



Camera Projecter

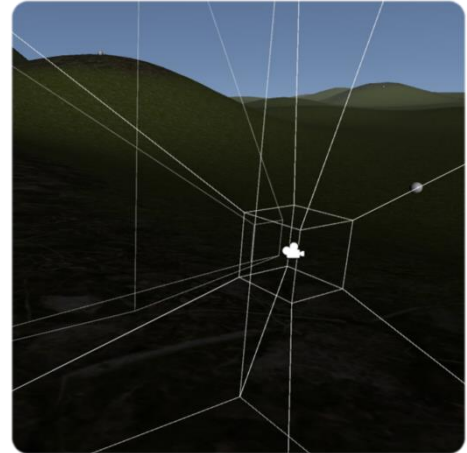
A simple shader projection trick to let camera render beyond 180°

V2.0

Introduction

Camera Projecter is an asset with purpose to add more different kinds of projection into Unity built-in camera. It works by capturing cubemap view of a camera then map to the screen using sphere projection techniques found in different sources on the internet.

The resulting image is largely depends on what kind of map projection is used. Different kind of projection has pros and cons and so it each has its unique purposes.



Some projection technique like Stereographic projection is used popularly to render beyond 180° and resembling the so-called Fish eye effect, some other like Geographic project is used to build a panorama image.

All projections is working in realtime and can be used on all devices which supports cubemap rendering.

How it works (technical detail)

Camera Projecter uses [Camera.RenderToCubemap\(\)](#) to render full 360 degree of current view relative from camera. If VR is enabled, the script will render cubes for each eye.

After capturing the cubemap, it will be rendered to screen according to selected sphere projection technique. Rough shader code would be like:

```
fixed4 frag(v2f i) : SV_Target
{
    // _MainTex is cubemap texture, while
    // map() projects 2D screen UV into
    // cubemap UV direction
    return texCUBE(_MainTex, map(i.uv));
}
```

In the end this looks simple, but there's a lot of other things to consider. Camera Projecter helps lift the technical difficulties in implementing this such that it is "just works". In this manual we'll guide you in how to use the asset and customizing things to suit your project needs.

Installation

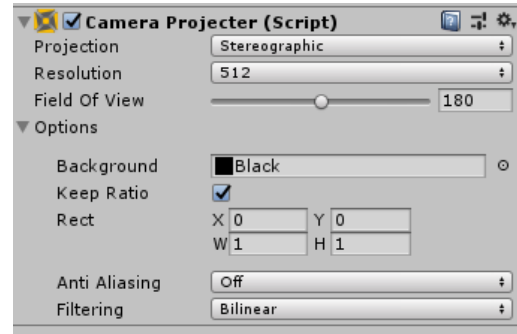
These steps below is most recommended to follow when using this asset for your project.

1. Select your main camera
2. Add CameraProjecter component via Inspector
3. Done! The rest of work is done automatically for you!

For convenience, we have ready-made GameObject in form of prefab located in Plugins/CameraProjecter/Prefab. You can choose between prefab called CameraProjecterAsMainCamera or CameraProjecterAsSecondCamera depending in your usage.

Customization

All parameters is exposed in the Camera Projecter component. When this component is enabled the camera will set to render zero batches (while leaving camera enabled so Camera.main still work) and make it self control whole rendering process with subset cameras that already made before.




Projection Specify current projection mode.

 See [kinds of projection](#) for more detail

Resolution Resolution of RenderTexture before projected to the screen/target texture. Note that bigger resolution will take up more GPU memory.

Field of View (FOV) Vertical angle of view

 Some projection types only can get until 180° even you still managed to force it futher until 360°

More options is shown after the expanding the folded options:

Background Optionally draw a texture before rendering the screen

Keep Ratio Should we account the screen ratio?

Rect Customized scale for output image

Anti Aliasing Anti aliasing amount applied to internal render texture.

