ENTROπ GAMES

GPU BASED LINE OF SIGHT VISUALIZATION

1. PREFACE

A line of sight system visualizes which parts of the game world can be seen from the standpoint of for example a 3rd person in-game character. The area outside of the line of sight is obscured and remains hidden from the player.

This system was developed as an alternative to CPU based line of sight visualizations. CPU based systems use a combination of ray casting and dynamic meshes to compute the line of sight and are often CPU intensive, resulting in a serious impact on performance.

This new GPU based line of sight system uses a technique very similar to shadow mapping and is significantly faster than any CPU based system, freeing up valuable CPU time. It has been designed from the ground up with performance, ease-of-use and customizability in mind. The effect can be tweaked to achieve any desired visual style. The user can control the look of the effect by selecting which image effects are applied to the area outside of the line of sight. It is perfectly suited for top-down stealth or action games, but also works from other viewpoints and in other type of games.

This system is compatible with both the forward and deferred rendering pipelines and supports DX11. Because this system makes heavy use of render textures, it is currently not supported on mobile platforms.

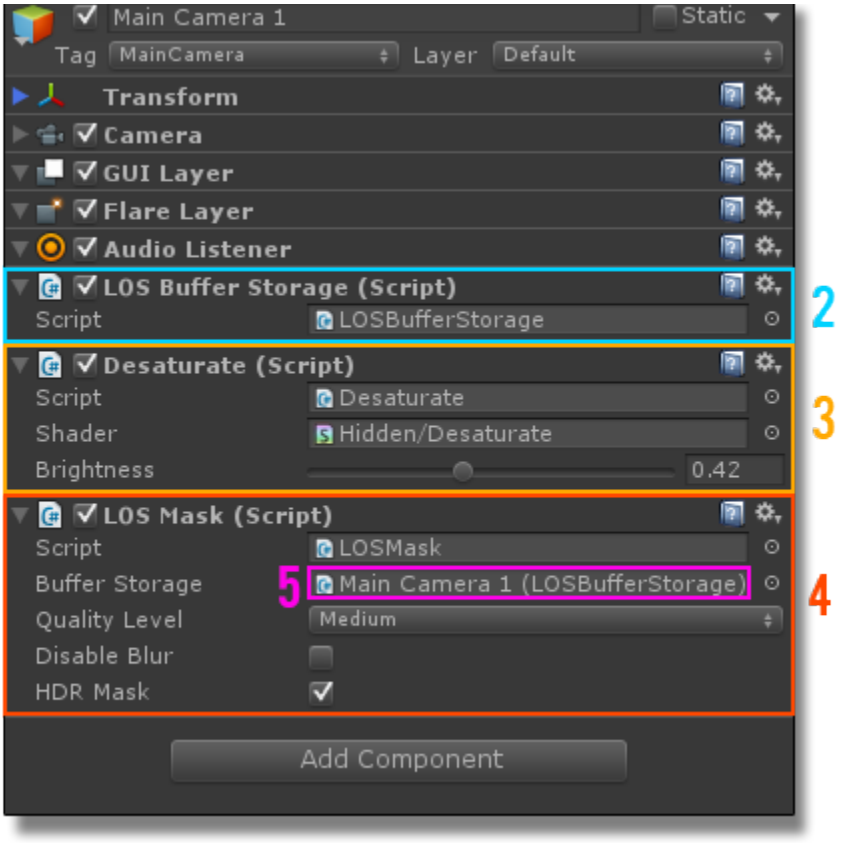
2. HOW-TO

2.1 QUICK START

TL;DR? Don't have time to read the whole manual? This step-by-step guide will teach you the basics in setting up the line of sight system and will get you started in no time!

