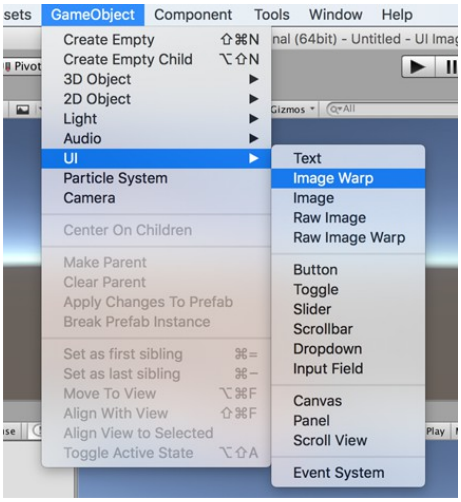
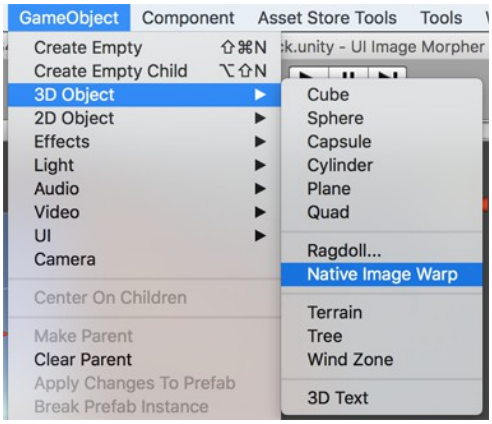
Table of Contents

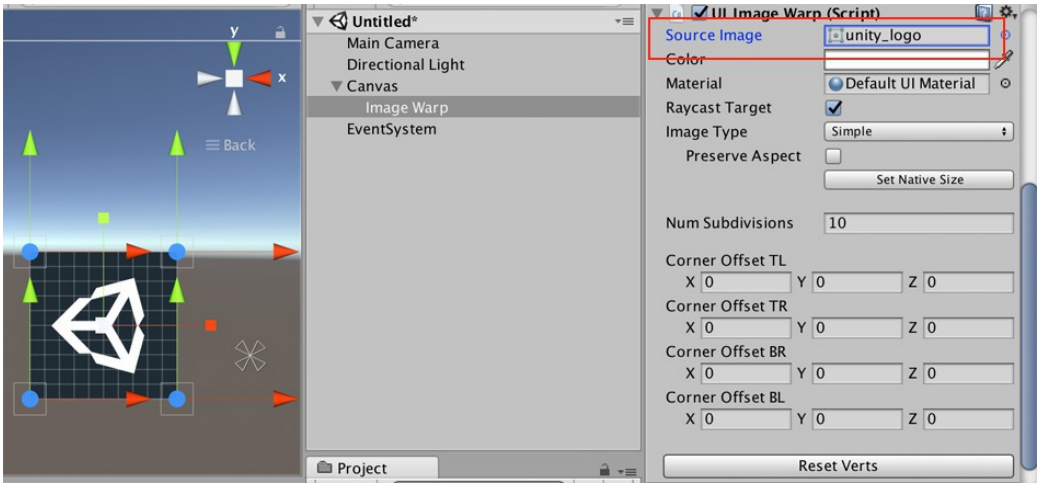
Getting Started

Create a new Image Warp object.

