Aerial Robotics Kharagpur Documentation Template

Neha Dalmia

***Abstract*— In this take I have tried to use Dijkstra's algorithm to navigate a player about a gamespace with severzl obstacles and jump restrictions.**

Path planning algorithms and C-Space rendering are important components of autonomous robotics. Applications of path planning can range from simply trying to navigate a robot in a given direction to using drones to navigate through obstacles for military purposess as well. Path planning is an important component which determines how your robot navugates through obstacles in real world. Path planning algorithms involve dividing an entire game space into nodes such that the path obtained by following any of their connections is valid and free of obsactles. In a C space, the obstacles are or areas that should not be traversed are marker by 0s and the valid path is marked by 1s. Navigating through this space involves choosing the shortest path that can be obtaines by following the 1s.

I. INTRODUCTION

For this task we were given a gave environment and a basic strategy to navigate about the same environment. We had to come up with a strategy to make it move in the given environment and also train it to pick up a weapon and fight. I was not able to make much progress in this task due to lack of knowledge about path planning algorithms. I have tried to make the player reach the top of the game board after which it can go everywhere. The code that I have come up with works for simple game boards but only for a certain cases of the complex game board due to two reasons. I have not coded anything with respect to the jump pads and I my player gets stuck in cases when it has to make a backward movement ie it is capable of reaching its goal only when it has to move in the same direction. These are very major problems. I have scattered **nodes** across the gameboard taking into consideration the positions **walls, platforms and ladders** that are scattered across the game. The bot then chooses the **node which is closest to it and can be accessed via jumping**. Thus platform and wall nodes and ladder are treated differently.

II. PROBLEM STATEMENT

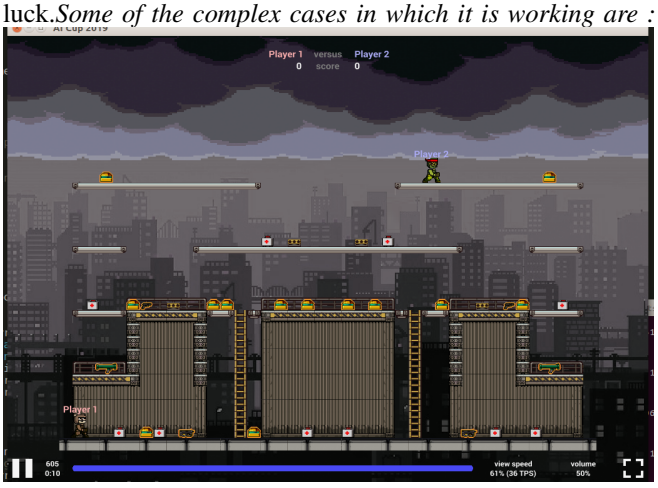
The problem statement provided us with a game environment from Russianaicup 2019 and we had to program the player using path planning algorithms and other techniques to navigate about the game space. Gravity had ti be taken into consideration which heavily limited the areas the player can access in a given turn. The game board was highly complex with obstacles like walls and components like jump-pads, ladders platforms, walls and empty spaces. We had to create a strategy to win the game. There were weapons and an enemy unit as well. The default strategy given to us was very simple first we had to approach the nearest weapon and once we acquired it we had to move towards the enemy. If the target's height y coordinate was more than us , we jump and our velocity is towards the target.