2.1.1 SETTING UP THE MAIN CAMERA

1. Select the *Main Camera* in your scene, it can be either perspective or orthographic
2. Add a *LOS Buffer Storage* component to the camera's *GameObject* (Add Component > Line of Sight > LOS Buffer Storage)
3. Add the image effects you want applied to the area outside of the line of sight (ex. the included Desaturate effect)
4. Add the *LOS Mask* component (Add Component > Line of Sight > LOS Mask)
5. Assign the previously added *LOS Buffer Storage* component to the *Buffer Storage* property in the *LOS Mask* component



The order in which the LOS components are placed is very important. The *LOS Buffer Storage* Component should always come before (above in the inspector view) the image effects you want applied to the area outside of the line of sight. The *LOS Mask* component should always come after (below in the inspector view) these image effects.

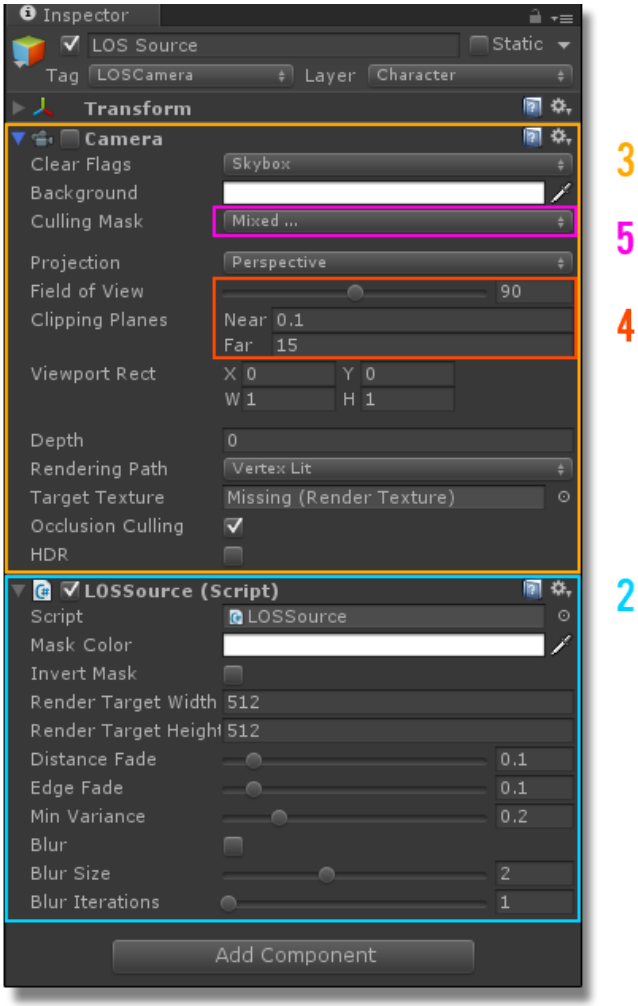
The image effects you add in between the *LOS Buffer Storage* and *LOS Mask* components only influence the area outside of the line of sight. If you want to apply image effects to the whole screen (ex. bloom or SSAO), you can always add them after the *LOS Mask* component.

The *LOS Buffer Storage* component doesn't necessarily have to be attached to the same *GameObject* as the *LOS Mask*, but it has to be rendered **before** the *LOS Mask*. This allows you more freedom in customizing the effects look, and doesn't limit you to using Image Effects only.
You could for example render the scene with 2 separate cameras; the first camera (ex. depth 0) has *the LOS Buffer Storage* attached and renders a subset of the scene, the second camera (ex. depth 1) renders another subset of the scene and uses the *LOS Mask* to combine the image rendered by both cameras.

2.1.2 ADDING A LINE OF SIGHT SOURCE

A line of sight source reveals parts of the world.

1. Create a new Empty *GameObject* (GameObject > Create Empty)
2. Add a *LOS Source* Component (Add Component > Line of Sight > LOS Source)
3. The previous step should have added a new *Camera* component to the same *GameObject* automatically
4. Adjust this cameras *far clipping plane* and *FOV* to tweak the area covered by the line of sight source
5. If you're source camera is inside an object, make sure to exclude that object using the source cameras culling mask properties



The effect should now be visible in the game tab or when you start the game. Make sure the camera attached to the LOS source *GameObject* is visible to the main camera. For a 360 line of sight visualization, it's recommended to add a *LOS Source Cube* component instead of the *LOS Source* component. You can also use one of the prefab LOS sources (located in the *Line Of Sight/Prefabs* folder) instead.

