MadGoat

Super Sampling

MadGoat SSAA & Resolution Scaling Documentation

Table of contents

1. INTRODUCTION

Thank you for purchasing our asset!

MadGoat Supersampling brings one of the best anti-aliasing techniques to your Unity3D project: **be it a game, a VR application or an archviz presentation, now you can achieve stunning image quality with just one click!**

What is SSAA?

SSAA, short from super sampling anti-aliasing (not to be confused with screen space anti aliasing) is one of the highest quality anti aliasing methods, since it works by rendering the whole image at a higher resolution and down sampling it to the screen resolution.

Then why don't you see it in many games?

SSAA works by directly changing the internal render resolution of your game, therefore the higher the setting is, the higher the performance hit is. Because of that, it uses more resources and needs more processing power than other anti-aliasing options, but it also looks better. This anti aliasing solution is great for high end GPUs where there is performance headroom left, and also for high resolution, sharp screenshots.

2. SSAA SETUP

2.1 CHOOSING THE RIGHT SSAA COMPONENT FOR YOUR PROJECT

MadGoat SuperSampling now contains 3 different SSAA components for your cameras:

- **MadGoatSSAA**
- **MadGoatSSAA_Adv**
- **MadGoatSSAA_VR**

MadGoatSSAA is made to be used at runtime, being perfect for use inside 3d/2d games and real-time archviz applications. The way it operates makes it non-intrusive to the editor and easy to configure at runtime.

MadGoatSSAA_Adv is an advanced version of the MadGoatSSAA, that also runs in edit mode, and allows usage of other scripts and components directly on the internal camera of the system, proving it useful for offline rendering systems and for better compatibility with more complex image effects and camera tools.

MadGoatSSAA_VR is the VR version of the MadGoatSSAA script. Unlike MadGoatSSAA, this script doesn't make use of render textures in order to bypass the render resolution of the camera. Instead it relies on Unity VR Device api for changing the device's resolution before applying the downsampling filters. This ensures the max performance for this use case.

NOTES

- MadGoatSSAA_Adv inherits the MadGoatSSAA script, resulting in all the API being the same for both.
- Using image effects or the post processing stack on the internal render camera is supported, however effects that rely on the depth buffer will not work as depth is not passed to the internal renderer.
- **When removing the MadGoatSSAA_Adv component from the camera, the RenderCameraObject child of the camera also has to be deleted to avoid issues.**
- **By design, MadGoatSSAA_VR affects the resolution scale of all the cameras in the scene. This is a drawback of using the VR device api that is available directly into Unity3D.**

