Projection Rendering Toolkit

Unity plugin – Version 1.4

Documentation

Table of content

1 Introduction

Camera Projection Rendering Toolkit (CPRT) is a Unity plugin that includes C# objects, shaders and some resources as sample scenes and assets. It allows you to use advanced projection (non-linear perspectives, oblique orthographics), add SSAA (Super-Sampling AntiAliasing) and take high quality screenshot.

The property inspector is designed to be interactive and supports multi-selection, it only requires to add one script to your camera.

2 Known issues / Upcoming features

- Picking is not supported with ObliqueOrthographic.
- Rendering is broken on Unity 5.6.0, just update to 5.6.1 version.
- Some harmless warnings can appear in the console on Unity version 5.6.x :
    - “OnRenderImage() possibly didn't write anything to the destination texture”.
    - “Lighting data asset ‘LightingData’ is incompatible with the current Unity version. Please use Generate Lighting to rebuild the lighting data. Realtime Global Illumination cannot be used until the lighting data is rebuilt.”

3 Setup

3.1 Camera Prefabs

For starters, you can begin with a prefab that you’ll modify as you see fit (CameraProjectionRenderingToolkit/Prefabs -> drag and drop in your Hierarchy). These prefabs are actual cameras, so you may have to replace yours.

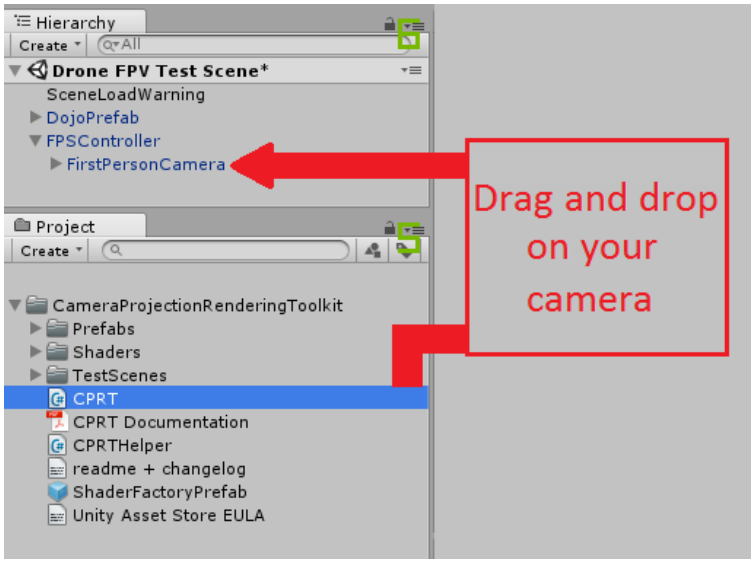
Available prefabs are:

- Camera SSAAx2: Creates high quality render by using Super Sampling AntiAliasing.
- Camera Pannini Simple: Removes horizontal distortions thanks to the Pannini projection.
- Camera Pannini Adaptive: Removes horizontal distortions and stays synchronized with the horizon with the Adaptive Pannini.

- Camera Fisheye: Removes objects distortion when they are far from the center of the screen, thanks to Stereospherical projection.
- Camera Oblique Isometric: Create an Oblique isometric projection.
- Camera Oblique Topdown: Create an Oblique topdown projection.
- Camera Oblique Cavalier: Create an Oblique cavalier projection.
- Camera Oblique Military: Create an Oblique military projection.

3.2 Add the script to a camera

If you already have a functional camera that you don't want to replace, you'll have to add the script and its resources.



3.3 General settings

3.3.1 Projection Type

3.3.1.1 SimpleOversampling

No projection modification, but like every other modes, gives access to supersampling/downsampling. See Oversampling Factor.

3.3.1.2 UnityBuiltinFisheye

Simple postprocessing effect that mimics a lens distortion effect (but it's quite inaccurate). This effect was in the unity built-in image effects, but wasn't able to oversample, so the center pixels were down-sampled, which can be fixed with a higher value of oversamplingFactor.