2.2 EXCLUDING OBJECTS FROM THE LINE OF SIGHT MASK

Excluding a set of objects from the line of sight mask requires using multiple cameras, each with different *Culling Mask* properties. The first camera will render the objects that need to be affected by the line of sight mask and afterwards a second camera (with identical settings) renders the remaining objects on top.

The *LOS Layer Excluder* script component sets up the multiple cameras for you, so you don't have to do it manually.

Please follow these steps to setup multiple cameras using the *LOS Layer Excluder* script:

1. If you haven't already, setup the main camera (adding the components needed for the line of sight system to work) following the steps in the quick start guide
2. Select the main camera
3. Add a *LOS Layer Excluder* script component to the main camera's *GameObject* *(Add Component > Line of Sight > LOS Layer Excluder)*
4. Set the layers you want to exclude from the line of sight mask using the *Exclude Layers* property in the *LOS Layer Excluder* script component
5. This will **automatically** remove the selected layers from the main cameras *Culling Mask* property (but if you **remove** a layer from the *Exclude Layers* property, you have to include it again manually in the main cameras *Culling Mask*)

The LOS Layer Excluder script will create a child object with a second camera component. This camera won't be deleted and can be used to apply Image Effects.

2.2.1 UNITY BUG WARNING 1

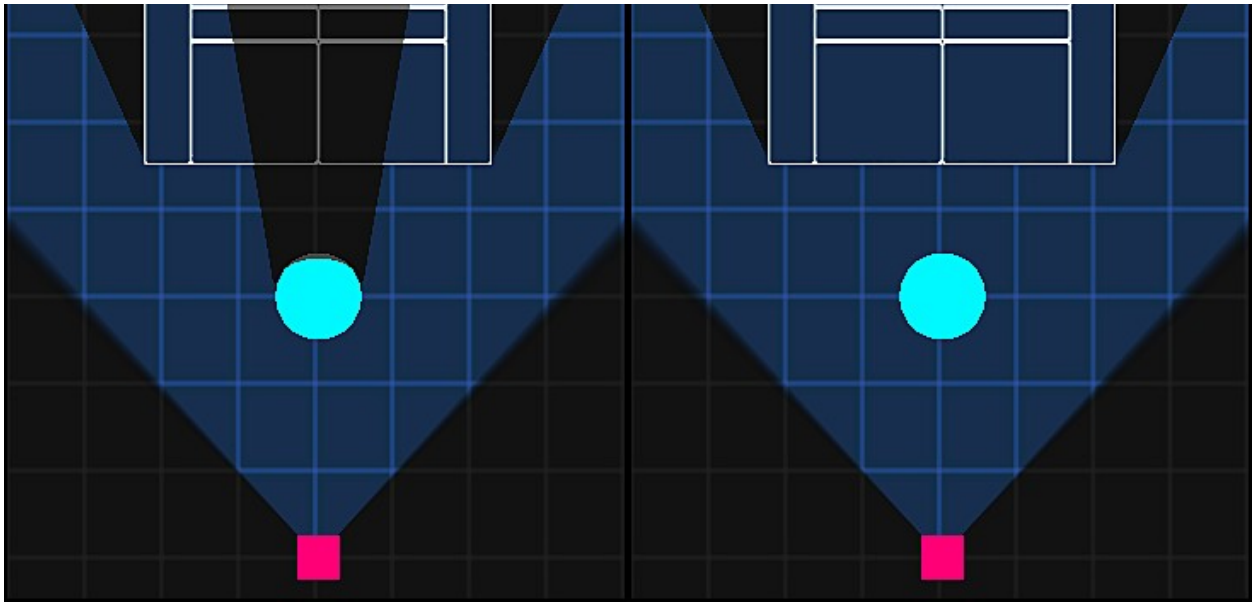
When using multiple cameras together with forward or deferred rendering, there is a bug in unity which makes the second camera (ex. depth 1) sometimes render incorrectly if it's *Clear Flags* property is set to Don't clear and if there's an image effect applied to the first camera (ex. depth 0).
There are 2 ways to fix this problem:

