# Game Documentation

To insure a quick response to any issues with the asset please send all support requests to the following e-mail address:

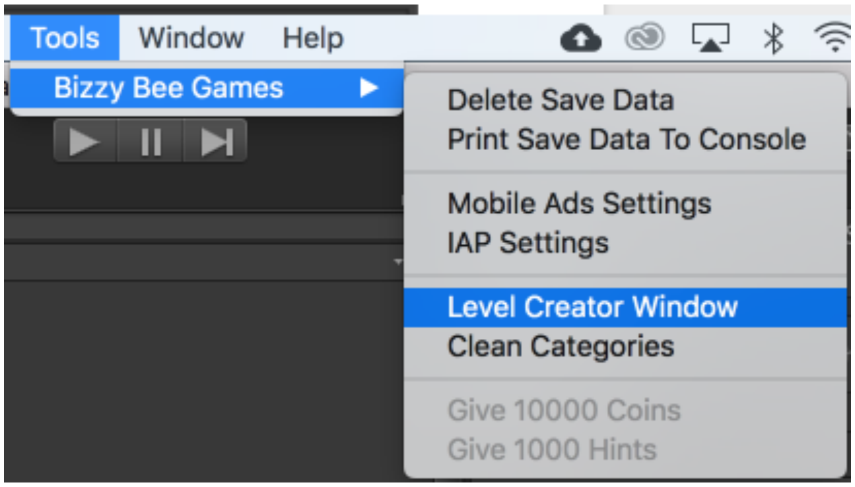
**support@bizzybeegames.com**

Please include the asset name and Unity version you are using. Thank you!

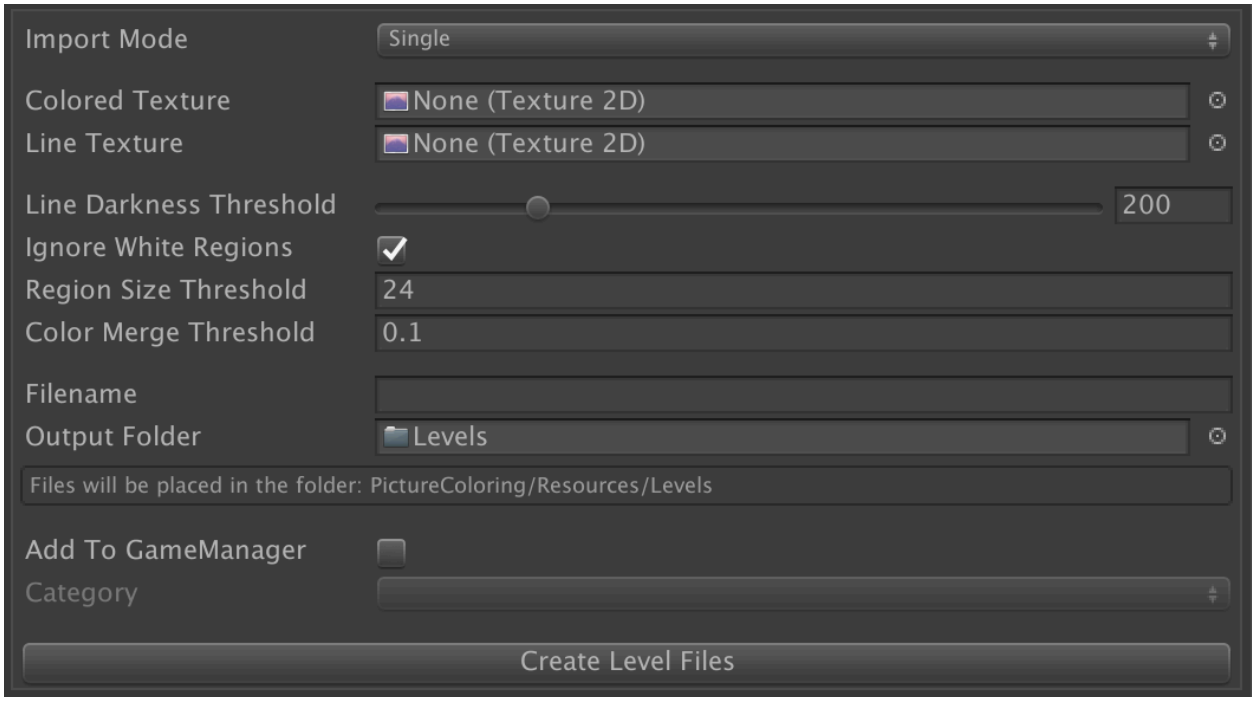
**Table of Contents:**

## Creating Level Data Files

The game works by using text and byte level file which are generated using the Level Creator editor window. To open the window select the menu item **Tools -> Bizzy Bee Games -> Level Creator Window**.



## Level Creator Window



| | |
|---|---|
| Import Mode | Single |
| Colored Texture | None (Texture 2D) |
| Line Texture | None (Texture 2D) |
| Line Darkness Threshold | 200 |
| Ignore White Regions | ✓ |
| Region Size Threshold | 24 |
| Color Merge Threshold | 0.1 |
| Filename | |
| Output Folder | Levels |

Files will be placed in the folder: PictureColoring/Resources/Levels

| | |
|---|---|
| Add To GameManager | |
| Category | |

**Import Mode**: Single lets you import one image from the project window. Batch lets you import multiple images at once by selecting the folder where the images are located (The folder does not need to be in you project, it can be anywhere on your computer)