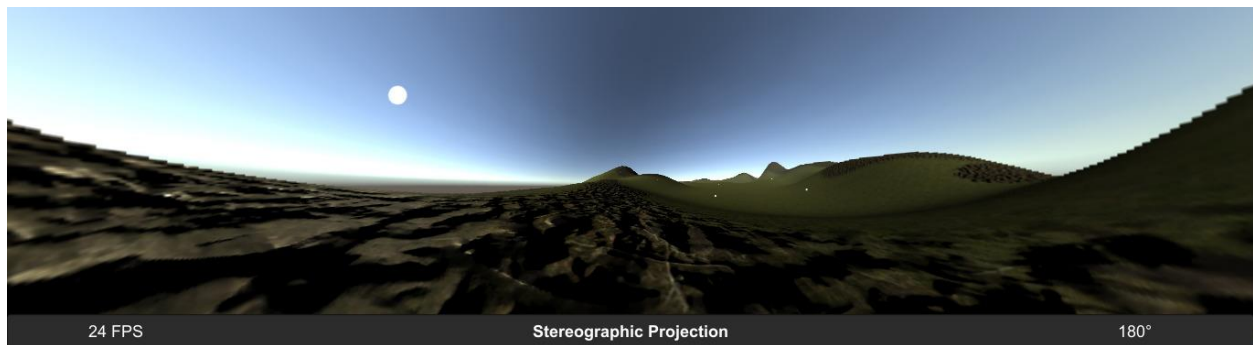
Filtering Filtering mode (Point/Bilinear) applied to internal render texture.

Purpose of each Projections

There are 7 types of projections available in CameraProjecter. First four of them is Azimuthal, means the projection works as a sphere into a plane, like shadow. The rest three is cylindrical, like a plane wraps around the sphere.

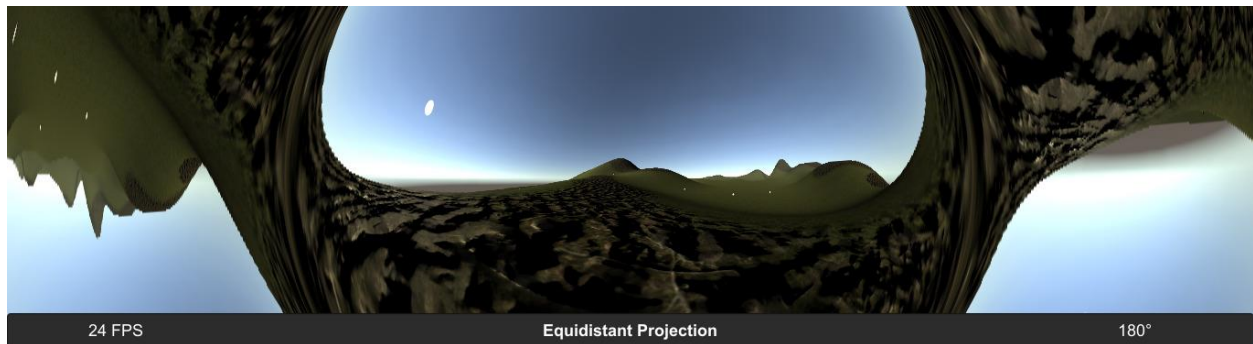
Stereographic

This projection is most widely known because it is preserves angle so no distortion will occur in the edge. Stereographic allows you to have FOV angle beyond 180° but distorted when near 360° . One of interesting imagery effect "little planet" can be produced using this projection.



Equidistant

This Azimuthal Equidistant projection is ideal for sensitive measurement like minimap because any angular distance and direction in produced image is correctly preserved in relative to the center point.



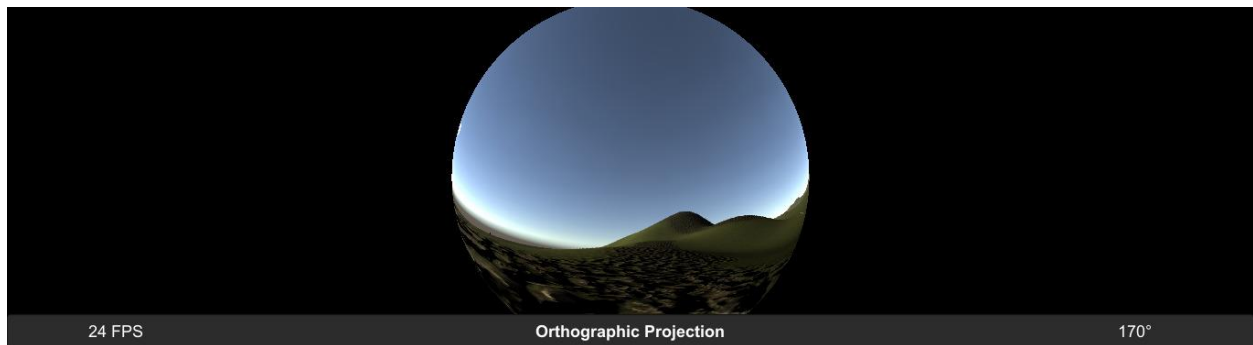
Equisolid

This projection is also known as Lambert's Azimuthal equal-area projection. It preserves an equal area on any given surface even near the edge. The result is similar to reflective sphere like shown in reflection probes.