- Set the *Clear Flags* property of the **first camera** (depth 0) to *Depth only*
- Enable anti-aliasing in the quality settings.

2.2.2 UNITY BUG WARNING 2

Due to changes made to the **deferred renderer** in Unity 5 and new bugs that were introduced with these changes, it's currently not possible to use the *LOS Layer Excluder* script in combination with **deferred rendering in Unity 5**! I'm still looking for an alternative solutions and filed an issue tracker with Unity regarding this problem.

2.3 PREVENTING OBJECTS FROM AFFECTING LINE OF SIGHT



In some cases you will want objects to be obscured by the line of sight, but not affect the line of sight itself. This can be achieved by assigning the object to a layer and then excluding that layer from the *Culling Mask* property in the Camera component linked to the *LOS Source*.
Have a look at the "NPCs" in the example scene for an example of how to do this.

2.4 HOW TO HIDE OBJECTS WHEN THEY'RE OUTSIDE THE LINE OF SIGHT?

Even though objects will be obscured when they're outside of the line of sight (depending on the line of sight image effects you've setup), in some cases you'll want to hide them completely (ex. to not give away their position to your player). To do this, just add the *LOS Object Hider* component to the *GameObject* containing the objects *Mesh* or *Skinned Mesh Renderer*. This component will enable or disable this objects renderer depending on if the object is visible or not to any of the LOS sources in the scene.

