FOV Mapping

FOV Mapping by *StupaSoft* is an advanced approach to Field of View (FOV) and Fog of War (FOW) systems for Unity. Leveraging the power of the GPU, it stands out as a high-performance solution that offers exceptionally efficient field of view system. FOV mapping has the following strengths.

1. **High Performance** - Running on the GPU, FOV mapping does not enforce the CPU to check the visibility of each pixel, nor does it fire numerous rays toward all directions. You can increase the number of units as you want without sacrificing the performance. FOV mapping can handle sights of hundreds of units at once.
2. **Terrain Adaptiveness** - FOV mapping can deal with terrains with highly variative elevation. With FOV mapping, you will find the fog of war harmonizes with your own terrain in a seamless manner.
3. **Rich Features** - FOV mapping provides various functionality to make your intention feasible. You can fine-tune the property values to adjust the visibility of enemy, sight range, and even precision of the FOV maps.

Originally tailored for real-time strategy games characterized by expansive environments, FOV Mapping's optimization enables smooth and seamless experiences in large-scale scenarios. Although it was initially optimized for real-time strategy (RTS) with a large scale, it can also be applied to various other projects seeking to incorporate a field of view system without compromising performance.

For a comprehensive understanding of the intricate operational mechanisms at play, delve into the in-depth descriptions provided within the posts at reference [1].

How to Set Up

If You Are Using The Universal Render Pipeline (URP)

If you're using Unity's **Universal Render Pipeline (URP)**, start by importing additional assets. Double-click `FOVMapping/RenderPipelinePackages/URPPackage.unitypackage` to do so. If you're not using URP, you can skip this step and move on to the next section.

1. Setup a Projector

1. Drag and drop the `FOVMapping/Prefabs/FOWPlane` (`FOWMapping/Prefabs/FOWPlaneURP` if you are using the URP) prefab to place in the scene.
2. Adjust the `FOWPlane`; modify the following properties to make the `FOWPlane` encompass the boundary of your map.

2. Prepare an FOV map

1. Open an FOV map baker window through the menu bar.
2. Once `FOVMappingEditorWindow` shows up, fill in the empty slots of `generationInfo` and modify other properties to fit in your intention. **Never forget to specify `Level Layer` to the layer of the level and assign `projector`**. The meanings of each property is described in the following section.
3. Press `Create an FOV map` button to commence generation.

4. After several minutes later, the process finishes and an FOV map asset is created in the path we specified.

3. Deploy `FOVAgent`s

1. Add `FOVAgent` component to all playable friendly and hostile units.
2. Modify properties of `FOVAgent` so that each unit behaves correctly as a sight agent. The following demonstrates how to set up units in general.
    1. For friendly units, check `Contribute To FOV` and uncheck `Disappear In FOW`. Adjust `Sight Range` to modify the visual extent, but ensure that it does not exceed `Sampling Range` we hired to generate the FOV map.
    2. For hostile units, uncheck `Contribute To FOV` and check `Disappear In FOW`. `Sight Range` does not matter as it is not going to contribute to the sight.

3. Press the play button and you may see a marvelous fog of war with great performance.

Scripting API

FOVMapGenerationInfo

`FOVMapGenerationInfo` specifies how to generate an FOV map.

Fields & Properties

Name	Description
`string path`	Path to save the generated FOV map
`string fileName`	Name of the FOV map file
`Transform FOWPlane`	Plane for with which the FOV map is baked upon and that shows the fog of war
`LayerMask levelLayer`	Layer of the level to be sampled; this field must be set to some layer other than `Nothing`.
`int FOVMapWidth`	Width of the generated FOV map
`int FOVMapHeight`	Height of the generated FOV map

Name	Description
`int layerCount`	Number of layers in the generated FOV map
`float eyeHeight`	Height of the 'sampling eye'
`float samplingRange`	Maximum sampling range; the sight system will not work beyond this boundary.
`float samplingAngle`	Vertical angular range from the sampling eye
`int samplesPerDirection`	How many rays will be fired toward a direction at a location?
`int binarySearchCount`	How many iterations for a binary search when finding a silhouette point?
`float blockingSurfaceAngle`	Surfaces steeper than this angle are considered vertical and there will be no further sampling toward the direction at the location.
`float blockedRayAngleThreshold`	Surfaces located below this vertical angle are never considered vertical.

