Aggronator Documentation

1. Introduction

Aggronator is a scripting asset for Unity created by Thiago Kühn with art from Maximilian Costa, both students of electronic games at University of Vale do Rio Sinos, Porto Alegre, Brazil.

Its objective is to provide an AI system in which some agents can choose a target according to a number of factors, and also to choose an action according to the target (like fighting or running). To do this, you'll need a class for the Agents used in combination with the Aggronator script.

2. How it Works

The Aggronator has a very simple idea in its essence: You call the Aggronator from an agent, and send a number of factors from the agent and its enemies, and set a weight for each of these factors. The Aggronator is going to do one calculation to determine which will be the target, and another to determine which will be the action to the target. The target and the action will be then returned to the agent. Then, you should call the respective funcion according to the way your agent works.

You can check the functions of the "Enemy" class, which is the basic agent used in the example scene, to see how it works with the Aggronator, and how to get the results and run the right function.

3. The 'enemy' class

In the Aggronator example scene, the 'Enemy' class represents a generic agent whose actions are defined by the Aggronator. It works like this: The enemy is initially in a 'passive' state (represented by the light warriors), and it just remains still. When you click on a 'passive' enemy, its Sprite is going to change to a dark warrior, and it will be changed to an 'altered' state. The altered state makes them hostile against all the 'passive' enemies. Note that the 'passive' and the 'altered' enemies call different functions to the Aggronator: The 'altered' enemy is more aggressive than the 'passive', and instead of running when on disadvantage (action for the 'passive' enemy), it tries to regroup with his allies.

4. The Aggronator

As stated in 2, the Aggronator has a very simple idea: send some inputs to the Aggronator, and it will return a target and an action. Now, you'll be able to see the Aggronator functions, and what each of these do. Note that all of the agents must have a reference to the Aggronator, so they can call the Aggronator functions.

calculatePassiveEnemy and *calculateAlteredEnemy*: These two public functions are called, respectively, by the 'passive' and the 'altered' agent, on every frame, when there are hostile agents in the scene. They receive a number of parameters from the agent, and

iterate over every hostile enemy in the scene to check who will be the targeted agent, by calling the functions *calculatePassiveTarget* or *calculateAlteredTarget*. Then, after choosing the best target, the Aggronator calls the function *calculatePassiveAction* or *calculateAlteredAction* to choose what will be the action related to that target.

calculatePassiveTarget and *calculateAlteredTarget*: These functions are private, and they are called by the above mentioned functions. Their objective is to set who will be the target of the agent; to do that, it checks some factors, like: the own agent life, the hostile agent life, the distance between the two, and some other factors. Then, a calculation is made by using these factors and a weight attributed to them, which is then returned to the above functions. This function is called for every hostile agent, so it can check all of the potential targets.

calculatePassiveAction and *calculateAlteredAction*: These functions are also private, and they are called once the target is already defined. They receive some parameters to check what will be the action to the target. They return a number, which will be used by the 'Enemy' class to define what will be the desired behavior. Note that the example has only two possible reactions for different returned values, but you can add as many as you want.

5. Conclusion and possibilities of use

The 'Aggronator' has a big potential of use on many different genres. Some of the countless possibilities are:

- When defining the AI for an RTS, use the 'Aggronator' to make the behavior of the units when there is no intervention from the player
- In an MMORPG, use the 'Aggronator' to define which player will be the target for a monster AI
- In a survival horror, an increasing horde of zombies, to define which will be the target for the zombies
- Etc.

For future versions, we intend to create more examples, with different game genres. We also intend to make use of the Inspector for the Aggronator script, so some changes will be possible without having to access the script itself.