Table of contents

What is Scratch Card?

This is an easy-to-use asset, which allows you to create scratch card objects. All you need is to add prefab to the scene:

- **ScratchCard.prefab**

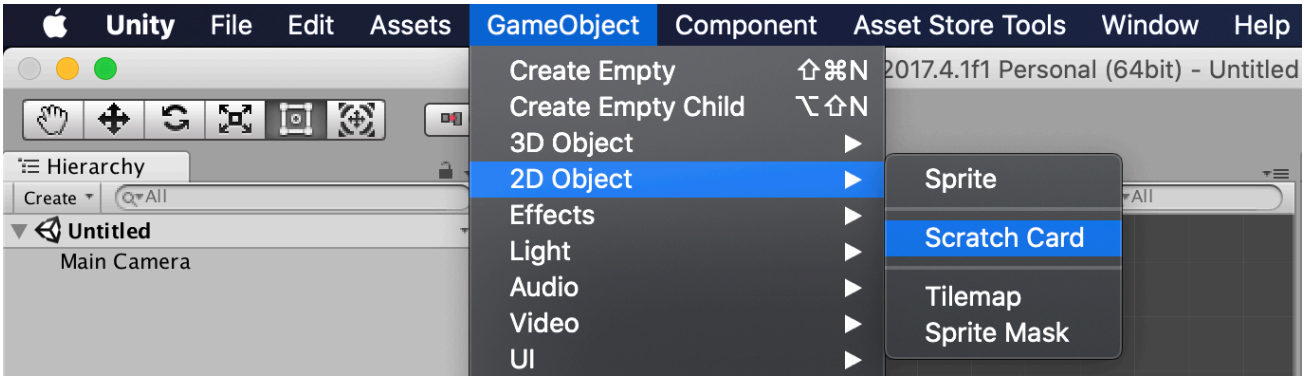
choose Main Camera and set Sprite - your scratch card sprite.

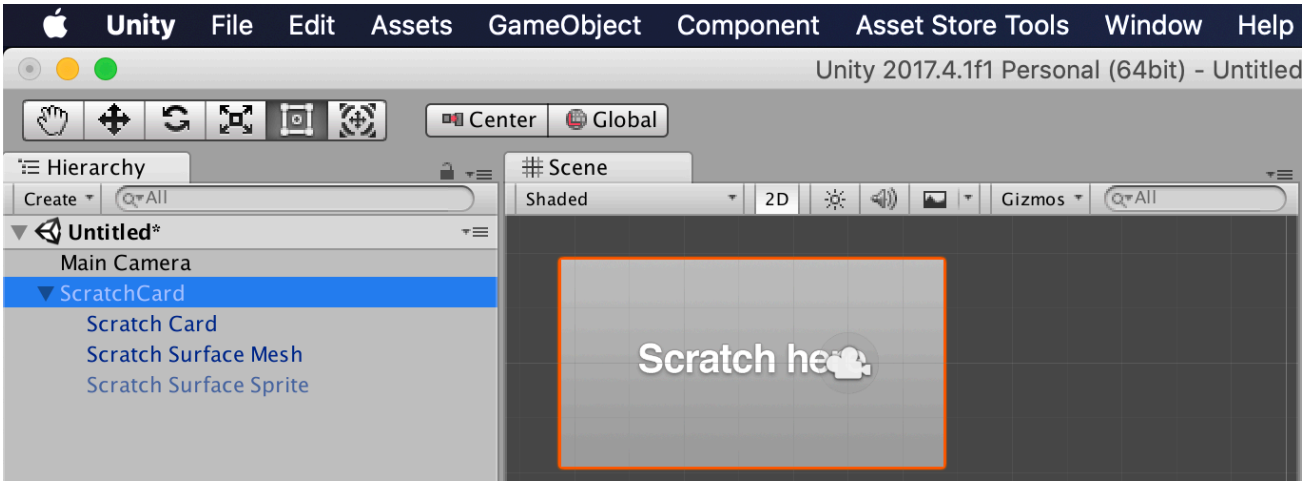
This works well on Personal and Pro Unity, suitable for all platforms.