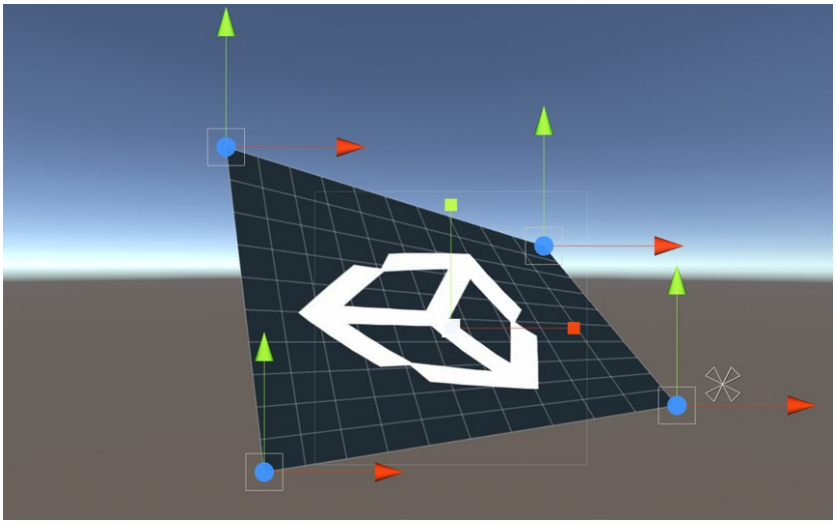




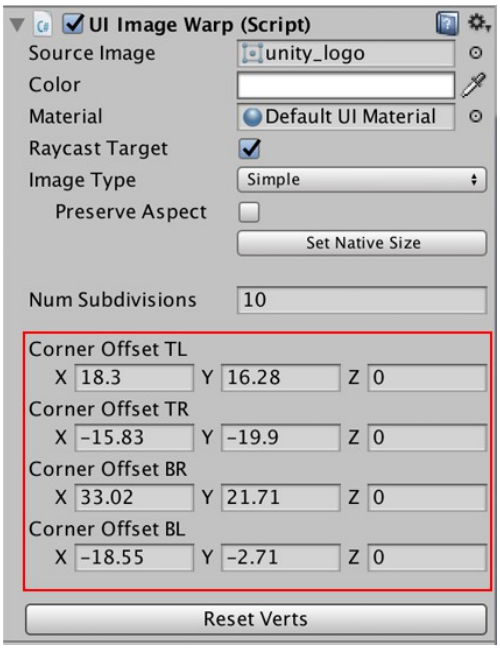
Assign a Sprite/Texture.



Warp it by dragging the corner handles directly in Scene View!



Or change the values precisely in the inspector.



Scripting With Image Warp

The Basics

- Include the '***Fenderrio.ImageWarp'*** namespace.
- Keep a reference of an *ImageWarp* component from your scene.
- Set warp properties!

```
1  using UnityEngine;
2  using Fenderrio.ImageWarp;
3
4  public class ImageWarpTest : MonoBehaviour {
5
6      public ImageWarp m_imageWarper;
7
8      void Start ()
9      {
10         m_imageWarper.cornerOffsetBL = Vector3.zero;
11         m_imageWarper.cornerOffsetTL = new Vector3(-20f, 20f, 0);
12         m_imageWarper.cornerOffsetTR = new Vector3(20f, 20f, 0);
13         m_imageWarper.cornerOffsetBR = Vector3.zero;
14
15         m_imageWarper.numSubdivisions = 12;
16     }
17 }
```

Public Properties

```
bool flipX { get; set; }
bool flipY { get; set; }

Vector3 cornerOffsetTL { get; set; }
Vector3 cornerOffsetTR { get; set; }
Vector3 cornerOffsetBL { get; set; }
Vector3 cornerOffsetBR { get; set; }

int numSubdivisions { get; set; }
bool bezierEdges { get; set; }

Vector3 topBezierHandleA { get; set; }
Vector3 topBezierHandleB { get; set; }
Vector3 rightBezierHandleA { get; set; }
Vector3 rightBezierHandleB { get; set; }
Vector3 bottomBezierHandleA { get; set; }
Vector3 bottomBezierHandleB { get; set; }
Vector3 leftBezierHandleA { get; set; }
Vector3 leftBezierHandleB { get; set; }

float cropLeft { get; set; }
float cropTop { get; set; }
float cropRight { get; set; }
float cropBottom { get; set; }

Vector3 cornerWorldPositionTL { get; }
Vector3 cornerLocalPositionTL { get; }
Vector3 cornerWorldPositionTR { get; }
Vector3 cornerLocalPositionTR { get; }
Vector3 cornerWorldPositionBR { get; }
Vector3 cornerLocalPositionBR { get; }
```

```
Vector3 cornerWorldPositionBL { get; }
Vector3 cornerLocalPositionBL { get; }

Vector3 topBezierWorldPositionHandleA { get; }
Vector3 topBezierLocalPositionHandleA { get; }
Vector3 topBezierWorldPositionHandleB { get; }
Vector3 topBezierLocalPositionHandleB { get; }

Vector3 rightBezierWorldPositionHandleA { get; }
Vector3 rightBezierLocalPositionHandleA { get; }
Vector3 rightBezierWorldPositionHandleB { get; }
Vector3 rightBezierLocalPositionHandleB { get; }

Vector3 bottomBezierWorldPositionHandleA { get; }
Vector3 bottomBezierLocalPositionHandleA { get; }
Vector3 bottomBezierWorldPositionHandleB { get; }
Vector3 bottomBezierLocalPositionHandleB { get; }

Vector3 leftBezierWorldPositionHandleA { get; }
Vector3 leftBezierLocalPositionHandleA { get; }
Vector3 leftBezierWorldPositionHandleB { get; }
Vector3 leftBezierLocalPositionHandleB { get; }
```