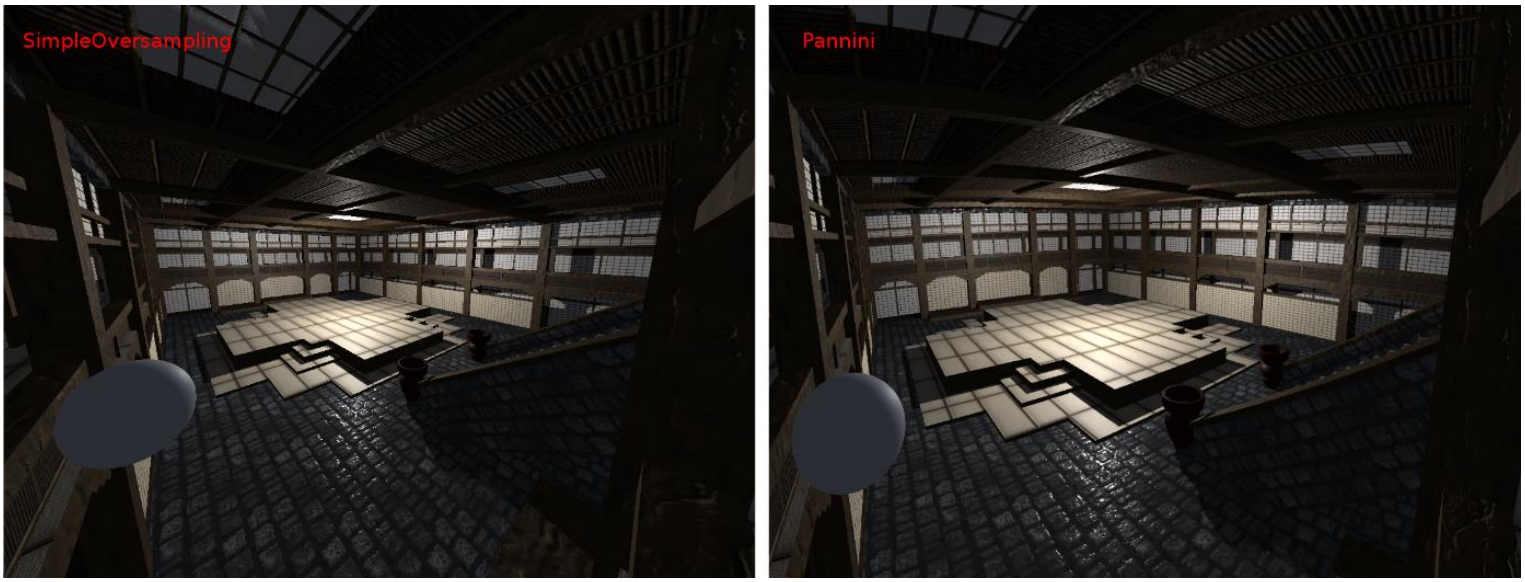
3.3.1.3 Pannini

Inspired by the painter Giovanni Paolo Pannini, this perspective projection isn't very realistic (still more than the classic one!) but allows the drawing of wide panorama and gives a pretty good feeling of being small relative to the architectural structures.

It's a stereo cylindrical projection (classic game projections are called rectilinear).

It's useful for reducing the distortion of objects near of the screen X borders when the field of view is high.

Furthermore, this projection gives more space to the center objects, which has the effect of "zooming" the center pixels. This is why you'll need to oversample the rendering to avoid blurred/pixelated viewport center.

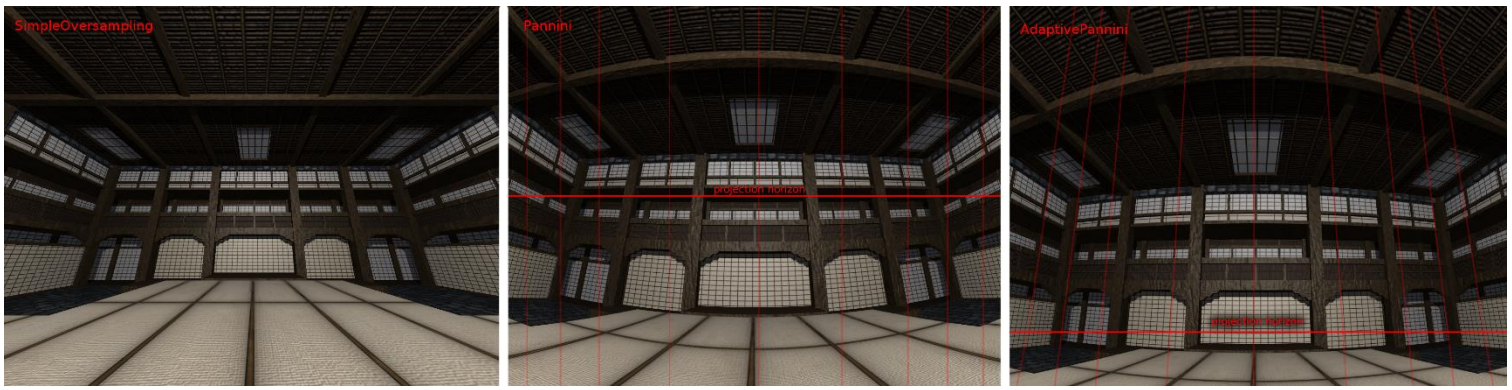


Screenshot with a 100° Y field of view. Note how the classic projection deform the sphere and the room borders, while the Pannini projection restore the shape in the X axis.

3.3.1.4 AdaptivePannini

This perspective projection is more suited to FPS and other "natural", frequently moving, not horizon-aligned views.

It keeps the horizon of the projection conform to the X-Z plane. The Pannini projection involves a singularity just below the camera (which is never visible unless the Y field of view is over 180°), so the AdaptivePannini will attenuate itself when the camera will be able to see it.



Screenshot with a 100° Y field of view. The Adaptive Pannini sets a coherent horizon, the projection mesh is now synchronized with the camera pitch.

The default values are quite good for a ~120° maximum Y field of view, otherwise you can adjust isAdaptiveAutomatic, adaptivePower and adaptiveTolerance.

Standard FPS settings would be 90° field of view, with intensity set to ~0,47.

Pannini painting "Piazza Navona Roma".

3.3.1.5 Stereospherical

"Fisheye effect", this perspective projection is the mathematically most accurate imitation of what the human eyes see.

It reduce objects distortion when they are far of the center of the screen (while Pannini only does it horizontally).

Like Pannini's projection, it gives more space to the objects in the center, which has the effect of "zooming" the center pixels. This is why you'll need to oversample the rendering to avoid blurred/pixelated image at viewport center.

In the inspector, you have an "Auto oversampling quality" button to automatically set quality related settings (oversamplingFactor and projectionPrecision attributs). For C# scripting, refer to the AutoOverSamplingFactor() and AutoProjectionPrecision() methods.

This projection require a pretty big mesh to be drawn which will make the default mesh precision quite GPU-heavy, so don't hesitate to halve it (31 is pretty good for real-time application).

3.3.1.6 ObliqueOrthographic

This orthographic projection can imitate what you see in classic RPGs, like Gameboy Pokemons (Top-Down) or Final-Fantasy Tactics (Isometric). It can draw Top-Down, Isometric, Military and Cavalier projections.

Please refer to Oblique orthographics settings to see typical settings.

