[http://dictionary.sensagent.com/Fog%20of%20war/en-en/#Grand\_strategic](http://dictionary.sensagent.com/Fog%20of%20war/en-en/" \l "Grand_strategic)

<https://en.wikipedia.org/wiki/Wargaming#Early_years_to_Kriegsspiel>

<https://www.giantbomb.com/fog-of-war/3015-14/>

http://dictionary.sensagent.com/Kriegsspiel%20(wargame)/en-en/

https://www.amazon.com/War-Made-New-Weapons-Warriors/dp/1592403158

<https://archive.org/details/Tanktic-s_1981_Avalon_Hill>

Fog of War:

0.Introduction:

Fog of war is a common concept found in many combat-based real-time and turn-based strategy games. It simulates the unknowns of the battle and provides an exploration incentive to uncover them, forcing the player to rely on imperfect and incomplete information of the battlefield and sometimes improvise based on intuition and common sense.

In this document we are going to talk about the concept of the Fog of War and all its variables, the origins, how board games and videogames adapt the concept to engage the user, and finally a tutorial of how to do a fog of war for an RTS game.

1.Concept:

The **fog of war** is a term used to describe the uncertainty in the situation experienced by participants in military operations. The fog of war isn’t limited at the own capability, so it determinates the adversary capability too.

So fog of war is directly related with the fog in a real battle camp and the ambiguity that it generates, but when this concept is transferred to the world of games is not necessary simulated with fog. For example, in Battleship the famous board game there’s no fog but the concept of war ambiguity is present, in fact, is the principal feature of the game.

[Battleship foto]

In videogames is practically the same. It’s true that a lot of videogames simulate the fog of war concept directly with fog but some videogames use other methods. For example, BRAIN / OUT is a shooter that simulates the war ambiguity hiding the players that aren’t doing noisy actions.

[BRAIN / OUT foto]

Now that we know what really means Fog of War let’s talk about the different ways in which the ambiguity can affect the user and change the game flow.

1.2 Types:

The ambiguity that the fog of war generates can vary according to the level at which participants are engaged. The different ambiguity levels are:

* Grand Strategic:

Ambiguity is related to the political intent, capabilities and logistical strengths of an adversary. Basically we know that there’s an adversary but we don’t know nothing about it. [Age of Empires develop]

* Military Strategic:

Ambiguity is related to the structure, strength, capability, and disposition of own and adversary offensive and defensive assets. The own assets ambiguity can be generated by a bad report of it or unwillingness to watch for them. [Cod Hardcore develop]

* Operational:

Ambiguity is related with the adversary like the grand strategic but is coupled with own directive ambiguity. So the commander doesn’t have the full sight of the strategic imperative. [Dragons of Atlantis develop]

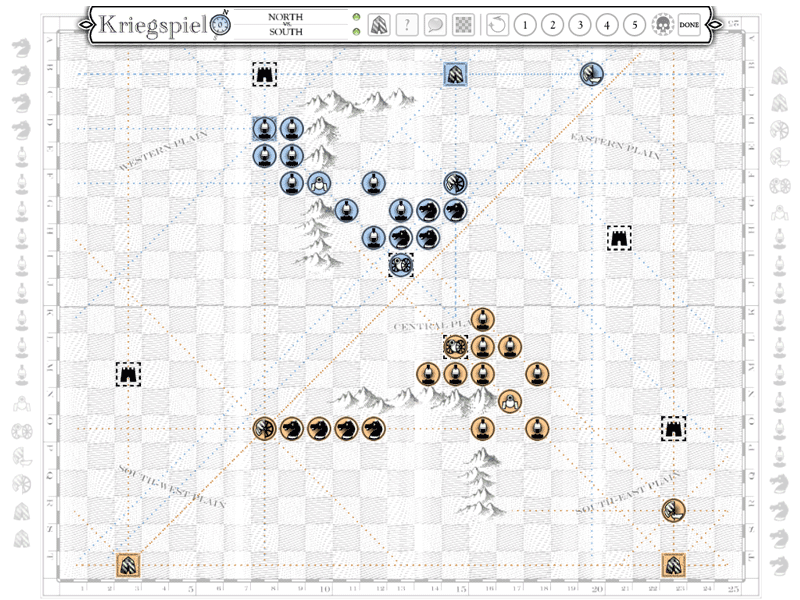
* Tactical:

In this case the ambiguity is around several factors at the tactical level. The ambiguity can be related with the interaction between allied troops, their status and their intentions. This lack of comprehension of the tactical environment by the commander aren’t necessary generated by factors inherent to battle. [??? develop]

2.Origins:

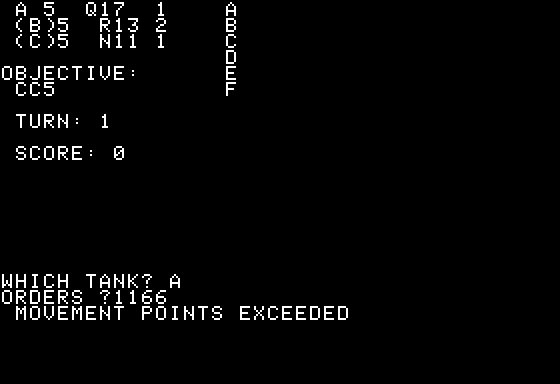
Okay, we have seen all the different features of the fog of war and which mechanics use the games to adapt the concept. But who was the first person that thinks that use fog of war in a game should be a good idea?

