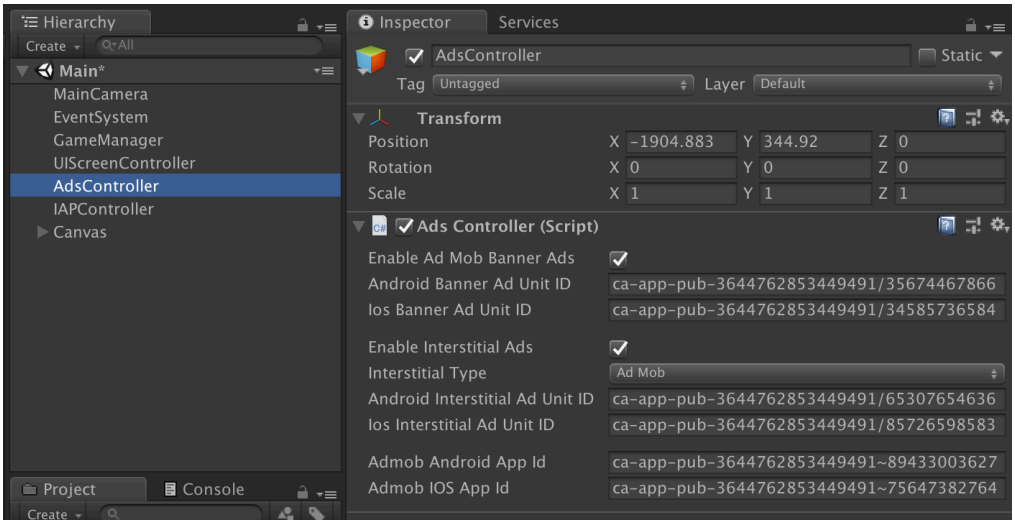
2.2 AdMod Setup

Step1. Download and import the AdMob Unity SDK by clicking on this link **https://github.com/googleads/googleads-mobile-unity/releases** and clicking the GoogleMobileAds.unitypackage

Step2. Open Player Settings and add **ADMOB** to the Scripting Define Symbols. (Make sure you hit the Enter key on your keyboard after typing it in so the changes are applied)



Step3. Select the AdsController in the Main scenes hierarchy and fill in your App Ids and Unit Ids



Step4 [Android Only]. Make sure the play services resolver that comes with the GoogleMobileAds plugin has executed by selecting the menu item **Assets -> Play Services Resolver -> Android Resolver -> Resolve**.

***** If using AdMob SDK version 3.18 or above *****

Step6 [Android only]. Select **Assets > Google Mobile Ads > Settings** from the menu. Enable AdMob by clicking Enabled checkbox under Google AdMob section. Then enter your Android and iOS AdMob app ID in each field.

***** If using AdMob SDK version 3.17 or below *****

Step6 [Android only]. Open the AndroidManifest located in the folder Plugins/Android/GoogleMobileAdsPlugin and add the following lines in the **application** element, replace ADMOB_APP_ID with your App Id:

<meta-data android:name="com.google.android.gms.ads.APPLICATION_ID" android:value="“ADMOB_APP_ID"/>