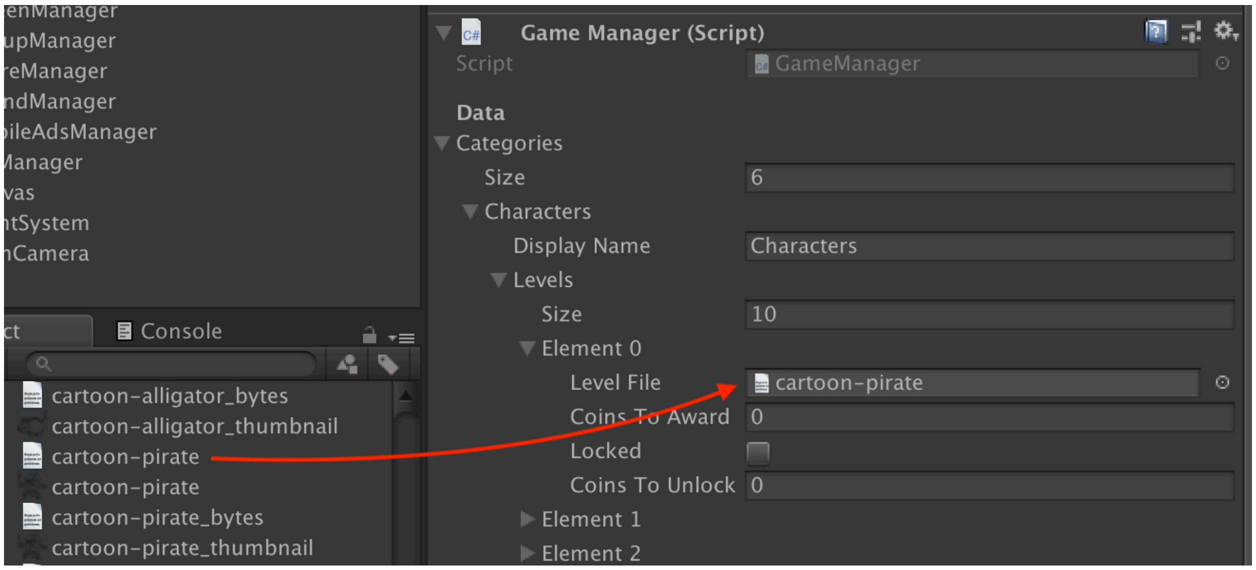
**Colored Texture**: The texture from the project folder which should be used as the colored in image. (See requirements below)
**Line Texture**: The texture from the project folder which should be used as the lines. (See requirements below)
**Line Darkness Threshold**: The “darkness” threshold for a pixel to be considered a line or not. For instance, setting to 255 means only solid black (#000000) pixels will be considered line pixels.
**Ignore White Regions**: If selected, any white areas on the colored texture will be ignored, ei. the player will not need to color them in to complete the level.
**Thumbnail Size**: The size of the thumbnail image to be used on the library screen.
**Region Size Threshold**: The minimum number of pixels in a region to be considered a region the player must color in.
**Color Merge Threshold**: The threshold for similar colors to be merged together (EX. Two shades of red which look the same to the naked eye). Set to 0 for no color merging.
**Set Power Of 2**: Sets the “Non Power Of 2” setting on the images for power of 2 compression.
**Filename**: The name to get the generated files. If blank then the name of the Colored Texture file will be used.
**Output Folder**: The folder to place the generated files into. The generated files must be placed in the Resources folder, if the selected folder is not in a Resources folder then a Resources folder will be created and the images will be placed into that folder instead.
**Add To GameManager**: If selected, the generated level file will be added as a level to the selected Category in the GameManager.
**Category**: The Category from the GameManagers Categories list to add the generated level to.

## Create Level Files

The **Create Level Files** button will start the process of taking the given images and converting the pixel information into vector information and exporting 2 files which are place in the Output Folder:

1. **level.txt** : Small text file with the id of the level and the path to the other level files in the Resources. This file is added to the level in the GameManager Categories list.
2. **level_bytes.bytes** : Contains color and vector information used to display the image in the game.

If you did not selected **Add To GameManager** then you will have to manually drag the **level.txt** from the Resources folder into a level on the GameManager:



enManager
upManager
eManager
ndManager
bileAdsManager
Manager
vas
tSystem
nCamera

ct | Console

cartoon-alligator_bytes
cartoon-alligator_thumbnail
cartoon-pirate
cartoon-pirate
cartoon-pirate_bytes
cartoon-pirate_thumbnail

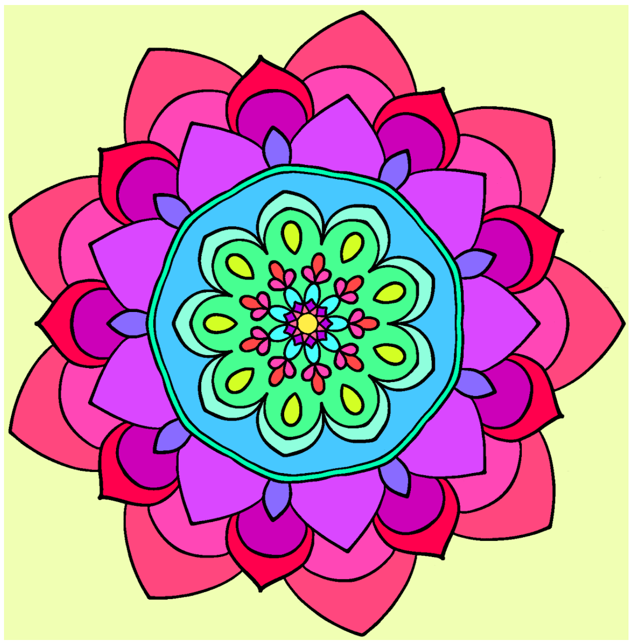
Game Manager (Script)
Script: GameManager
Data
Categories
Size: 6
Characters
Display Name: Characters
Levels
Size: 10
Element 0
Level File: cartoon-pirate
Coins To Award: 0
Locked:
Coins To Unlock: 0
Element 1
Element 2