Quick Start

As mentioned above, to create scratch object, add prefab «**ScratchCard**» from «**Assets/ScratchCard/Prefabs/ScratchCard.prefab**», set Camera, Sprite for scratching, Brush Texture and Scratch Surface will work! That's all!

In another way, prefab «**ScratchCard**» can be created using Unity menu: “GameObject -> 2D Object -> Scratch Card”:





How does it works?

ScratchCardManager component manages **ScratchCard** and **EraseProgress** components.

ScratchCard component creates RenderTexture and paint on it with user input.

EraseProgress component calculates average alpha color of ScratchCard RenderTexture to its own progress RenderTexture with size 1x1 pixel. Color of progress RenderTexture pixel - it is scratch progress value.

API Help

Content

ScratchCardManager

ScratchCardManager component creates and configures ScratchCard. Note that for changing ScratchCard parameters in runtime, user should refer to ScratchCard component.

ScratchCardManager script has parameters:

- **Main Camera** - Main Camera of scene;
- **RenderType** - render type of scratch card: MeshRenderer, SpriteRenderer or CanvasRenderer;
- **ScratchSurfaceSprite** - sprite for scratching;
- **ScratchSurfaceSpriteHasAlpha** - whether sprite for scratching has alpha-channel;
- **MaskProgressCutOffValue** - value for comparing sampled RenderTexture alpha values with sampled Source Texture alpha values, using by MaskProgressCutOff shader only with ScratchSurfaceSprite without alpha-channel;
- **EraseTexture** (BrushTexture) - texture for erasing (erasing brush);
- **EraseTextureScale** (BrushTextureScale) - brush scale;
- **InputEnabled** - whether input is enabled;
- **Card** - reference to **ScratchCard** component;
- **Progress** - reference to **EraseProgress** component;
- **MeshCard** - reference to GameObject with MeshRenderer;
- **SpriteCard** - reference to GameObject with SpriteRenderer;
- **ImageCard** - reference to GameObject with Image;

After creating instance of **ScratchCard.prefab**, you need set **Main Camera**, **RenderType**(optional), **ScratchSurfaceSprite** (Sprite) and **EraseTexture** (Brush Texture)**.**

ScratchCardManager script has methods:

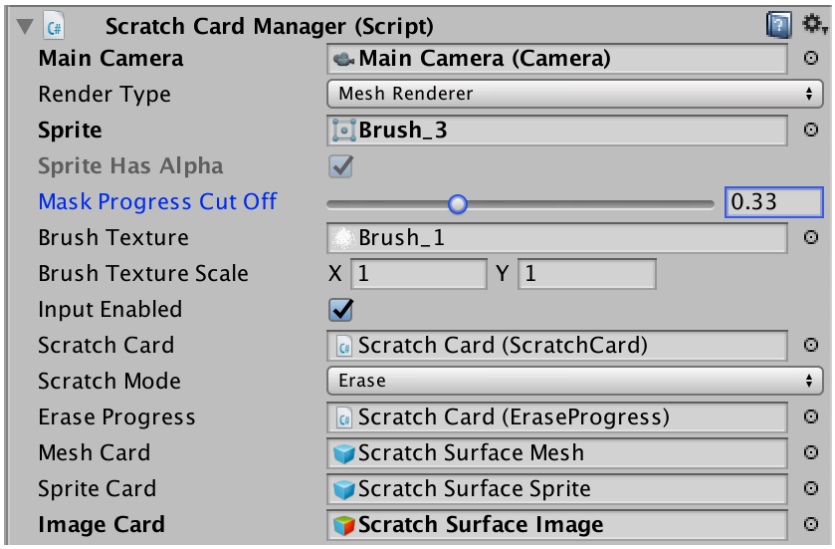
`void Awake()` - setting up the scratch card;

`public void SetEraseTexture(Texture texture)` - setting up the erase texture(brush) for the scratch card;

`public void ResetScratchCard()` - reset scratch card.

Note that if you use Canvas, ScratchCardManager.ImageCard object must be child of Canvas.