3.3.1.7 PseudoOrthographic

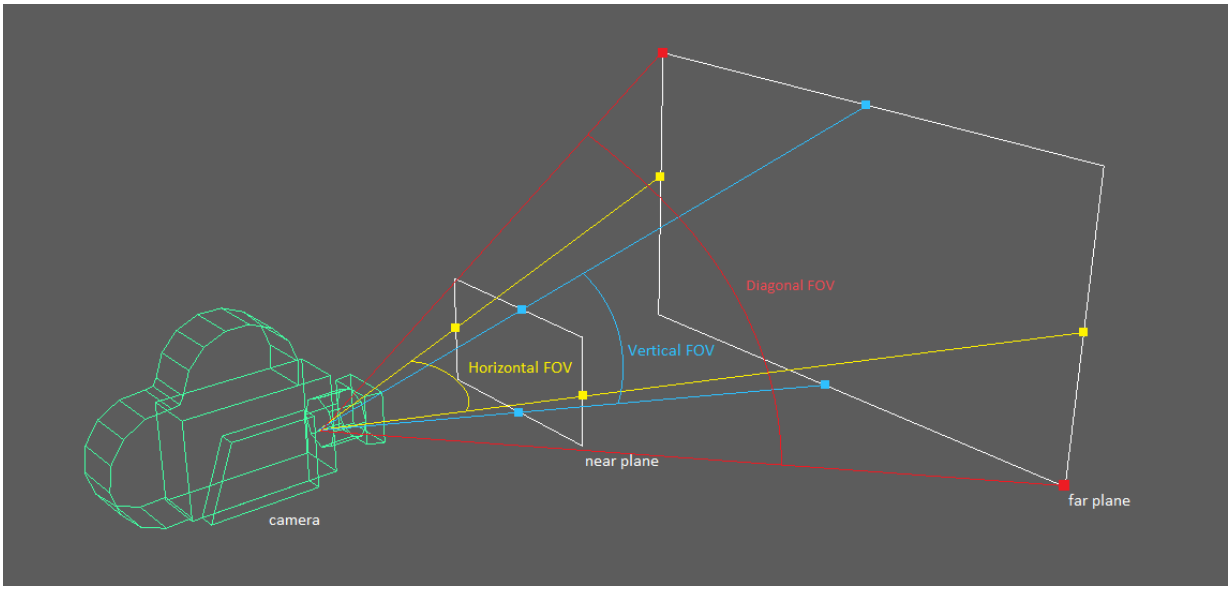
This projection mimics an orthographic projection with a perspective in order to allow deferred rendering.

Please refer to Pseudo orthographics settings to see typical settings.

3.3.2 Field Of View

Use this setting instead of the Unity Camera native one (which is disabled).

The FOV define the camera aperture, in degrees, either in vertical, horizontal or diagonal way.



Vertical FOV is the common unit for video games while Diagonal FOV is the real-life lens one.

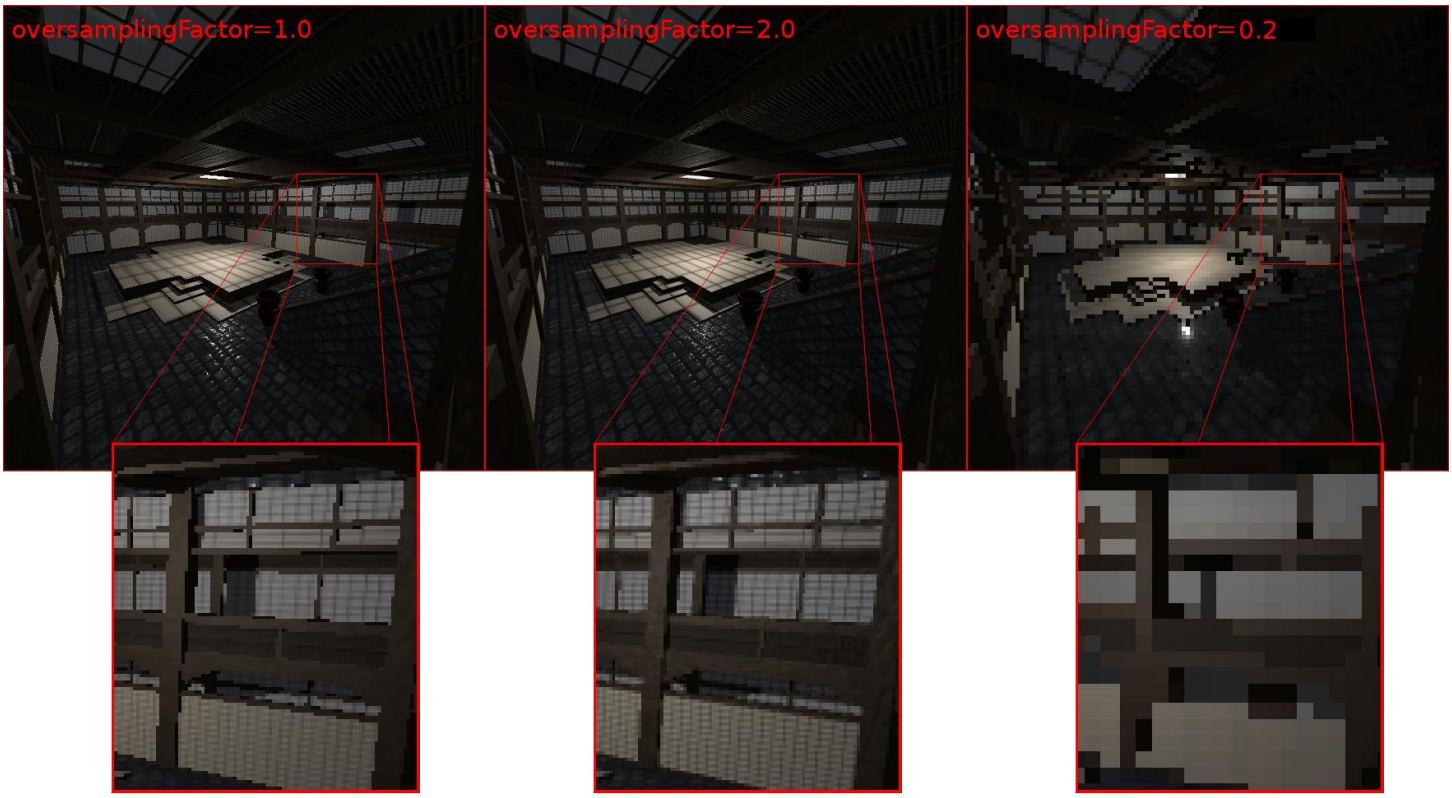
3.3.3 Oversampling Factor

- In Simple Oversampling mode

Super-Sampling Anti-Aliasing (SSAA, also called oversampling) has for effect to render the screen into a higher resolution then to average pixels to get a smaller image. This smaller image will have more samples per pixels, which will result in less blurry textures and anti-aliased edges.