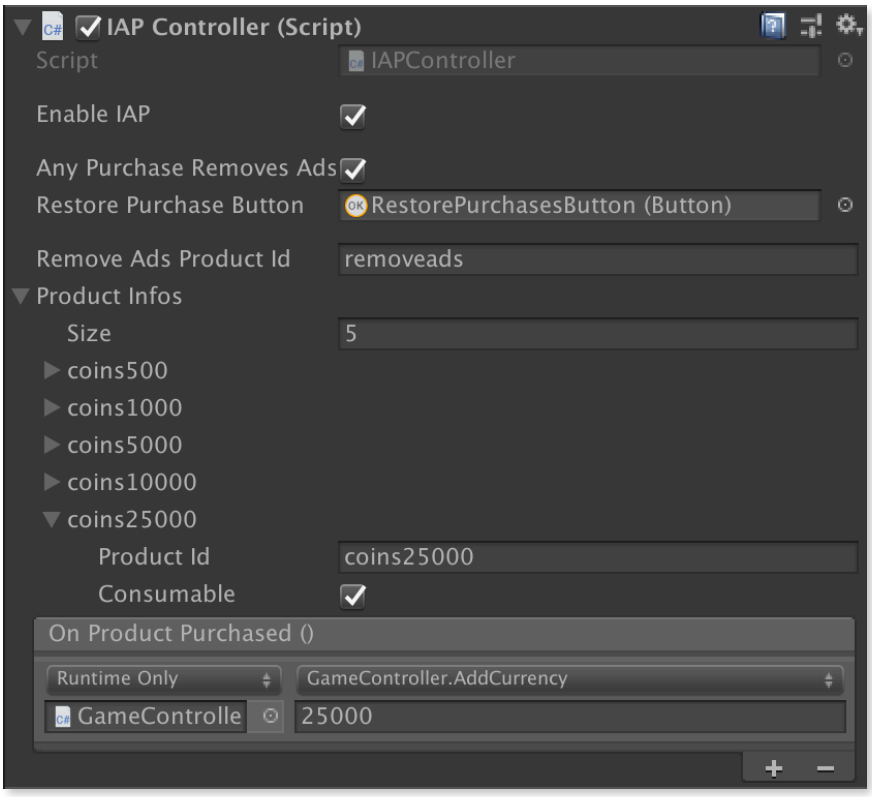
Your AndroidManifest should look something like this:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.unity.ads"
    android:versionName="1.0"
    android:versionCode="1">
  <uses-sdk android:minSdkVersion="14"
      android:targetSdkVersion="19" />
  <application>
    <meta-data
        android:name="com.google.android.gms.ads.APPLICATION_ID"
        android:value="ca-app-pub-3644762853449491~2999379837"/>
  </application>
</manifest>
```

3 IAP

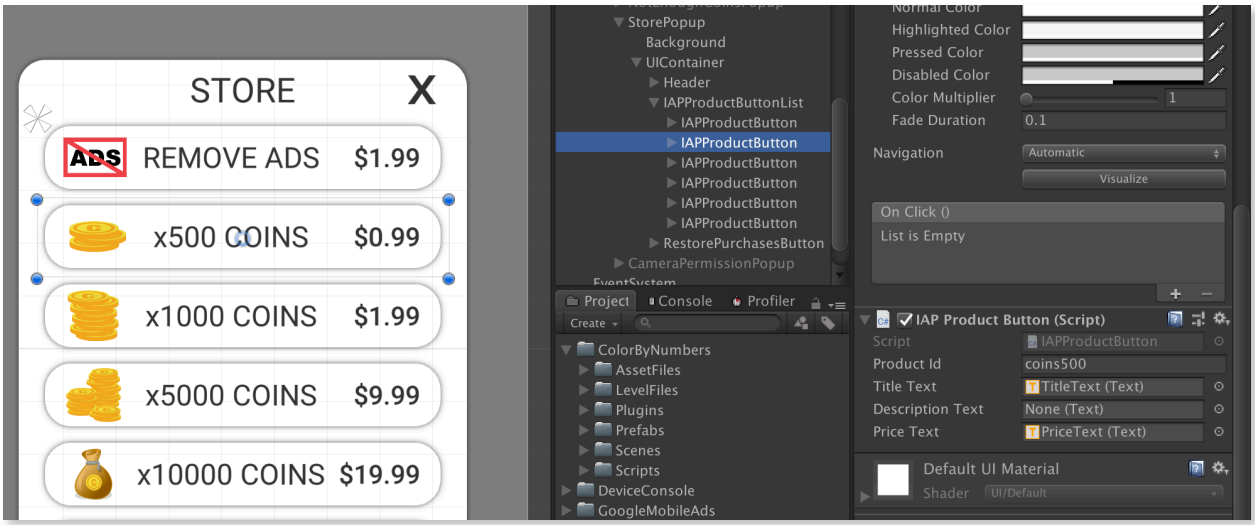
You can enable / disable IAP in the **IAPController** inspector. Currently the Color By Numbers asset is setup to handle two types of IAP purchases: removing ads and purchasing consumable coins.

To add products to purchase, simply increase the **Size** of the **Products Info** list and set the **Product Id**. You can then check if the item is consumable or not. The On Product Purchase event is invoke when the product is successfully purchased.



To purchase a product call method **IAPController.Instance.BuyProduct(productId)** passing it the product id to purchase. If the product was purchased successfully then the IAPControllers **OnProductPurchased** event will be invoked along with the OnProductPurchased event set in the IAPControllers inspector.

The **IAPProductButton** component can be used to make purchasing products simple. Attach the script to a Button and set the Product Id for the product you would like purchased when the button is clicked. The IAPProductButton component will automatically attach a click listener when the game runs and call IAPControllers ButProduct method when clicked. An example of it's usage can be seen in the StorePopup object in the scenes hierarchy.

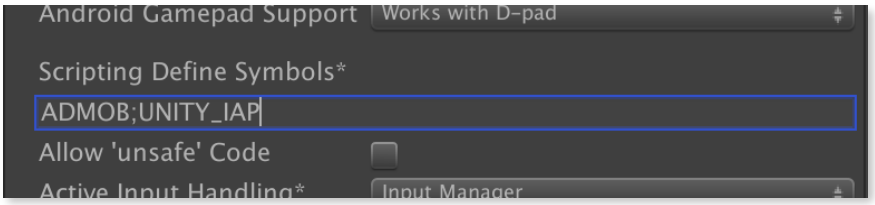


3.1 IAP Setup

To enable Unity IAP, first open the Services window by navigating to **Window -> Services** and enable **IAP**.

Next navigate to the **Player Settings** window and under the **Other Settings** add **UNITY_IAP** to the **Scripting Define Symbols**. (If there are already scripting define symbols, you can add others by simply separating them with a semi-colon)

NOTE: This setting is not shared between platforms. You need to add it to the Player Settings on both Android and iOS platforms.

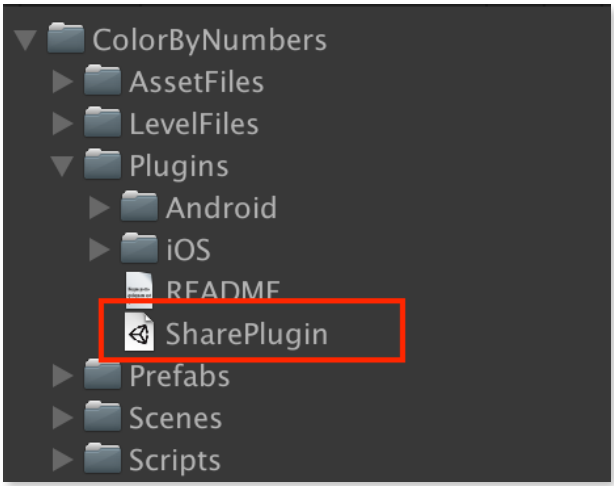


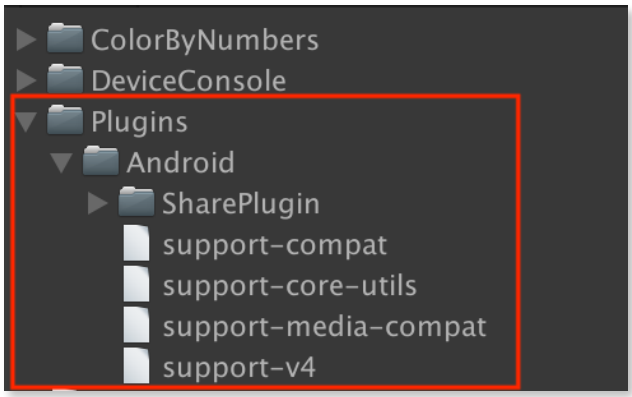
4.0 Sharing

Sharing is handled using native plugins for Android and iOS. The project contains a class called **NativePlugin.cs** which is the bridge between the C# code and the Android/iOS code. The SharingController is used to invoke the actual methods inside NativePlugin.

4.1 Android Plugin Setup

To setup the Android plugin to enable sharing simply double click the **SharePlugin.unitypackage** located in the folder **ColorByNumbers/Plugins/Android**. This will import the unity package and place the Android Library in the proper place. The Android plugin **MUST** be located in the root Plugins folder **Assets/Plugins/Android** or it will not work properly.





***** IMPORTANT *****

The SharePlugin uses the package name as part of the authority for the FileProvider.
If you ever change the package name in Player Settings you must also change the package name in the **AndroidManifest.xml** file located in folder **Assets/Plugins/SharePlugin** after importing the SharePlugin.unitypackage.

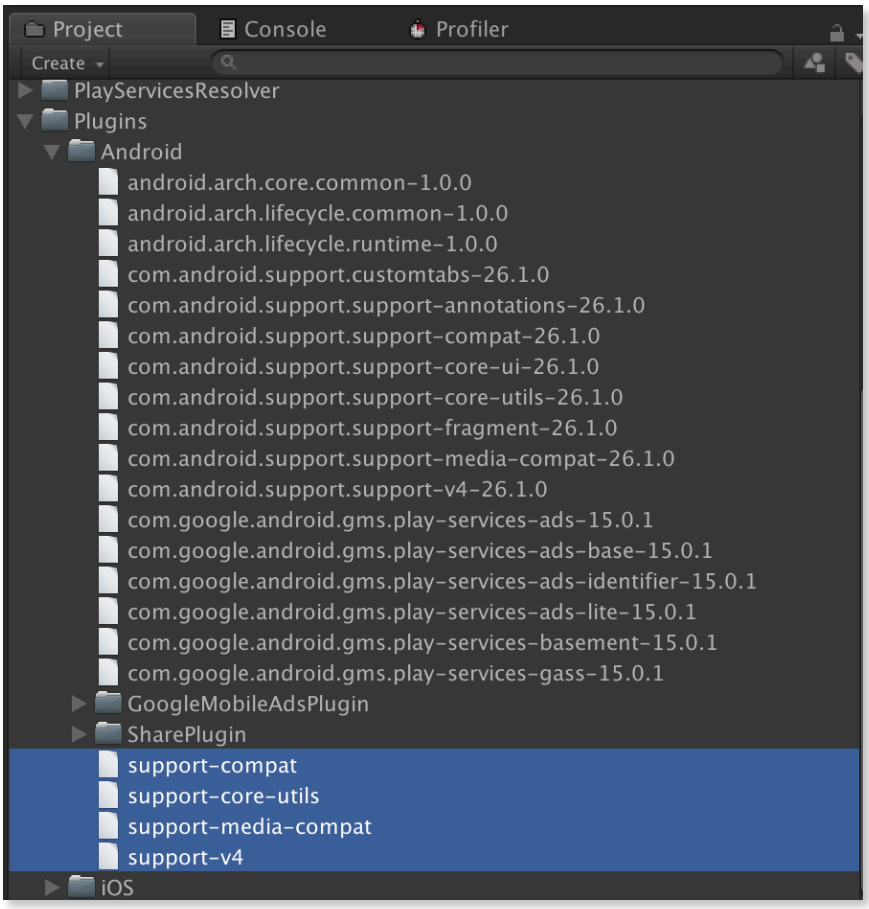
Do not change the “package” in the AndroidManifest, only the authorities:

Line 9: android:authorities="**your.package.name**.fileprovider"

(See screenshot on next page for exact locations)

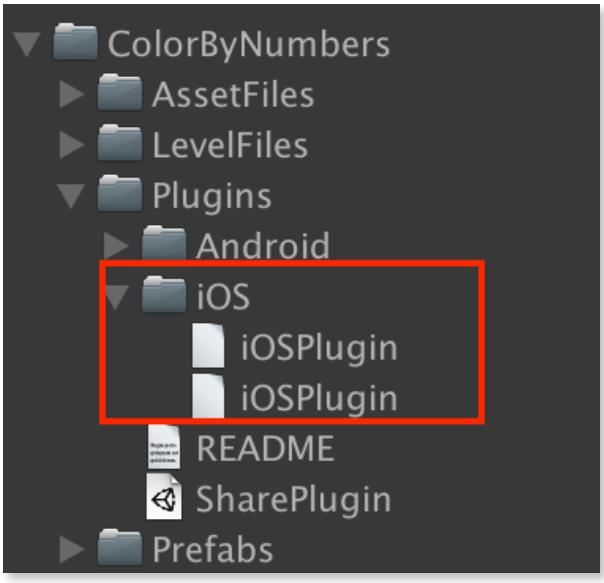
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.plugin.share">

    <application>
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.bizzybeegames.colorbynumbers.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/provider_paths"/>
        </provider>
    </application>
</manifest>
```