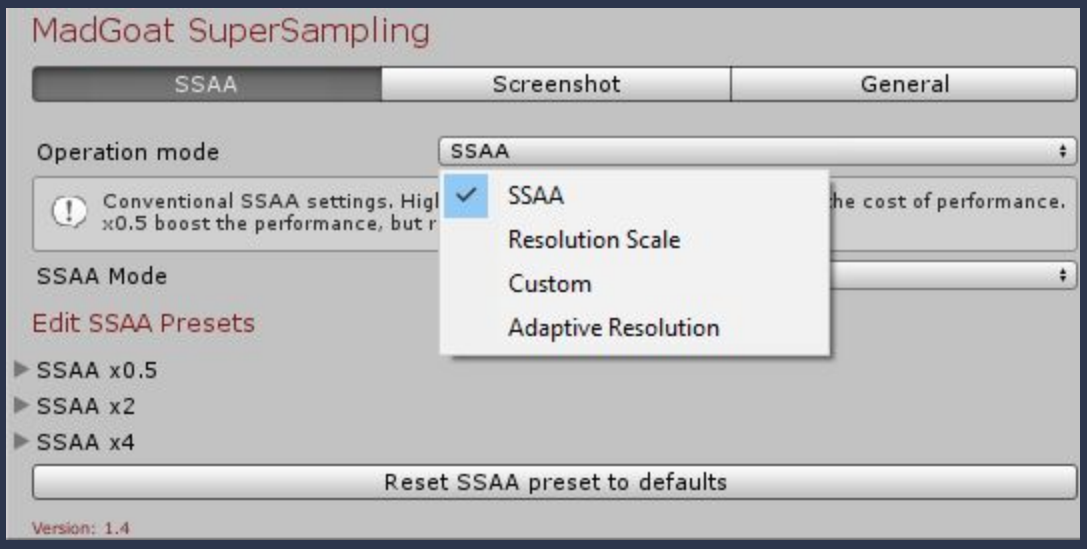
2.2 SETTING UP THE CAMERAS

MadGoat Supersampling is easy to integrate into your projects. Just drag and drop the desired SSAA script on your camera and you are good to go. If you have multiple cameras, you can add the script to each one of them and control them via the global c# api.

2.3 THE SETTINGS

There are 4 operation modes that can be chosen in the inspector:

- **SSAA:** Works by using the typical SSAA options (x2, x4 and x0.5)
- **Resolution scale:** Scales the current screen resolution based on a percent (from 50% to 200%)
- **Per axis scale:** Allows to setup a multiplier for both horizontal and vertical resolution values.
- **Custom setting:** Allows to setup a custom resolution scaling setting.
- **Adaptive Resolution:** Allows for automatic resolution adjustment based on a target frame rate.

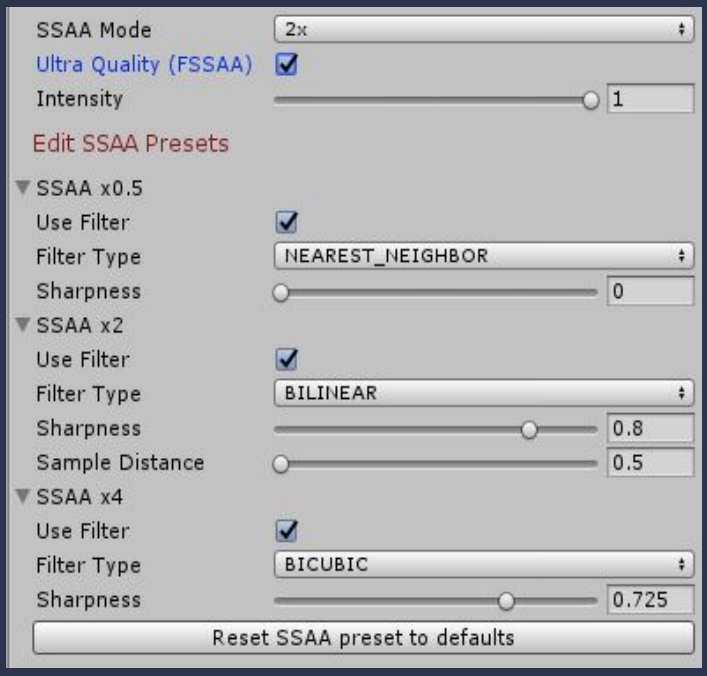


2.4 CUSTOMIZING EACH MODE

Every operation mode has its own set of settings. These settings can be very important in achieving the best quality image.

For the **SSAA mode**, each profile can be configured independently.

- **Use Filter:** If enabled the final rendered image will be passed through a custom down sampler.
- **Filter Type:** The type of filter used to down sample the image.
- **Sharpness:** The sharpness of the down sampled image.
- **Sample Distance:** Distance between the samples.



In addition to that, **Ultra Quality (FSSAA)** can be enabled. **Only available for SSAA x2, x4 and resolution multipliers over 1 on other modes.**

*FSSAA is inspired from the **Filtering Approaches for Real-Time Anti Aliasing** slides by **Timothy Lottes (NVIDIA)** and uses the FXAA v3 to filter the render image before resizing the frame buffer to screen resolution.*

The **Resolution Scaling**, **Custom Setting** and **Adaptive Resolution** modes also have these settings, but in addition they have their own specific settings:

Resolution Scale

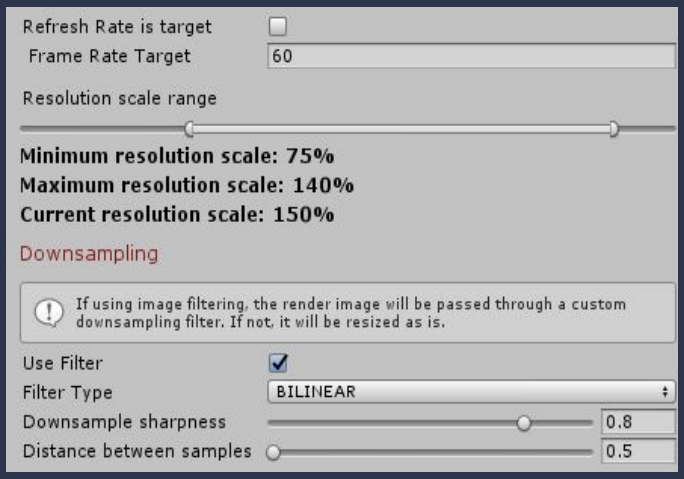
The resolution scale slider is used to set the size of the rendered image relative to the size of the screen. Options from 50% to 200% are available.

FSSAA is also available when scale is greater than 100%

Custom Setting & Per Axis Scale Modes

On the Custom Setting and Per Axis modes, the Resolution Multiplier is the value with which the screen resolution is multiplied (From 0.2f to 4f). If you choose “Don’t limit the multiplier”, you can set any values.

Be aware that multiplier values over 2 are not recommended.



Adaptive Resolution

If Refresh rate is target is enabled, the used target rate will be the screen refresh rate. Otherwise a custom Frame rate target can be set.

The resolution scale range represents the interval in which the resolution can be modified (by percent).

FSSAA is not available for this mode.

Notes

The Custom Setting mode can be “dangerous” when setting high multiplier values while the screen resolution is already high (1080p or more) and can result in memory leaks, engine crashes and even system hangs.

The adaptive resolution mode can cause frame rate stuttering at high resolutions/high multiplier due to the time needed for the system and engine to allocate the new render size. This is due to the fact that the render image used to store the internal screen buffer needs to be reallocated every time it’s dimension changes.

2.5 SSAA AND VIRTUAL REALITY

The settings of the SSAA for VR are identically in every aspect to the regular SSAA component's settings, the only difference between the components being in the Screenshot and General tabs.

When using MadGoat SuperSampling for VR, make sure the VR device is connected. Currently tested devices are Oculus Rift and HTC Vive, which should work on both single pass and multi pass rendering techniques.

Recommended settings for best VR performance are:

- SSAA 2X + FSSAA enabled.

Or

- Resolution scale 120% - 150% + FSSAA enabled.

However those are just some guidelines. Hardware performance, especially the GPU performance pays a big role when it comes to SSAA and VR, so the performance for the recommended settings may vary from system to system: some lower end systems might struggle to run it at acceptable frame rate, while high end systems might actually run great even at higher SSAA settings.

Reading Section 5 might also help in further optimizing the performance when using SSAA and virtual reality.

Note: Due to the implementation of VR compatibility, the screenshot module is unavailable in VR Mode. Not only the inspector tab regarding the screenshot module is disabled, but also the code API.

2.6 THE FILTER TYPES

Nearest Neighbor

The simplest and crudest filtering method — it simply uses the colors of the texel closest to the pixel center for the pixel colors. While simple to implement, this results in a large number of artifacts

When to use Nearest Neighbor: When looking to get a sharp image while rendering at a smaller resolution. Nearest Neighbor can also be used to achieve a “pixelated” image

Bilinear Filtering

In this method four texels in the neighborhood of the center pixel are sampled, and their colors are combined by weighted average. This removes the artifacts seen while rendering at higher resolution, as there is now a smooth gradient of color change from one texel to the next, instead of an abrupt jump between the texels.

When to use Bilinear Filtering: When looking to get good down sampling quality without a big performance hit. It can also be used when rendering at lower resolutions to create a “smoother” image.