This algorithm is basically the same than the Multi-Sampling Anti-Aliasing (MSAA) one, except it is applied everywhere (MSAA only oversamples pixels that are on an edge) during the whole rendering path, but has the drawback of using more memory and computing power. Unlike MSAA, SSAA is compatible with deferred rendering.

2x SSAA is equivalent to a 4x MSAA on the polygons-edges. This setting can be GPU-Heavy.



- For non-linear projections (Stereospherical / Pannini)

Non-linear projections usually give more space to the objects in the center, which has the effect of "zooming" the center pixels. This is why you'll need to oversample the rendering to avoid blurred/pixelated image at viewport center.

"Auto oversampling quality" and "SSAA 2x" buttons are available in the inspector to automatically set quality related settings.

3.3.4 Intensity

Useful with Stereospherical and Pannini projection, defines the interpolation between the transformed projection and the rectilinear one. The value range is between 0 and 1. Lower values decrease the projection oversampling needs.

0 = rectilinear (classic); 1 = fully transformed projection.

3.3.5 Target Texture

Use this setting instead of the Unity Camera native one (which is disabled).

If you want to draw the scene in a Render Texture, set it here (otherwise, let it empty). This can be useful in the following cases (non-exhaustive):

- You want to make real-time surveillance cameras, set this setting to the Render Texture used by the screen's material.
- You want to optimize post-processing effects: using post-processing effects on the target texture means that it will be calculated on the final resolution, which is smaller. This way, you are effectively working after the last item on the camera FX chain.

  Please refer to the FX chain part for the exact process.

Be wary that the default camera behaviour is to render in the target each frame. To change this, you must disable the camera, and manually call CPRT.RenderToTexture() when you want the rendering to happen.

3.3.6 Viewport Rect

Use this setting instead of the Unity Camera native one (which is disabled).

Defines the viewport's drawing rectangle in normalized coordinates (0->1).

3.3.7 Filter / Filter Sharpening

Filter Mode: Filter mode used when viewport pixels are smaller or bigger that screen pixels, useful for producing good intermediate SSAA results (SSAAx1.5 for exemple), or to have correct antialiasing on non-linear projection (Stereospheric / Pannini).

Possible values:

- NearestNeightboor

No filter applied, the nearest sample is selected. Useful to get a pixelated rendering when oversamplingFactor<0.

- SimpleBilinear

This filter just mixes the 4 nearest samples. Sometimes a "blurring" pattern can appear with SSAA. The projection distortion isn't taken into account.

- DistordedBilinear

Same as SimpleBilinear, but also take into account the projection distortion, useful with Pannini and Stereospherical projection when aliasing is present on the border on the screen.

The distortion tolerance can be set with filterSharpen (1.414 is a pretty good value).

- DistordedBilinearOptimized

Faster and less accurate version of DistordedBilinear.

- UniformBilinear

Take into account the projection distortion and remove the blur pattern, but is blurrier and more GPU demanding than the other modes. It can be sharper with a higher value of filterSharpen (1.414 is a pretty good value).

Filter Sharpen: Sharpening of the filter, used on whole screen when filterMode is on UniformBilinear, or on distorded zones when filterMode is on DistordedBilinear or DistordedBilinearOptimized. Default: 1.414.

3.3.8 Render Depth Bits

Depth rendering precision. You can refer to RenderTextureDepthBitCount for more explanations.

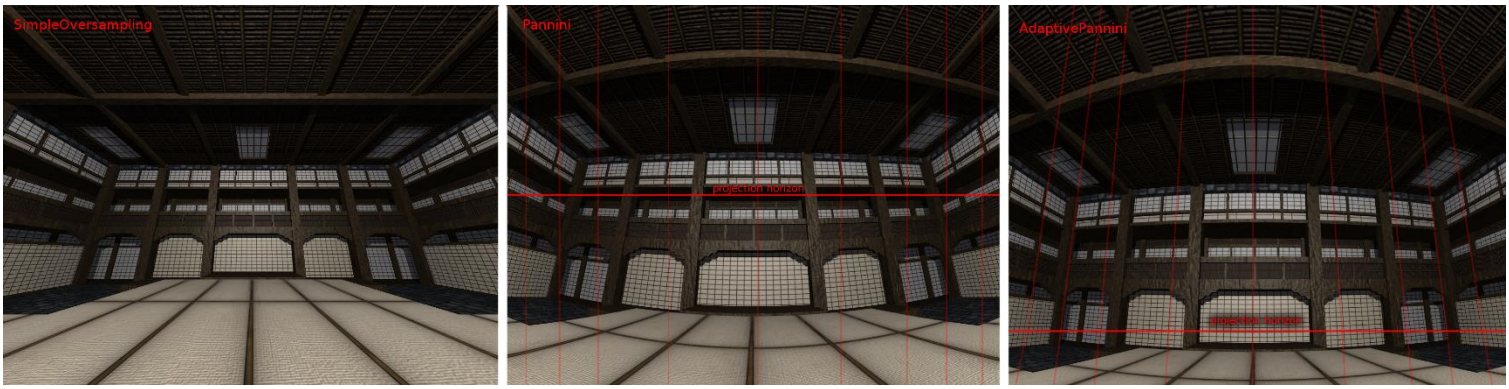
3.3.9 Screen Independant Screenshot

Screenshot Size: Final size of the screenshot.

Screenshot Oversampling Factor: Screenshot rendering size factor. Useful for making high quality screenshots with SSAA. This parameter can be GPU-Heavy.