NOTE: If you are using AdMob in the project and have imported the Unity AdMob SDK from section 2.2 then after importing the SharePlugin you must delete the four support files highlighted in the image below of you will encounter build errors when building to Android.



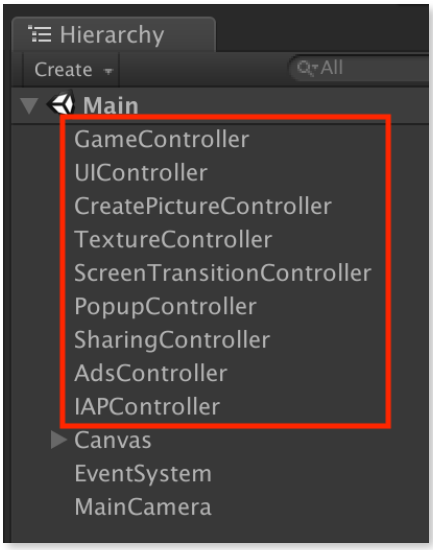
4.2 iOS Plugin Setup

The iOS share plugin is already ready to go in the asset. The **ShareController.cs** script contains a constant variable called **LibraryUsageDescription** which is added to the iOS Info.plist file when building to iOS from Unity. This field is required in order for the application to display the permission dialog if the user selects “Save Image” after clicking the share other button.



5.0 Controller Classes

The Controller scripts handle the majority of the functionality throughout the game. You can get a reference to any of the controller classes using the static “Instance” property. (EX. To get a reference to the GameController simply type **GameController.Instance**). All of the Controller class are defined at the root of the scenes hierarchy.



5.1 GameController

The GameController handles starting and playing a level. It’s responsible for setting up the game UI to play an active level. It also gets events from the PictureScrollArea (The component which handles input) for when the user taps / drags the screen and colors the pixels in the active level.

Public Properties	
PictureInfos	Gets a list of PictureInformation objects. Each picture file will have a PictureInformation object created for it which contains all the information about the level and any save data for the level.
PlayedPictureInfos	Gets a list of PictureInformation objects that have been played. If a level has been started it will be added to this list.
ActivePictureInfo	The PictureInformation for the level that is currently being played.
CurrencyAmount	The amount of coins the player has.

Public Methods	
StartLevel	Starts the level for the given PictureInformation. This will setup the GameScreen so when the GameScreen is shown the levels textures are displayed and the color palette list has all the colors in it.
OnBombSelected	Toggles the bomb power up on/off. If the bomb power up is selected then the next time the user taps the grid the bomb power up will be used.
OnMagicWandSelected	Toggles the magic wand power up on/off. If the magic wand power up is selected then the next time the user taps the grid the magic wand power up will be used.
UnlockLevel	Unlocks the given PictureInformation so the user can start it. If the level costs coins the the amount of coins will be deducted from CurrencyAmount.
AddCurrency	Adds the given amount of coins to CurrencyAmount
DeleteLevelProgress	Clears any saved progress for the given level.
CompleteActiveLevel	Completes the ActivePictureInfo as if the user clicked all the cells at once.

5.2 UIController

The UIController handles most things on the MenuScreen related to UI. It handles switching between menu screens (Library, My Works, Create). It also handles filling the Library and My Works list items and pooling the items when they scroll off screen.

Public Constant Variables	
CategoriesMenuScreenId	The Id given to the categories MenuScreen item.
LibraryMenuScreenId	The Id given to the library MenuScreen item.
MyWorksMenuScreenId	The Id given to the my works MenuScreen item.
CreateMenuScreenId	The Id given to the create MenuScreen item.

Public Methods	
OnLibraryScrolled	Should be assigned to the ScrollRects OnValueChanged event property for the library list.
OnMyWorksScrolled	Should be assigned to the ScrollRects OnValueChanged event property for the my works list.
ShowMenuScreenById	Shows the MenuScreen with the given id.

5.3 CreatePictureController

The CreatePictureController handles everything on the CREATE screen. It handles getting an instance of WebCamTexture and transforming that texture to the pixelized grayscale texture that is displayed on the screen. Once the “accept” checkmark button is clicked it processes the image to reduce the color palette and automatically starts the level. The created picture level will then appear on the my works screen.

Public Methods	
StartCapturing	Starts capturing frames from the devices camera and displaying them on the screen. The device must have the proper permissions for this to work.
StopCapturing	Stops capturing frames.
SwitchCamera	Switches cameras if there are more then one camera on the device.
Preview	When called this stops capturing and displays the final captured frame that will be used for the level. In order to play the level the Process method must be called.
Process	Takes the last captured frame and reduces the color palette based on the settings set in the inspector. Processing is done on a separate thread and once finished the level is automatically started.
GetUserCreatedPictureContents	Gets the JSON for all the user created picture files.

5.4 TextureController

The TextureController handles taking a PictureInformation and generating the necessary textures required when playing the level. The controller caches the grayscale and completed images so if they are request again and are cached they don't have to be re-loaded. The maximum number of images that can be cached is set in the controllers inspector.

Public Methods	
LoadGrayscale	Loads the grayscale image for the given PictureInformation. If the level has progress then this will also set the pixels that have been colored.
LoadCompletedTexture	Loads the completed picture with all pixels colored their proper color.

Public Methods	
LoadGameTextures	Loads all the textures required by the GameController to play the game. The contents of the list are as follors: [Mask Texture, Grayscale Texture, Colored Texture, Selection Overall Textures…]. The Mask Texture is a texture where all blank pixels are white, this is used as a mask on the picture to hide grid lines. The Grayscale Texture is the pictures grayscale (without colored pixels), the Colored Texture is a texture where all the already colored pixels are set, the Selection Overlay Textures are all the textures that highlight where the selected colors pixels are located in the picture.

5.5 ScreenTransitionController

The ScreenTransitionController handles transitioning between full screens in the game. There are only two screens in Color By Numbers, the MenuScreen and GameScreen. The MenuScreen contains 3 sub-screens and the UIController handles showing / hiding those screens.

Public Methods	
Show	Shows the screen with the given id. The screen must be in the Screens list in the controllers inspector.
TransitionScreens	Animates the fromScreen off the screen and the toScreen onto the screen. Does not keep track of what screen is currently showing. This is used to transition the 3 sub screens in MenuScreen.
SetOffScreen	Automatically sets the screen so its not visible, does not animate it.

5.6 PopupController

The Popup controller provides a simple way to show popups in the game and get notified though a callback when the popup closes and gets any important data / information when it closes.

To show a Popup call the method **PopupController.Instance.Show(id, inData, popupClosed)**. The **id** is the id of the popup set in the Popup Infos list, the **inData** is an array of objects that the specific popup is expecting in order to display what it needs to display. (See LevelLockedPopup.cs OnShowing method for an example of how to extract the data when the popup shows). The **popupClosed** is a callback that is invoked when the popup is hidden, the popup can set the outData that will be set when invoking the callback.