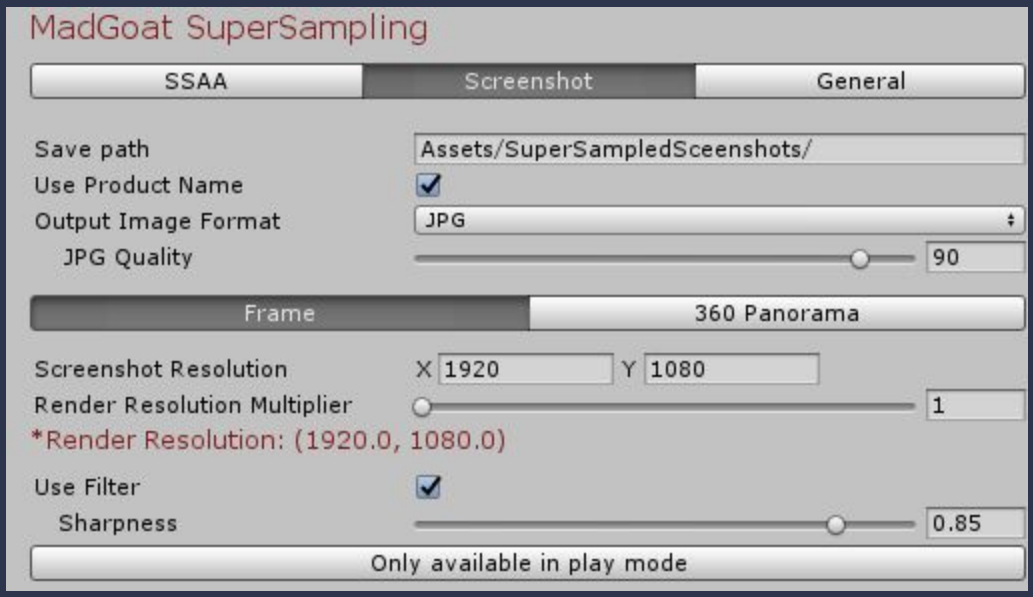
Bicubic Filtering

Bicubic filtering is often resulting in higher down sampling quality than bilinear filtering or nearest neighbor, but with the cost of slower computation. In contrast to bilinear filtering, which only takes 4 pixels into account, bicubic uses more samples in order to create a smooth curve based interpolation between the texels colors. Bicubic filtering also produces less image artifacts than the bilinear filtering.

When to use Bicubic Filtering: When the best quality is desired and the performance is not an issue.

3. THE SCREENSHOT MODULE

The screenshot module can take super sampled in-game screenshots at custom resolutions without the need to change the actual screen resolution and without affecting the aspect ratio.



- **Save Path:** The path where all the screenshots made by this plugin will be saved. The path is relative to the application's data path.

- **Use Product Name & File Name Prefix:** Only available in inspector. Prefix to be added to the file name. If the Use Product Name option is enabled, the prefix will be the product name set in the Player Settings. The structure of the file name is PREFIX_TIME_RESOLUTION.png

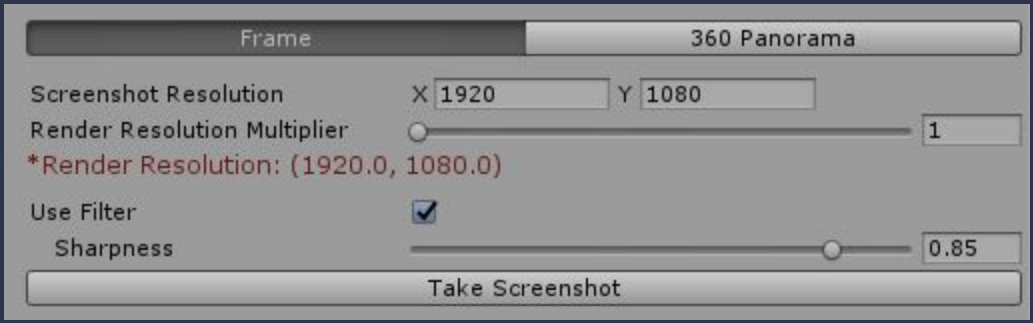
- **Output Image Format:** The format in which the image will be saved. Supported formats are png, jpg and exr (Unity 5.6 and newer)

There are 2 types of screenshots that can be taken using this plugin, Frame and 360 Panorama.