3.4 Adaptive Pannini Settings

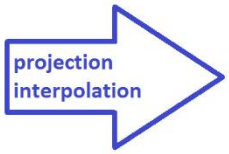
Adaptive Pannini keeps the horizon of the projection conform to the X-Z plane. But the Pannini projection involves a singularity just below the camera (which is never visible unless the Y field of view is over 180°), so the AdaptivePannini will attenuate itself when the camera will be able to see it and become a rectilinear projection (like with Intensity parameter).



Screenshot with a 100° Y field of view. The Adaptive Pannini sets a coherent horizon, the projection mesh is now synchronized with the camera pitch.

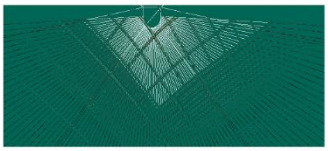
The default values are quite good for a ~120° maximum Y field of view. Standard FPS settings would be 90° field of view, with Intensity set to ~0,47.

With the following parameters, you can erase the singularity just below and above the camera, and give a natural projection interpolation when you variate the camera pitch.



Adaptive Pannini Settings

Adaptive Pannini Settings



(The projection mesh cannot be constructed in this area)

Is Adaptive Automatic: Does the AdaptivePannini work on automatic settings? (Not recommended but easier to configure).

Adaptive Power: When projectionType is on AdaptativePannini, defines the smoothness of the perspective adaptation. Should be greater than or equal to 1.

Adaptive Tolerance: When projectionType is on AdaptativePannini, defines perspective artifact tolerance. Set to 1 by default but tune it down if there is visible artifacts (looks like a distortion above or below the camera). Lower values means more visible distortion when looking up or down.

3.5 Oblique orthographic Settings

Oblique orthographic projections can imitate what you see in classic RPGs, like Pokemon games (Top-Down) or Final-Fantasy Tactics (Isometric). It can draw Top-Down, Isometric, Military and Cavalier projections.

They translate vertices in X/Y proportionnally to the depth from a focus point.

Oblique Bias: Defines the amount of x/y bias per depth unit.

Zero Bias Distance: When projectionType is on ObliqueOrthographic, defines the distance from the camera where the bias is 0. Typically the distance between the camera and the floor in a Top-Down perspective.

Presets buttons are available for quick access to common projections.

3.6 Pseudo orthographic Settings

This projection mimics an orthographic projection with a perspective in order to allow deferred rendering.

To have good results, make sure the center of interest is around halfway of the camera near & far clip plane.

It introduces:

- The orthographicSize parameter, which replaces the Camera.orthographicSize parameter. This parameter exists because when the camera is on perspective mode, orthographicSize isn't visible in the editor.
- The perspectiveOffset parameter, which defines the distance of the perspective from the camera. The higher it is, the closer the perspective is to an orthographic projection.

3.7 FX Chain / Rendering pipeline

When CPRT is enabled, the rendering is done in the target resolution multiplied by the oversampling factor with a linear projection. Post processing effect can be applied just after this step (those are Pre-CPRT effects). The CPRT plugin will then apply its non-linear projection distortion and downscaling, which will bring back the target resolution. At this point, pixel information other than color is lost (pixel

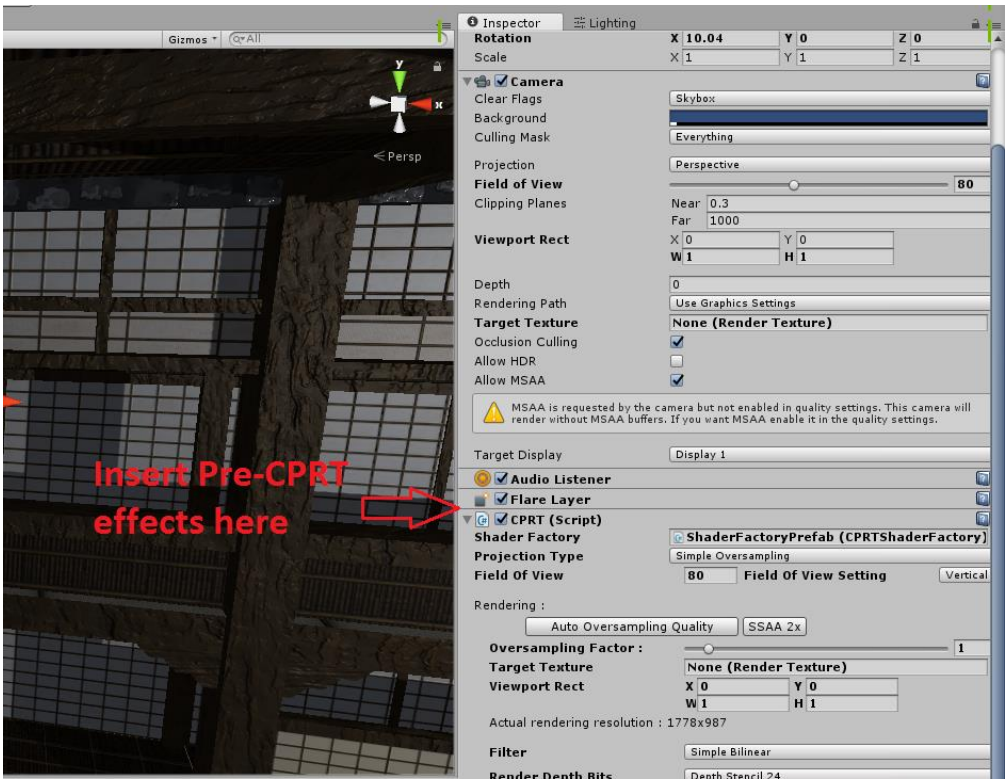
depth, diferred rendering information like pixel position, normal, albedo, reflectivity…), meaning you can still add color related effects after this step (Post-CPRT effects).

About GUI elements (not the legacy ones), if the GUI Canvas is in « Screen Space Overlay » mode, standard GUI elements can be added as usual and will be drawn after the post-processing.

3.7.1 Pre-CPRT effects

Post-processing effects which need other information than the pixel color must be applied on the camera before the CPRT effect.