3. COMPONENT OVERVIEW

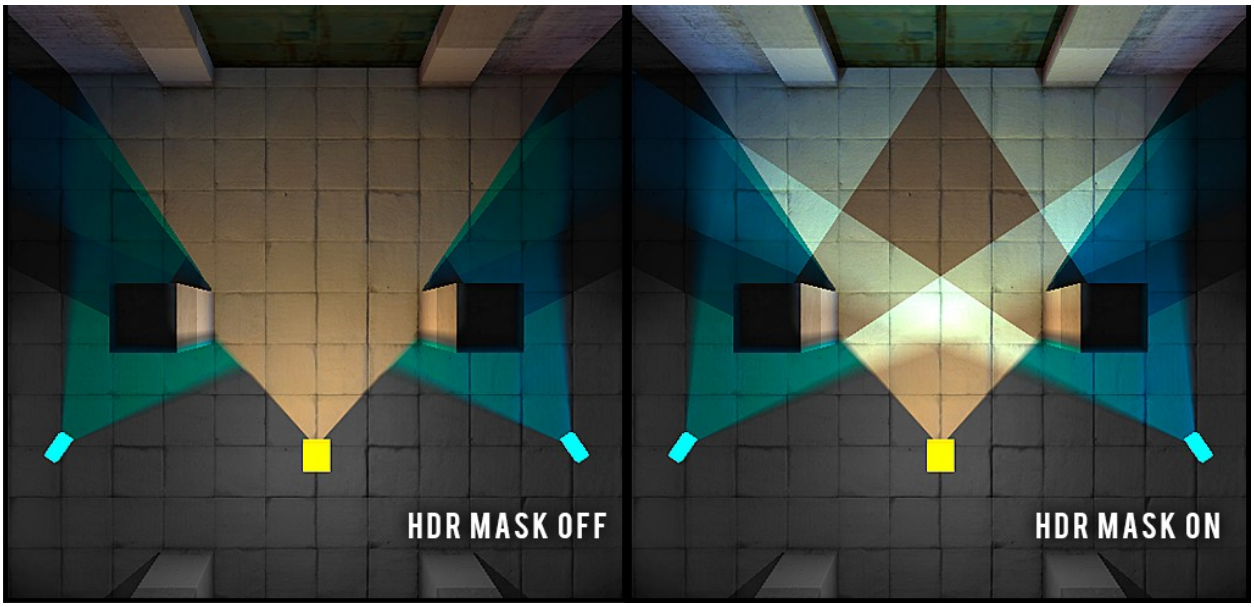
3.1 LOS MASK

This component is the heart of the line of sight system. It renders the line of sight mask and uses it to create the final image. This component is implemented as an image effect and therefore can only be added to a *GameObject* containing a camera component.

3.1.1 SETTINGS

- Buffer Storage: keeps a reference to the *LOS Buffer Storage* component used by this *LOS Mask*
- Quality Level: Sets the level of quality
    - High: doubles the resolution of all LOS source render textures
    - Medium: default, no changes
    - Low: halves the resolution of the line of sight mask buffers
- Disable blur: Global switch to disable all blur, overriding blur setting on LOS source components
- HDR Mask: Toggles HDR rendering of this mask. With HDR enabled the mask will be rendered into a **ARGBHalf** format render texture instead of **ARGB32**. To achieve the most accurate results I recommend using **Linear** *Color Space* instead of Gamma for your project.

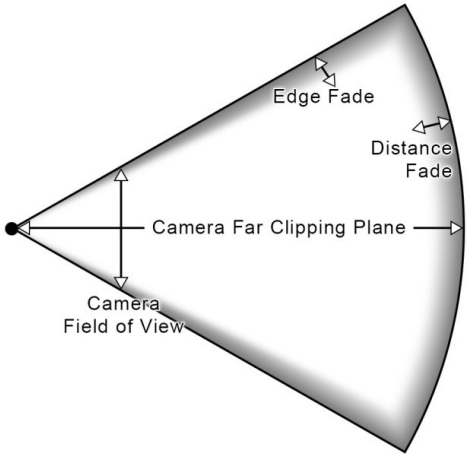
The image below illustrates the difference between HDR Mask off and on.



3.2 LOS SOURCE

This component should be added to each camera which is used as a line of sight source. The LOS source component has several settings which allow you to tweak the visual appearance of the line of sight cone.

3.2.1 SETTINGS



- **Mask Color**: sets the color
- **Mask Intensity**: sets the intensity (most noticeable in HDR mode)
- **Mask Invert**: inverts the mask
- **Render Target Width**: controls the width of this sources render target
- **Render Target Height**: controls the height of this sources render target
- **Distance fade**: controls how much the line of sight cone fades over distance
- **Edge Fade**: controls how soft the left and right edges of the cone are
- **Min Variance**: helps reduce artifacts in mask, best kept as low as possible
- **Out Of Bound Area**: controls how pixels above and below the cameras frustum are handled Please refer to section 5.1 for more details.
    - **Clamp:** uses the clamped depth value to calculate visibility
    - **Include:** area always visible
    - **Exclude:** area never visible
- **Blur**: blurs the mask created by this LOS source (experimental)
- **Blur Size**: controls the size (strength) of the blur
- **Blur Iterations**: controls the quality of the blur, more iterations impacts performance

Other settings like the length and width of the line of sight cone are controlled by the camera directly. The cameras *far clipping plane* controls the length of the cone, while the *Field of View* (together with the LOS source resolution settings) controls the width of the cone.

Keep in mind that in Unity the FOV controls the height of the camera and not the width. This component has a custom gizmo which displays the correct camera frustum in the same color as the *Mask Color* setting. The frustum of the camera displayed in the editor (by the camera component) is not correct because it assumes it's render target is the same resolution as the resolution set in the game mode.

