Fog of War

User Guide

v1.9

Overview

FogOfWar allows you to easily add fog to any 3D RTS.

Features:

- Works with 2D and 3D
- Varying map size
- Different fog colors
- Different filters
- Queue-friendly tools
- Good performance
- Line of Sight (occluding objects)
- View cones (using angles)
- Clear Fog (ie see through to background)
- See-through ground
- Works for both Orthographic and Perspective cameras
- Render to multiple cameras
- Save and load fog between plays

Updates

v1.9

- Fixed bug where screen would go black when the unity editor loses focus
- Improved performance by converting shader property ids to ints on initialization
- Made fog values public

v1.8

- Made fogValues setter public to enable manual setting of fog (for loading games)

v1.7

- Improved test scenes with enemies, better trees and smoother movements
- Put all code in FoW namespace
- Added toggle for max fog distance
- Made fog values publicly accessible via property
- Added method to calculate explored percentage
- Split fog planes and physics (ie 2D and 3D)

v1.6

- Fixed bug where fog was rendering on camera's far plane
- Fog can have textures applied to them
- Unit vision angles/cones

v1.5

- Add Secondary FoW to allow for multiple cameras

v1.4

- Added support for 2D (along the X and Y axes)

v1.3

- Fog now works with orthographic cameras

v1.2

- Added Clear Fog
- Included TransparentShadowCaster (see-through ground) shader

v1.1

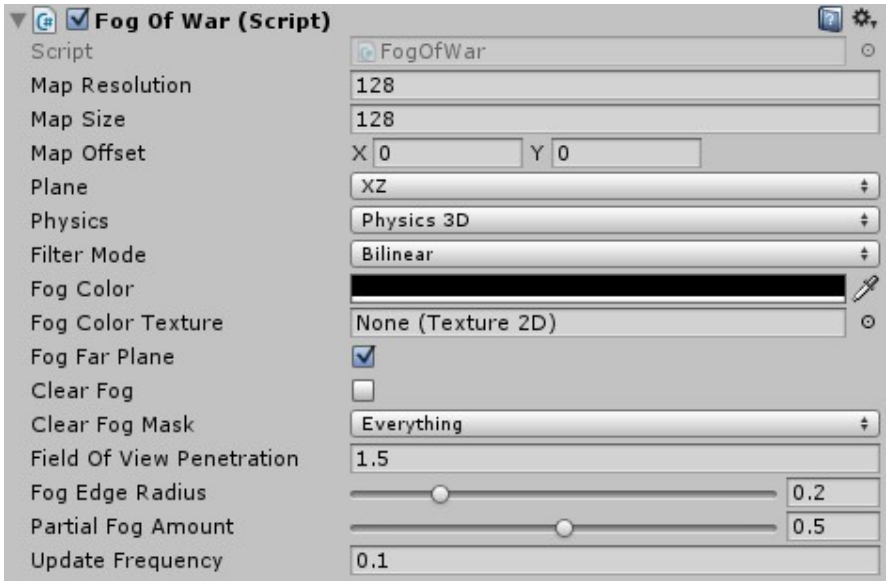
- Implemented Line of Sight
- Added new assets to demo
- Demo minimap now displays units
- Updated user guide considerably
- Added new method: Unfog(Rect)
- Camera zoomimg in the demo scene

v1.0

- Basic Fog of War implemented
- Basic demo

Components

FogOfWar



FogOfWar is the main component to handle the script. **It must be attached to the camera!**