Typical pre-CPRT effects : Ambient Occlusion, Screen Space Reflects, FXAA, some motion blur implementations…



3.7.1 Post-CPRT effects with GUI

If you need to render post-processing effects in the target resolution (to get better performance for exemple) or if don't want them to be distorded by non-linear distortions, you'll have to apply them after the CPRT effect.

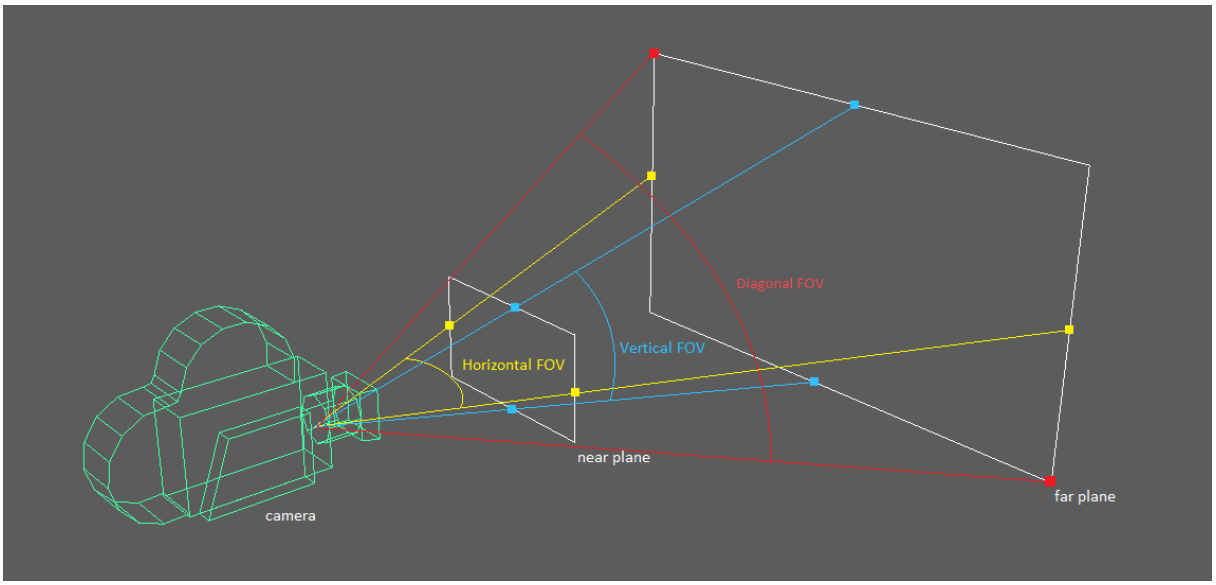
You'll need to create your own Render Texture (and reference it in the Target Texture setting), and then apply your effects on it.

Typical post-CPRT effects : Glow, Sun Rays, Screen Motion Blur …

With the following technique which uses Unity GUI elements (not the legacy ones), you can setup post-CPRT effects and your GUI.

You'll need:

- A Camera rendering the scene without the « UI » layer,
- A second Camera only rendering the « UI » layer, where you can attach post-CPRT effects / scripts,
- A fullscreen Raw Image with a material drawing the Target Texture (alternatively you can apply your post-CPRT effects on it instead of the second camera),
- A script attached on the second camera with the following events:
    - in Start() event, disable the first Camera,
    - in OnPreCull() event, create / configure the target texture, possibly with RenderTexture.GetTemporary(), then call the CPRT.RenderToTexture() method from the first camera,
    - in OnPostRender() event, will eventually release the target texture by calling RenderTexture.ReleaseTemporary().
- Standard GUI elements in the UI layer in front of the fullscreen Raw Image,



3.7.1 HDR remark

If you're using HDR on your cameras, is highly recommended to use a Color Grading effect to get the full quality of SSAA. Without color grading, edge with high brightness difference will produce jagged lines regardless of SSAA activation.

4 Objects / Components

Every classes of the plugin are into the CameraProjectionRenderingToolkit namespace.

4.1 CPRT

This is the main object to implement custom projections (perspective and orthographic).

To transform a camera projection, you have to add the CustomCameraProjection script to the camera. Move it down below any post-processing image effect in the inspector (glow, tone mapping, antialiasing…).

This script add a super-sampling (SSAA) / down-sampling functionality, various re-projection effects, it can also modify the camera projection matrix.

4.1.1 Attributes

Inspector variables / object attributes of CameraProjectionRenderingToolkit*, documented by type, name, default value and description.*

These attributes will be in the inspector if they are used. Furthermore, some errors, warnings, and their fixes will show up if needed.

Major features are in bold.

Name	Type	Default	Description
			Projection attributes
projectionType	CCPType	SimpleOversampling	Must be coherent relative to the Camera.orthographic projection mode. If applicable, the projection will match the Camera.fieldOfView property. A big field of view result in more oversampling needs (see oversamplingFactor). RefreshEffect() and RefreshViewport() must be called if it has been changed by C# scripting. **Please refer to the CCPType enum for more explanations.**
intensity	float	1.0	Power of the projection deformation. The value range is between 0 and 1. Lower values decrease the projection oversampling needs (see oversamplingFactor). 0 = rectilinear (classic); 1 = transformed projection.
isAdaptiveAutomatic	bool	false	Does the AdaptivePannini work on automatic settings? (Not recommended but easier)
adaptivePower	float	4.0	When projectionType is on AdaptativePannini, defines the smoothness of the perspective adaptation. Should be greater than or equal to 1.
adaptiveTolerance	float	1.0	When projectionType is on AdaptativePannini, defines perspective artifact tolerance. Set to 1 by default but tune it down if there is visible artifacts (looks like a distortion above or below the camera). Lower values means more visible distortion when looking up or down.
obliqueBias	Vector2	{0.0 ; 1.0}	When projectionType is on ObliqueOrthographic, defines the amount of x/y bias per depth unit. Typical settings : - Top-Down : {0 ; 1} with camera rotation at {90 ; 0 ; 0} - Isometric : {0 ; 0.1709} with camera rotation at {45 ; 45 ; 0} - Cavalier : {0.707106 ; -0.707106} with camera rotation at {0 ; 0 ; 0} - Military : {0 ; 1} with camera rotation at {90 ; 45 ; 0}