3.2.2 PERFORMANCE WARNING

Having several different resolutions for your *LOS Source* components will increase the video memory needed. Try to use the **same resolution** for as many LOS sources as possible, so they can share the same render texture as their render target.

3.3 LOS BUFFER STORAGE

This component is used to store the screen buffer before rendering the line of sight image effects. The screen buffer stored by this component is used by the *LOS Mask* Component to create the final image. This component doesn't require any settings.

3.4 LOS LAYER EXCLUDER

This component is used to exclude layers from the line of sight. Please refer to section 2.2 for more info on how to set it up.

3.4.1 SETTINGS

- **Exclude Layers**: select which layers to exclude from the line of sight

3.5 LOS FINAL RESOLVE

This component should only be used when using **deferred rendering** and multiple cameras to exclude a set of objects from the line of sight mask (see section 2.2 *Excluding Objects from the line of sight mask*). It is used to fix a bug in Unity when using the deferred rendering pipeline in combination with multiple cameras (rendering to same render target) and image effects. Please refer to this Unity forum post for more details: *http://forum.unity3d.com/threads/deferred-renderer-multiple-cameras-posts-ssao-bugged-ufps-onrenderimage.198632/*

3.6 LOS CULLER

This component checks if the *GameObject* it is attached to, is visible to any of the LOS Sources. You can check if this object is visible by retrieving the *Visible* property. This component is used by the *LOS Object Hider* and *LOS Object Revealer* components to hide object outside of the line of sight area.

This component will first check if the mesh bounds of the attached Renderer component fall inside one of the LOS sources camera frustum. If the bounds are inside a camera frustum, it will check if another object blocks the line of sight using raycasts. Because this component uses raycasts, the objects that need to block it require a collider.

3.6.1 SETTINGS

- **Raycast Layer Mask**: select which layers block raycasts used for visibility calculations

3.6.2 WARNING

This component must be attached to a *GameObject* containing a *Mesh* or *Skinned Mesh Renderer*, because it uses the mesh bounds to calculate its visibility.

3.7 LOS VISIBILITY INFO

This component provides a list of all the LOS sources the *GameObject* (it is attached to) is visible to. You can check which LOS sources the object is visible to by retrieving the *VisibleSources* property.
This component also exposes several C# events you can subscribe to, which are triggered when the *GameObject* enters, exits, or stays inside a LOS sources field of vision.

This component use the same functions to calculate its visibility as the *LOS Culler* component.

3.7.1 SETTINGS

- **Raycast Layer Mask**: select which layers block raycasts used for visibility calculations

3.7.2 EVENTS

- **OnLineOfSightEnter**: Triggered when object enters LOS source field of view
- **OnLineOfSightStay**: Triggered when object remains inside LOS source field of view
- **OnLineOfSightExit**: Triggered when object exits LOS source field of view

3.7.3 WARNING

This component must be attached to a *GameObject* containing a *Mesh* or *Skinned Mesh Renderer*, because it uses the mesh bounds to calculate its visibility.

3.7.4 PERFORMANCE WARNING

This component calculates the objects visibility in relation to every LOS source in the scene, so it can be very resource intensive if you have a lot of LOS sources inside your scene.

3.8 LOS OBJECT HIDER

This component is used to hide objects outside the line of sight. It simply disables the *GameObjects* renderer, when the *LOS Culler* components *Visibility* property is set to false.
This component requires the *LOS Culler* component to be attached to the same *GameObject*. When you add this component, it will be added automatically if it doesn't already exists.

3.8.1 WARNING

This component must be attached to a *GameObject* containing a *Mesh* or *Skinned Mesh Renderer*.

3.9 LOS OBJECT REVEALER

This component is used to hide objects until they are revealed by the line of sight. Once revealed the objects will remain visible.

3.9.1 WARNING

This component must be attached to a *GameObject* containing a *Mesh* or *Skinned Mesh Renderer*.

