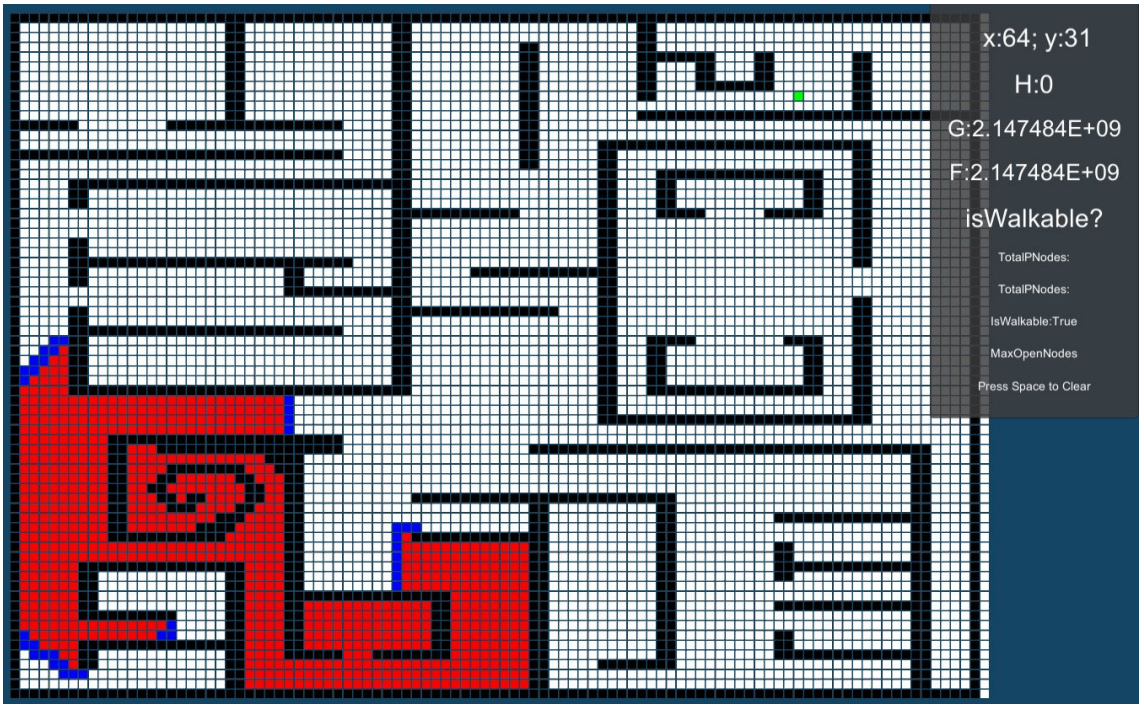
Inteligência Artificial para Jogos

2020-2021

TÉCNICO LISBOA

2nd IAJ Project – Efficient pathfinding in Room based scenarios and Path Following



In the second practical project of the IAJ Course, we want you to create a very efficient pathfinding algorithm for room based scenarios (where the traditional distance based heuristics fail to properly guide the search algorithm). The second goal is then to create a simple pathfollowing algorithm to make the character follow the path.

For this second project, you will need to write a report up to 4 pages, explaining and justifying the decisions made by your group in terms of implementation.

You should submit a zip file with both Unity source code and with the report (a pdf/doc file) via Fenix, **until 23:59 of November 6th**. We will accept submissions after the deadline, but with a 0.5 value penalty for each hour after the deadline. If the size of the Unity Project is too large for Fenix please, just like the past project, send an email with a download link to manuelguimaraes@edu.ulisboa.pt with the subject “Project 2 Group XX”

Level 1– Traditional A*.

- Implement the A* search algorithm.

Level 2– Node Array A*

- Implement the Node Array A* search algorithm.

Level 3 – JPS+

- Jump point search (JPS+) is a recently devised optimal pathfinding algorithm that can speed up searches on uniform cost grid maps by up to an order of magnitude over traditional A*
- Implement the JPS+ according to the slides and the paper in the course's website.
- You can change the NodeRecord class at your will or create a similar one.

Level 4 – Comparing the pathfinding algorithms

- Analyse the differences in performance between the original A* algorithm (w euclidean distance), the original A* algorithm (w euclidean distance and tie-breaking), the NodeArray A* algorithm (w Euclidean distance), the JPS+ algorithm. Consider aspects such as fill, nodes visited, size of the open and closed list, total processing time and processing time per node (and other aspects if relevant). You can also use the tables from Lab4. Which version has the best performance and why? Include this analysis in the report.

  PS: Make sure you use a grid with a lot of nodes to better compare the results between algorithms.

Level 5 – Path following

- This level can be implemented in parallel with levels 2-5. The goal is to implement the dynamic Follow Path movement algorithm to make the character to follow the path returned by the pathfinding algorithm.
- You can use your previous project to help you. Once the pathfinding is finished, spawn a car/character/object in its initial position and iterate through the positions using DynamicArrive/DynamicSeek.

Level 6 – Optimizations [Bonus Level]

- Select 2-3 relevant method optimizations that can be made in your code and implement them. Justify why they are important or not, create a table with the method's execution time and number of calls before the optimization and after. Finally, briefly discuss the results obtained.