Public Methods	
Show	Shows the popup with the given id.

5.7 SharingController

The Sharing Controller handles sharing an image (Texture2D) to Twitter, Instagram, and any other application that can handle the image.

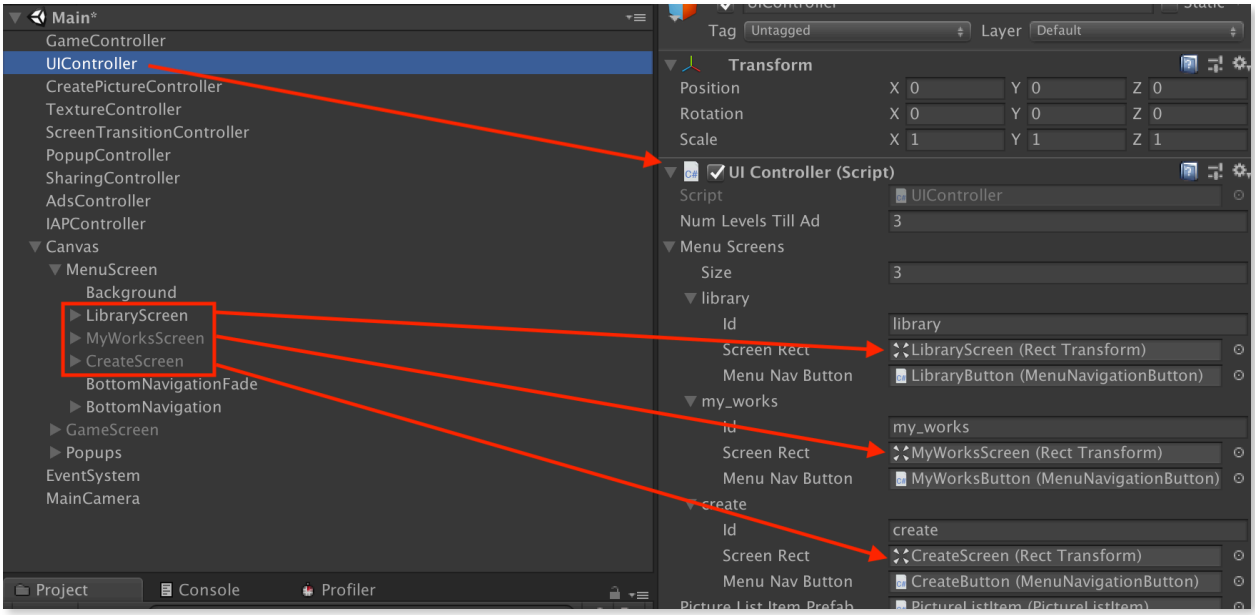
Public Methods	
ShareToTwitter	Shares the given texture to the Twitter application. Returns false if the application is not installed on the device. (Always returns false in editor)
ShareToInstagram	Shares the given texture to the Instagram application. Returns false if the application is not installed on the device. (Always returns false in editor). **NOTE:** On iOS it is not possible to directly open the Instagram application, instead it opens the share view with only the Instagram icon in it, the user must then click the Instagram icon to open Instagram)
ShareToOther	Opens the devices sharing view to allow the user to select what application they want to share the image to. This list is populated by the devices OS and cannot be controlled by the asset.

6.0 Project Setup - Menu Screen

There are two main scenes in the asset located in the Scenes folder. **Main_NoCategories** is a version of the game that does not use categories. All levels are displayed at the same time in the Library. **Main_Categories** does use categories. In this scene there is an extra CategoryScreen that appears first, selecting one of the categories then shows all the levels that are part of that category only.

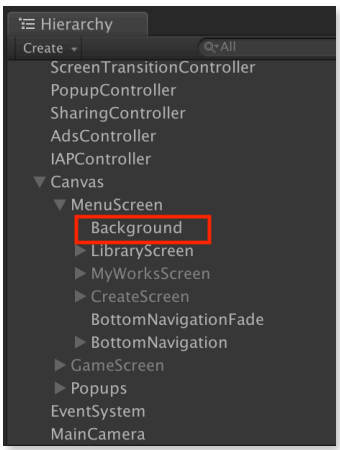
6.1 Menu Screens

The MenuScreen has three sub-screen: LibraryScreen, MyWorksScreen, and CreateScreen. These sub-screens are controlled by the UIController and can be shown/hidden using the UIControllers ShowMenuScreenById method.



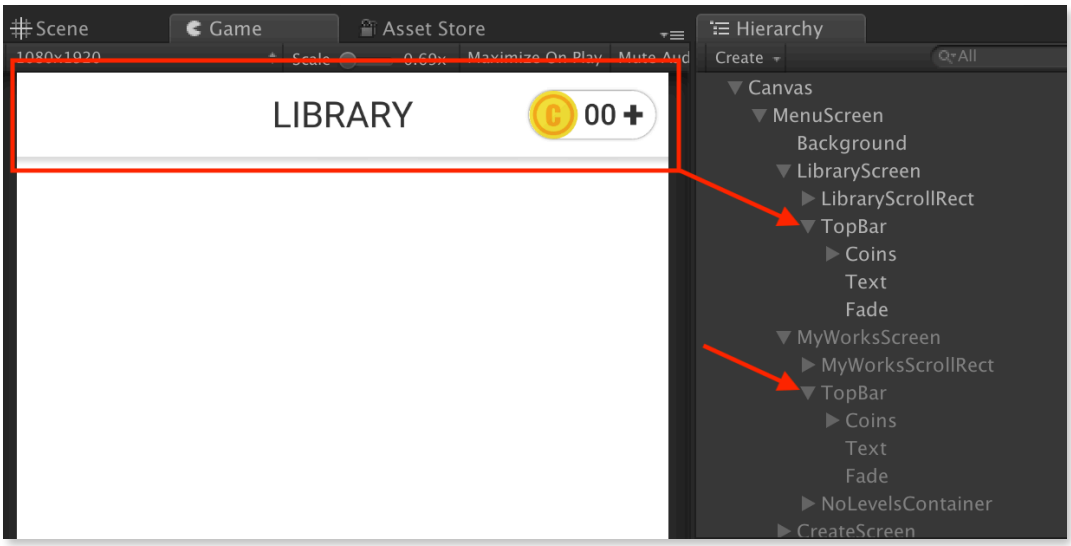
6.2 Background

The background for the MenuScreen is set on the Background object, currently is is a white Image with no Sprite.