The Frame mode can be used to take a single frame screenshot (conventional screenshot) at any resolution and aspect ratio, regardless of the current screen resolution.

The 360 Panorama can be used to take "Spherical" pictures. This mode outputs 6 images representing sides of a cube map, that can be later composed into a single panorama image using third party software. The resulting images can also be used for setting up cubemaps inside unity (for VR archviz visualization etc.)

3.1 FRAME CAPTURE SETTINGS



- **Screenshot Resolution:** the target resolution of the screenshot to be taken.
- **Render Resolution Multiplier:** multiplier used for supersampling the image.
- **Use Filter:** if enabled the image will be passed through a shader based downsampling filter.
- **Sharpness:** Parameter of the downsampling shader. Represents the sharpness of the resulting image.

3.2 360 PANORAMA SETTINGS

- **Panorama Size:** The resolution of a side of the panorama cube map.
- **Panorama Multiplier:** The multiplier used for the supersampling of the panorama.
- **Use Filter:** If enabled the image will be passed through a shader based downsampling filter.
- **Sharpness:** Parameter of the downsampling shader. Represents the sharpness of the resulting image.

The button available in the inspector can be used to take a screenshot directly from the editor while in play mode. **When using MadGoatSSAA_Adv it is also possible to take screenshots while in edit mode.** The screenshot module is also available in-game, and screenshots can be easily taken by calling the built in methods which we'll discuss in the API section.

4. DRAWBACKS AND LIMITATIONS

The **MadGoat SSAA and Resolution Scale** system mostly relies on workarounds to expose and modify the engine's render resolution setting, which means the plugin has its drawbacks and limits:

- The SSAA System makes high use of RenderTextures, which means that the maximum resolution is limited to the maximum resolution the RenderTexture can hold (which sometimes is also limited by hardware) The maximum recommended internal resolution is 7680*4320 - 8K. However there have been cases when 16K worked as well.
- The VR Module makes use of VRDevice (or XRDevice on newer unity versions) API. This results in the inability to control the resolution of independent cameras. Even if the script is applied to only one camera, the render resolution will be the same for every camera.
- MadGoat SSAA doesn't currently support overlapping cameras (camera transparency) except for the VR module which doesn't use target render textures to bypass the resolution setting.
- 360 Panorama screenshot mode is currently limited at 8K resolution per each side. This is a limitation related to the engine's cubemap allocation.
- SSAA might highly affect performance when paired with complex post processing effects such as screen space reflections and ambient occlusion. It is recommended to run those with downsampling enabled when SSAA is enabled.

5. TROUBLESHOOTING, KNOWN ISSUES, TIPS AND TRICKS

Unity3D crashes when taking high resolution pictures.
- Happens when there is not enough video memory to allocate the needed resources for given resolution. Reduce the resolution or the multiplier in the screenshot module.

My Unity3D project uses too much video memory when running SSAA and crashes.
- Since SSAA changes the render resolution, it will also affect the mip level of the loaded textures in your scene, resulting in materials' textures to occupy more space in memory. To avoid this, you can manually tune the player's mip bias when changing SSAA settings. This can happen in complex scenes with high resolution textures (2048x2048, 4096x4096) when running on lower end platforms.

Some transparent objects such as transparent objects are rendered in incorrect positions.
- This is the result of a compatibility issue between SSAA and TAA. Makes sure that no temporal anti aliasing is turned on while SSAA is enabled.

SSAA enabled on iOS platforms causes black screen or screen glitching (depending on device and render API)
- This is a bug that will be fixed in the next update (Due to lack of testing hardware).

HDR post processing effects are not working.
- Make sure the Render Image Format in the general tab is set to a HDR ready format such as **ARGB Half**, **ARGB Float** or **Default HDR**.