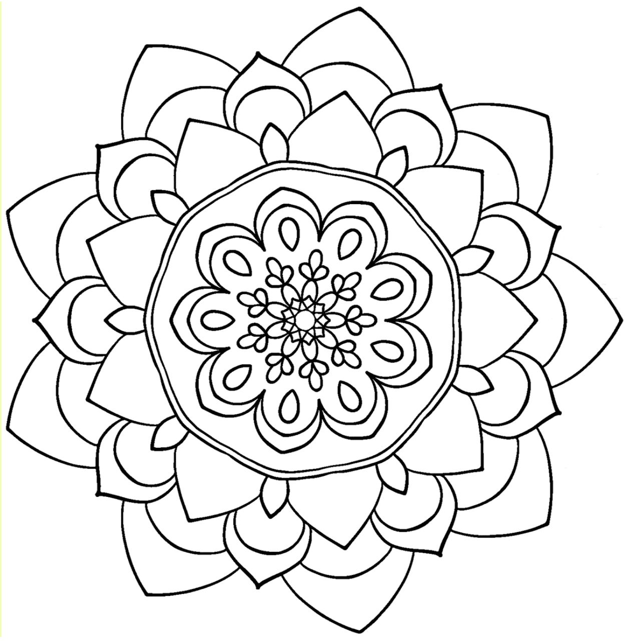
## Image Requirements

Requirements for the **Colored Texture**:
1. Needs to be the exact same size as the Line Texture.
2. Needs to have Read/Write enabled. (Not necessary for Batch mode)
3. Fully colored in, white regions can be set to be ignored (Not needed to be colored in)
4. No alpha, any regions with alpha pixels will default to ignored and appear as white in the game.

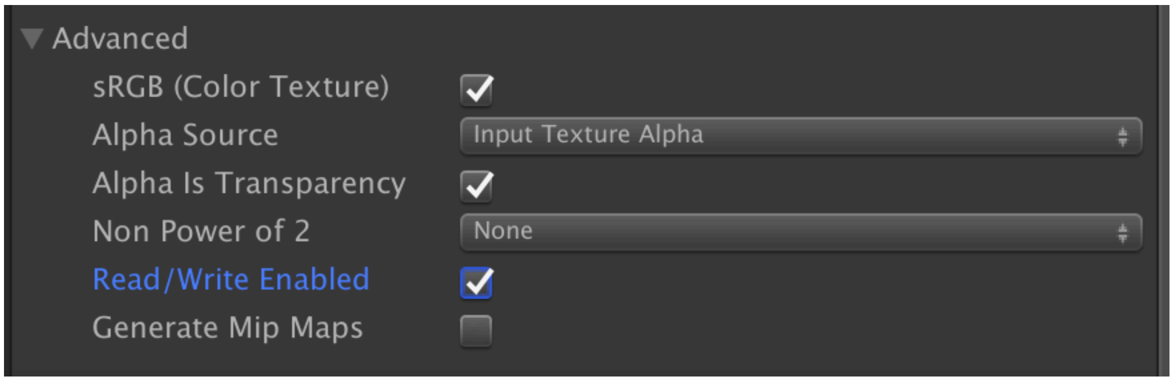
Requirements for the **Line Texture**:
1. Needs to be the exact same size as the Colored Texture.
2. Needs to have Read/Write enabled. (Not necessary for Batch mode)
3. Needs solid black lines with either white background. NO COLOR.

Example **Colored Texture** and **Line Texture**:





To set the textures as Read/Write enabled, select the image in your Project window and then in the Inspector, expand **Advanced** and select **Read/Write Enabled**. Then click apply at the bottom of the Inspector:

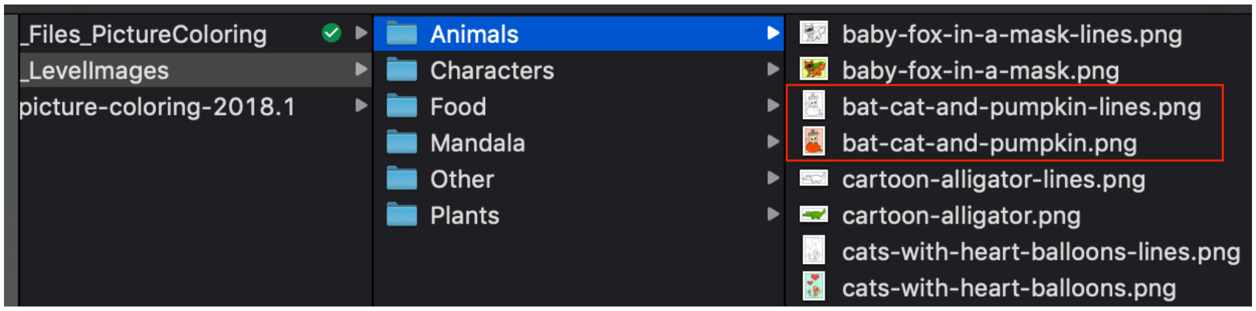


Advanced
sRGB (Color Texture) ✓
Alpha Source: Input Texture Alpha
Alpha Is Transparency ✓
Non Power of 2: None
Read/Write Enabled ✓
Generate Mip Maps