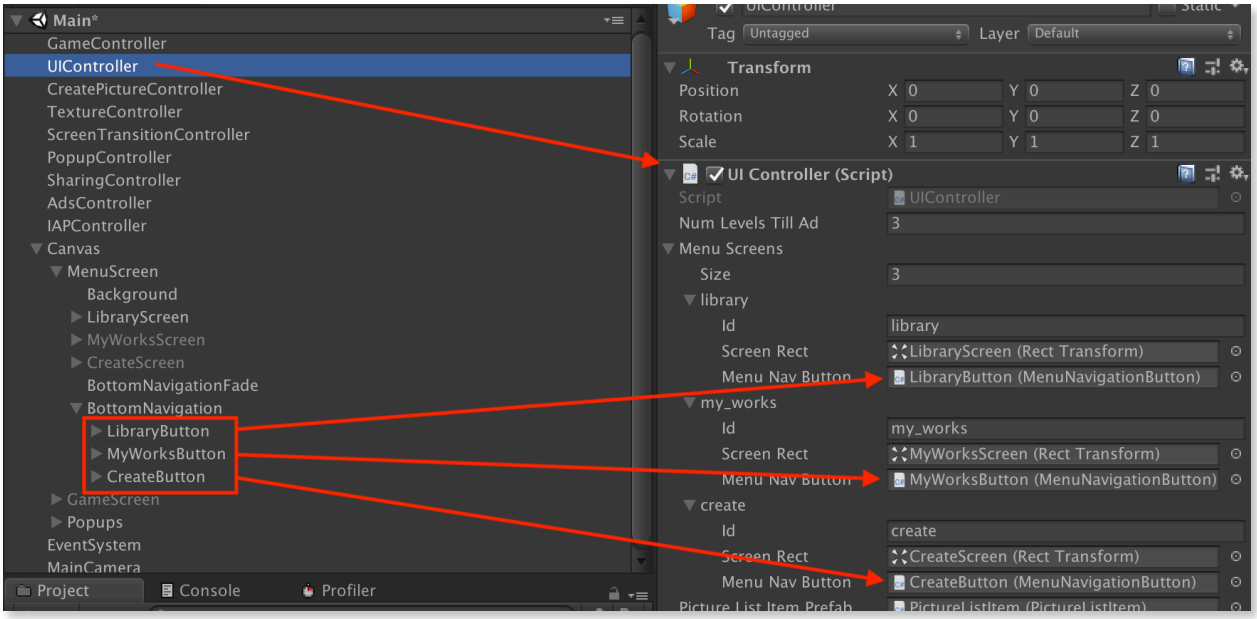
6.3 Top Bar

There is a TopBar object for both the LibraryScreen and MyWorks screen. They are identical expect for the text that appears on the bar. The coins amount text is set using the **CurrencyAmountText** component. If this component is attached to an object with a Text component whenever the CurrencyAmount changes this component will update the text.



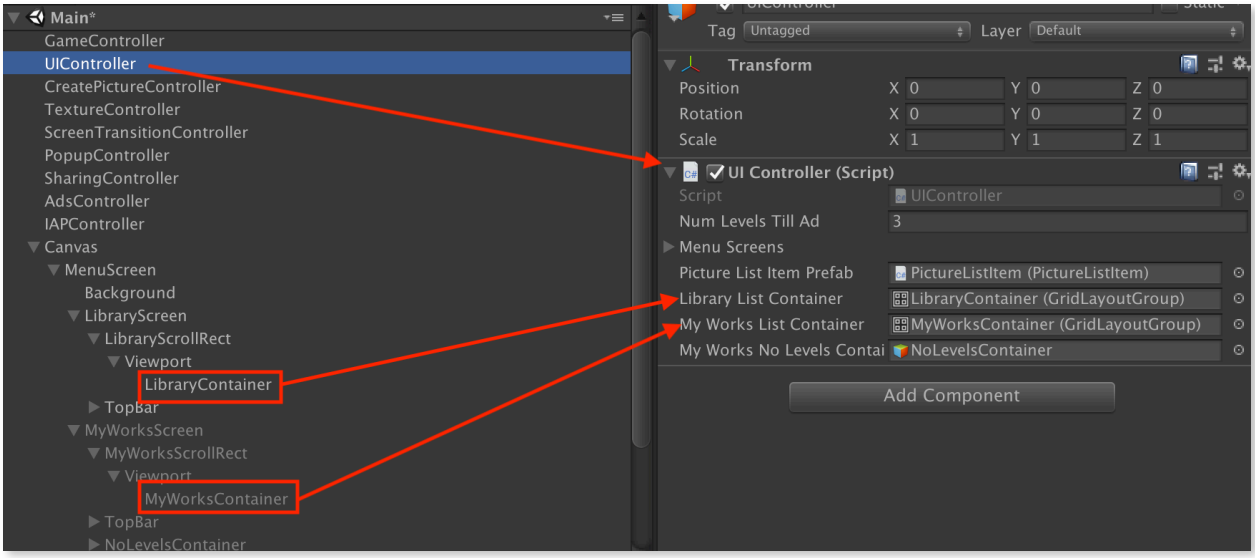
6.4 Bottom Navigation

The BottomNavigation are the 3 buttons that when clicked show the 3 sub-screens. They are assigned to the Menu Nav Button field in the Menu Screens list on the UIController. Each button object has a **MenuNavigationButton** component attached to it. On the component set the Id of the screen it should show. The Icon Image and Button Text will have their colors set to the Normal color when the screen is not showing is the Selected color when the screen is showing.



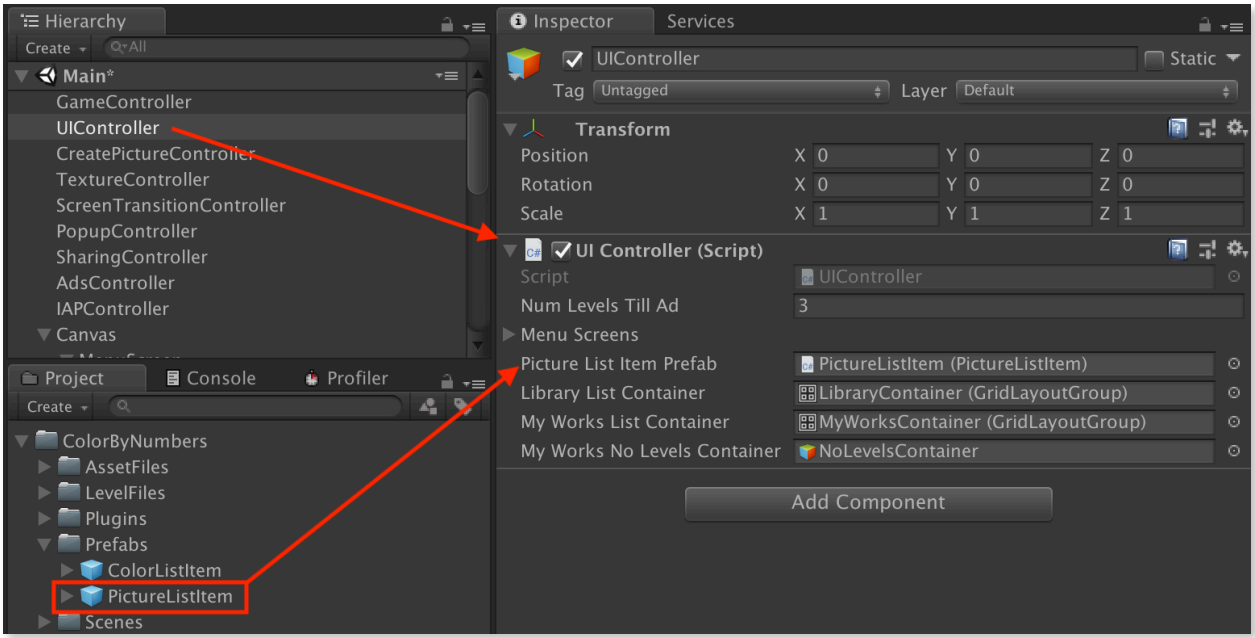
6.5 Library / My Works List

The **LibraryContainer** and **MyWorksContainer** are list Transforms that the PictureListItems prefab is instantiated and placed in. They both have **GridLayoutGroups** attached to them that control the size of the items. The UIController handles instantiating the PictureListItem prefab and placing them in the list containers. It also keeps track of what items are visible on the screen and what items aren't. As items scroll on/off the screen it pools the items, only ever instantiating enough items to fill the screen and re-using items that scroll off the screen. This increase the performance of the list and allows a very large number of items without any lag. The UIController also adjusts the cell size of the GridLayoutGroups for difference screen sizes.



