Fog of War in videogames

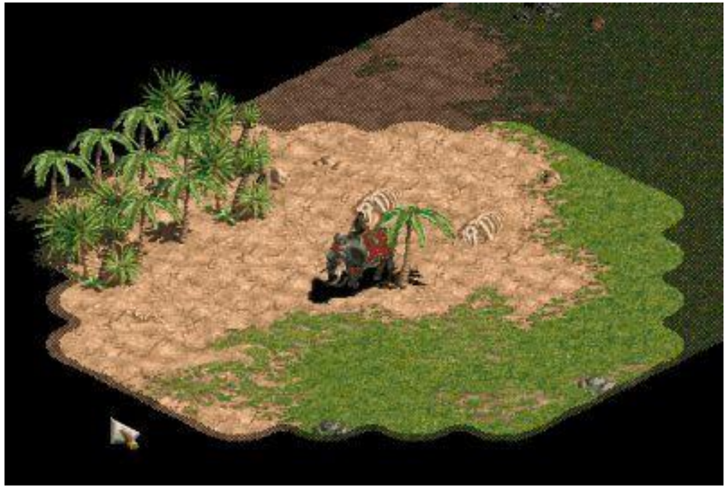
Oscar Pérez Martín

Fog of War (FoW)

What does FoW mean?

What is it used for in video games?

- **Strategy**-> Prediction, Intel Gathering
- **Adventure/Roleplay**-> Exploration



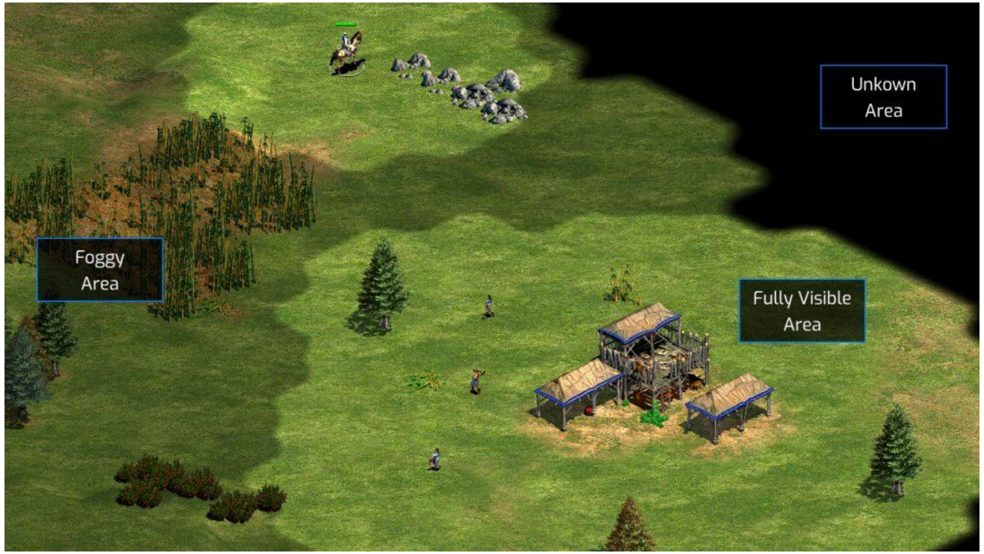
Age of Empires

FoW Characteristics - Distinctive features

Gameplay related: fog & shroud

- *Fog:* Some (little) information, half opacity. Seen back in time.

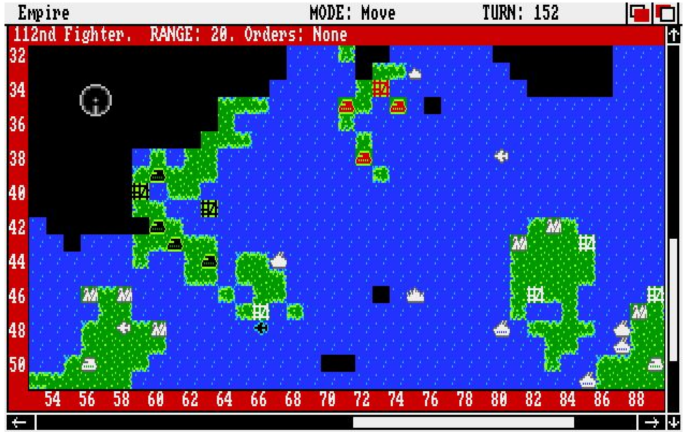
- *Shroud:* No information, fully occluded map.