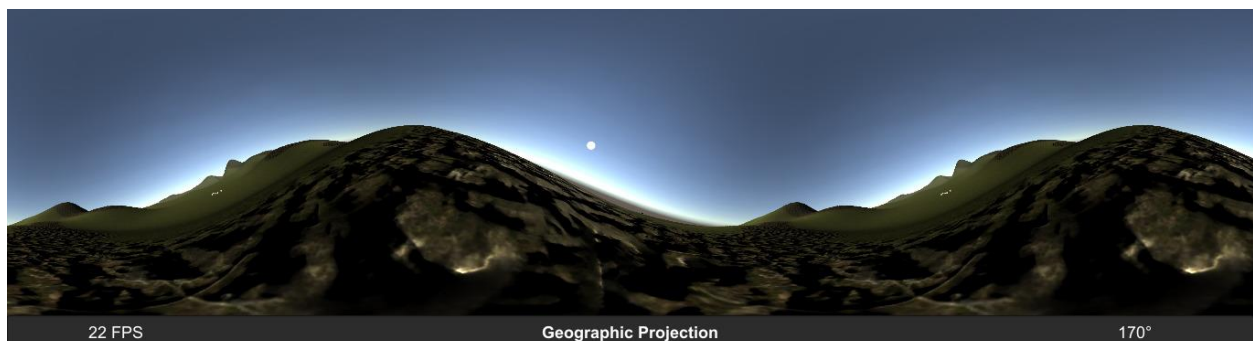
Orthographic

This *circular* orthographic projection is similar to most 180° wide lenses. The result image in 180° FOV is circular which means it is ideal for rendering to objects with convex lenses.



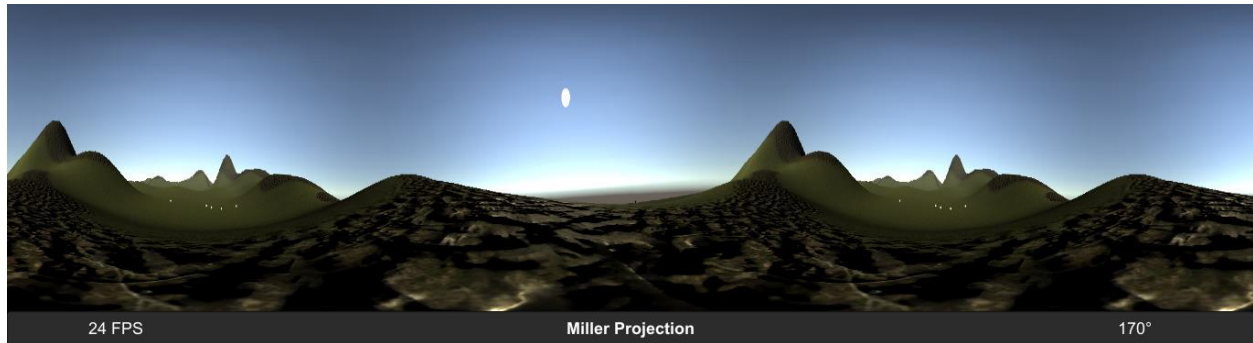
Geographic

This geographic (aka. Equidistant-Cylindrical/Equiarectangular/Longitude-Latitude) projection is the most simple and yet popular. The projection is equidistant to and have no distortion in the equator. This is the default projection for 360° panorama image.



