



MULTIRESOLUTION 2D

Thank you for using MultiResolution 2D Unity asset :)

Feel free to get in touch with suggestions, questions, or issues you might have.

Contact: nicolas@bananapps.com

Index

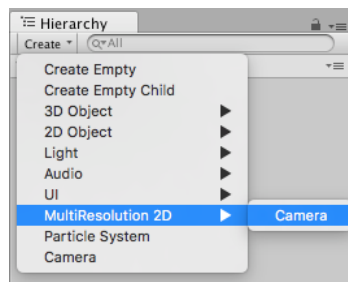
The MultiResolution Camera	3
Creating a MultiResolution Camera	3
What is an Auto Scaling Group?	4
Add sprites	5
Camera Scaler	6
Properties	6
Auto Scaling Group	7
Properties	7
Details	8
Scale Mode - Manual	8
Scale Mode - Aspect Fit Width	9
Scale Mode - Aspect Fit Height	9
Scale Mode - Aspect Fit	10
Scale Mode - Aspect Fill	10
Scale Mode - Closest Whole Number	10
Scale Mode - Pixel Perfect	11
Matching Criteria	12
Camera Anchor.....	13
Properties	13
Hints	14
Pixel Art Game	14

The MultiResolution Camera

Creating a MultiResolution Camera

1. Create a new scene and:

- Add the CameraScaler component to the Unity "Main Camera"
- or
- Delete the Unity "Main Camera"
 - Create a MultiResolution Camera by clicking on menu "GameObject / MultiResolution 2D / Camera". You can also add the camera from the hierarchy view via "Create / MultiResolution 2D / Camera"

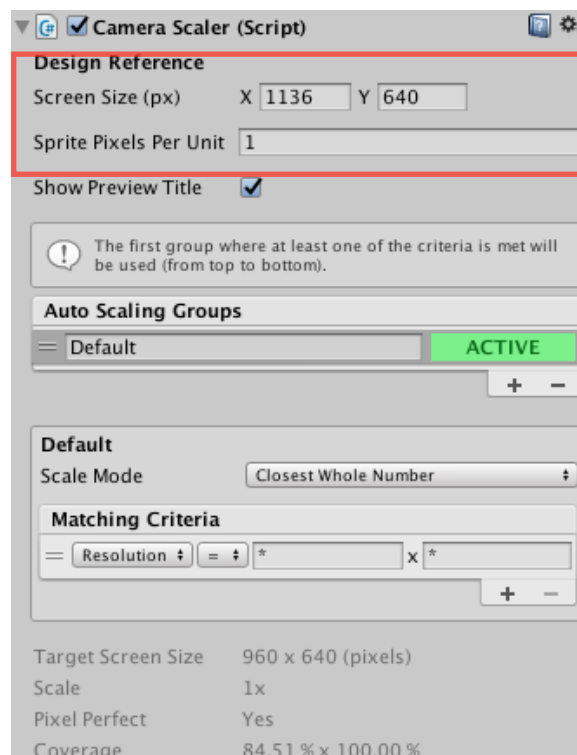


2. Select your camera. You can see the "Camera Scaler" component in the inspector view.

3. Set up your design screen size. This is the resolution your game is designed for.

4. Set up the Pixel per Unit parameter.

In this example, we want a **1136 x 640** design resolution, and we want **1** Pixel per Unit. It is a convenient value as if you move a sprite from 10 world units it will move the sprite by 10 pixels on your design resolution.



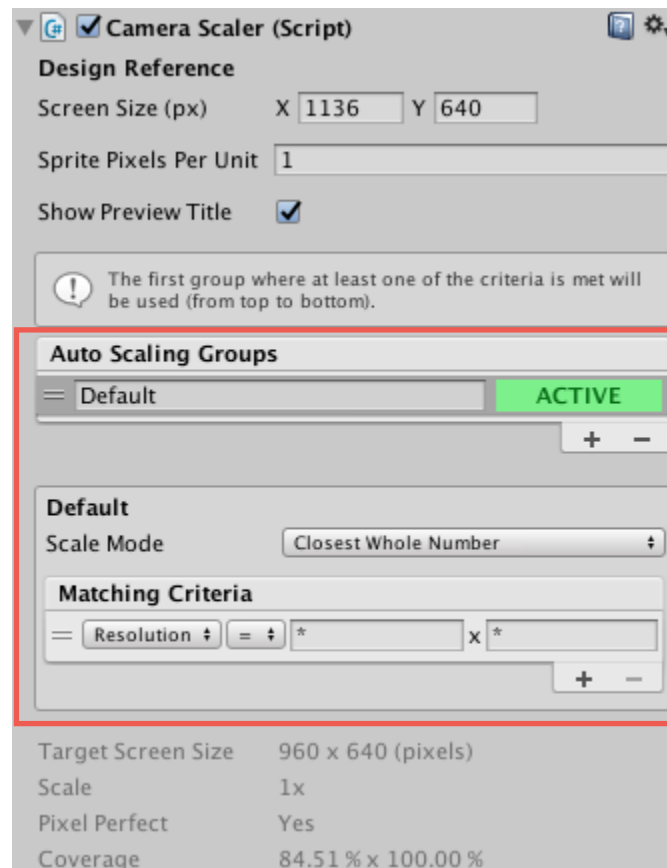
What is an Auto Scaling Group?