If you want to use ScratchCard images with transparent areas, for getting better result of progress calculations setup material, using Cut Off field. Cut Off field value is using to compare sampled RenderTexture alpha values with sampled Source Texture alpha values.



ScratchCard

ScratchCard component creates and configures RenderTexture then draws the quads in RenderTexture. You can use SpriteRenderer, MeshRenderer or Image for scratch effect.

ScratchCard script has such parameters:

- **Main Camera** - Main Camera of scene;
- **Surface** - transform of Surface object, which can contain Render Component;
- **RenderTextureQuality** - quality(size) of RenderTexture texture: High, Medium, Low;
- **Eraser** - Material of Eraser(brush);
- **Progress** - Material of Progress;
- **Scratch Surface** - Material of Scratch Surface.
- **RenderTexture** - RenderTexture for scratching;
- **BrushScale** - scale of brush;
- **InputEnabled** - whether input is enabled.

ScratchCard script has properties:

`public ScratchMode Mode` - scratch card mode: erase or restore;

`public bool IsScratching` - returns if user is tried scratch surface currently (input processing);

`public bool IsScratched` - returns if user is scratched surface currently (input processed and some part of texture was scratched).

ScratchCard script has methods:

`public void FillInstantly()` - fills RenderTexture with white color instantly (100% scratched surface);

`public void ClearInstantly()` - fills RenderTexture with clear color instantly (0% scratched surface);

`public void Clear()` - fills scratch card with clear color in the next Update;

~~`public void Reset()` - re-creates RenderTexture and clears it in the next Update.~~ This method was removed from version 1.8, please, use `ResetRenderTexture()` instead;

`public void ResetRenderTexture()` - re-creates RenderTexture and clears it in the next Update;

`public void ScratchHole(Vector2 position)` - scratches hole using texture position;

`public void ScratchLine(Vector2 startPosition, Vector2 endPosition)` - scratches line using texture positions;

`public Texture2D GetScratchTexture()` - returns scratch texture;

`public void SetScratchTexture(Texture2D texture)` - sets new scratch texture.

EraseProgress

EraseProgress component creates and configures RenderTexture then using shader calculates the average alpha-channel value of ScratchCard.RenderTexture.

Component samples ScratchCard.RenderTexture 225 times (15 times by horizontal and 15 times by vertical) in MaskProgress and MaskProgressCutOff shaders. It is not recommended to increase samples count, because a few users reported problems when shaders with 16x16 and more samples on some Android devices.

EraseProgress script has events:

`public event ProgressHandler OnProgress;` - invokes, when user scratches/restores surface;

`public event ProgressHandler OnCompleted;` - invokes, when user completed scratching/restoring surface.

EraseProgress script has methods:

`public float GetProgress()` - returns scratch erase progress from 0 to 1;

`public void UpdateProgress()` - updates scratch progress;

`public void ResetProgress()` - resets isCompleted flag for for further interaction.

Tips and tricks

User can scratch surface and restore it, to do that, set scratch mode to Restore using code:

```
var scratchCard = … //ScratchCard component reference
scratchCard.Mode = ScratchCard.ScratchMode.Restore;
```

For switching back to erase mode, set mode:

```
scratchCard.Mode = ScratchCard.ScratchMode.Erase;
```

You can clear and fill scratch surface using following methods:

```
//(100% scratched surface)
scratchCard.FillInstantly();
//(0% scratched surface)
scratchCard.ClearInstantly();
```

You can scratch holes and lines from code:

```
//draws hole in [100, 100] texture pixel position
scratchCard.ScratchHole(new Vector2(100, 100));
//draws line from [100, 100] to [200, 100] texture pixel positions
scratchCard.ScratchLine(new Vector2(100, 100), new Vector2(200, 100));
```

You can get and set scratch texture in runtime from code:

```
//gets scratch texture
scratchCard.GetScratchTexture();
//sets scratch texture
Texture2D someTexture = … //texture
scratchCard.SetScratchTexture(someTexture);
```

You can subscribe to scratch progress event and get erase progress in range [0, 1]:

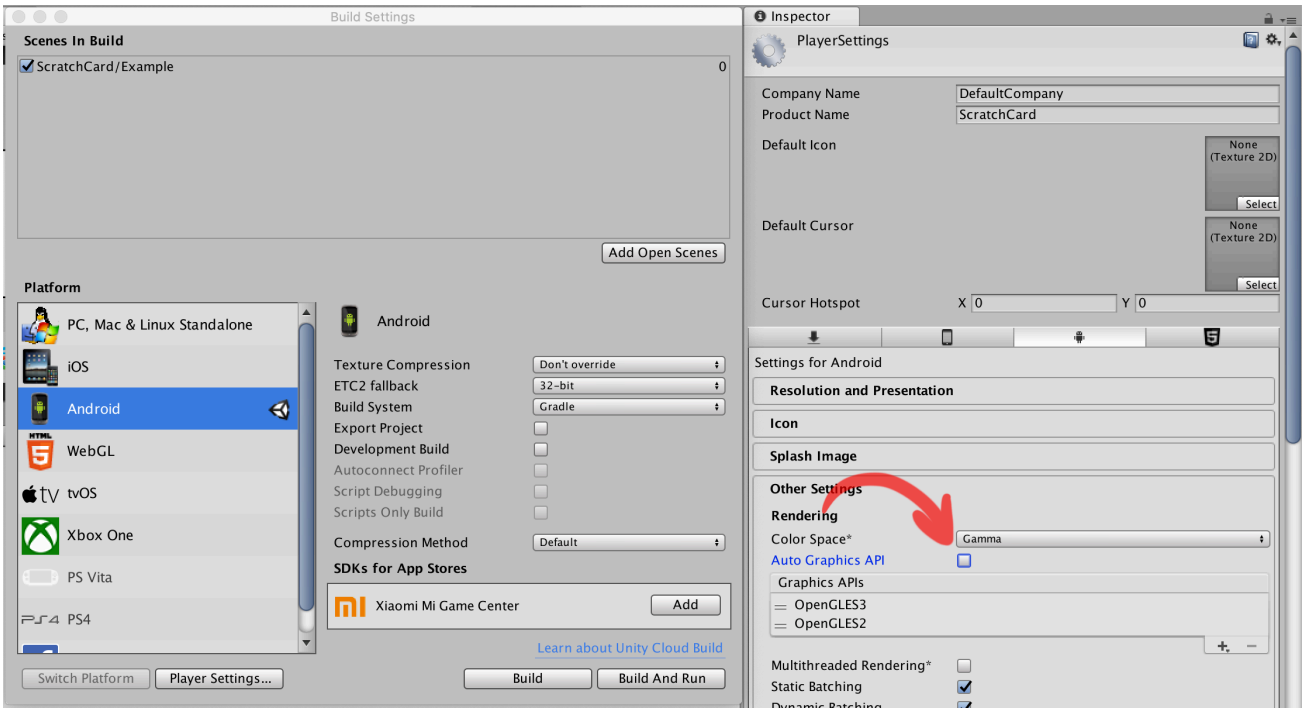
```
EraseProgress EraseProgress = … //EraseProgress component reference
//subscribe to OnProgress event, that invokes when texture scratches
EraseProgress.OnProgress += OnEraseProgress;

private void OnEraseProgress(float progress)
{
    Debug.Log("Erase progress: " + progress);
    if (progress >= 0.9f)
    {
        //fills scratch card when progress greater or equal 90%
        scratchCard.FillInstantly();
    }
}
```

Troubleshooting

1. By some reason, WebGL may ignore .shader and doesn't add files into build. To fix it, add shaders to Graphics Settings or move them to «Assets/Resources/» folder.

2. Unity earlier than 2019.3.0 has a bug on Android, when Graphics API in Project Settings -> Other is set as "*Auto Graphics API*" or has "*Vulkan*" in Graphics API list. To fix this, use Unity 2019.3.0 and newer or turn off **Auto Graphics API** using menu "**File -> Build Settings -> Player**

Settings -> Other -> Auto Graphics API”. Make sure that there is no “*Vulkan*” in Graphics API list.



Please let me know if you have any questions.
E-mail: unitymedved@gmail.com