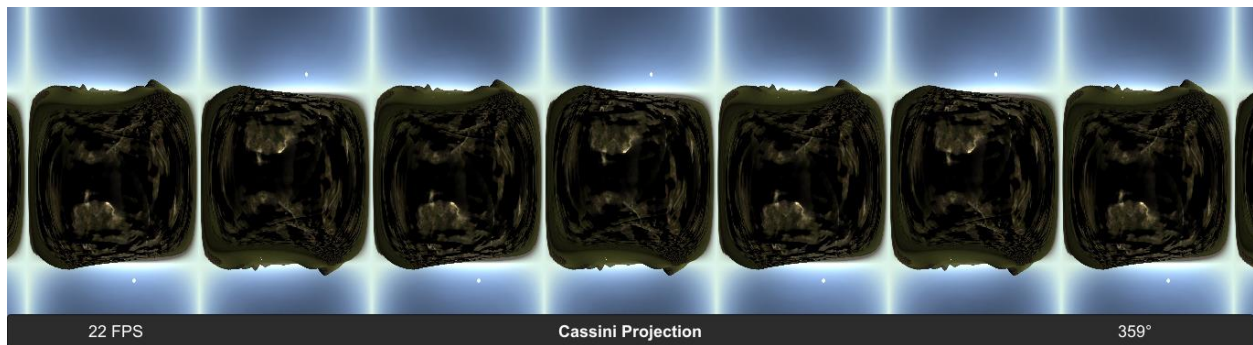
Miller

This Miller Cylindrical projection is the derivation of original Mercator projection and has its purpose to extend the equator so reducing distortion in polar area. Nothing is preserved in this projection.



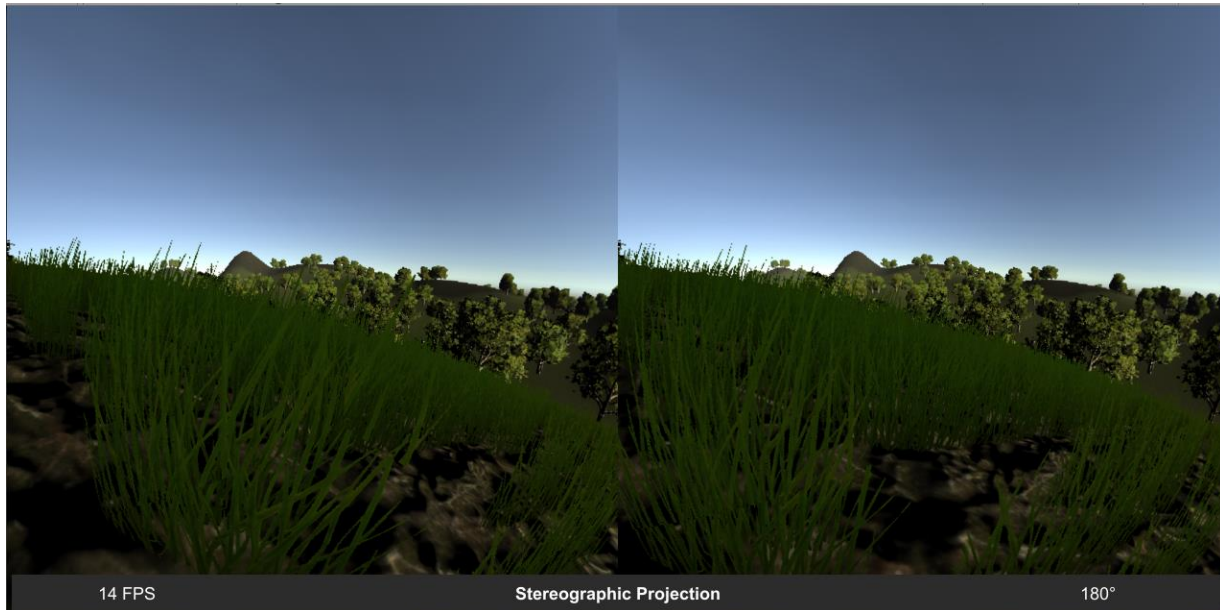
Cassini

Cassini Cylindrical projection is the transversed version of equirectangular projection and has its unique purpose that it doesn't distort the poles but the surrounding longitudes instead. To see in 360° the screen ratio must be 1:2 longer in vertical.



VR Support

Camera Projecter supports VR out of the box – No additional setup is necessary.