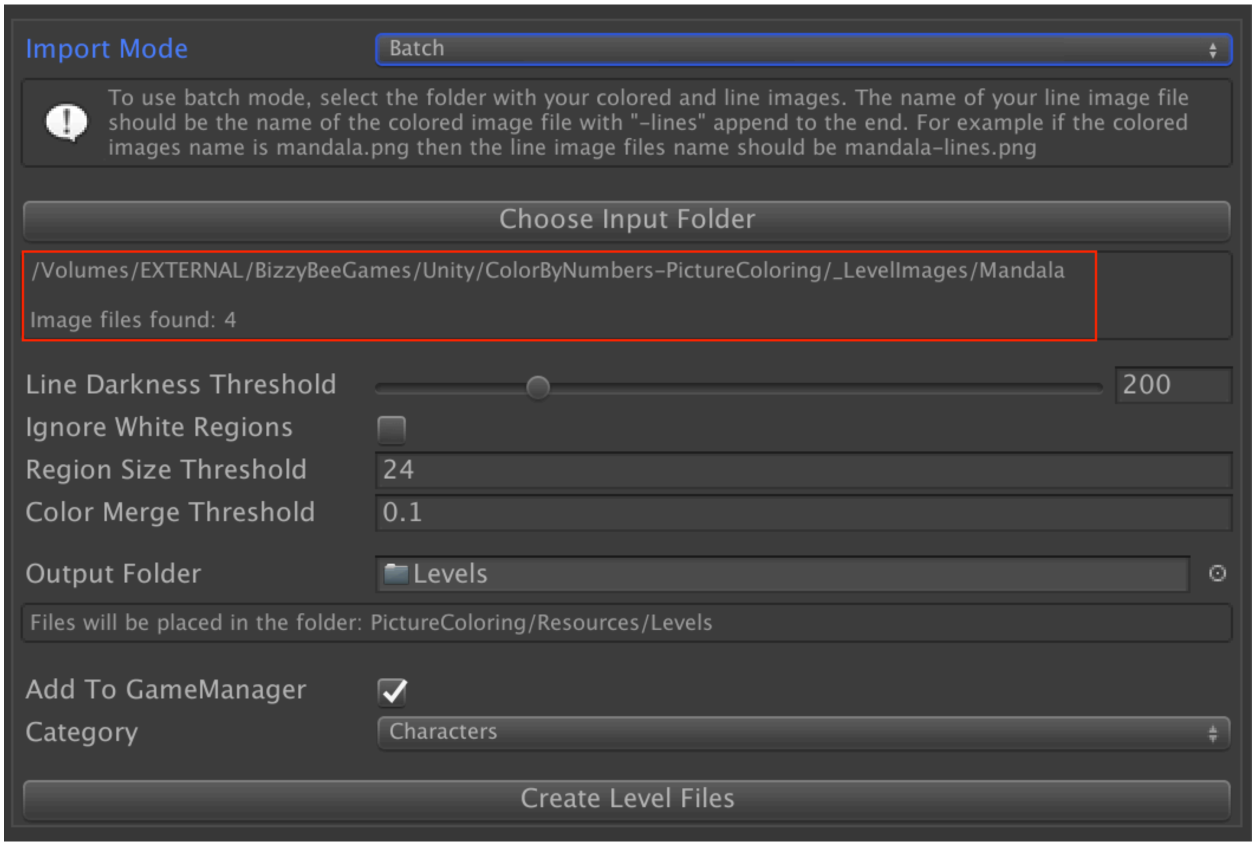
## Batch Mode

Setting the window to Batch import mode will allow you to select a folder from your computer and have all PNG images in that folder imported one after each other.

The line image will need to be the same name as the colored image with “-lines” append to the end. For example, if you have a colored image called “mountains.png” then the line image will need to be named “mountains-lines.png”.



When you select a folder the window will tell you how many level images it found, it will also print an error message to the Unity Console if it couldn’t find the lines image for any of the images.



Import Mode: Batch

To use batch mode, select the folder with your colored and line images. The name of your line image file should be the name of the colored image file with "-lines" append to the end. For example if the colored images name is mandala.png then the line image files name should be mandala-lines.png

/Volumes/EXTERNAL/BizzyBeeGames/Unity/ColorByNumbers-PictureColoring/_LevelImages/Mandala

Image files found: 4

Line Darkness Threshold: 200
Ignore White Regions: ☐
Region Size Threshold: 24
Color Merge Threshold: 0.1
Output Folder: Levels

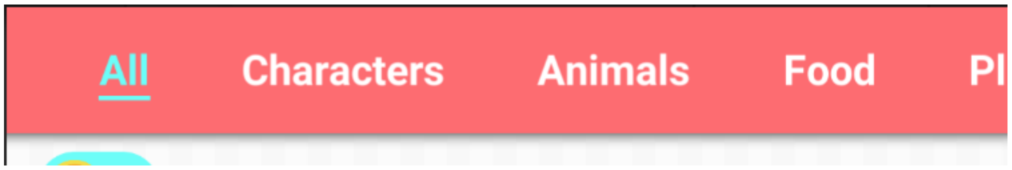
Files will be placed in the folder: PictureColoring/Resources/Levels