The CameraScaler component will resize the camera based on target resolution.

With an *Auto Scaling Group*, you can tell the CameraScaler what to do (i.e. how to scale) when running the game at a specific resolution, aspect or density.

You have an Auto Scaling Group by default, which tell the CameraScaler to use the *Closest Whole Number* scale mode on any resolutions.

It's up to you to add, remove, reorder, rename the auto scaling groups!

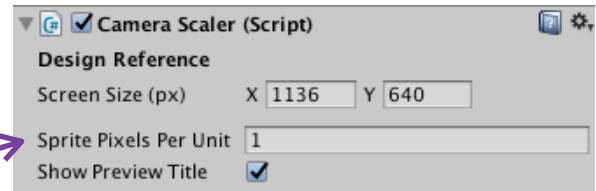
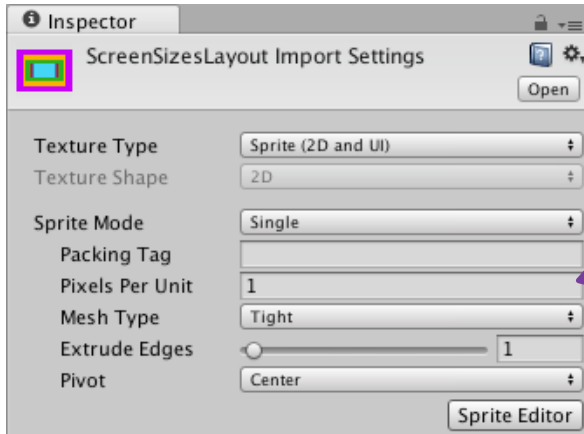


In the scene view you can see your design resolution camera bounds and, in same time, what the game will looks like on the target resolution selected (using the game window resolution). So you can test different scale mode to use the one that fit the best for your game.

See the [Auto Scaling Group](#) section for more details.

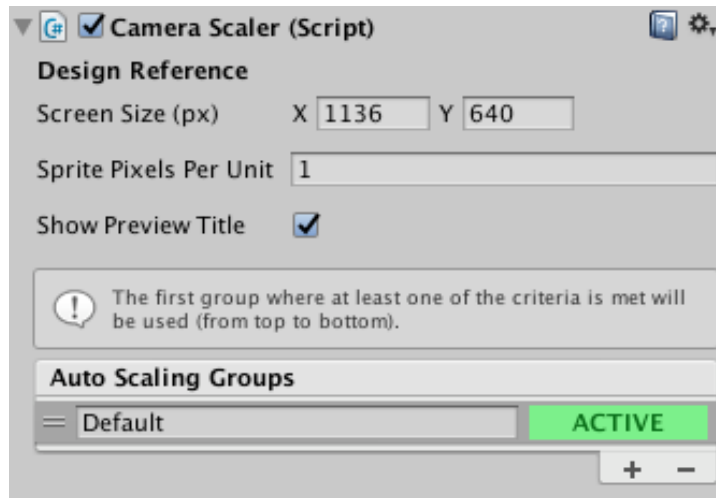
Add sprites

1. Add a sprite to your project and select it in the project window.
2. Set the *Pixels Per Unit* value to be **exactly** the same than the *Sprite Pixels Per Unit* value from the CameraScaler script.



Camera Scaler

The *Camera Scaler* component is used to resize the Unity orthographic camera, or in other terms: to scale your game at any resolutions with very little effort.