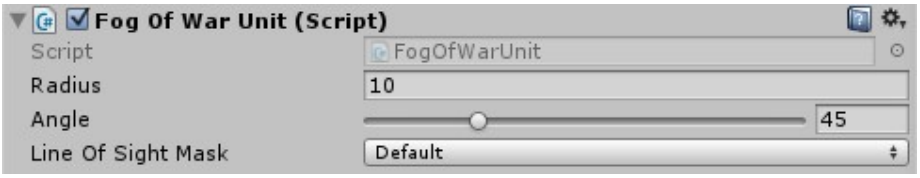
Map Resolution	The resolution of the texture used to render the fog. Setting this to a high value can have a significant performance impact, so keep it as low as possible. It should always be a power of 2.
Map Size	The size of the map that the fog will cover in world coordinates.
Map Offset	If the map is not centered at the origin (0, 0) you can offset it with this.
Plane	Which plane the fog will be rendered to. 3D is usually XZ and 2D is usually XY
Physics	What physics will be used for raycasting and collision detection. This is Unity's built-in 2D or 3D.
Filter Mode	The filtering applied to the fog. Bilinear will have a smoother look, Point will have a pixelated feel to it.
Fog Color	The color of the fog.
Clear Fog	If true, a secondary camera will render the background content using the clear fog mask (see the Clear Fog section).
Clear Fog Mask	Determines which objects will be rendered in the clear fog (see the Clear Fog section).
Field of View Penetration	How far units can see through an object when Field of View is enabled. Setting this to non-zero can have an impact on performance.
Fog Edge Radius	How much the fog will be interpolated from where a unit is located. Setting this to 0 will make the edge hard a pixelated.
Partial Fog Amount	When a unit moves, the old visible areas will be partially fogged by this amount. This should probably not be changed at runtime.
Update Frequency	How often the map texture will be recalculated. Having this set to a low value can have a significant performance impact when there are large amounts of units.

Useful Functions

void Reinitialize()	Reinitializes fog texture. Call this if you have changed the mapSize, mapResolution or mapOffset during runtime. This will also reset the fog.
float ExploredArea(int skip = 1)	Returns how much of the map has been explored/unfogged, where 0 is 0% and 1 is 100%. Increase the skip value to improve performance but sacrifice accuracy.
void SetAll(byte value)	Sets the fog value for the entire map. Set to 0 for completely unfogged, to 255 for completely fogged.
Vector2i WorldToFogPlane(Vector3 position)	Converts a world position to a fog pixel position. Values will be between 0 and mapResolution.
Vector2 WorldPositionToFogPositionNormalized(Vector3 position)	Converts a world position to a normalized fog position. Values will be between 0 and 1.
Vector3 FogPlaneToWorld(Vector2 position, float v = 0)	Does the inverse of WorldToFogPlane().
byte GetFogValue(Vector3 position)	Returns the fog amount at a particular world position. 0 = completely unfogged, 1 =

	completely fogged.
bool IsInCompleteFog(Vector3 position)	Returns true if the world position is completely fogged.
bool IsInPartialFog(Vector3 position)	Return true if the world position is at least partially fogged.
Unfog(Vector3 position, float radius, int layermask = 0)	Unfogs a part of the map. Position and radius are both in world coordinates. If a layermask is specified, Field of View will be checked
Unfog(Rect rect)	Unfogs a rectangular area on the map. Positions are all in world coordinates.

FogOfWarUnit



FogOfWarUnit should be placed on every unit that will have a range of vision.

Radius	How far the unit can see.
Angle	View cone angle. 180 degrees is full vision, 0 degrees is no vision. The forward direction will always be Y+ for 2D and Z+ for 3D.
Line Of Sight Mask	A layer mask that specifies which objects will occlude the unit's vision. Be sure that the occluding objects have a collider on it and that they are at the same vertical height as the unit to be properly occluded. Enabling this can have a significant impact on performance with large radii.

Known Issues

- Transparent shaders (ie blending) don't behave well with the fog. I am yet to find an efficient solution to this.

Future Features

Let me know if you're interested in any of these so I can bump them up the priority list.

- Penumbra for field of view
- Lots of optimization
- Multithreading (or maybe render fog texture map on GPU?)
- Custom methods for field of view (can allow for varying height of occluded objects)
- A method in FogOfWar that detects if an area of fog is visible (this would be a more efficient solution to the penetration depth when using Field of View)
- When an occlusion object is detected, a message is sent to the object so that it can clear that part of the map or have other in-game uses
- Custom component GUI?