Add To GameManager: ✓
Category: Characters

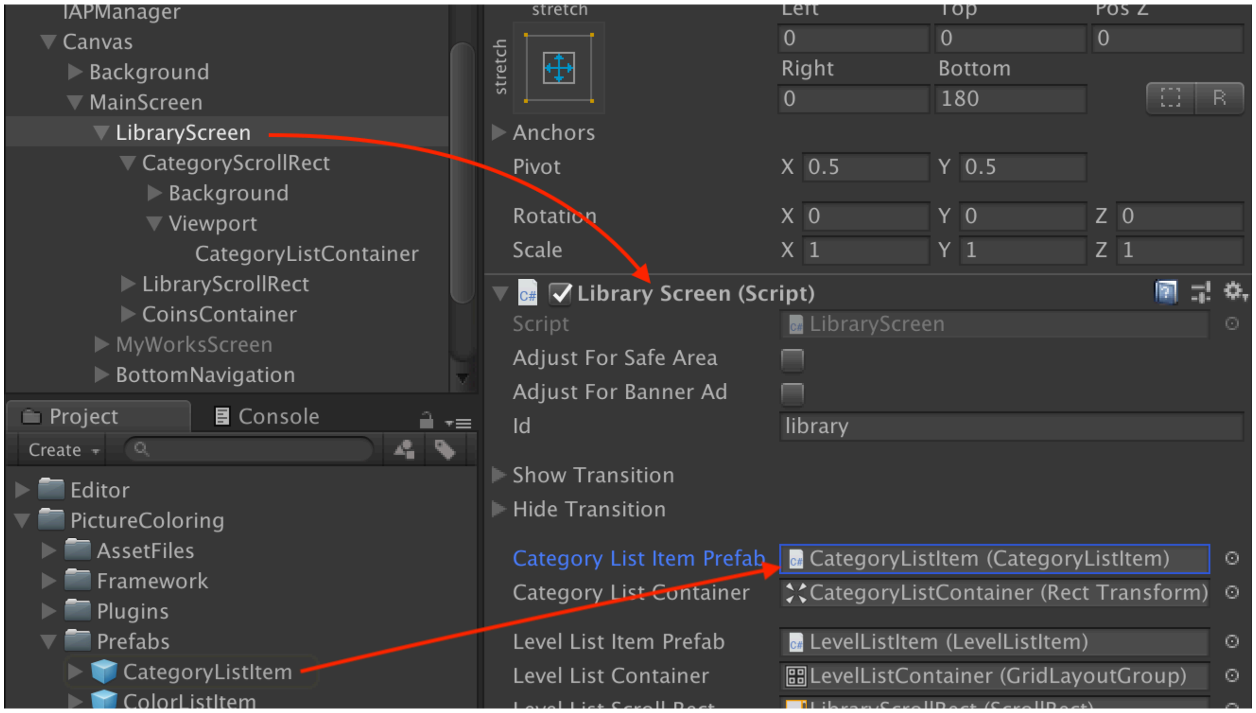
## Project

This section will show you where UI elements that are changed / instantiated by scripts are located and how to edit them.

## Category List Items



The category list items are generated at run time by the **LibraryScreen** component which is attached to the LibraryScreen GameObject. The LibraryScreen will instantiate a new **Category List Item Prefab** for each Category in the GameManager. The Category List Item Prefab that is being used in the asset is located at **PictureColoring/Prefabs/CategoryListItem**

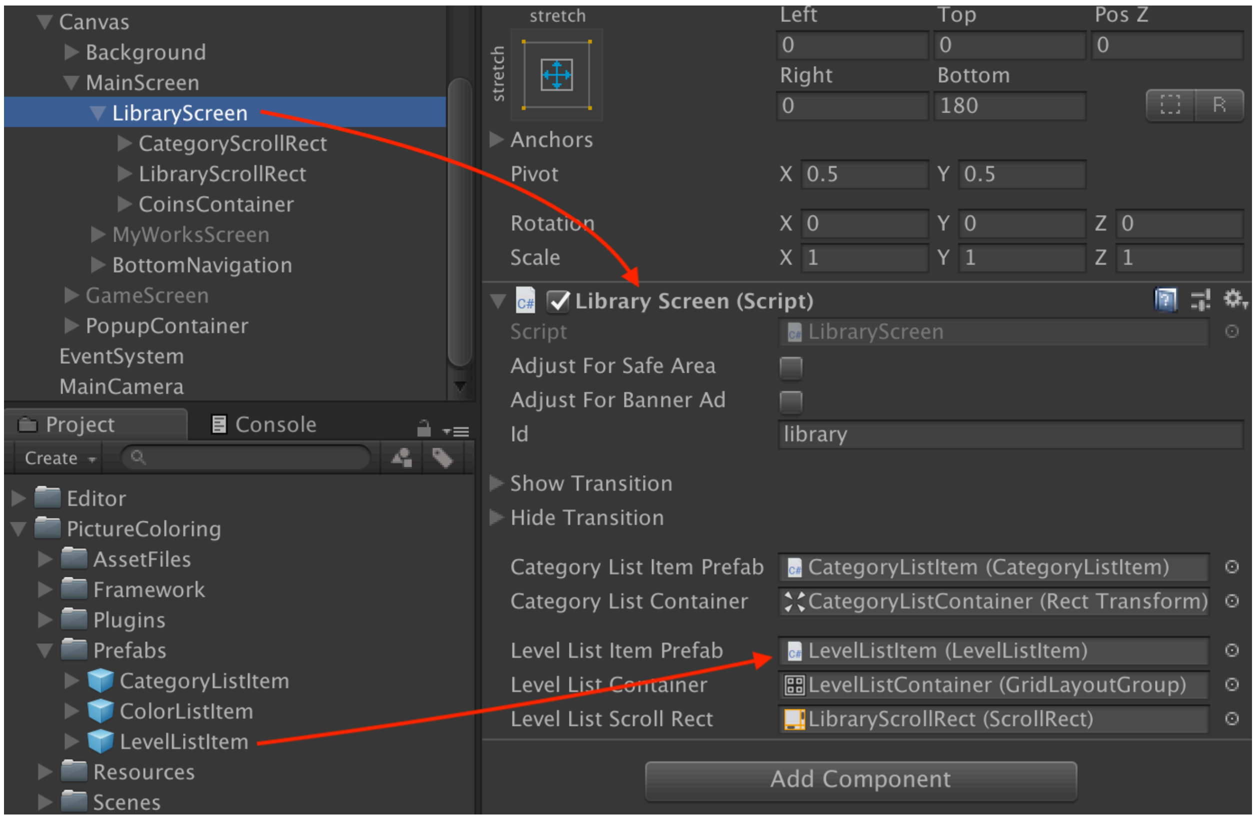


```
IAPManager
▼ Canvas
   ▶ Background
   ▼ MainScreen
      ▼ LibraryScreen
         ▼ CategoryScrollRect
            ▶ Background
            ▼ Viewport
               CategoryListContainer
         ▶ LibraryScrollRect
         ▶ CoinsContainer
      ▶ MyWorksScreen
      ▶ BottomNavigation

Project    Console
Create

▶ Editor
▼ PictureColoring
   ▶ AssetFiles
   ▶ Framework
   ▶ Plugins
   ▼ Prefabs
      ▶ CategoryListItem
      ▶ ColorListItem
```

```
stretch        Left      Top       Pos Z
               0         0         0
               Right     Bottom
               0         180
▶ Anchors
  Pivot        X 0.5     Y 0.5
  Rotation     X 0       Y 0       Z 0
  Scale        X 1       Y 1       Z 1

▼ ✓ Library Screen (Script)
  Script                      LibraryScreen
  Adjust For Safe Area        ☐
  Adjust For Banner Ad        ☐
  Id                          library
▶ Show Transition
▶ Hide Transition
  Category List Item Prefab   CategoryListItem (CategoryListItem)
  Category List Container     CategoryListContainer (Rect Transform)
  Level List Item Prefab      LevelListItem (LevelListItem)
  Level List Container        LevelListContainer (GridLayoutGroup)
```