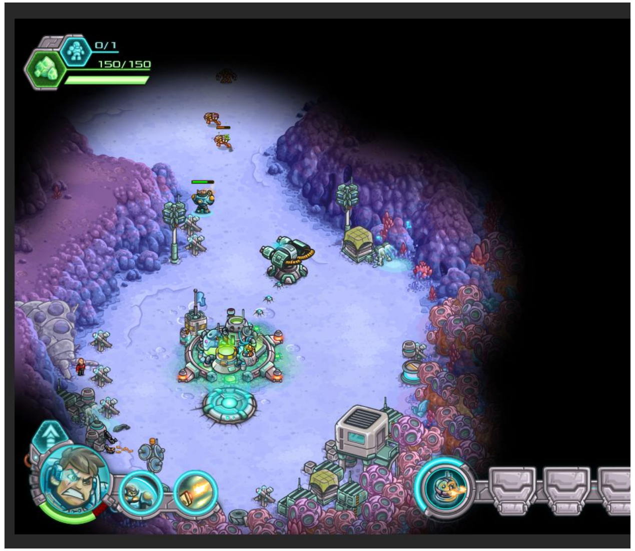
Age of Empires

FoW Characteristics - Distinctive features

Aesthetics related: “Chunky” vs “Smooth” fog



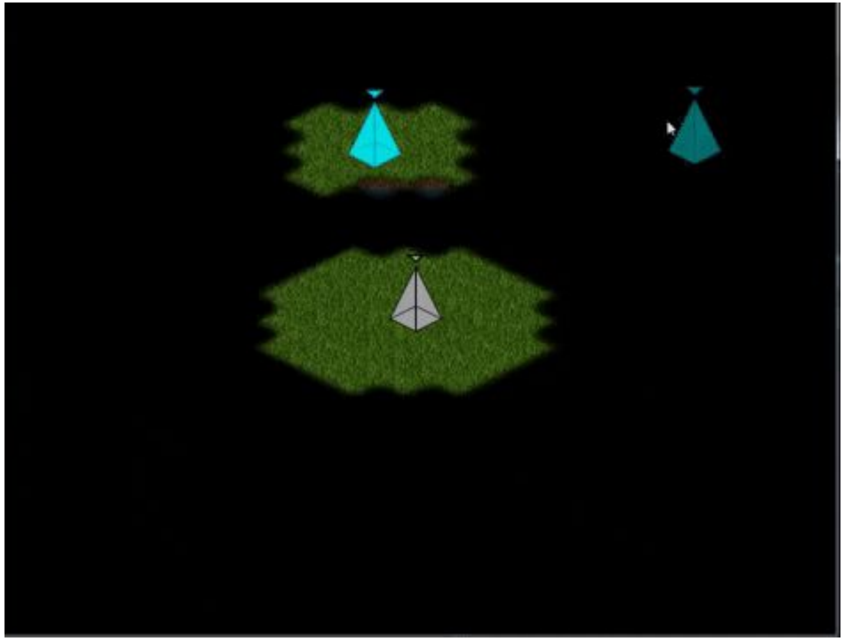