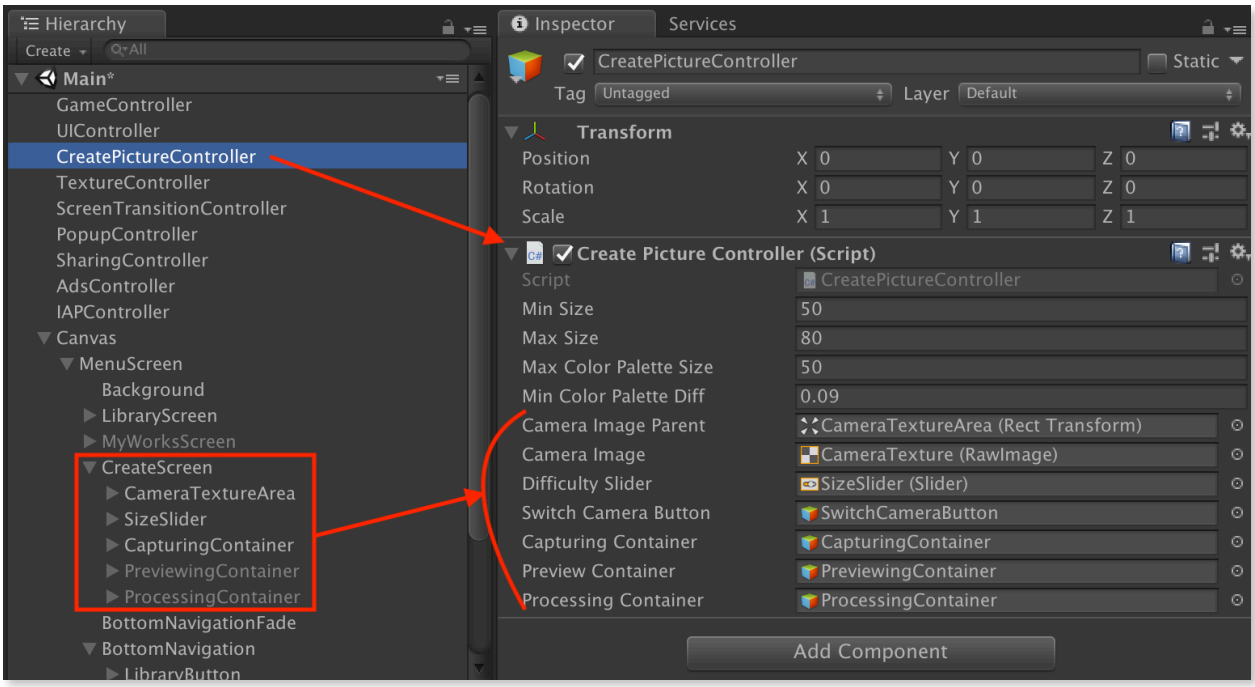
6.6 Picture List Item

The PictureListItem is a prefab located in the Prefabs folder. If is assigned to the UIController and the UIController uses the prefab to instantiate copies of the list item and places them in the list containers.



6.7 Create Screen

The CreateScreen is controlled by the CreatePictureController. The CREATE nav button has an on click event that calls the controllers StartCapturing method. The other two nav buttons call the controllers StopCapturing method so if the user navigates away from the create screen the controllers stops capturing camera frames.



7.0 Project Setup - Game Screen

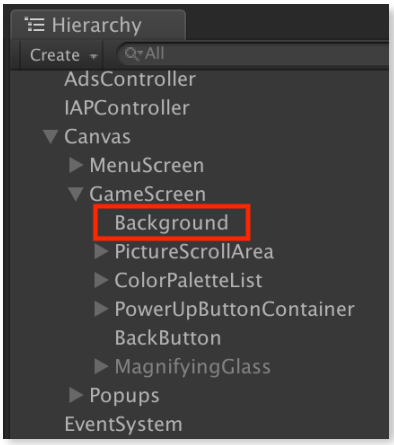
7.1 Zoom In Editor

You can zoom a picture in and out in the Unity editor by first clicking and holding the picture to drag it. Then, at the same time, hold down the ‘Z’ key on the keyboard. While both the mouse and z key held down you can then move the mouse to zoom in/out.

NOTE: You must first click and hold with the moues before the ‘Z’ key. If you are hold the ‘Z’ key before clicking with the mouse then the picture wont zoom.

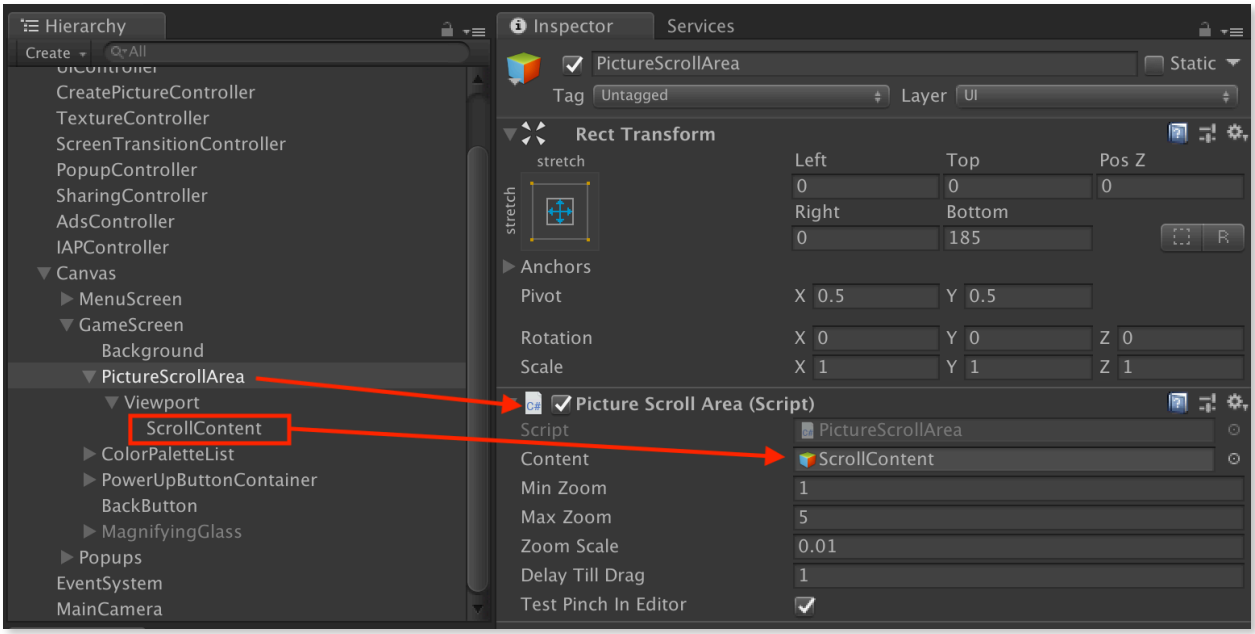
7.1 Background

The background for the GameScreen is set on the Background object, currently is is a white Image with no Sprite.

