Little Rogue Project Report

Francisco [fc55555@alunos.fc.ul.pt](mailto:fc55555@alunos.fc.ul.pt)

Vasco Cruz [fc48760@alunos.fc.ul.pt](mailto:fc48760@alunos.fc.ul.pt)

Artificial Intelligence in Games

The final project for the Artificial Intelligence in Games class involves developing artificial intelligence components for a game of our choosing.

That game is Little Rogue, a "top-down role-playing dungeon-crawler rogue-like" filled with items, enemies, and different levels with procedurally generated maps.

The game in question is relatively simple in terms of mechanics and design components of it so, to achieve simplicity of implementation and modularity of alteration and iteration of the artificial intelligence component, we decided to implement it with the use of "Behaviour Trees".

To define it, we started by determining the essential actions to be able to play and advance in this game: pick up items, use such items, attack, move, progress in level.

After defining the essential actions, we define conditions to perform them and, through them, we define the checks to be done before performing actions.

Checking the hit points and the number of potions available before deciding to use a potion to restore health points to the character, or checking if the character does not find itself on top of the “stairs” object before performing the action of progressing to the next level of the game are examples of checks made each turn to decide what action to perform.

By optimizing both actions and checks, we arrived at the following decision tree:

* In terms of behavioral flaws, we found nothing that could be classified as such. However, we were able to identify situations where the character's behavior could be improved and made more complex and adaptable to various situations in the game.
* The identification of the superiority of an enemy, apart from its rank, in terms of damage caused by it could be taken into consideration to be able to decide on the use of certain items such as scrolls to increase the damage done to the enemy, so as to end the battle faster in order to avoid taking too much damage and prepare for future confrontations, It's an example of increased complexity and adaptability to the game, but we didn't consider it a behavioral flaw.

We were able to add an extra component to the project so that we could demonstrate how the implemented code works. The visualization of the operation of the search algorithm A\* demonstrates how the "pathmaking" was implemented and how it performs, also as well as its performance:

* In order to improve the path created, this process is applied to each turn, making the pathmaking more able to adapt to changes in the terrain, such as the elimination of an enemy that was between the player character and the stairs that give access to the next level. Before, the algorithm would make the player character circumvent this space even after eliminating the enemy, because the path would have been created before the event and was not able to respond to it. With the update of the path at each turn, it is possible to change the same, making it able to respond and adapt to such changes.