## Library Level List Items

The level list items are generated at run time by the **LibraryScreen** component which is attached to the LibraryScreen GameObject. The Level List Item Prefab that is being used in the asset is located at **PictureColoring/Prefabs/LevelListItem:**



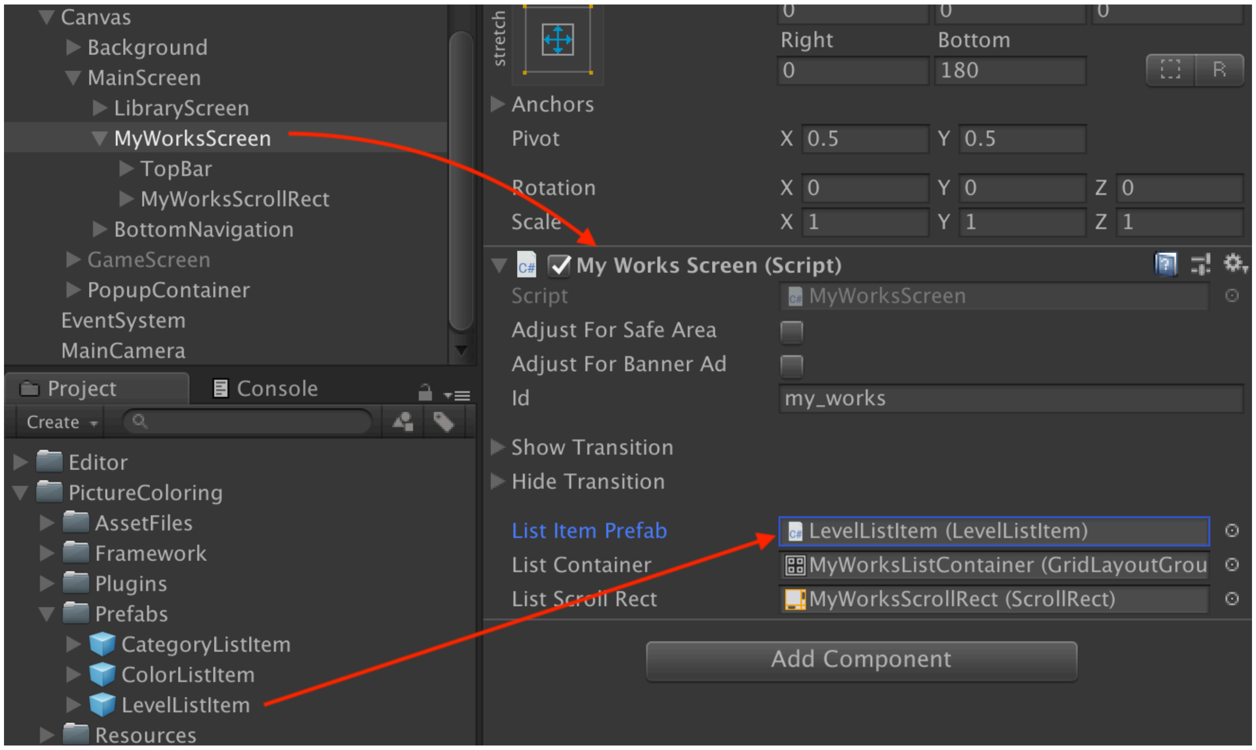
| | | | |
|---|---|---|---|
| | Left | Top | Pos Z |
| | 0 | 0 | 0 |
| | Right | Bottom | |
| | 0 | 180 | |
| Anchors | | | |
| Pivot | X 0.5 | Y 0.5 | |
| Rotation | X 0 | Y 0 | Z 0 |
| Scale | X 1 | Y 1 | Z 1 |

**Library Screen (Script)**

| | |
|---|---|
| Script | LibraryScreen |
| Adjust For Safe Area | ☐ |
| Adjust For Banner Ad | ☐ |
| Id | library |
| Show Transition | |
| Hide Transition | |
| Category List Item Prefab | CategoryListItem (CategoryListItem) |
| Category List Container | CategoryListContainer (Rect Transform) |
| Level List Item Prefab | LevelListItem (LevelListItem) |
| Level List Container | LevelListContainer (GridLayoutGroup) |
| Level List Scroll Rect | LibraryScrollRect (ScrollRect) |

## My Works List Items

The level list items on the My Works screen are generated at rum time by the **MyWorksScreen** component which is attached to the MyWorksScreen GameObject. The **List Item Prefab** is used to instantiate copies of the level list items and uses the same prefab as the LibraryScreen which is located at **PictureColoring/Prefabs/LevelListItem:**

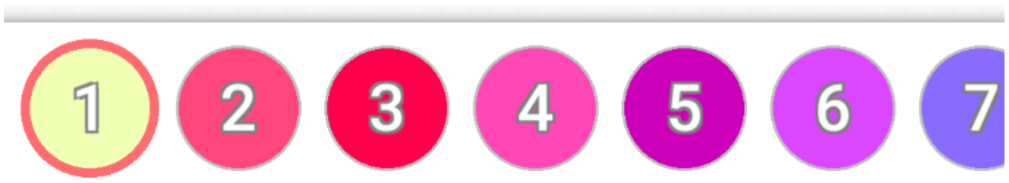


| | | | | | |
|---|---|---|---|---|---|
| Right | 0 | Bottom | 180 | | |
| Pivot | X 0.5 | Y 0.5 | | | |
| Rotation | X 0 | Y 0 | Z 0 | | |
| Scale | X 1 | Y 1 | Z 1 | | |