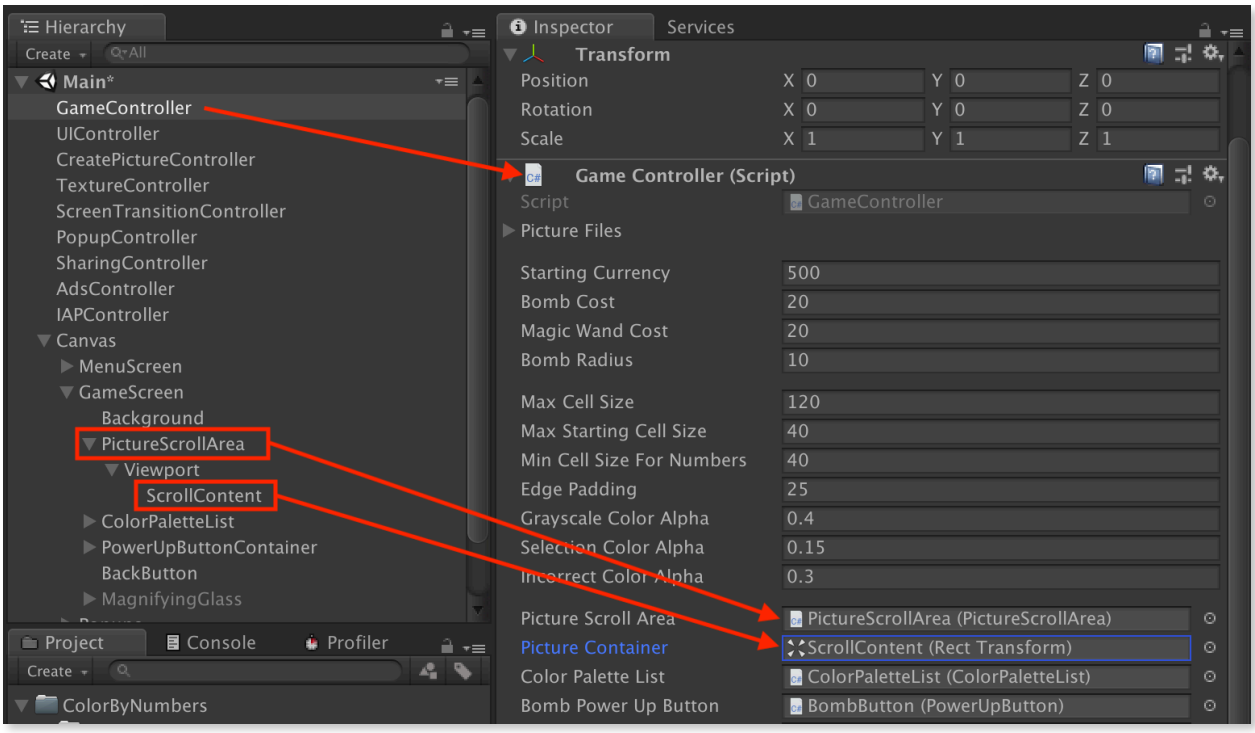


7.2 Picture Scroll Area

The PictureScrollArea handles mouse input for dragging / zooming the picture when playing the game. The **ScrollContent** object is the Transform that is scaled up/down when zooming occurs and is the object that is moved when dragging occurs. When a game starts the picture objects are scaled down to fit within the ScrollContent so when the ScrollContent is zoomed / dragged the picture content is zoomed / dragged with it. The PictureScrollArea has a number of public events that are invoked and the GameController listens to those events so it knows when to color pixels or update the visibility of numbers and grid lines.

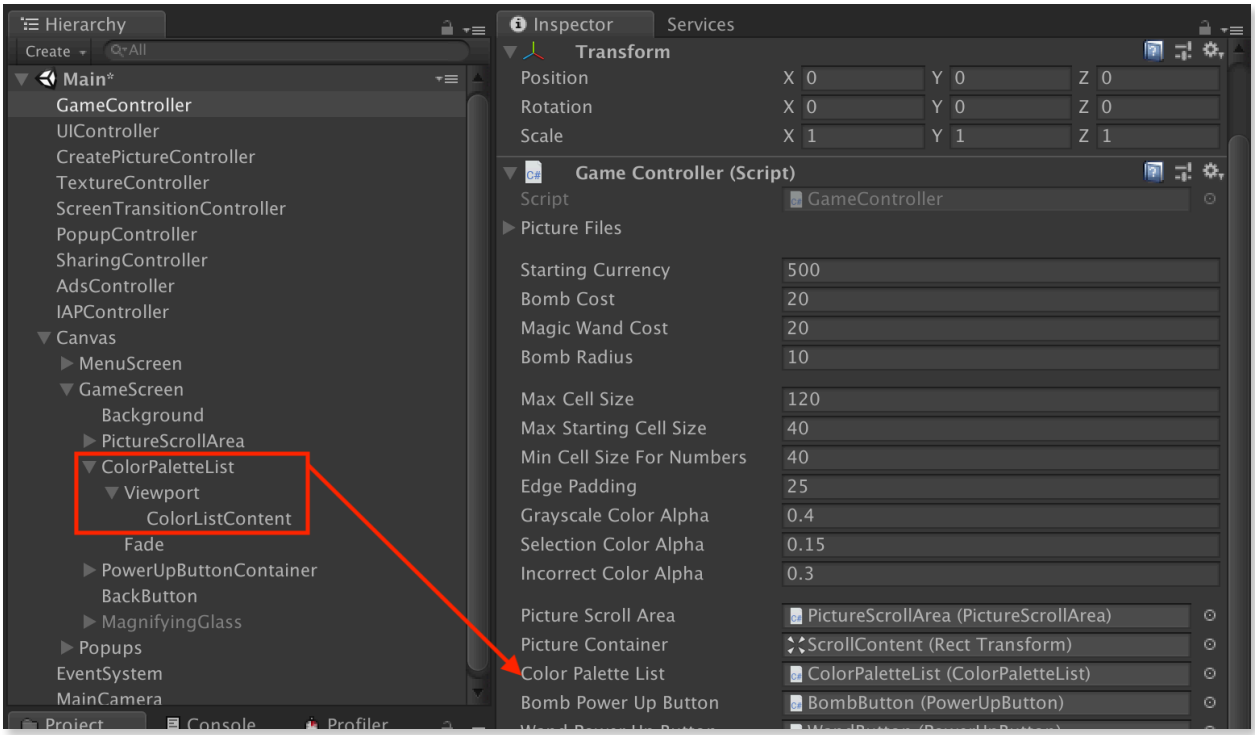


The PictureScrollRect and ScrollContent are then assigned to the GameController so it can receive events from the PictureScrollRect. The ScrollContent is used as the Transform that the picture objects which are generate when a level starts are placed inside



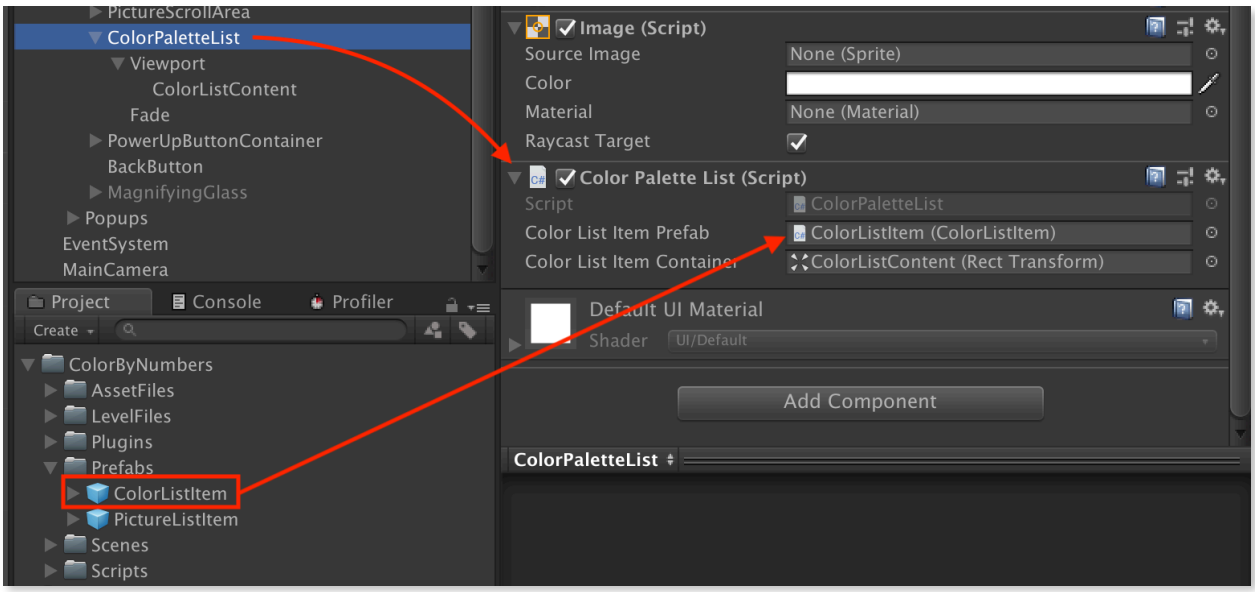
7.3 Color Palette List

The ColorPaletteList handles displaying the list colors that are in the picture. It is assigned to the GameController and when a level starts the GameController passes it the PictureInformation so that it can add the proper colors. When a color is selected the ColorPaletteList invokes an event which the GameController is listening to so if knows when a color is selected.