4. DEPRECATED COMPONENTS

4.1 LOS CAMERA SYNC (DEPRECATED IN VERSION 1.1)

Deprecated in version 1.1 in favor of the new *LOS Layer Excluder* script. Removed in version 1.2.

This component should only be used when using multiple cameras to exclude a set of objects from the line of sight mask (see Excluding Objects from the line of sight mask). It syncs the properties from the main camera to the secondary camera containing this script component, so that both cameras render the scene in the exact same way. This component also sets the Clear Flags property to Don't Clear (The secondary camera shouldn't clear what the first camera rendered and should use the same depth buffer) and makes sure the Depth of the second camera is setup correctly, so that it renders after the main camera.

4.2 LOS SOURCE CUBE (DEPRECATED IN VERSION 1.2.2)

Deprecated in version 1.2.2 because of compatibility issues on Mac platforms

This component is designed specifically for **fast 360 degrees line of sight** visualization.

This source renders into a cube map texture (instead of a regular render texture) reducing the amount of shader passes needed for full 360 degrees line of sight from 6 to 1, improving performance significantly. There are a couple of drawbacks; this line of sight source is less precise than the normal *LOS Source* component (it uses encoded 16bit floats instead of 32bit) and also does not support blurring.

Most of this components settings are identical to the *LOS Source* component, with a few exceptions. Changing the camera's *Field of View* has no effect, as it will always render the entire world around it. Only one resolution can be specified through the *Cube Map Resolution* property, since a cube maps height and width need to be the same. This resolution also needs to be a power of two (64, 128, 256, 512, 1024, etc...). If you enter a none power of two number, it will be rounded up to the nearest number of two number (ex. 300 would become 512).

4.2.1 SETTINGS

- **Mask Color**: sets the color
- **Mask Intensity**: sets the intensity (most noticeable in HDR mode)
- **Mask Invert**: inverts the mask
- **Cube Map Resolution**: controls the width and height of this sources cube map render target.
- **Distance fade**: controls how much the line of sight fades over distance
- **Min Variance**: helps reduce artifacts in mask, best kept as low as possible

The **radius** of the line of sight is controlled by the cameras *far clipping plane* directly. Keeping this as low as possible, will result in the highest precision.

This component has a custom gizmo. The gizmo's sphere represent the area influenced by the line of sight. The color of this sphere matches the color set in the *Mask Color* property.

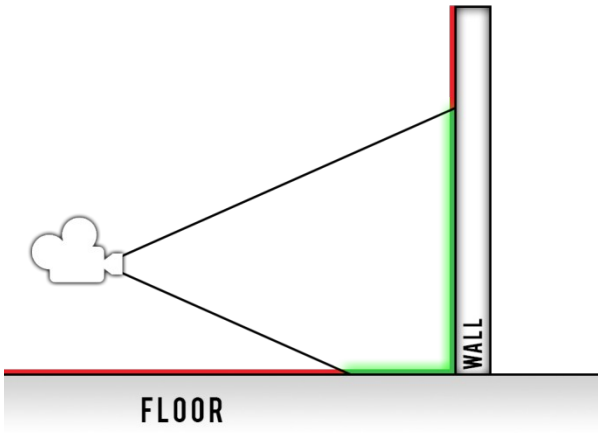
4.2.2 PERFORMANCE WARNING

Having several different resolutions for your *LOS Source Cube* components will increase the video memory needed. Try to use the **same resolution** for as many *LOS Source Cube* components as possible, so they can share the same cube map render texture as their render target.

5. KNOWN ISSUES

5.1 CAMERA FRUSTUM

The pictures below illustrates a "problem" that occurs when a line of sight source comes close to a wall. The top part of the wall, and a part of the floor, won't be visible to the camera attached to this source. In reality, a person standing close to a wall and looking straight ahead will also only see a part of the wall.

