

# Docs - AOS Fog of War (v1.1)

 [Youtube Tutorial](#)

 [API](#)

 [Adjustable Properties](#)

[Basic Properties](#)

[Fog Properties](#)

[Level Data](#)

[Scan Properties](#)

[Debug Options](#)

 [Support](#)

 [Contact](#)

 [Major Credits & References](#)

 **Youtube Tutorial**

<https://youtu.be/MnER3bD7LbA>



**AOS Fog of War** Main Page Namespaces Classes Files Search

AOS Fog of War  
Namespaces  
Namespace List  
FischlWorks\_FogWar  
BigHeaderAttribute  
csFogVisibilityAgent  
csFogWar  
FogRevealer  
LevelColumn  
LevelData  
CheckLevelGridRange  
CheckVisibility  
CheckWorldGridRange  
GetUnitVector  
GetUnitX  
GetUnitY  
GetWorldX  
GetWorldY  
WorldToLevel  
\_UnitScale  
levelData  
shadowcaster

## FischlWorks\_FogWar.csFogWar Class Reference

The non-static high-level monobehaviour interface of the AOS Fog of War module. [More...](#)

Inheritance diagram for FischlWorks\_FogWar.csFogWar:

```
graph TD
    MonoBehaviour --> FischlWorks_FogWar_csFogWar
```

### Classes

- class [FogRevealer](#)
- class [LevelColumn](#)
- class [LevelData](#)

### Public Member Functions

- bool [CheckLevelGridRange](#) (Vector2Int levelCoordinates)  
Checks if the given level coordinates are within level dimension range.

AOS Fog of War: AOS Fog of War API

<https://fischlworks.github.io/>

## Adjustable Properties

### Basic Properties

#### Basic Properties

Fog Revealers 2

Element 0

- Revealer Transform
- Sight Range 10
- Update Only On Move ☒
- Debug
  - Last Seen At X 0 Y 0

Element 1

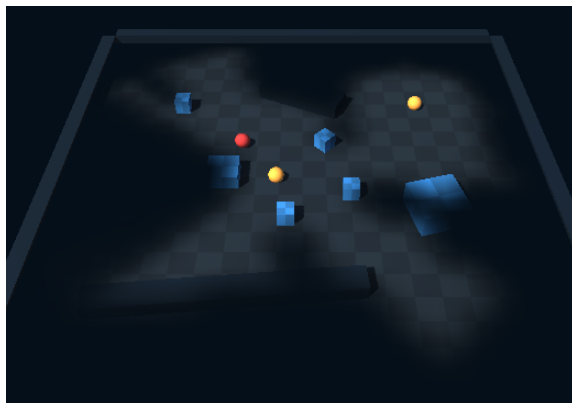
- Revealer Transform
- Sight Range 5
- Update Only On Move ☒
- Debug
  - Last Seen At X 0 Y 0

Level Mid Point

Fog Refresh Rate 10

**Fog Revealers** are entities that provides vision to the player, which usually refers to player characters, minions, turrets, or wards. Having zero revealers is OK.

Note that through ***AddFogRevealer()*** public method of ***csFogWar***, you can add more entities as the revelaters dynamically during runtime.

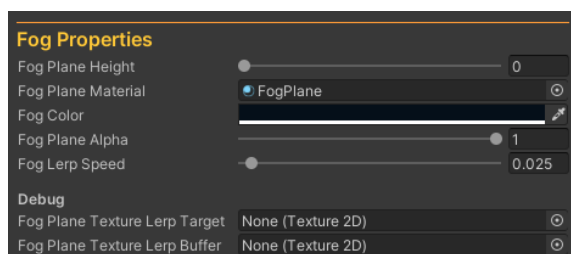


Changes in **Sight Range** property during runtime is OK, and the **Update Only On Move** property will stop the module from being updated (it will just return shortly after each ***Update()*** call) when the player is not moving.