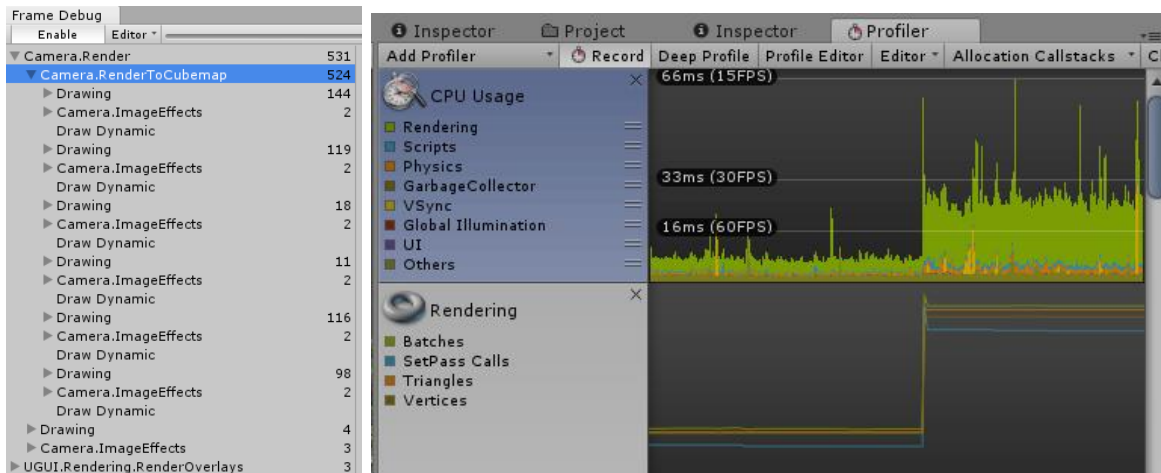
Please note that when VR enabled, the rendering time is doubled, because CP internally renders for both in a single frame – one for left eye then right one. Single pass rendering is also not supported at this time.

Please see the next page for performance considerations.

Performance Consideration

Camera Projecter suffers a major problem in performance. It became noticeable when you're using Camera Projecter in complex scenes.

The major problem is in `Camera.RenderToCubemap()`. Internally it's equivalent with using separate six camera doing rendering in different directions. This overhead can be much worse if VR is enabled.



So what can we do to avoid lag? Not so much, but we have small tips:

Don't Aim to be Perfect

Try to aim for 20-30 Hz instead of 60 Hz. It is still acceptable for most users.

Use Smaller Cubemap Resolution

Using smaller resolution can make it faster by 4x for each level down. So aim for fine resolution, like 256 for mobiles, 512 for HD monitors and 1024 for 2K monitors.

Use Smaller FOV

Yes, smaller FOV can help because Camera Projecter has a special feature where when FOV is small enough, it will only render the cubemap faces which shown on screen, generally saving 15%-50% render time.

The algorithm only applies for azimuthal projections for now, and it will improve over the next updates.

Use GPU Instancing

GPU instancing helps reduce batch count by batching together objects which share same material without cost. Read [Unity Manual](#) for this.

Use Lighter Quality

This could sacrifice game experience, but using lowest quality as default can also reduce render time quickly.

Use Other Graphics Goodie Tips

This includes graphics best practices, like using LODs, using narrower camera far clip, baking lightmap, using fewer shader features, and so on.

Release Notes

V2.0 – June 15, 2018

- NEW: VR Support (separate eye rendering)
- NEW: Selective rendering (results in performance improvements for small FOV)
- CHANGE: Removed separate six camera setup – now only use cubemap rendering
- CHANGE: Improved some projection offsets – some adjustment may needed after upgrade.
- CHANGE: Minimum unity version increased to 2018.1

V1.2 – June 5, 2017

- NEW: Undefined regions in some projection is explicitly masked out
- FIX: OnPreRender/Cull confusion. Significant improvement in performance expected.

V1.1 - May 23, 2017

- NEW: Cubemap rendering via Camera.RenderToCubemap()
- NEW: Tetrahedral camera setup which uses only 4 camera instead of 6!
- NEW: Support for rendering to RenderTexture
- NEW: Filter and Antialiasing choices for internal render textures
- NEW: Individual camera (aka. grabbers) now can be disabled or change the resolution separately
- NEW: Grabbers can be adjusted manually in scene
- NEW: CP now can be statically updated instead of every frame
- NEW: Scene example improvements with trees and bushes
- FIX: Shader doesn't compile in OpenGL ES 2.0

V1.0 - May 11, 2017

- First Release

About This Package

This package is built with love and curiosity in Asset Store published by Wello Soft.

Documentation is CC-BY 4.0

(C) Wello Soft 2017

[Website](#) | [Email](#)

[Asset Link](#) | [Documentation](#) | [Demo](#) | [Forum Thread](#)