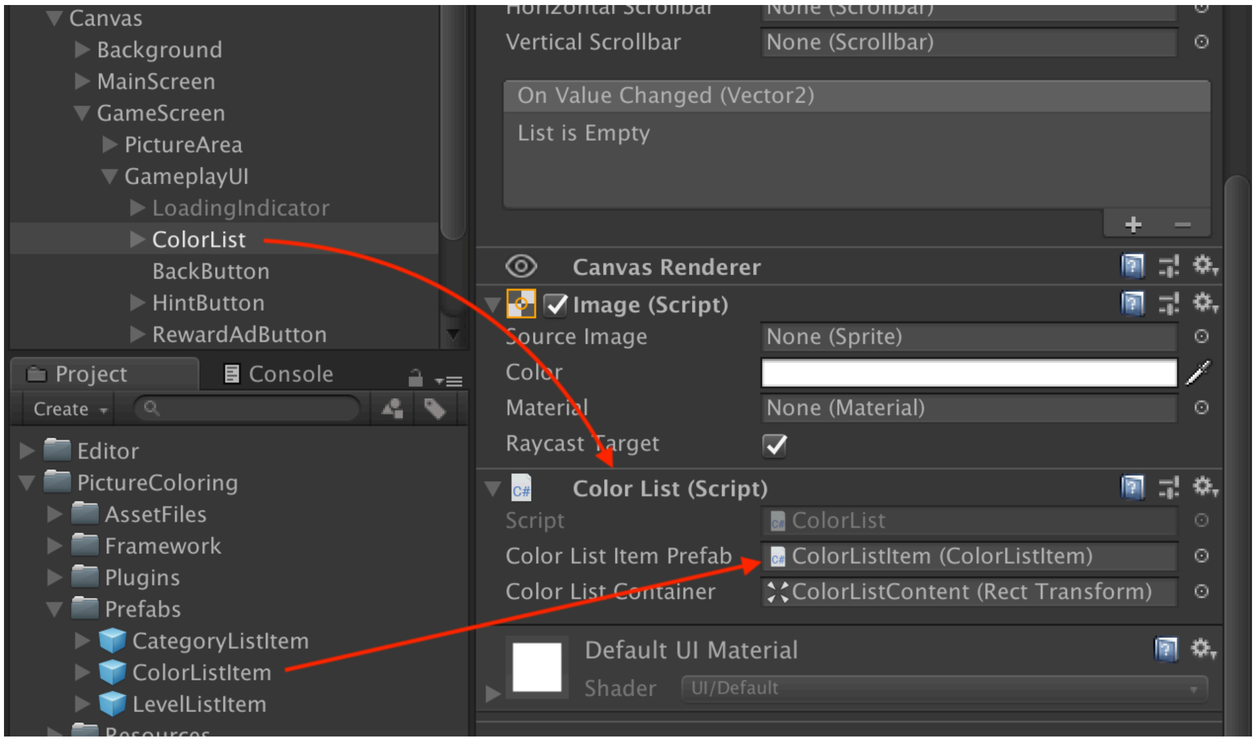
**My Works Screen (Script)**

| | |
|---|---|
| Script | MyWorksScreen |
| Adjust For Safe Area | |
| Adjust For Banner Ad | |
| Id | my_works |
| List Item Prefab | LevelListItem (LevelListItem) |
| List Container | MyWorksListContainer (GridLayoutGrou |
| List Scroll Rect | MyWorksScrollRect (ScrollRect) |

