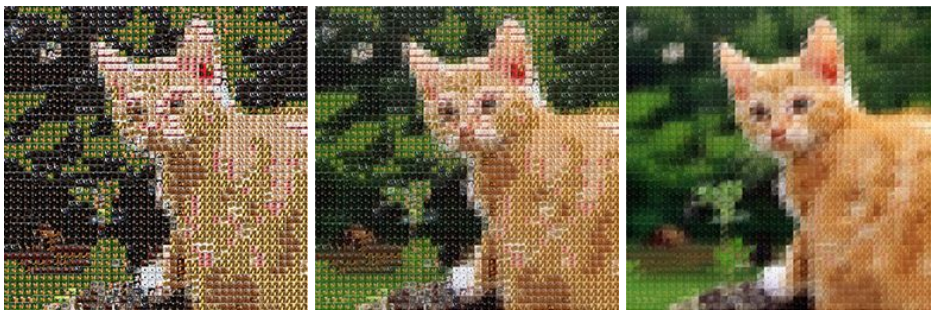
**Photo Mosaic (v1.0)**

Photo Mosaic is a library for you to real time create photographic mosaic. Because photographic mosaic is an heavy process, to make it works in real time we have to use some tricky way to cut corner. We get two methods for you to do this effect. One method (PhotoMosaic) is using CPU to generate a matrix of square images, one image one GameObject. Another method (PhotoMosaicImageEffect) is working like image effect, using GPU to real time render pre-processed data to mosaic effect.

**Cheating Tricks**

As everything has to be done in realtime, there is some limit you have to load. For example, it's not recommanded to load too many images. I think the suggested maximum won't be <100 for GameObject-based method, and 256 for Image-Effect-based method. It's sure that with only a hundred images, the final result may look quite repeative, that's why i offer several "cheating" to fix this problem.

ColorCheat (0,0.4,0.8)



PhotoCheat (0,0.4,0.8)

Also there is some randomize features you can use to make it less repeative.

There are two steps of caching in my library: Texture Cache and Index Map Cache. Texture cache is about render every loaded images into one RenderTexture.  Index Map Cache is creating a map pairing which image should a color pixel use. And you have to do it in an order (1. Texture Cache, 2. Index Map Cache).

| | Rare | Texture Cache | Index Map Cache |
|---|---|---|---|
| GameObject-based | work | work<br>better performace | work<br>but no benefit |
| Image-Effect-based method | not work | not work | work |

**PhotoMosaicDataSet**

This is a ScriptableObject but you are not supposed to edit it by yourself. You can use Tools->PhotoMosaic Wizard to load images and create a PhotoMosaicDataSet object. Or using script to create one, and call addTexture() to load image.

```
PhotoMosaicDataSet(bool useCache = false)
        if useCache is true, everything will be render to a RenderTexture first.
public TextureData addTexture(Texture2D texture)
        add a texture to the dataset. texture must be readable.
public void makeCache()
        render everything to a RenderTexture
public void makeIndexMap(int sizePow = 3)
        cache all color comparation data a Texture
public IEnumerator makeIndexMapAsync(int sizePow = 3,Action onComplete = null)
        same as makeIndexMap() by async
        you have to call it with StartCoroutine()
public bool isReadyForImageEffect()
        To check if the data set is ready to be used by PhotoMosaicImageEffect
        if not you can call makeCache() and makeIndexMap() to enable it
```

**PhotoMosaic**

You should add this component to an empty GameObject. Each grid will be created as child GameObject. There will be lots of objects, but it’s more flexible for different ideas.
Please remind that the perforamce depends on the number of images source and the number of grids. The higher the setting is, the slower the app will be. So it’s important to know how much your hardware can handle.

```
public SpawnOrder spawnOrder;
        if you set to not spawn every pieces at once, this is how it spawn piece by piece
public Material material;
        the material reference for each image, the material should use PhotoMosaicShader
```

```
public float spawnPerSecond = 1000;
        spawn per second based on game timer
public Vector2 gridSize = Vector2.one;
        the size of each piece
public int columns=10;
        horizontal number of pieces
public int rows=10;
        vertical number of pieces
public float colorCheat = .5f;
        the color cheating
public float photoCheat = .5f;
        the photo cheating
public float tryAvoidRepeat = .1f;
        for each image get used, it will add a value to avoid it get used again.
        setting this value higher will make things less repeative,
        but it may use images that is not suitable for that pixel
public PhotoMosaicDataSet defaultDataSet;
        so that you can drag in a pre-generate dataset
public bool usePool;
        this is for memory management
        if true, the script will keep all pieces that is created but not using
        and then re-enable them when need
        if false, it will just release everything when the pieces are not need anymore
public void generate (Texture2D target, PhotoMosaicDataSet dataSet=null)
        call an IEnumerator to generate all pieces
        if dataSet is null, it will use defaultDataSet;
public event SpawnEventHandler(GameObject piece, PhotoMosaicDataSet.TextureData
data) onSpawn;
        it's for you to get feedback when a piece is spawned,
        so that you can add your own tween or other effect.
```

**PhotoMosaicImageEffect**

The workload is static, the number of images and the number of grids won't influence the performace. But this method is less flexible. First, you must do cache before using this. Second, the resolution is fixed, and quite depends on the hardware. The library will cache images on RenderTexture as in a 16x16 table. If your hardware can only handle 2048 size texture, the resolution for each images will be 128.

```
public PhotoMosaicDataSet dataSet;
        there will be no effect if there is no dataSet ready
public Shader shader;
        link this to PhotoMosaicEffectShader
public float colorCheat = .5f;
        the color cheating
public float photoCheat = .5f;
        the photo cheating
```

```
public float scale = 24;
```
the verticle number of pieces
```
public PhotoMosaicDataSet.Randomness randomness;
```
the index map store 4 most suitable images for a color
this parameter is to make the image effect randomly use the 2rd-4th images

**Misc**

- For GameObject effect, you need PhotoMosaicShader material to make it work prefectly. It has function to blend the pixel texture and the overall texture.
- All texture has to be read/write enable, so that my script can analyze the color. Select the texture file, change “Texture Type” to “Advanced”, set “Read/Write Enabled” to “yes”
- Example folder is just for example, concerning the file size, you can delete it if you want. (I tried to pack it as another untiypackage but asset store doesn’t allow me to do that)

**Future**

If there are people buying this library, i will keep enrich this library. Please make a comment in asset store or forum if you think there is some feature you think it’s good to include.

1. non square ratio

if you read my source, you can see it was planned to have non square ratio. for now, i want to make sure it works well with square size. but i can keep working on this feature.

2. multi cache texture with dynamic tiling

Now everything is quite hardcode. There is only one RenderTexture, the size is not customziable, the tiling is always 16x16. But I may be able to change it. For example, if we use multi RenderTexture, we can have high resolustion image in weak device.

3. alpha channel

The coding part shouldn’t be hard. The challenging part is how to look good. If you want, i can take a look about it.