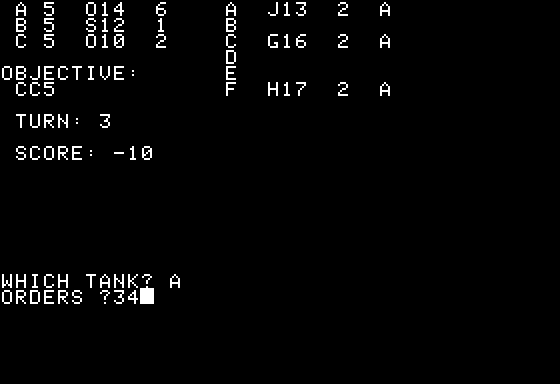
The first board game that uses fog of war was Kriegsspiel, that means wargame in German. This game was created by Lieutenant von Reisswitz in 1824 for training officers of the Prussian army. The invent that Lieutenant craft wasn’t directed only for playing but is considered the grandfather of modern wargames because he established several conventions for wargaming.



Getting closer to today and entering to the digital era, we have Tanktics. The first digital war-game created in 1977 by Chris Crawford’s and published for Commodore PET in 1978. The game simulates a two-player tank battle on a hex grid. This was the first videogame that uses fog of war and for that was criticized. The fog of war in Tanktics generates a Military Strategic ambiguity because the game doesn’t show the enemy coordinates since your tanks don’t see it.

This two Tanktics screenshots shot three ally Tanks (A,B,C) that the player controls and the enemy tanks (A,B,C,D,E,F) that are in unknown coordinates in the first frame. In the second frame the player has discovered A, C and F tanks. To know more about Tanktics location system and how it works go here: <https://archive.org/details/Tanktics_1981_Avalon_Hill>





3.Lets craft our fog of war:

Now we are experts of fog of war so let’s craft our fog of war for a RTS videogame.

This tutorial of how to make a basic fog of war for a RTS videogame is focused to people that knows the basic concepts of c/c++ programming.

First we download the base code from this “link” where there’s also a release with the final result of the tutorial. We have the material, let’s work with it!

TODO 1: Create fog layer

Allocate an array of FOG\_TYPE for the fog\_layer.

Array size is the same as a map layer so get map width and map height from App->map.data.

Iterate the allocated array and fill it with DARK\_FOG type.

When all the past objectives are cleared discomment the code of ClearFogLayer method in fog of war module.

TODO 2: Map optimization

Improve the GetFogID method to get the FOG\_TYPE of the map tiles.

If the FOG\_TYPE is DARK\_FOG the tile is covered of opaque fog so its not necessary to draw it.

TODO 3: Entities draw optimization

Improve entity Draw method to draw the entity only when is necessary.

Ally entities are always drawn.

Neutral entities are not drawn when FOG\_TYPE is DARK\_FOG.

Enemy entities are only drawn when there’s NO\_FOG.

TODO 4: Draw alpha layer

Collect the candidates of the fog\_quadtree to get the alpha cells in camera.

CollectCandidates quadtree method fills the vector you pass with the alpha cells inside the specified area.

Iterate the filled vector and call FogBlit method from render module.

The cell size is the size of the cell walls in pixel calculated when the fog alpha layer is generated.

TODO 5: Make ally entities clear alpha layer

In entity draw method check if the entity is an ally. If is an ally, clear alpha layer around vision area with 0 alpha value.

To clear alpha layer use ClearAlphaLayer method from fog of war module.

TODO 6: Make player check his vision area

Fill CheckVisionArea entity method and call it from player update.

Use the public entities quadtree from the entities manager to collect the candidates inside the vision area. CollectCandidates quadtree method fills the vector you pass with the entities inside the specified area.

TODO 7: Clear fog area when a neutral entity is detected

Iterate the entities collected in the last TODO and check if there’s a neutral entity.

If there’s one, clear alpha layer around its vision area with MID\_ALPHA value and fog layer around its render area with GRAY\_FOG.

TODO 8: Update alpha layer

To update alpha layer, you just need to check if there’s any alpha cell in camera with a higher value than the MID\_ALPHA value defined before.

if( alpha > MID\_ALPHA )alpha = MID\_ALPHA

Extra Work!

Now that you have a functional RTS fog of war system let’s upgrade it! Update the fog alpha layer clear method to have a smoothest result.

Check the solution release and active the glow mode( F2 ) to see one possible solution.

TODO 8: Smooth