## Color List Items

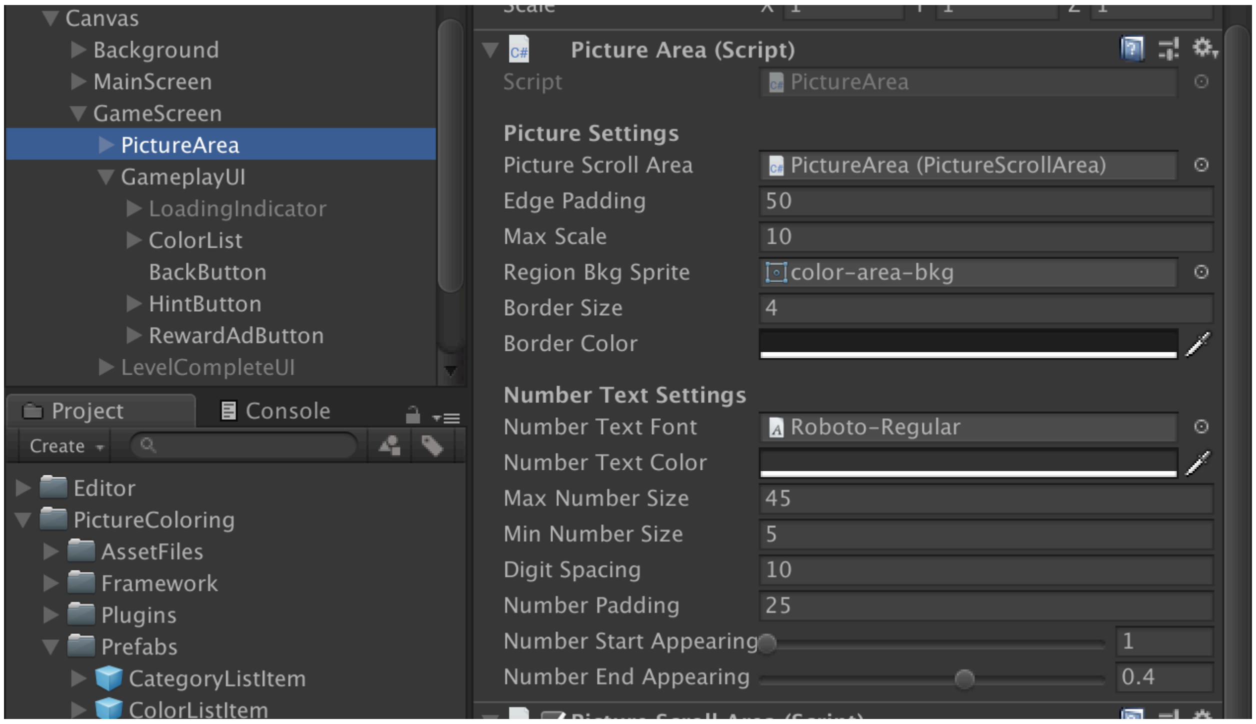


The color items on the GameScreen are generated when a level starts by the **ColorList** component attached to the ColorList GameObject. It uses the **Color List Item Prefab** to generate a new color list item for each color in the level. The Color List Item prefab that is being used in the asset is located at **PictureColoring/Prefabs/ColorListItem:**



Horizontal Scrollbar None (Scrollbar)
Vertical Scrollbar None (Scrollbar)
Source Image None (Sprite)
Color
Material None (Material)
Raycast Target ✓
Script ColorList
Color List Item Prefab ColorListItem (ColorListItem)
Color List Container ColorListContent (Rect Transform)
Shader UI/Default

## Picture Settings



```
▼ Canvas
  ▶ Background
  ▶ MainScreen
  ▼ GameScreen
    ▶ PictureArea
    ▼ GameplayUI
      ▶ LoadingIndicator
      ▶ ColorList
        BackButton
      ▶ HintButton
      ▶ RewardAdButton
    ▶ LevelCompleteUI
```

```
Project   Console
Create
▶ Editor
▼ PictureColoring
  ▶ AssetFiles
  ▶ Framework
  ▶ Plugins
  ▼ Prefabs
    ▶ CategoryListItem
    ▶ ColorListItem
```

**Picture Area (Script)**

| | |
|---|---|
| Script | PictureArea |
| **Picture Settings** | |
| Picture Scroll Area | PictureArea (PictureScrollArea) |
| Edge Padding | 50 |
| Max Scale | 10 |
| Region Bkg Sprite | color-area-bkg |
| Border Size | 4 |
| Border Color | |
| **Number Text Settings** | |
| Number Text Font | Roboto-Regular |
| Number Text Color | |
| Max Number Size | 45 |
| Min Number Size | 5 |
| Digit Spacing | 10 |
| Number Padding | 25 |
| Number Start Appearing | 1 |
| Number End Appearing | 0.4 |

The appearance of the picture on the GameScreen can be changed using the settings on the **PictureArea** component which is attached to the PictureArea GameObject.

**Edge Padding:** The amount of space between the image and the edge of the PictureArea when the image is zoomed all the way out (At it's min scale / the start of the level).
**Max Scale:** The maximum amount that the image can be zoomed in by. In the above screenshot the image can be zoomed in to 10x is size.
**Region Bkg Sprite:** This is the sprite to use for the Image that appears when a color is selected. The sprite that comes with the asset is a repeating checkered pattern.
**Border Size:** The thickness of the border that appears around the image.
**Border Color:** The color of the border that appears around the image.

**Number Text Font:** The font to use for the numbers that appear on the image.
**Number Text Color:** The color to use for the numbers that appear on the image.
**Max Number Size:** The maximum size in pixels that a number can be on the image, this is to control large regions from having super big numbers.
**Min Number Size:** The minimum size in pixels the a number can be on the image.
**Digit Spacing**: The amount of spacing in pixels between digits in a number if the number is larger than 9. (Ex. The number 12, the amount of space between the 1 and 2).
**Number Padding**: The amount of padding to apply to a number in it's region so it's not super close to the lines.
**Number Start Appearing**: Controls when the numbers start appearing relative to the zoom/ scale of the image.
**Number End Appearing**: Controls when the smallest numbers appear on the image, settings to 0 means the image will have to be zoomed in all the way to see the smallest numbers.

# Sounds

Sounds in the game are controlled using the SoundManager. On the SoundManager's inspector you will find a number of Sounds Infos already created and used in the game.

## Sound Info fields

**Id** - The Id used to play the sound in the game.
**Audio Clip** - The sound file from your project.
**Type** - The type of sound (Sound Effect or Music), this is used to turn on/off all sounds of a particular type.
**Play And Loop On Start** - If selected the Audio Clip will play when the game starts and will loop forever unless it is stopped.
**Clip Volume** - Sets the volume of the sound when it is played.

## Playing Sounds

Sounds can be played by calling the **Play** method on the SoundManager like so:

```
SoundManager.Instance.Play(string id);
```

You can easily play a sound when a Button is clicked by adding the **ButtonSound** component to a GameObject with a **Button** component. The ButtonSound will play the sound with the specified Id every time the button is clicked.

## Share Plugin

**iOS Setup:**

No extra setup is required for iOS

**Android Setup:**

To enable sharing on Android, follow the steps below:

1. Download the **AndroidSharePlugin.unitypackage** here: https://www.bizzybeegames.com/plugins/AndroidSharePlugin.unitypackage

2. Import the unity package into your project.

3. The plugin will be imported to **Assets/Plugins/Android/SharePlugin**. The SharePlugin cannot be moved to another Plugins folder, it must reside in the root Plugins folder: Assets/Plugins/Android or the plugin will not work.

4. Open the **AndroidManifest.xml** located in Assets/Plugins/Android/SharePlugin and change YOUR.PACKAGE.NAME (on line 4 and line 9) to the package name in your Player Settings.

5. Download and import the google play services resolver using the link: https://github.com/googlesamples/unity-jar-resolver and selecting the **play-services-resolver-x.x.x.unitypackage** file and clicking the Download button.

6. If the play services resolver does not automatically run when you import it into the project select the menu item **Assets -> Google Play Resolver -> Android Resolver -> Resolve**