Empire



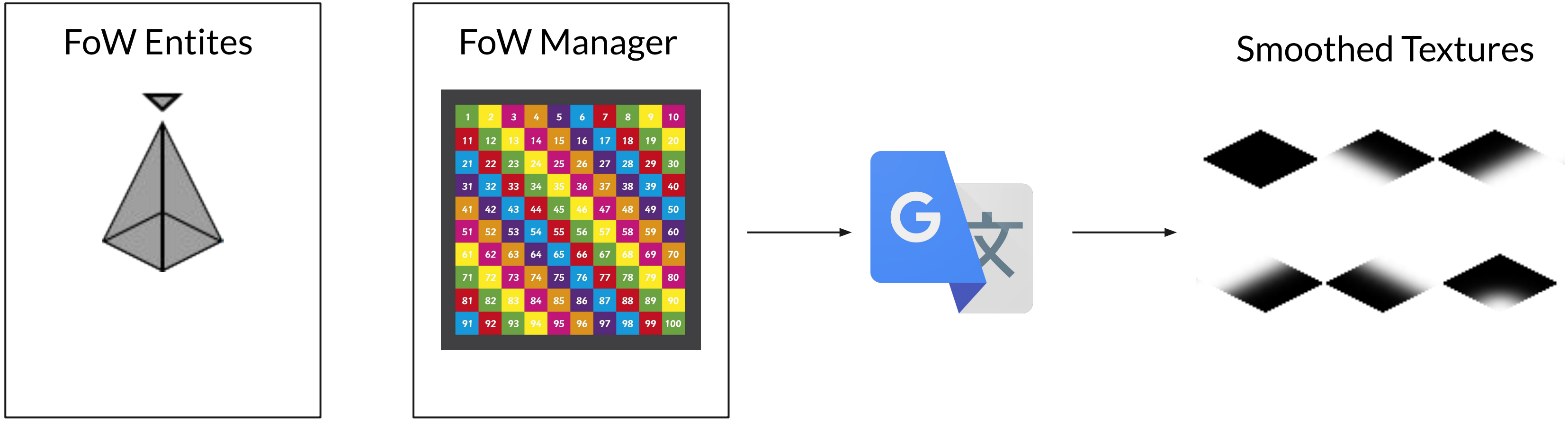
Iron marines

My implementation

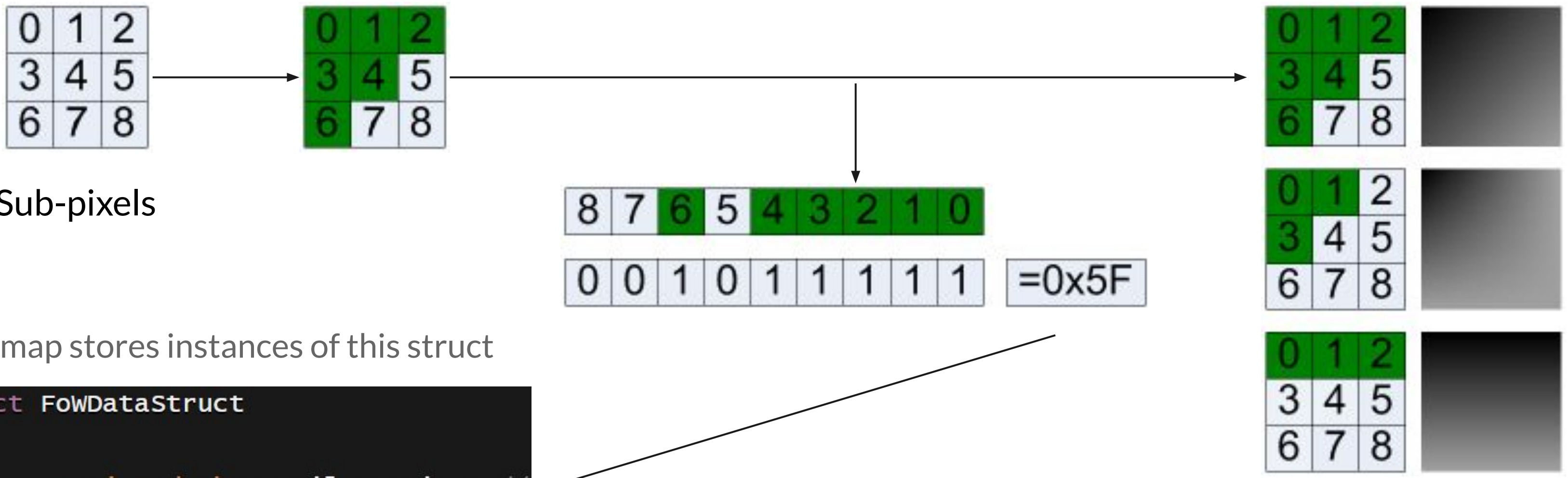
- Smooth FoW
- Fog & Shroud
- Bitmasking