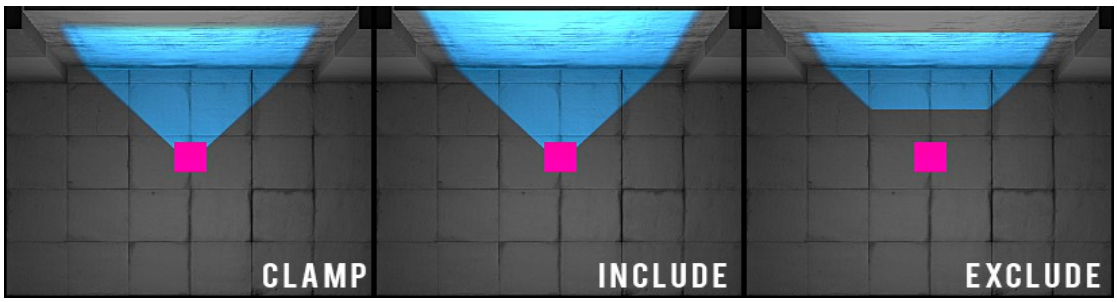


Even though this system accurately visualizes what can be seen from a certain position, this might not always be the desired outcome. The *Out of Bound Area* property of the *LOS Source* component allows you to control what happens to pixels above and below the cameras frustum (colored red in the image).

The default setting is **Clamp**, which will use the clamped depth value to calculate visibility. This setting works well in most cases, but starts producing artifacts when you are close to a wall. One way to reduce those artifacts is by increasing the height of the *LOS Source*'s render texture target relative to the width.

The **include** setting will mask the out of bound area as visible. This will cause some artifacts, when for example the wall in the image above is viewed from the right side.

The **exclude** setting is the most accurate one, it will mask the out of bound area as not visible.



5.2 SCENE SAVING

Because some of the scripts in this asset use the *ExecuteInEditMode* attribute (to allow direct previewing of changes in the editors game window), scenes that use these components will be marked as unsaved even though no changes have been made.

6. EXAMPLE SCENE

The example scene uses all of the available LOS components and can serve as a reference for how to set them up correctly. You can safely delete the example scene folder (or don't import it at all) if you have no need for it.

The scene does not use any custom layers or tags to avoid interfering with your projects settings, so keep in mind that the layers which are used in this scene might look illogical. The *TransparentFX* layer is used for objects that should not affect the line of sight. The *Water* layer is used for objects that affect the line of sight but should be excluded from the mask.

The blue capsules represent very simple AI controlled NPC's. They use the *LOSObjectHider* script to disable their renderer (effectively hiding them) when outside the line of sight.

When you start the scene in game mode, all of the furniture will be hidden until it is revealed by the line of sight. Afterwards it will remain visible even when outside the line of sight. This effect is achieved by adding the *LOSObjectRevealer* script to the object.

6.1.1 CONTROLS

- Character movement: Arrow or WASD keys
- Look around: Mouse
- Place camera with LOS source: Space bar
- Reset level: R

7. CHANGELOG

7.1 VERSION 1.2.3

- Added LOS Visibility Info component
- Improved LOS Culler component accuracy
- Fixed issues with Unity version 5.3.2f

7.2 VERSION 1.2.2

- Deprecated LOSSourceCube

7.3 VERSION 1.2.1

- Cubemap Fix for LOSSourceCube

7.4 VERSION1.2

- Added LOSSourceCube component for faster 360 line of sight
- Refactored and optimized existing code base
- Fixed world space reconstruction issues with orthographic cameras
- Added gizmos
- Added tooltips
- Updated example scene
- Improved error handling

7.5 VERSION 1.1

- Added HDR mode to LOS Mask
- Added Color Mask property to LOS Source
- Added Intensity property to LOS Source
- Added Out of Bound Area property to LOS Source
- Added Invert Mask option to LOS Source
- Added Layer Mask property to LOS Culler
- Added example prefabs
- Added LOS Layer Excluder script component
- Simplified setup process for multiple cameras
- Improved performance

8. SUPPORT

Feedback, suggestions or questions? Please send an email to: info@entropi-games.com