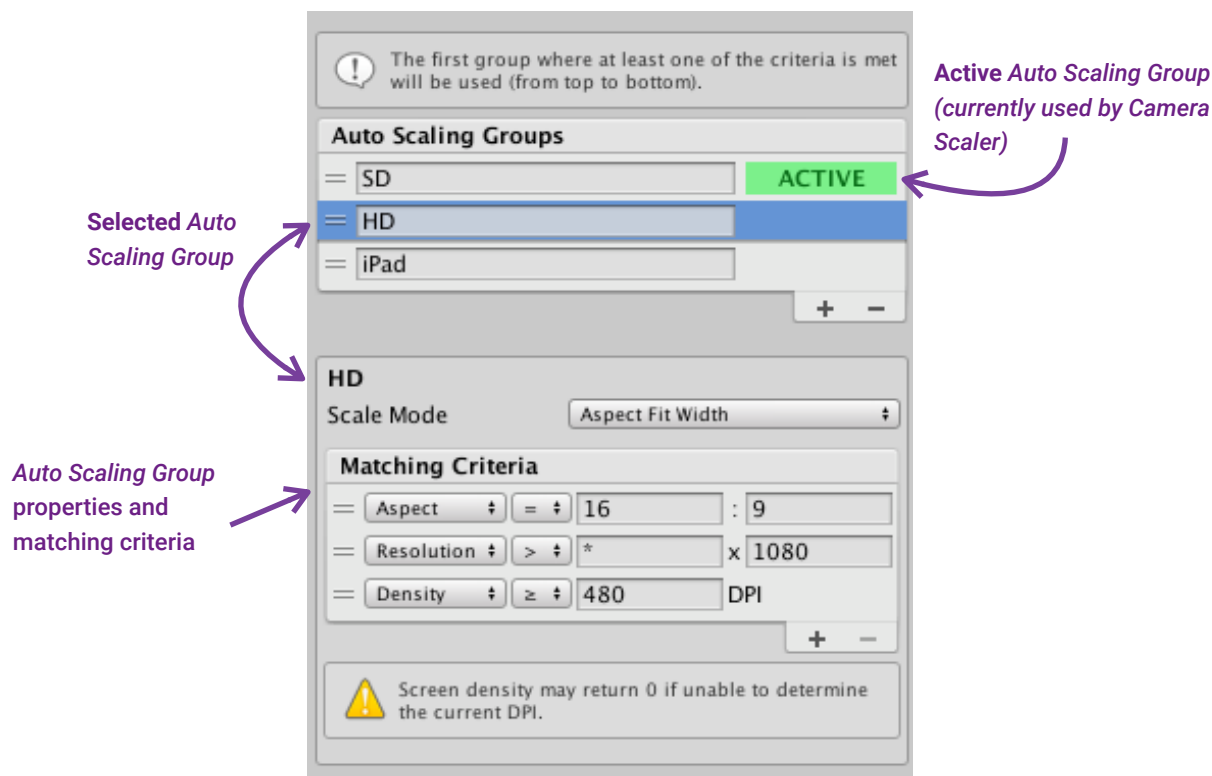
Properties

Property:	Function:
Design Reference - Screen Size (px)	The resolution in pixels the game is designed for (width x height).
Sprite Pixels Per Unit	How many pixels in your sprites correspond to one unit in the world. It is the equivalent of property Pixels Per Unit (aka PPU) on your sprite textures and it should have the same value.
Show Preview Title	Do you want show the title of <i>Target</i> and <i>Design</i> camera bounds in Scene view?
Auto Scaling Groups	You can create, remove, reorder and name an <i>Auto Scaling Group</i> .

Auto Scaling Group

The *Auto Scaling Group* defines how to scale your design resolution when it is running at different resolutions. The *Auto Scaling Group* is active if the current screen resolution matches at least one of the criteria.

The *Camera Scaler* use the first active *Auto Scaling Group* from his list to apply the scale mode defined (matching is done from top to bottom).



Properties

Property:	Function:
Name	The name of the auto scaling group (from <i>Camera Scaler</i> list)
Scale Mode	Determines how the camera is scaled.
Manual	Scale the camera using the scale factor provided.
Aspect Fit Width	Scale the camera to fit the width to the current resolution by maintaining the aspect ratio. Some portion of the design resolution height may be clipped.
Aspect Fit Height	Scale the camera to fit the height to the current resolution by maintaining the aspect ratio. Some portion of the design resolution width may be clipped.
Aspect Fit	Scale the camera to fit the design resolution to the current resolution by maintaining the aspect ratio.