Code Structure

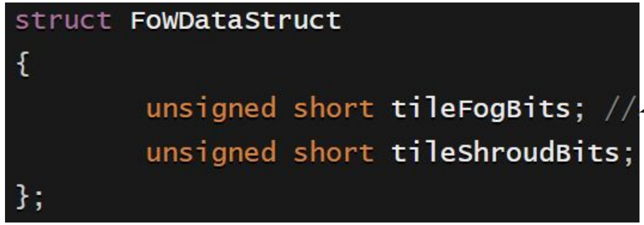


Bitmasking: Sub-Pixels



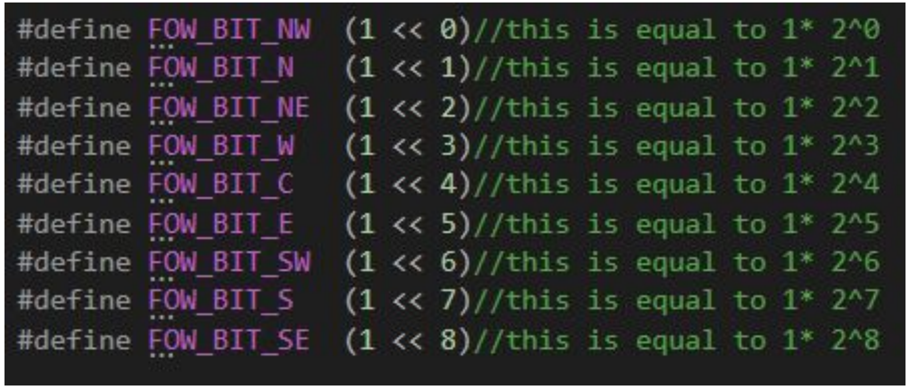
Sub-pixels

The map stores instances of this struct



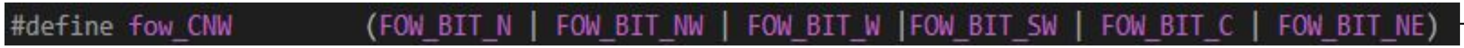
```
struct FoWDataStruct
{
        unsigned short tileFogBits; //
        unsigned short tileShroudBits;
};
```

Bitmasking: Bit definitions

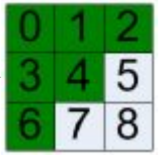


```
#define FOW_BIT_NW  (1 << 0)//this is equal to 1* 2^0
#define FOW_BIT_N   (1 << 1)//this is equal to 1* 2^1
#define FOW_BIT_NE  (1 << 2)//this is equal to 1* 2^2
#define FOW_BIT_W   (1 << 3)//this is equal to 1* 2^3
#define FOW_BIT_C   (1 << 4)//this is equal to 1* 2^4
#define FOW_BIT_E   (1 << 5)//this is equal to 1* 2^5
#define FOW_BIT_SW  (1 << 6)//this is equal to 1* 2^6
#define FOW_BIT_S   (1 << 7)//this is equal to 1* 2^7
#define FOW_BIT_SE  (1 << 8)//this is equal to 1* 2^8
```

Just for simplicity & readability

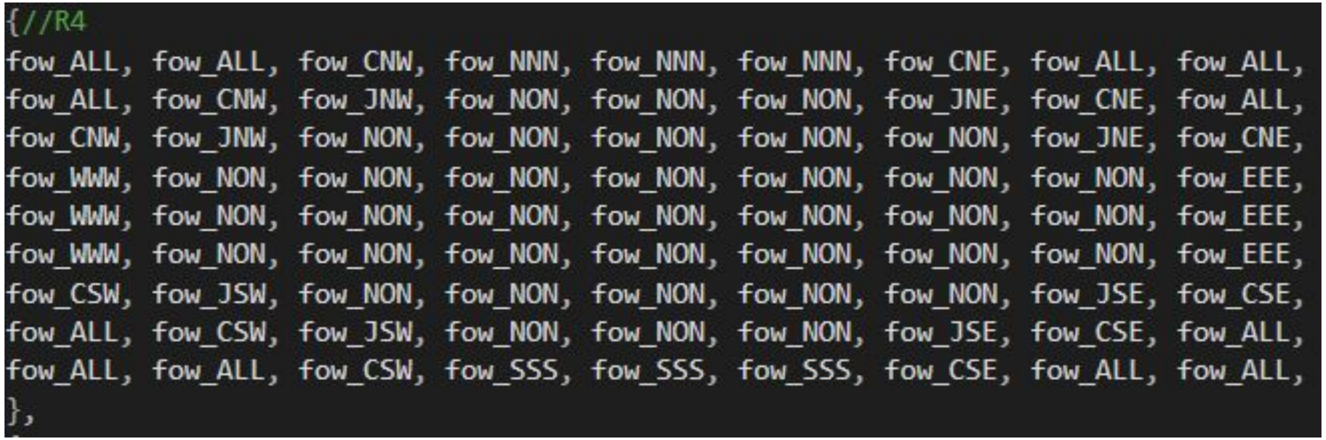


```
#define fow_CNW          (FOW_BIT_N | FOW_BIT_NW | FOW_BIT_W |FOW_BIT_SW | FOW_BIT_C | FOW_BIT_NE)
```