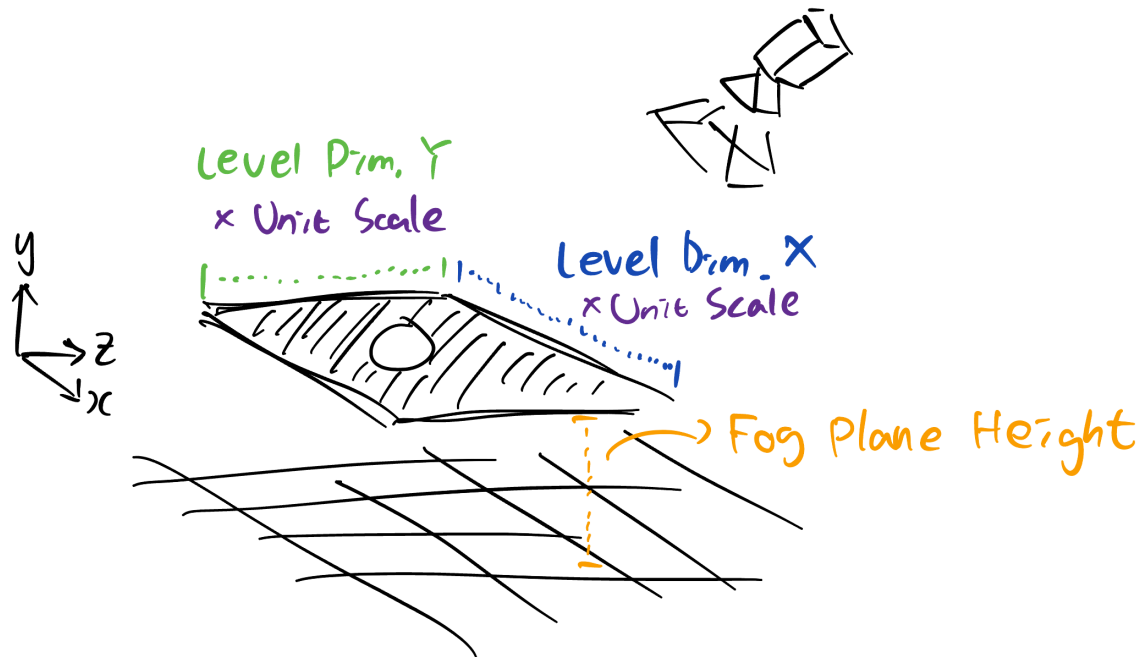
**Level Mid Point** property **MUST** be set to the middle of the level, and it is recommended that it is not moved during runtime. All scans and unit conversion happens around this specific transform's position, so setting it right is very important.

**Fog Refresh Rate** designates the number of checks that is done per second. It is based on ***Time.deltaTime***, so it is independant from target framerates.

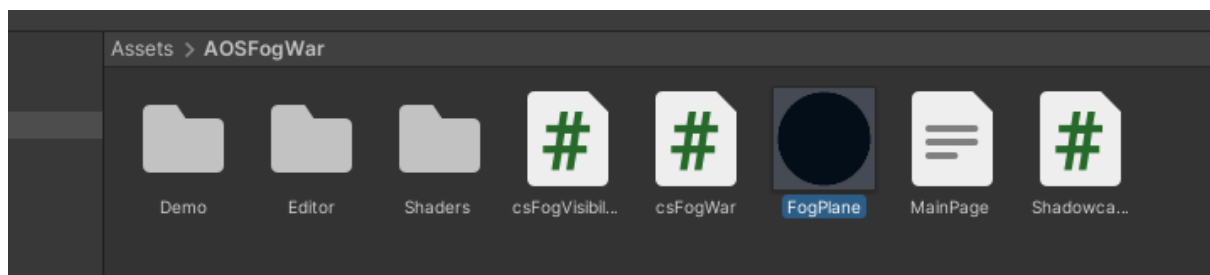
## Fog Properties



**Fog Plane Height** property designates the height of the fog plane.



This drawing may help you understand better.

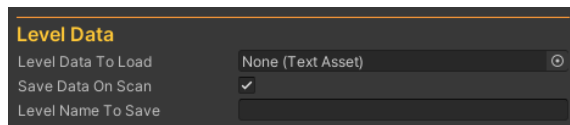


**Fog Plane Material** is a property section that **REQUIRES** the specific material named ***"FogPlane"*** that comes along with the module to be assigned manually.

With the **Fog Color** and **Fog Plane Alpha** property, you can control the color and the alpha value of the fog plane texture.

**Fog Lerp Speed** literally designates the speed that the shape of the fog texture changes. The closer it is to 0, the smoother the transition will be.

## Level Data

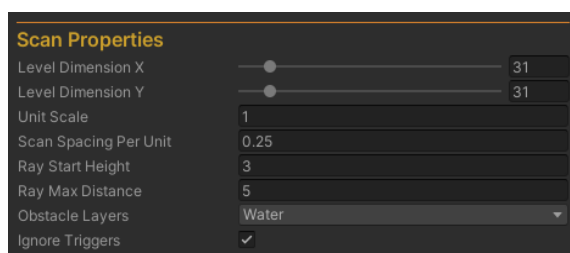


Inside the **Level Data To Load** property, you can assign a pre-scanned & saved level data located in ***Assets/Level/Data/*** path. The private path name property is called **levelScanDataPath**.

Save Data On Scan option designates whether to save the scanned data or not. Note that if there are no pre-assigned level data, the csFogWar module will perform a scan based on the **Level Mid Point** property and the **Scan Properties**.