Property:	Function:
Aspect Fill	Scale the camera to fill the current resolution with the design resolution by maintaining the aspect ratio. Some portion of the design resolution width may be clipped.
Closest Whole Number	Upscale the camera by the closest whole numbers (1x, 2x, ...) or Downscale the camera by the closest whole numbers fractions (1/2x, 1/3x, ...). Some portion of the design resolution may be clipped.
Pixel Perfect	Upscale the camera by the closest whole number (1x, 2x, 3x, ...). Some portion of the design resolution may be clipped.

Settings for Scale Mode - Manual:

Property:	Function:
Scale	The scale factor to use.

Details

For this we're going to consider 2 screen resolutions:

- Phone HD in landscape (960 x 640 pixels with a 3:2 aspect ratio)
- Tablet HD in landscape (2048 x 1536 pixels with a 4:3 aspect ratio)

The design screen size is the Phone HD Landscape resolution.

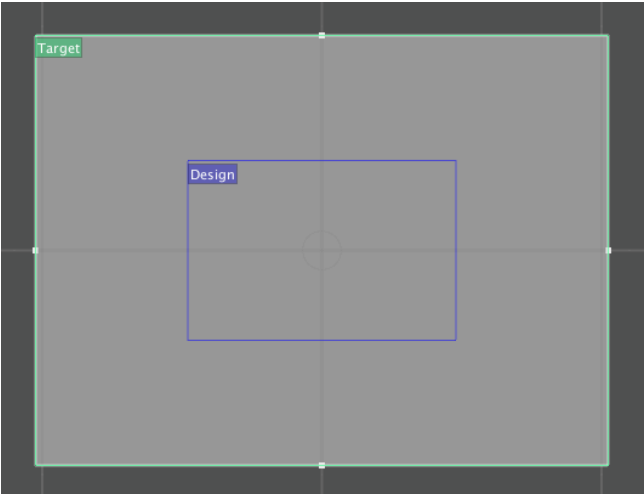
Design Reference
 Screen Size (px) X Y

The design camera bounds is in blue.

The target camera bounds (game window resolution) is in green.

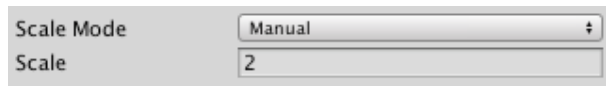
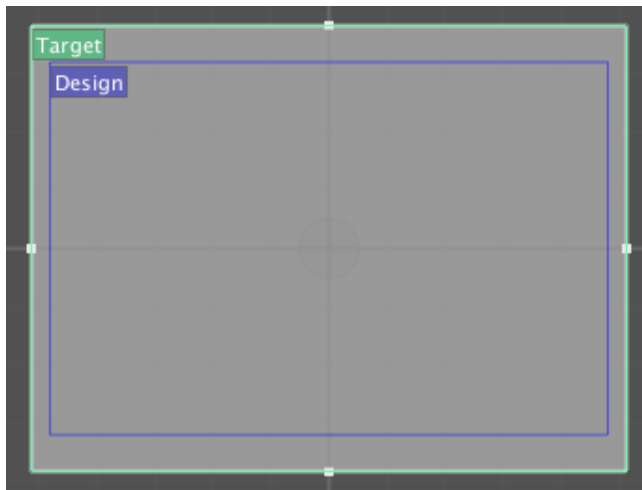
Scale Mode - Manual

In this mode, the Camera is scaled using the scale factor property given by the user.



Scale Mode
 Scale

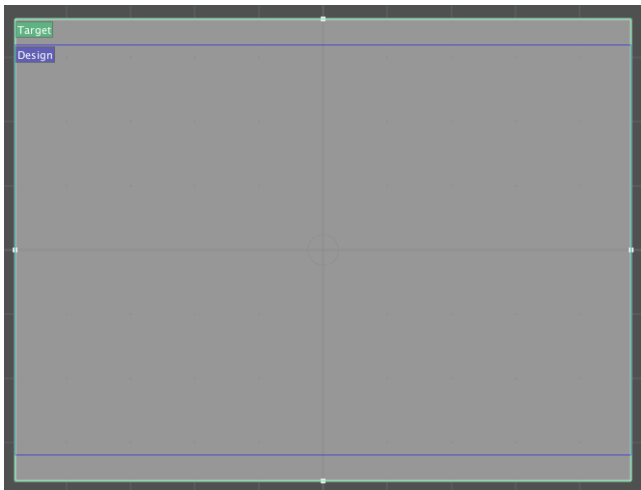
The target will render the design at 1x.
 There is a 1:1 pixel ratio (1 design pixel = 1 target pixel)
 It is pixel perfect.
 It is giving you a coverage of 214% x 240% of your design resolution.