There are shader / null reference errors in the error log
- Make sure that all of the shaders and shader includes have been correctly imported.

Pointer events such as OnMouseEnter(), OnMouseDown(), etc. are not triggering.
- This is a design flaw that is caused by the usage of RenderTextures. As an easy workaround, OnClick() Compatibility Mode can be enabled in the general tab, at the cost of lower overall performance. Alternatively, raycasting from the pointer position on screen can be used for pointer controls.

Raycasting and Camera Screenspace API have incorrect coordinates.
- The Screenspace coordinates of the camera are getting affected by the internal render resolution. MadGoatSSAA and MadGoatSSAA_Adv have their own wrapper methods for getting the correct screen positions **(further discussed in the Code Api section of this documentation)**

Dynamically adjusting the super sampling parameters such as the multiplier from another script causes stuttering and high performance loss.
- Changing the multiplier at runtime implies re-allocating the RenderTexture used by the internal renderer of the SSAA system. As this can be a time consuming operation for most computers, it can result in performance hiccups. It is not recommended to change the multiplier in short time periods.

Getting the best performance out of MadGoat SSAA
- In order to further optimize the performance while using SSAA, it is recommended to tweak the mipmap and the LOD biases accordingly, since the LOD and mip distances are directly affected by the render resolution. While the mipmap bias tweaking can help lowering the memory usage, tweaking the LOD bias will also lower the triangles and vertices count at higher resolutions without noticeable detail loss.

6. CODE API

This section describes the several functions that can be used to change the parameters of the SSAA component from your own code. Those functions can be used in-game, for example for your settings menu code.

In order to gain access to those functions, you first have to reference the SSAA component from the camera you want to change. For example:

```
MadGoatSSAA ssaaComponent = GetComponent<MadGoatSSAA>();
```

Alternatively, the global setting functions can be called directly from the MadGoat_SSAA class without referencing, but they will affect all the SSAA components found in the scene if more than one.

Take note that all of the asset's scripts are contained in the MadGoat_SSAA namespace. Add *using MadGoat_SSAA;* to your code.

6.1 THE API METHODS

1) SetAsSSAA(SSAAMode mode)

- Sets the rendering mode to a given SSAA mode. Available SSAA modes as defined in the enum are SSAA_OFF, SSAA_HALF, SSAA_X2 and SSAA_X4.

2) SetAsScale(int percent)

- Set the rendering mode to scale, and the resolution scale to a given percent. When using this function, the down sampling shader is disabled!

3) SetAsScale(int percent, Filter FilterType, float sharpnessfactor, float sampledist)

- Set the rendering mode to scale, and the resolution scale to a given percent. When using this function, the down sampling shader is enabled. The FilterType parameter can be NEAREST_NEIGHBOR, BILINEAR and BICUBIC. The sharpnessfactor and the sampledist parameters are FilterType dependent as stated before.

WARNING: While it is possible to set any value to the percent, avoid big values as the renderer can cause engine instability at high resolutions and even crashes

4) SetAsCustom(float Multiplier)

- Set the rendering mode to custom, and the resolution multiplier to a given number. When using this function, the down sampling shader is disabled!

5) SetAsCustom(float Multiplier, Filter FilterType, float sharpnessfactor, float sampledist)

- Set the rendering mode to custom, and the resolution multiplier to a given number. When using this function, the down sampling shader is enabled. As stated before, the SetAsScale function, the sharpnessfactor and sampledist parameters FilterType dependent, and they are clamped to correct values automatically.

WARNING: While it is possible to set any value to the Multiplier, avoid values greater than 4, as the renderer can cause engine instability at high resolutions and even crashes.

6) SetDownsamplingSettings(bool use)

- Set the downsampling shader parameters. The use parameter specifies whether or not the down sampling shader should be used. If use = true, default settings will be given to the shader. It is important to know that this function should be always called after calling any of the functions above, since they always override the downsampler settings.