- Smoothed edges when calling Unfog(Rect)
- Reduced aliasing on view cones

FAQs

Why does the fog appear to update at a slow frame rate?

You can easily change the update frequency in the FogOfWar component.

Updating the fog every frame for every unit can have a massive performance impact. To combat this, units update the fog individually. After a set amount of time, the fog is rendered and then the loop starts again.

If you only have a few units, updating the fog each frame may be viable. But if you have hundreds of units, the frame rate will become unbearable.

What is Clear Fog and how do I get it to work?

Clear Fog enables you to replace the fog with a different background. This can be used to create weird effects where the fog won't be just a single color.

To get it to work, there are a few steps you must take:

1. Make sure the **Clear Fog** is set to true.
2. Set the **Fog Color**'s alpha to zero.
3. Set the **Clear Fog Mask** to the objects that will appear in the background.

How can I improve performance for Clear Fog?

The main thing you want to do is make sure the **Clear Fog Mask** is set correctly. The mask should ONLY contain objects that will appear in the background (ie in the fog). Also, the camera's **Culling Mask** should contain only the objects that are NOT in the background (ie not in the fog).

You don't need to set the camera's **Clear Flags** to skybox as the skybox will be automatically set on the clear fog. This can help performance a tiny bit.

How can I have a see-through ground (ie a space scene)?

You need to have a surface for the fog to project on. But it is possible to make that surface transparent. There is a shader in the FogOfWar folder called **TransparentShadowCaster** which will allow you to do this.

One downside to this method is that objects in the background may not receive shadows.

How do I get Line of Sight working?

There are a few things you must do:

1. On each unit, make sure the Line of Sight Mask is set.
2. Make sure all occluding objects are of the same mask as set in step 1.
3. Make sure all occluding objects have a collider.
4. Make sure all occluding objects are level with the unit (ie they must be at the same height, or the unit will see right through it).

5. (Optional) Tweak the Field of View Penetration variable on the Fog of War component so that units can see the objects they are looking at.

What can I do to improve performance?

There are a number of things:

- Reduce the Map Resolution or the Unit's vision radius. This should always be the first port of call!
- Set a higher value for the Update Frequency.
- If you're using Line of Sight:
    - Make sure the ground/terrain is not on the same layer as the unit's Line of Sight Mask.
    - Make the Line Of Sight occlusion object's colliders as small as possible and space out the objects as much as possible. The more objects a single unit can see, the worse performance will get.
    - Consider setting the Field of View Penetration value to zero. An alternative to the penetration is to let the occlusion objects clear the fog themselves as opposed to letting the units do it.
- Beg me to add more features and optimize things better☺

How can I implement a minimap?

Fog of War will not generate a minimap for you, as there are many different parts to a minimap. But you can get the fog texture quite easily.

Fog is stored in a normal 2D texture with each pixel begin an 8-bit byte. A value of 0 means it is unfogged, 255 means it is completely fogged. If you want to convert the fog map to something more usable, you can find an example in the FogOfWarTestGUI.cs file (in the OnGUI() method).

How can I render fog to multiple cameras?

One camera needs to have the FogOfWar component attached. For all other cameras, you can use the FogOfWarSecondary component. It should automatically detect the original FogOfWar camera in your scene.

How do I save/load the fog?

You can access all of the fog values by **FoW.FogOfWar.current.fogValues**. This is a byte array and the size will depend on the **mapResolution** variable. You can read/write to this at any time.

Can I mix Unity's 3D physics with 2D art?

Yes! Set the FogOfWar plane to XY and physics to Physics3D.

Contact

Email: stu_pidd_cow@hotmail.com

Donations

PayPal: stu_pidd_cow@hotmail.com