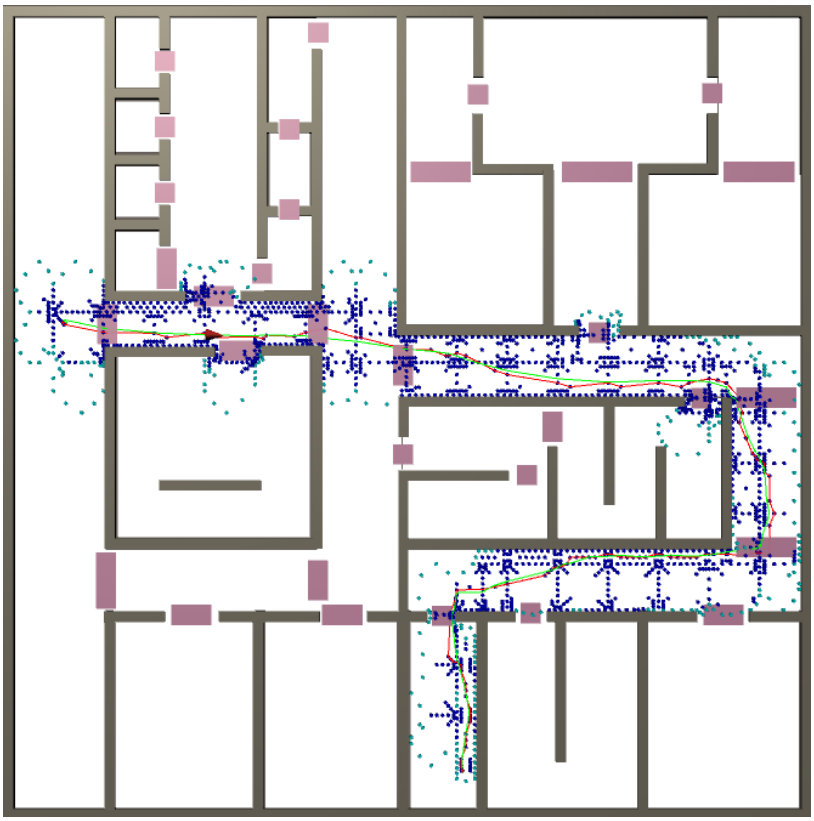
TÉCNICO LISBOA

Inteligência Artificial para Jogos

2016-2017

2.º 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. You can (and should) reuse part of your work from the Lab classes from 1-4. For this second project, you will need to write a report up to 3 pages, explaining and justifying the decisions made by your group in terms of implementation. For instance, describing and justifying the strategy used for the quantization of nodes in the Cluster Graph.

You should submit a zip file with both Unity source code and with the report (a pdf/doc file) via Fenix, **until 24:00 of October, 31**. We will accept submissions after the deadline, but with a 0.5 value penalty for each hour after the deadline.

This project includes a new Unity Project. You should implement your code in the new Unity Project provided because it contains a scene with manually created GameObjects that are used to represent Clusters and Gateways.

Level 1– Traditional A*.

- Implement the A* search algorithm[1].

Level 2– Node Array A*

- Implement the Node Array A* search algorithm.

Level 3 – Gateway Heuristic

- Implement the Gateway heuristic according to the theoretical classes. To do this, you need to create a cluster graph that includes amongst other things a table with shortest distances between gateways. Part of the cluster graph is already built, but you need to implement the rest. This is done in the Assets/Editor/IAJMenuItems class. To better understand the structure of the cluster graph take a look at the classes inside the pathfinding/datastructures/hpstructures. You also need to implement the Quantize method inside the ClusterGraph class. Please note that you can save additional information to the ClusterGraph if you would like.
- Once the cluster implementation is completed you can test it by activating the editor's top menu IAJ/Create Cluster Graph. It will take a while to compute the Cluster Graph with the Gateway distance table. We recommend you to use the NodeArray A* search for this since it involves nearly 1000 searches. At the end of this process, a new Asset named ClusterGraph will be created (usually inside the Assets folder). **Move** the created ClusterGraph Asset to the Resources folder.
- After the Cluster Graph Asset is created, you can focus on creating the actual Gateway Heuristic function.

Level 4 – Comparing the pathfinding algorithms

- Analyse the differences in performance between the original A* algorithm (w euclidean distance), the NodeArray A* algorithm (w Euclidean distance), and the NodeArray A* with Gateway Heuristic. Consider aspects such as fill, nodes visited, total processing time and processing time per node (and other aspects if relevant). Which version has the best performance and why? Include this analysis in the report.

Level 5 – Path following

- This level can be implemented in parallel with levels 2-4. The goal is to implement the dynamic Follow Path movement algorithm to make the character to follow the path returned by the pathfinding algorithm.
- For the implementation of this level you will need to use an auxiliary method in the MathHelper class. Take a look at it.
- You will need to first complete the implementation of the Path classes in the Pathfinding/Path directory. The idea is that a local path corresponds to the line segment between two graph nodes, while the global path is the collection of all of these line segments from the start node to the goal node.
- This is a very frequently asked question: **What is the param for a global path/local path?** The param represents the current distance traversed in the path. In the case of a global path, you can use a value such as 2.45 as the param. The integer part will represent the local path inside the global path that the param refers to (e.g in the case of 2.45, 2

[1] If you implemented it in the previous Labs, you can just copy the pathfinding classes to the new Unity Project. Don't forget to copy the Open/Closed data structures. However, **do not replace** the code for the **Path classes**.

represents the third local path, starting at 0), while the decimal part represents the percentage of the local path traversed so far. So, param 2.45 would represent that 45% of the third local path has been traversed. You can use other representations for the param if you would like, but this is a valid one.

- Once the Path classes are completed, copy your DynamicArrive movement implementation from the initial labs to the DynamicMovement folder. Implement the DynamicFollowPath class.

Level 6 – Optimizations

- Select 1 -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.