Bitmasking: Shape masks

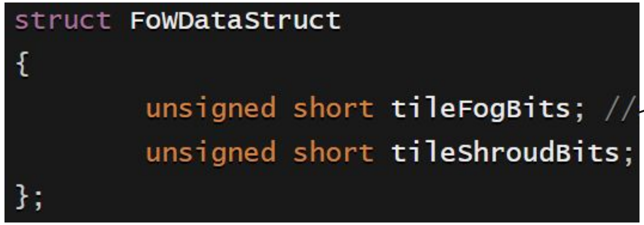
Shape mask for a circle of radius 4



```
{//R4
fow_ALL, fow_ALL, fow_CNW, fow_NNN, fow_NNN, fow_NNN, fow_CNE, fow_ALL, fow_ALL,
fow_ALL, fow_CNW, fow_JNW, fow_NON, fow_NON, fow_NON, fow_JNE, fow_CNE, fow_ALL,
fow_CNW, fow_JNW, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_JNE, fow_CNE,
fow_WWW, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_EEE,
fow_WWW, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_EEE,
fow_WWW, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_EEE,
fow_CSW, fow_JSW, fow_NON, fow_NON, fow_NON, fow_NON, fow_NON, fow_JSE, fow_CSE,
fow_ALL, fow_CSW, fow_JSW, fow_NON, fow_NON, fow_NON, fow_JSE, fow_CSE, fow_ALL,
fow_ALL, fow_ALL, fow_CSW, fow_SSS, fow_SSS, fow_SSS, fow_CSE, fow_ALL, fow_ALL,
},
```

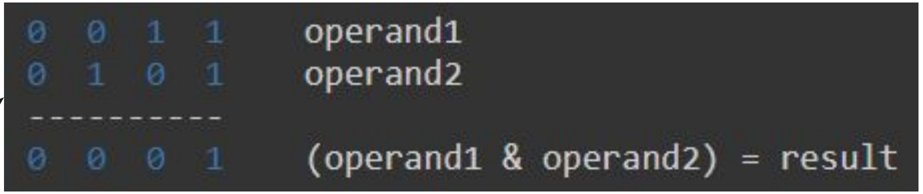
Bitmasking: Applying the mask

1 on 1 bitwise ANDing

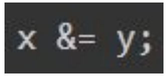


```
struct FoWDataStruct
{
        unsigned short tileFogBits; //
        unsigned short tileShroudBits;
};
```

Mask



```
0   0   1   1     operand1
0   1   0   1     operand2
-----------
0   0   0   1     (operand1 & operand2) = result
```

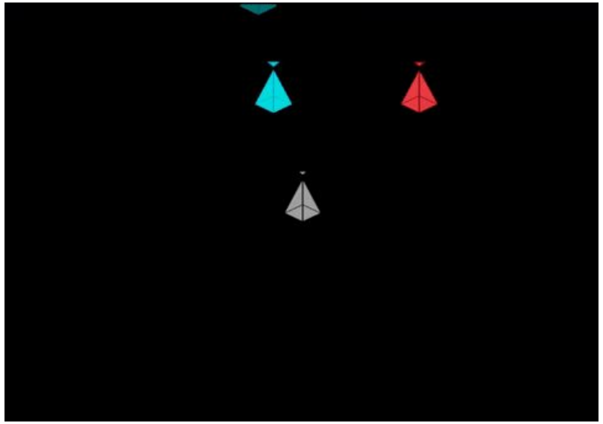


```
x &= y;
```

TODO 1 - Creating the FoW Map

To start making fog of war we first need a map in which we will store the fog of war data of every tile. So go ahead and create your first FoW map!