Name	Type	Default	Description
obliqueZeroDistance	float	0.0	When projectionType is on ObliqueOrthographic, defines the distance where the bias is 0. Typically the distance between the camera and the floor in a Top-Down perspective.
perspectiveOffset	float	2000.0	When projectionType is on PseudoOrthographic, defines the distance of the perspective from the camera. The higher it is, the closer the perspective is to an orthographic projection.
orthographicSize	float	100.0	When projectionType is on PseudoOrthographic, this parameter replaces the Camera.orthographicSize parameter. This parameter exists because when the camera is on perspective mode, orthographicSize isn't visiblein the editor.
Quality/rendering attributes			
oversamplingFactor	float	1.0	Real rendering size factor. Useful for making SSAA. Non-linear projections need a bigger rendering precision. Values greater that 1 result in oversampled rendering. Values lower that 1 result in downsampled rendering, which results in pixelated image. Values greater than 2 aren't currently supported for oversampling and are just useful for projection purpose. Warning: This value multiply the rendering resolution, **which can dramatically be GPU-heavy** (keep an eye on the "Actual rendering resolution" in the inspector), it may be clamped if the render size cannot be supported. High values may not be supported on every system. Check the **IsResolutionSupported** property after a call to **RefreshViewport**() or after a render, but basically, lower resolution than 4096x4096 are supported. Can be auto-adjusted with AutoOverSamplingFactor() or with the button just above in the inspector.
projectionPrecision	int	64	Precision of the rendering mesh, can be auto adjusted with AutoProjectionPrecision() or with the button just above in the inspector. Typical value is 64 for Pannini, 32 for StereoSpherical (Higher values can be GPU-heavy).
viewportRect	Rect	Rect(0,0,1,1)	Viewport's drawing rectangle in normalized coordinates (0->1). **When the effect is activated, use this parameter instead of the native camera viewport-rect parameter**, there is an interactive button in the inspector to help you doing this when needed.
filterMode	CPRT FilterMode	Bilinear	Filter mode used when viewport pixels are smaller or bigger that screen pixels, useful for producing SSAA. You can refer to CPRTFilterMode for more explanations.
filterSharpen	float	1.414	Sharpening of the filter, used on whole screen when filterMode is on UniformBilinear, or on distorded zones when filterMode is on DistordedBilinear or DistordedBilinearOptimized.
msaaSetting	Render Texture MSAA	NoMSAA	Rendering MSAA sample count. **You can refer to RenderTextureMSAA enumerator for more precision about performance and compatibility.**
renderTextureDepthBits	Render Texture Depth BitCount	depth16	Depth rendering precision. You can refer to RenderTextureDepthBitCount for more explanations.
projectionWireframe	bool	false	This option draws the projection mesh wireframe (for debugging, or understanding how projection is made).
Offscreen screenshot			
screenshotOversampling Factor	float	1.0	Screenshot rendering size factor. Useful for making high quality screenshots with SSAA. This parameter can be GPU-Heavy.
screenshotWidth	float	1920	Screenshot viewport width. Values higher than 16384 are forbidden by the current hardware, moreover **screenshotOversamplingFactor** divide the limit. Example: a **screenshotOversamplingFactor** set to 2.0 limit the resolution to 8192x8192. Hardware limits can be added too.

Name	Type	Default	Description
screenshotHeight	float	1080	Screenshot viewport height. Values higher than 16384 are forbidden by the current hardware, moreover **screenshotOversamplingFactor** divide the limit. Example: a **screenshotOversamplingFactor** set to 2.0 limit the resolution to 8192x8192. Hardware limits can be added too.

4.1.2 Inner types

4.1.2.1 enum CCPType

This enumerator defines a projection type. You can refer to the Projection Type setting for more explanations.

4.1.2.2 Enum CPRTFilterMode

This enumerator defines the filter used for blending rendered pixels when they are smaller or bigger than the screen pixels (happen with SSAA, Pannini and Stereospherical projections). You can refer to the Filter / Filter Sharpening settings for more explanations.

4.1.2.3 enum RenderTextureMSAA

This enumerator defines the sample count of each pixel in a render texture. Increasing it reduces aliasing.

It can be casted into an int value that gives the sample count per pixel.

For better performance and quality: don't set any MSAA setting in unity, just use this MSAA setting or one of the Unity Built-in AA Image Effects, but they must be BEFORE this effect. Disable Unity MSAA in "Edit->Project Settings->Quality".

This effect is compatible with HDR, but keep in mind that **Unity isn't compatible with MSAA when deferred rendering path is enabled**. When using deferred rendering, you can use SSAA by multiplying the oversamplingFactor by a value between 1 and 2 (GPU-Heavy), or add an antialiasing image effect. Antialiasing image effect is a post-processing effect which is more optimized than SSAA, but more blurry and more affected by temporal distortion. I personally recommend the Unity FXAA1 Preset B.

Possible values:

- Depth16

Recommended value for mobile platforms.

- DepthStencil24

This last bit count permit to use a stencil buffer, it can be required by some shaders.

- Depth32

4.1.2.4 enum RenderTextureDepthBitCount

This enumerator defines the bit count (quality) of the depth buffer of a render texture (that defines each object depth). Increasing it reduces the z-fighting artifact.

It can be casted into an int value to fill some Unity functions.

Possible values:

- Depth16

Recommended value for mobile platforms.

- DepthStencil24 (Default)

This last bit count permit to use a stencil buffer, it can be required by some shaders.

- Depth32

4.1.3 Properties

This part is useful if you need to change the effect in real-time by C# scripting.