If there's already an existent file having the same name, the file will be overwritten by default.

## Scan Properties




**Level Dimension X** and **Level Dimension Y** designates the dimension of the level. Keep in mind that larger dimension will result in a much heavier computational load for the CPU.

**Unit Scale** property determines the width and height of each cell, and **Scan Spacing Per Unit** property determines the space between each **BoxCast** passes.

Unity - Scripting API: Physics.BoxCast

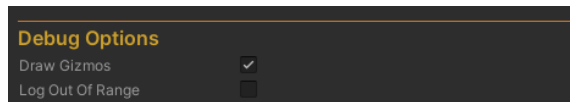


 [https://docs.unity3d.com/ScriptReference/Physics.BoxCast.h](https://docs.unity3d.com/ScriptReference/Physics.BoxCast.html)  
tml

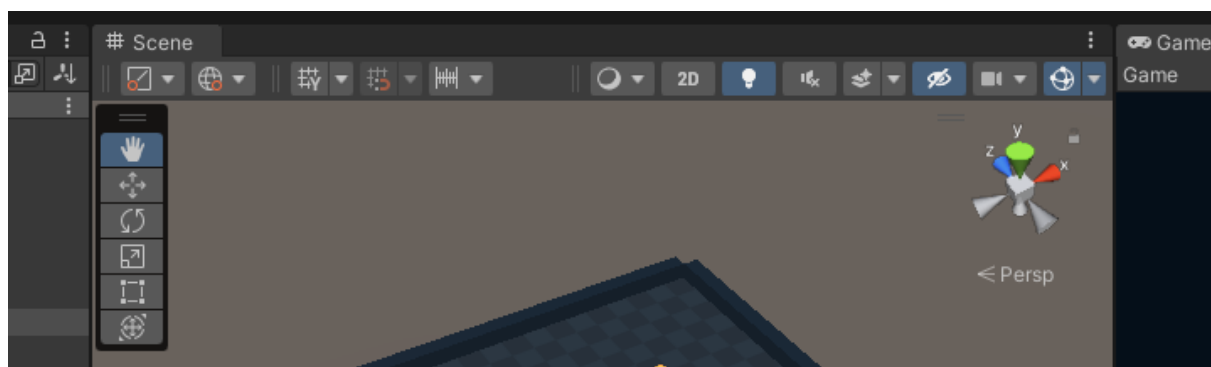
**Ray Start Height** and **Ray Max Distance** property are also passed onto the **BoxCast()** function as parameters, and you may have to optimize this value carefully the level is not flat. Note that the ray is casted downwards.

**Obstacle Layers** property is used to determine whether a hit object is an obstacle or not. **Ignore Triggers** option is pretty much self-explanatory.

## Debug Options



Toggling the **Draw Gizmos** option will let the module draw gizmos in the Scene view. **Log Out Of Range** option lets the module to log all level cell indexing request that is out of grid range.



Note that you have to toggle unity's show gizmos option on in the upper-right side of the scene view to actually display the gizmos.



## Support

*keithrek@hanmail.net*



## Contact

<https://www.linkedin.com/in/keithrek/>



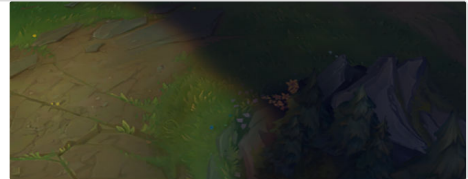
## Major Credits & References

### A Story of Fog and War

<http://riot.com/1NvPXyT> #RiotTechBlog



<https://technology.riotgames.com/news/story-fog-and-war>



### Dota 2 Interactive Map v5

Dota 2 Interactive Map. View, pan, and zoom in on any part of the Dota 2 map. Turn on various overlays for more information, place wards to see sight ranges, and measure distances.



<https://devilesk.com/dota2/apps/interactivemap/?x=-349&y=406&zoom=1&mode=observer&observer=-1485,94;-1960,2520;574,568>

### Symmetric Shadowcasting

<https://www.albertford.com/shadowcasting/>

### Implementing Fog of War for RTS games in Unity 2/2

As I said in the previous blog post, some time ago I started working on a new Fog of War / Vision System solution aiming the following features: Being able ...



<https://blog.gemserk.com/2018/11/20/implementing-fog-of-war-for-rts-games-in-unity-2-2/>

### 유니티 - 전장의 안개(Fog of War)

목차 1. 개념 2. 구현 방법 3. 타일맵을 이용한 구현 4. 구현 결과 5. 프로파일링, 최적화 6. Reference



<https://rito15.github.io/posts/fog-of-war/>