7) SetDownsamplingSettings(Filter FilterType, float sharpnessfactor, float sampledist)

- Set the down sampling shader parameters. In this case the down sampling shader is always enabled. Values for the FilterType, sharpnessfactor and sampledist have to be given. As for the function above this function should be always called after calling any of the functions above.

8) SetAsAdaptive(float minMultiplier, float maxMultiplier, int targetFramerate)

- Set the operation mode to adaptive, with a given targetFramerate. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set.

9) SetAsAdaptive(float minMultiplier, float maxMultiplier)

- Set the operation mode to adaptive. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set. Target frame rate is defaulted to the refresh rate.

10) SetAsAdaptive(float minMultiplier, float maxMultiplier, int targetFramerate, Filter FilterType, float sharpnessfactor, float sampledist)

- Set the operation mode to adaptive, with a given targetFramerate. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set. Downsampling filter is set to enabled and the conventional filter settings are available as parameters.

11) SetAsAdaptive(float minMultiplier, float maxMultiplier, Filter FilterType, float sharpnessfactor, *float* sampledist)

- Set the operation mode to adaptive. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set. Target framerate is defaulted to the refresh rate. Downsampling filter is set to enabled and the conventional filter settings are available as parameters.

12) SetUltra(bool enabled)

- Enable or disable the Ultra Quality (FSSAA)

6.2 THE GLOBAL FUNCTIONS (STATIC)

The global functions can be used to change the settings to all the SSAA components in the scene

1) SetAllAsSSAA(SSAAMode mode)

- Sets the rendering mode to a given SSAA mode. Available SSAA modes as defined in the enum are SSAA_OFF, SSAA_HALF, SSAA_X2 and SSAA_X4.

2) SetAllAsScale(int percent)

- Set the rendering mode to scale, and the resolution scale to a given percent. When using this function, the down sampling shader is disabled!

3) SetAllAsScale(int percent, Filter FilterType, float sharpnessfactor, float sampledist)

- Set the rendering mode to scale, and the resolution scale to a given percent. When using this function, the down sampling shader is enabled. The FilterType parameter can be NEAREST_NEIGHBOR, BILINEAR and BICUBIC. The sharpnessfactor and the sampledist parameters are FilterType dependent as stated before.

4) SetAllAsCustom(float Multiplier)

- Set the rendering mode to custom, and the resolution multiplier to a given number. When using this function, the down sampling shader is disabled!

5) SetAllAsCustom(float Multiplier, Filter FilterType, float sharpnessfactor, float sampledist)

- Set the rendering mode to custom, and the resolution multiplier to a given number. When using this function, the down sampling shader is enabled. As stated before, the SetAsScale function, the sharpnessfactor and sampledist parameters FilterType dependent, and they are clamped to correct values automatically.

6) SetAllDownsamplingSettings(bool use)

- Set the downsampling shader parameters. The use parameter specifies whether or not the down sampling shader should be used. If use = true, default settings will be given to the shader. It is important to know that this function should be always called after calling any of the functions above, since they always override the downsampler settings.

7) SetAllDownsamplingSettings(Filter FilterType, float sharpnessfactor, float sampledist)

- Set the down sampling shader parameters. In this case the down sampling shader is always enabled. Values for the FilterType, sharpnessfactor and sampledist have to be given. As for the function above this function should be always called after calling any of the functions above.

8) SetAllAsAdaptive(float minMultiplier, float maxMultiplier, int targetFramerate)

- Set the operation mode to adaptive, with a given targetFramerate. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set.

9) SetAllAsAdaptive(float minMultiplier, float maxMultiplier)

- Set the operation mode to adaptive. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set. Target frame rate is defaulted to the refresh rate.

10) SetAllAsAdaptive(float minMultiplier, float maxMultiplier, int targetFramerate, Filter FilterType, float sharpnessfactor, float sampledist)

- Set the operation mode to adaptive, with a given targetFramerate. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set. Downsampling filter is set to enabled and the conventional filter settings are available as parameters.