Result:



TODO 2 to 2.3 - Creating FoW Entites

In this series of TODO's you will learn how the FoW entities work:

- In TODO 2 you will learn how to create them and where are they stored
- In TODOs 2.1 to 2.3 you will learn how are they linked with normal entities

Result:

TODO 3 - Applying the Bitmask

The real deal! In this TODO you have to apply a precomputed mask to each affected tile in the FoW map data.

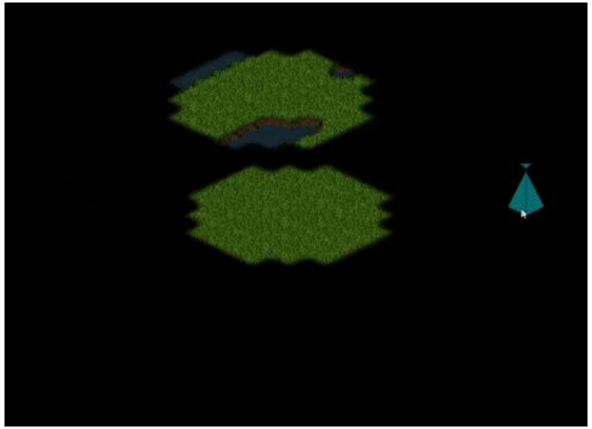
I recommend you to check TODO 3.1 in the code if you want to take a look to the sub-pixel definitions

If you have trouble with Bitwise operators check these links:

[Bitwise operators]

[How do Bitwise operators work]

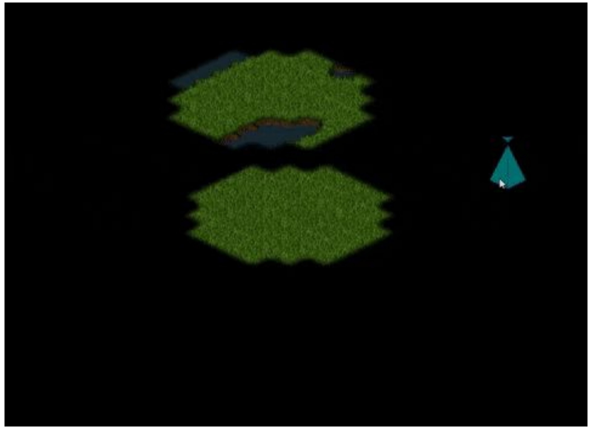
Result:



TODO 4 to 4.1 - Moving the FoW Entity

As you might have noticed, altough you move the player the masks stays still. We are going to change this now.

Result:

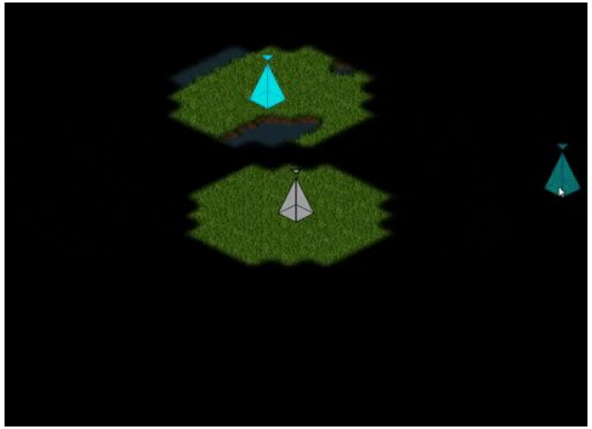


TODO 5 - Check tile visibility

We are almost done! There is only one thing left to do. We just need a way for the entities to know when they are visible or not.

So we are going to complete a function that tells the entities whether they are on top of a visible tile or not.

Result:



Homework

- Check the FoWManager function DrawFoWMap() and make sure you understand how it draws FoW tiles.
- Try to create your own precomputed shape mask

Improvements

I now have a FoW entity for every entity of the project, which keeps the project at a steady 60+ fps even with 100+ units. Nevertheless, you should consider grouping your entities and let them have just one FoW Entity for each group in case you want to have a lot of units together in one place to avoid re-checking the same tiles over and over again.

You can combine the unit move check that exists in my code with a timer to perform the check every now and then. Between 0.25 and 0.5 sec will be fine in most cases.

Thank you for your time!