Name	Type	Accessibility	Description
IsResolutionSupported	bool	Read-only	Returns if the needed rendering resolution is too high, RefreshViewport() must be called to update this value (it's updated on each render).
IntensityFactor	float	Read-only	Factor applied on intensity when projectionType is on AdaptativePannini
RenderTargetWidth	int	Read-only	Rendering buffer width (with oversampleFactor taken into account)
RenderTargetWidth	int	Read-only	Rendering buffer height (with oversampleFactor taken into account)
PixelRect	Rect	Read-only	Viewport rectangle in pixel coordinates
IsNonLinearProjection	bool	Read-only	Is the current projection type a non-linear projection?
ObserverFOV	float	Read-only	Projection "observer" field of view, mainly used for debugging. Currently read-only, but writing could be useful for creating a “cathode screen effect”, let me know if it’s needed.
ProjectionMaxPixel Scaling	float	Read-only	Get the approximate maximum pixel scaling. Can be used as an oversampleFactor value.

4.1.4 Methods

This part is useful if you need to change the effect in real-time by C# scripting.

Name	Return type	Parameters	Description
Methods to call when changing settings			
RefreshEffect	void	None	Refresh shader and antialiasing parameters and resources. Must be called if antialiasing settings or the projection type has been changed. Called by Unity's callback OnValidate() and by Start().
RefreshViewport	void	None	Refresh the necessary resources for the perspective. Must be called if there is any change in Viewport's size/AA/HDR settings, and after RefreshEffect() (if needed). Called by Unity's callback OnPreCull().
CheckResources	bool	None	Return true if the effect is compatible with the system. Refresh the post-processing shader if needed.
CheckViewportSize	void	None	Checks if oversamplingFactor can be supported, and change its value if not. Automatically called before each renderings.
Other methods			
RenderToTexture	void	- RenderTexture tex: Target Texture for this rendering. - float oversampling_factor: See Oversampling Factor.	Manually render the viewport into the specified target texture. See Target Texture.
OffsreenScreenshot	void	- string path : Relative/absolute path to the screenshot file to create/replace.	Take a screen-independent screenshot and save it to a file. Uses screenshotOversamplingFactor, screenshotWidth and screenshotHeight.
AutoOverSamplingFactor	void	None	Select a value of oversamplingFactor that avoids pixelization (all rendered pixels will never be bigger than one screen pixel).
AutoProjectionPrecision	void	None	Select a value of projectionPrecision that minimize projection distortion
IsScreenPointInViewport	bool	- Vector2 screen_position : Screen point in pixel coordinates, origin at upper-left.	Returns true if screen_position is in viewportRect.

Name	Return type	Parameters	Description
ScreenPointToRay	Ray	- Vector2 screen_position : Screen point in pixel coordinates, origin at upper-left.	Returns a normalized vector that go from the camera through a screen point while taking into account the correct perspective system. Does not work with the ObliqueOrthographic projection.
ViewportPointToRay	Ray	- Vector2 viewport_position : Viewport point point in [0.0 -> 1.0] coordinates, origin at upper-left.	Returns a normalized vector that go from the camera through a viewport point while taking into account the correct perspective system. Does not work with the ObliqueOrthographic projection.

4.2 CPRTToolkit

This static class regroups some functions that are useful for the custom projection system. Matrix convention is Row-Major (same as Unity).

Maybe you'll find a function that can be useful to you.

4.2.1 Methods

Name	Return type	Parameters	Description
		Camera Maths	
GetFovX	float	- float fov_y: Y field of view in radians (Camera.fieldOfView) - float aspect: Viewport width/height, (Camera.aspect)	Get the X field of view of a camera in radians
GetFovX	float	- Camera camera	Get the X field of view of a camera in radians
GetFovY	float	- float fov_x: X field of view in radians - float aspect: Viewport width/height, (Camera.aspect)	Get the Y field of view of a camera (number given by Camera.fieldOfView) in radians
		General Maths	
Cotan	float	- float x: angle in radian	Returns cotangent of x.
ACotan	float	- float x: Cotangent	Returns inverse cotangent (sometimes called cotan^-1) of x in radians.
Clamp	int	- int x: Value to clamp - int min: Min value - int max: Max value	Returns x clamped by min and max.
Sqr	int	- int x	Returns square of x.
Sqr	float	- int x	Returns square of x.

Name	Return type	Parameters	Description
Pow3	int	- int x	Returns cube of x.
Pow3	float	- int x	Returns cube of x.
SpherePoint	Vector3	- int pitch: Pitch angle in the sphere, in radians - int yaw: Yaw angle in the sphere, in radians	Returns a point on a 1 radius sphere (yaw is applied first).
SpherePoint	Vector3	- Vector2 pitch_yaw: Contains pitch and yaw angle in the sphere respectively in x and y, in radians.	Returns a point on a 1 radius sphere (yaw is applied first).
Vector4Transform	Vector4	- Matrix4x4 matrix - Vector4 vec	Returns a vector transformed (multiplied) by the matrix (Matrix4x4.MultiplyPoint Vector4 equivalent)
Vector4Transform Perspective	Vector3	- Matrix4x4 matrix - Vector4 vec	Returns a vector transformed (multiplied) by the matrix and apply perspective on it : - When vec.w==1.0, it is equivalent to Matrix4x4.MultiplyPoint. - When vec.w==0.0, it transforms a vector.
VectorAbs	Vector3	- Vector3 vec	Returns the vector 'vec' with positive coordinates.
RotationX	Matrix4x4	- float x: angle in radian	Create a matrix that rotates around X axis.
RotationY	Matrix4x4	- float x: angle in radian	Create a matrix that rotates around Y axis.
RotationZ	Matrix4x4	- float x: angle in radian	Create a matrix that rotates around Z axis.

5 Ideas

Let me know if the next functionalities are needed:

- Supporting higher OversamplingFactor values than 2 (more than 4 sample per pixel maximum).
- Giving access to the projection zoom and to the border clearing color, which can allow to create a “cathode screen effect”.
- A shader that gives to orthographic voxel scenes a 90s isometric feel.
- More screenshot file types.
- Cylindrical projection in any directions (not only vertical like Pannini).
- Tweakable Adaptive Pannini up vector.
- Supporting older Unity versions.

6 Contact

For any question, request, contact me at:

melvin.rey@viacesi.fr