FOVAgent

Fields & Properties

Name	Description
`bool contributeToFOV`	Is this agent an eye(set to true for friendly agents and false for hostile agents)?
`float sightRange`	How far can an agent see? This value must be equal to or less than the samplingRange of a generated FOV map.
`float sightAngle`	How widely can an agent see?
`bool disappearInFOW`	Will this agent disappear if it is in a fog of war(set to true for hostile units and false for friendly units)?
`float disappearAlphaThreshold`	On the boundary of a field of view, if an agent with `disappearInFOW` set to true is covered by a pixel in a fog of war whose opacity is larger than this value, the agent disappears.

Methods

Name	Description
`bool IsUnderFOW()`	Get if this agent is under a fog of war. If disappearInFOW is set to false, this agent is still visible regardless of whether it is being covered by a fog of war.

FOVManager

Fields & Properties

Name	Description
`int FOWTextureSize`	Size of the fog of war RenderTexture that will be projected with the Projector
`Color FOWColor`	Color of the fog of war
`int maxFriendlyAgentCount`	Maximum number of friendly agents (contributeToFOW == true)
`int maxEnemyAgentCount`	Maximum number of enemy agents (disappearInFOW == true)
`float updateInterval`	How frequently will the fog of war be updated?
`float blockOffset`	How much will the blocked sight be 'pushed away' to prevent flickers on vertical obstacles?
`float sigma`	Deviation of the Gaussian filter(larger value strengthens filtering effect to some extent)
`int blurIterationCount`	How many times will the Gaussian filter be applied? More iterations lead to a smoother fog of war, but with worse performance.
`float FOWExtrusion`	Extrusion value for the location of the fog of war
`Texture2DArray FOVMapArray`	(Essential) FOV map Texture2DArray for runtime FOV mapping
`Texture2D levelHorizonMap`	(Essential) Height map for showing the fog of war properly
`Shader FOVShader`	(Do not modify) FOV mapping shader
`Shader FOWProjectorShader`	(Do not modify) Fog of war projector shader
`Shader GaussianShader`	(Do not modify) Gaussian filter shader
`ComputeShader PixelReader`	(Do not modify) Pixel reader computer shader

Methods

Name	Description
`void EnableFOV()`	Enable the FOV system.

Name	Description
`void DisableFOV()`	Disable the FOV system.
`void FindAllFOVAgents()`	Find all `GameObject`s with an `FOVAgent` component.
`void AddFOVAgent(FOVAgent agent)`	Add an `FOVAgent` to the internal list of `FOVManager`. Once you spawn a new agent to the scene, you have to call this to make the agent take effect of FOV system.
`void RemoveFOVAgent(FOVAgent agent)`	Get rid of the specified `FOVAgent` from the internal list of `FOVManager`. The eliminated agent does not contribute to a sight(`contributeToFOV == true`) nor disappear in a fog of war(`disappearInFOW == true`).
`FOVAgent GetAgent(int idx)`	Retrieve an `FOVAgent` from the internal list with the specified index.
`int GetFOVAgentCount()`	Get the number of `FOVAgent` in the internal list.
`void ClearFOVAgents()`	Remove all `FOVAgent`s from the internal list. The FOV system will not work until another `FOVAgent` is added.

Pipeline Compatibility

FOV Mapping supports built-in pipeline and URP at this moment.

Pipeline	Support
Built-in	Yes
URP	Yes
HDRP	No

FAQ

1. **Question** - Enemy units are not disappearing even if they are under the fog of war.
   Answer - Did you check out the inspector values of `FOVAgent`?. The options are applied to each `FOVAgent` individually.
2. **Question** - How can I get just a traditional fog of war without the sight blocked by obstacles?
   Answer - You can achieve that by assigning `FOVMapping/FOVMaps/EmptyFOVMap` to the `FOV Map Array` property of `FOWManager`.
3. **Question** - Obstacles are not blocking the sight at all!
   Answer - Did you match `Level Layer` in the FOV map generation editor window with the layers of your obstacles? Select all the layers of which objects are intended to block the sight.
4. **Question** - `FOVAgent`s added to the scene through `Instantiate` are not working.
   Answer - You should add the new `FOVAgent`s manually to the agent pool inside a `FOVManager` instance through `FOVManager.AddFOVAgent`.

Links

[1] https://objectorientedlife.github.io/game%20project/FOVMapping1/

Contact

stupasoft@gmail.com