The target will render the design at 2x.
There is a 1:2 pixel ratio (1 design pixel = 2 target pixel).
It is still pixel perfect as we scale up by a whole number.
It is giving you a coverage of 107% x 120% of your design resolution.

Scale Mode - Aspect Fit Width

In this mode, the Camera is scaled automatically to fit the target resolution width with the design resolution width.



Some portion of the design resolution height may be clipped.
It is not pixel perfect as we could not have a whole number factor.
It is giving you a coverage of 100% x 112% of your design resolution.

Scale Mode - Aspect Fit Height

In this mode, the Camera is scaled automatically to fit the target resolution height with the design resolution height.



Some portion of the design resolution width may be clipped.
It is not pixel perfect as we could not have a whole number factor.
It is giving you a coverage of 89% x 100% of your design resolution.

Scale Mode - Aspect Fit

In this mode, it will choose the best fit between *Scale Mode - Aspect Fit Width* and *Scale Mode - Aspect Fit Height* to fully render the design resolution in the target resolution.

Your design resolution will not be clipped but you may have more world space to fill.

In this example, it will choose the *Scale Mode - Aspect Fit Width*: the design resolution is 100% visible in the target screen.

Scale Mode - Aspect Fill

It is the exact opposite of *Scale Mode - Aspect Fit*.

In this mode, it will choose the best fit between *Scale Mode - Aspect Fit Width* and *Scale Mode - Aspect Fit Height* to fully fill the target resolution with the design resolution.

Your design resolution may be clipped but you will have your screen completely filled.

In this example, it will choose the *Scale Mode - Aspect Fit Height*: the target screen is completely filled but the design resolution is clipped.

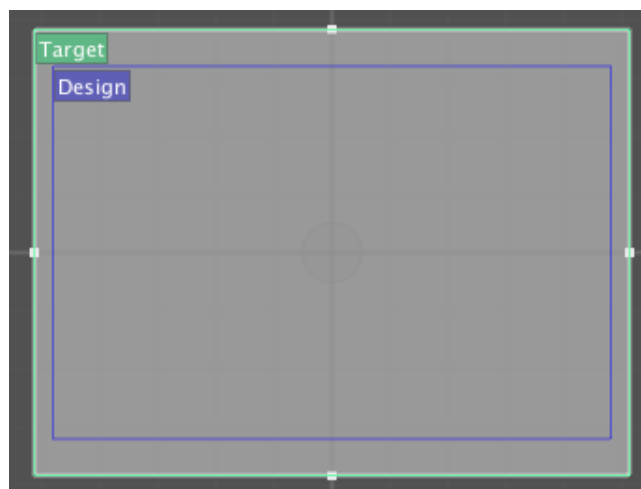
Scale Mode - Closest Whole Number

In this mode, it will scale the camera by a whole number to be as close as possible from the design resolution world space.

If the current resolution is bigger than the design resolution, it will **upscale** by a whole number: 1x, 2x, 3x, ...

If the current resolution is smaller than the design resolution, it will **downscale** by a fraction by a whole number: 1/2x, 1/3x, 1/4x, ...

To scale our design resolution (Phone HD) to our target resolution (Tablet HD), it will automatically apply a scale factor of 2.0x. With this scale factor we are closest from the original resolution with a coverage of 107% x 120% .



Another example: to scale our design resolution to a 480 x 320 pixels screen, it will automatically apply a scale factor of 1/2x (0.5x). With this scale factor we have a 100% x 100% coverage, **but** this is not pixel perfect as we can't divide original pixels by 0.5.

Scale Mode - Pixel Perfect

In this mode, it will scale the camera by a whole number to be as close as possible from the design resolution world space.

The difference from *Scale Mode - Closest Whole Number*, it is that it will only allow an upscale by a whole number (1x, 2x, 3x, ...) so you can be sure **you will always have a pixel perfect rendering**:

1 design pixel = 1, 2, 3 ... target pixels.

To use this mode, you should have your design resolution at 1x (your lowest possible target resolution).

If you want run your game on a smaller resolution, you can use *Scale Mode - Closest Whole Number*.

Matching Criteria

An *Auto Scaling Group* can only be active if there is at least one of the criteria that matches the current resolution, aspect ratio or density.

Accepted values are integers or the wildcard character (*) to means any values.