11) SetAllAsAdaptive(float minMultiplier, float maxMultiplier, Filter FilterType, float sharpnessfactor, float sampledist)

- Set the operation mode to adaptive. minMultiplier and maxMultiplier parameters represent the interval in which the resolution can be automatically set. Target framerate is defaulted to the refresh rate. Downsampling filter is set to enabled and the conventional filter settings are available as parameters.

12) SetAllUltra(bool enabled)

- Enable or disable the Ultra Quality (FSSAA)

12) SetUltraIntensity(float intensity)

- Set the intensity of the SSAA ultra effect (FSSAA intensity)

6.3 THE SCREENSHOT MODULE FUNCTIONS

---

1) TakeScreenshot(string path, Vector2 Size, int multiplier)

- Take a screenshot of resolution Size (x is width; y is height) rendered at a higher resolution given by the multiplier. The screenshot is saved at the given path in the currently set image format. Path is relative to Application data path.

2) TakeScreenshot(string path, Vector2 Size, int multiplier, float sharpness)

- Take a screenshot of resolution Size (x is width, y is height) rendered at a higher resolution given by the multiplier using bicubic filtering for downsampling, with a given sharpness. The screenshot is saved at the given path in the currently set image format. Path is relative to Application data path.

3) TakePanorama(string path, int size)

- Take a 360-panorama picture of resolution size x size. The panorama is saved at the given path in the currently set image format. Path is relative to Application data path.

4) TakePanorama(string path, int size, int multiplier)

- Take a 360-panorama picture of resolution size x size super sampled by the value of multiplier parameter. The panorama is saved at the given path in the currently set image format. Path is relative to Application data path.

5) TakePanorama(string path, int size, int multiplier, float sharpness)

- Same as above, but use the bicubic down sampling filter.

6) SetScreenshotModuleToPNG()

- Sets up the screenshot module to use the PNG image format. This enables transparency in output images.

7) SetScreenshotModuleToJPG(int quality)

- Sets up the screenshot module to use the JPG image format. Quality is parameter from 1 to 100 and represents the compression quality of the JPG file. Incorrect quality values will be clamped.

8) SetScreenshotModuleToEXR(bool EXR32)

- Sets up the screenshot module to use the EXR image format. The EXR32 bool parameter dictates whether to use or not 32 bit exr encoding. This method is only available in Unity 5.6 and newer.

The screenshot module API is not available in VR mode.

6.4 THE MISC FUNCTIONS

1) string GetResolution()

- Returns the internal render resolution of the plugin as string.

The following functions are wrappers for camera functions, that need to be used in order to get correct results since they depend on the camera render resolution.

2) Ray ScreenPointToRay(Vector3 position)

- Returns a ray from a given screen point

3) Vector3 ScreenToViewportPoint(Vector3 position)

- Transforms position from screen space into viewport space

4) Vector3 ScreenToWorldPoint(Vector3 position)

- Transforms position from screen space into world space

5) Vector3 WorldToScreenPoint(Vector3 position)

- Transforms position from world space to screen space

6) Vector3 ViewportToScreenPoint(Vector3 position)

- Transforms position from viewport space to screen space

Usage Example:

```
// Without Wrapper function
// ---------------
// Using ScreenPointToRay when the render resolution of the camera is different
// than the screen resolution results in offsetted mouse position.
Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
if (Physics.Raycast(ray, out hitInfo, 1000))
    Debug.Log("Offsetted Hit Info: " + hitInfo.point);

// With Wrapper function
// ------------
// MadGoatSSAA contains its own ScreenPointToRay method inside it's public api, using it instead
// of the camera's built in one will return correct ray position relative to the screen size
Ray ray2 = Camera.main.GetComponent<MadGoatSSAA>().ScreenPointToRay(Input.mousePosition);
if (Physics.Raycast(ray2, out hitInfo, 1000))
    Debug.Log("Correct Hit Info: " + hitInfo.point);
```

7. CONTACT

For any issue or bug report regarding the asset, please contact us at:

Support Email: madgoatstudio@gmail.com

Other contact methods:

Website: www.madgoat-studio.com

Facebook: www.facebook.com/madgoatstudio/

Twitter: www.twitter.com/MadGoatGames/