Public Methods

```
void ResetAll ();
void ResetCropping ();
void ResetCornerOffsets ();
void ResetBezierHandlesToDefault ();

void SetCornerWorldPositionTL (Vector3 a_worldPosition);
void SetCornerLocalPositionTL (Vector3 a_localPosition);
void SetCornerWorldPositionTR (Vector3 a_worldPosition);
void SetCornerLocalPositionTR (Vector3 a_localPosition);
void SetCornerWorldPositionBR (Vector3 a_worldPosition);
void SetCornerLocalPositionBR (Vector3 a_localPosition);
void SetCornerWorldPositionBL (Vector3 a_worldPosition);
void SetCornerLocalPositionBL (Vector3 a_localPosition);

void SetTopBezierWorldPositionHandleA (Vector3 a_worldPosition);
void SetTopBezierLocalPositionHandleA (Vector3 a_localPosition);
void SetTopBezierWorldPositionHandleB (Vector3 a_worldPosition);
void SetTopBezierLocalPositionHandleB (Vector3 a_localPosition);
void SetRightBezierWorldPositionHandleA (Vector3 a_worldPosition);
void SetRightBezierLocalPositionHandleA (Vector3 a_localPosition);
void SetRightBezierWorldPositionHandleB (Vector3 a_worldPosition);
void SetRightBezierLocalPositionHandleB (Vector3 a_localPosition);
void SetBottomBezierWorldPositionHandleA (Vector3 a_worldPosition);
void SetBottomBezierLocalPositionHandleA (Vector3 a_localPosition);
void SetBottomBezierWorldPositionHandleB (Vector3 a_worldPosition);
void SetBottomBezierLocalPositionHandleB (Vector3 a_localPosition);
void SetLeftBezierWorldPositionHandleA (Vector3 a_worldPosition);
void SetLeftBezierLocalPositionHandleA (Vector3 a_localPosition);
void SetLeftBezierWorldPositionHandleB (Vector3 a_worldPosition);
void SetLeftBezierLocalPositionHandleB (Vector3 a_localPosition);

void SetHandleWorldPosition (ImageWarpHandleType a_handleType, Vector3
a_worldPosition);
void SetHandleLocalPosition (ImageWarpHandleType a_handleType, Vector3
a_localPosition);
```

```
Vector3 GetHandleWorldPosition (ImageWarpHandleType a_handleType);
Vector3 GetHandleLocalPosition (ImageWarpHandleType a_handleType);
```

Backwards Compatibility

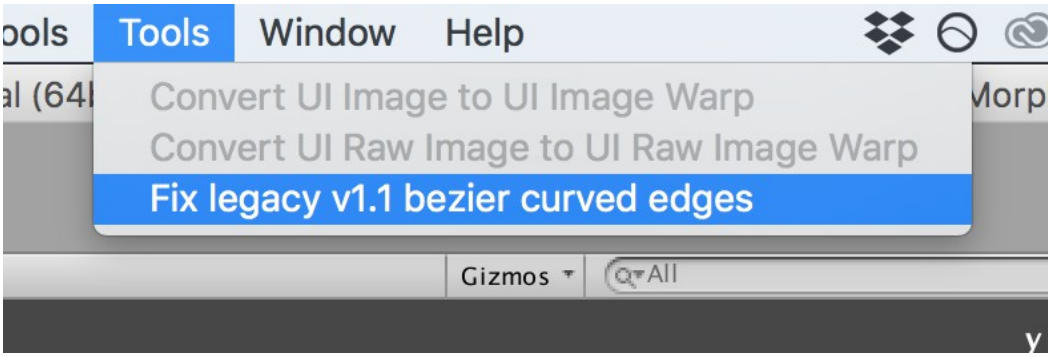
v1.1 Bezier Curve Data Correction Step

(Note: Only applicable to users who have upgraded from ImageWarp v1.1)

As part of the *v1.2* update, the underlying way in which the Bezier Curved edges are handled changed slightly (for the better!), which has unfortunately meant that existing legacy (pre v1.2) Image Warp implementations using Bezier edges could appear slightly differently after updating.

We've included an easy Menu Item shortcut to apply a 'correction' to your Image Warp Bezier data, returning it back to how it used to look!

Select your Image Warp gameObject in the scene, and go to ***'Tools->Fix legacy v1.1 bezier curved edges'***. Then save your scene.



Support

Support email: **fenderrio@gmail.com**

Thank you for buying Image Warp! :)

Changelog

v1.0.0 – 11/03/2017

- First release

v1.1.0 - 17/04/2017

- Added bezier curved edges functionality
- Fixed some bugs
- Optimisations

v1.2.0 – 11/02/2018

- Added image cropping support
- Added a 'Native Image Warp' implementation for use warping textures that aren't on a UI Canvas.
- Fixed some bugs
- Optimisations

v1.3.0 – 06/11/2018

- Added new API Properties & Methods for getting and setting warp handles using local and world positions
- NativeImageWarp now accepts 'Texture' class type, instead of just 'Texture2D'. Now accepts RenderTextures!
- Added options to Flip image horizontally and vertically.
- Added new Demo Scenes showing off 'Runtime Custom Warping' and 'Warping a Video Feed'
- Improved the Documentation; now a hyperlinked PDF instead of a PNG.