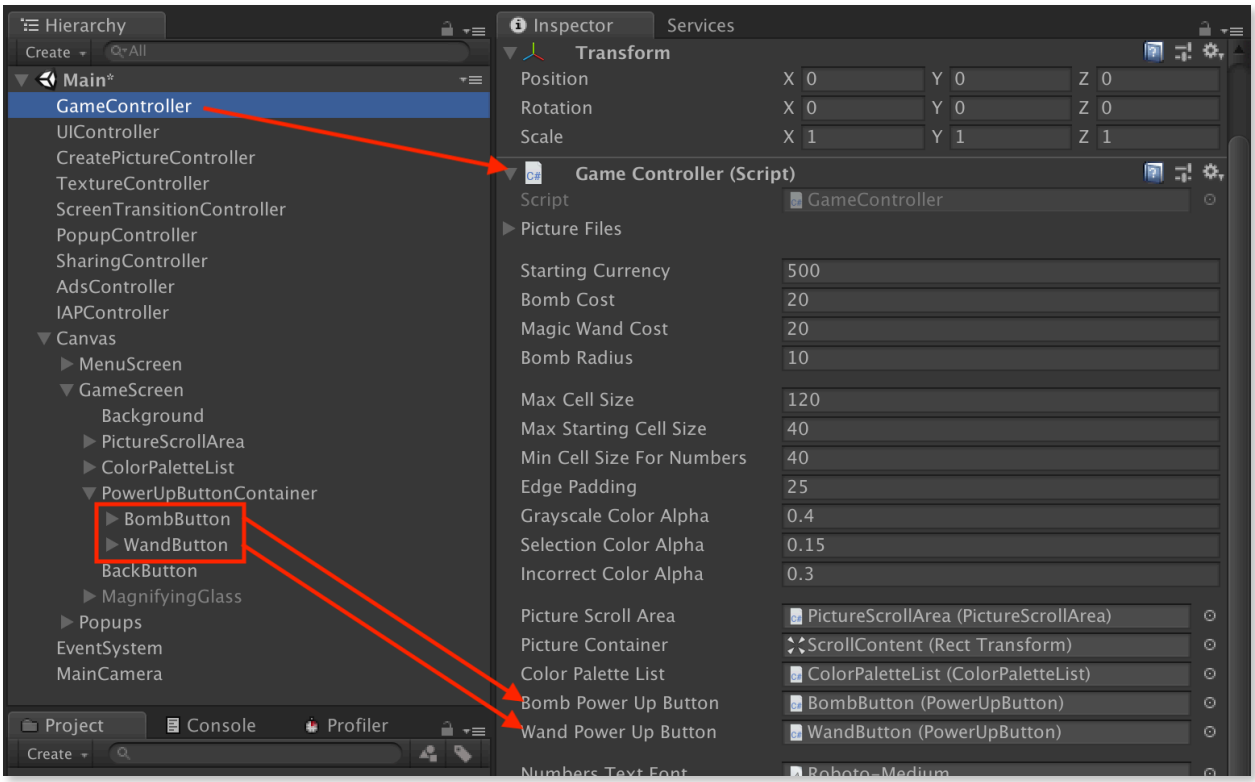
7.4 Color List Item

ColorListItem is the component that handles displaying a color in the color palette list. The ColorListItem prefab in the Prefabs folder is assigned to the ColorPaletteList component so the ColorPaletteList can instantiate copies of it to display in the list.

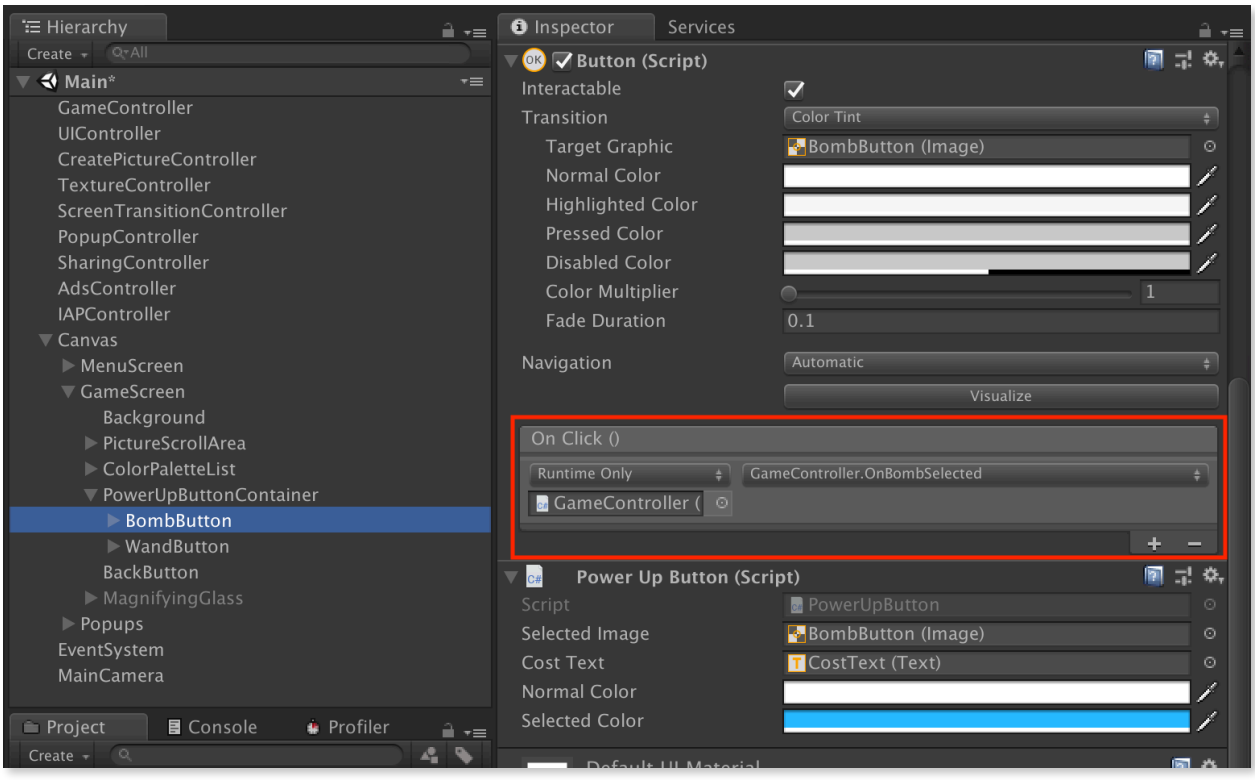


7.5 Power Up Buttons

The Bomb and Magic Wand power up buttons toggle when a power up is active in the GameController. The GameController sets the UI for the buttons when they are clicked, it also sets the amount of coins text on game start.



The On Click event in each of the power up buttons set the specific GameController method to call when they are clicked.



7.6 Magnifying Glass

The magnifying glass (Which appears when the user clicks and holds the picture) can be **enabled** / **disabled** on the GameController. Disabling will only make it so the magnifying glass no longer appears when you click and hold the picture, the user will still be able to then drag to color multiple cells at the same time.

The GameController handles setting up and positioning the magnifying glass. The **Magnifying Glass** field sets the Transform that must move with the mouse when dragging and the **Magnifying Glass Container** is the parent Transform for the picture objects that are created at run time. The Magnifying Glass Container is essentially the viewport of the magnifying glass.

The **Magnifying Glass Image** is simply the actual magnifying glass image that appears in top of whatever goes in the container.