The screenshot shows a 'Matching Criteria' dialog box with three rows of criteria. Each row has a dropdown menu for the criterion type, a comparison operator, and input fields for values. The first row is 'Aspect' with operator '=' and values '16' and '9'. The second row is 'Resolution' with operator '>' and values '*' and '1080'. The third row is 'Density' with operator '≥' and values '480' and 'DPI'. There are '+' and '-' buttons at the bottom right.

There are 3 different criteria types:

- Resolution

Specify the resolution: **width x height** in pixels

Matching Criteria	Target screen	Result
Resolution = 2048 x 1536	1024 x 768	False
Resolution < 2048 x 1536	1024 x 768	True
Resolution = *	1024 x 768	Always True

- Aspect ratio

Specify the aspect ratio: **width : height**

Matching Criteria	Target screen	Result
Aspect = 4 : 3	4 : 3 (1:1.33)	True
Aspect < 4 : 3	16 : 9 (1:1.77)	False
Aspect = *	3 : 2 (1:1.5)	Always True

- Density

Specify the screen density in DPI.

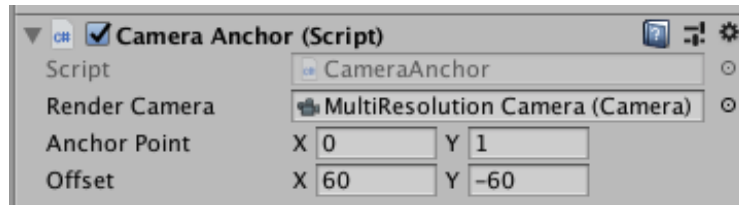
It will use the *Screen.dpi* from Unity to get the target screen value so it may return 0 if unable to determine the current DPI.

Matching Criteria	Target screen	Result
Density = 320 DPI	480dpi (xxhdpi)	False
Density < 320 DPI	240dpi (hdpi)	True
Density > *	640dpi (xxxhdpi)	Always True

Camera Anchor

The *Camera Anchor* component is used to anchor a *GameObject* position relative to a screen position. You can also specify an offset to this anchored position.

With this system, you can choose to always have a sprite at the lower left corner by example. The sprite will keep his position in any resolution.



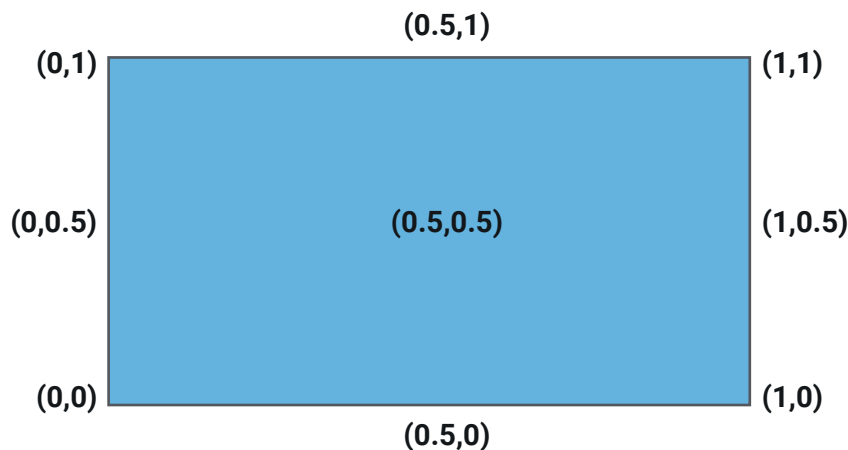
Properties

Property:	Function:
Render Camera	The camera that renders the screen rect.
Anchor Point	The anchor point defined as a fraction of the size of the screen rectangle. 0,0 corresponds to anchoring to the lower left corner of the screen rectangle, while 1,1 corresponds to anchoring to the upper right corner of the screen rectangle.
Offset	An offset in pixels from the anchor point.

You can create a *CameraAnchor* via the context menu in the hierarchy view.

The position of the *GameObject* will be updated to the anchored position with the offset scaled by the current scale of your *CameraScaler*.

Commons Anchors:



Hints

Pixel Art Game

To setup a projet for a pixel art game, use these tweaks:

On the Sprite texture:

- Texture Type: Sprite (2D and UI)
- Use a lossless compression e.g. True Color
- Generate Mip Maps: Turn off
- Filter Mode: Point (no filter)

In Render Quality Settings:

- Anisotropic Textures: Disabled
- Anti Aliasing: Disabled

(Optional) Create a custom material and set it up as follows:

- Shader: Use Sprites/Default
- Pixel snap: Turn on

Attach this material to the SpriteRenderer.