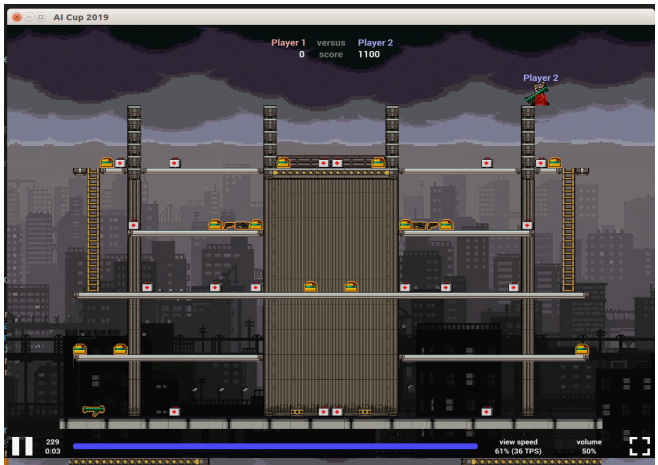
III. INITIAL ATTEMPTS

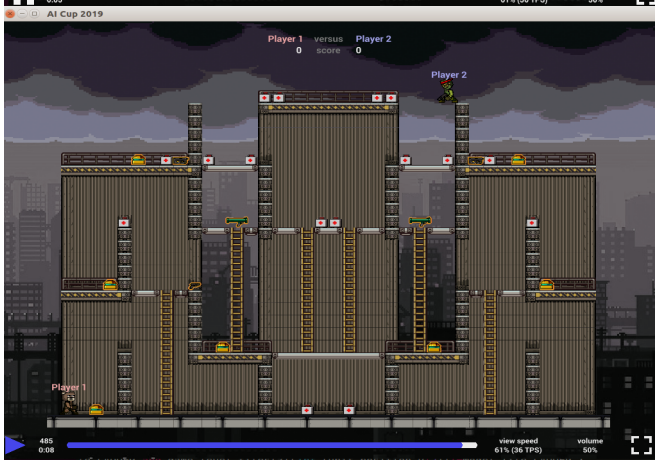
Initially I tried to approach the problem without using anu kind of path planning algorithm. The player will move in a straight line till it reaches a place it can jump to. If it can jump, it jumps and then the process repeats itself. This was extremely problematic as the player ended up getting stuff in several situation and could use only platforms and walls not ladders.

IV. FINAL APPROACH

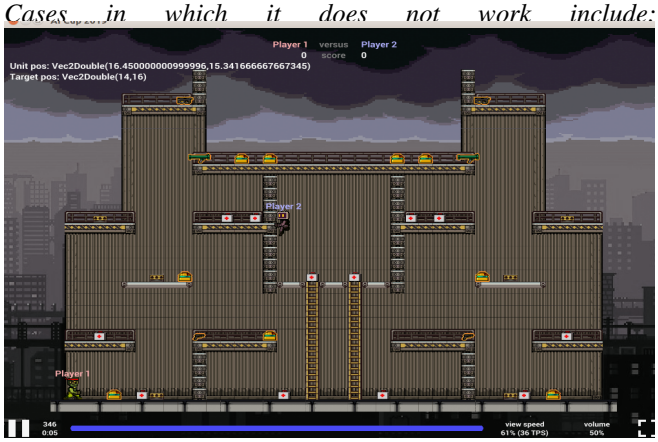
Finally, I decided to denote the various obstacles and components of the game play via nodes. Nodes exist at the both ends of a horizontal wall, in the middle of a horizontal platform and in the beginning of upward ladders. The program then computes the node which is closest to the position of the unit and which is reachable to a certain extent. I stored the nodes in an array and accessed them via loops. The detection of nodes included the ladders and platforms and walls. I also added a clause that if there is upward ladder in a row, which can be accessed ie it is not blocked by a wall then we go to that node first. Downward ladders are not considered as of now as as of now we are not allowing unit to move downwards. Platform nodes are the most easily reachable. There are numerous problems with this code thus it fails for many complex maps. Firstly, the inability to move downwards is problematic as there are situations where the player is stuck in a box like environment with only escape from a downwards ladder. Secondly, it gets stuck under walls and is unable to escape. Thirdly, it cannot use the jump-pad and its usage happens only by luck.*Some of the complex cases in which it is working are :*

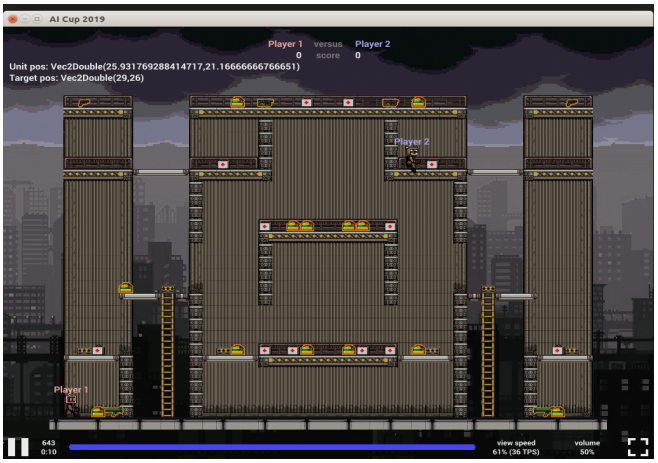






Cases in which it does not work include:





V. Results and Observation

The result of this task, the way I have implemented it gives very bad results. It is unpredictable, and for complex cases it rarely works. It can be improved upon by creating a **C-Space** consisting of all the points that it can access and then appointing nodes based on the 1s in the C space and the making the player navigate through these nodes. A lot more cases involving jump pads and walls need to be considered which I have not done yet. The accessibility of the nodes is more important than the actual configuration of the nodes, and usually all path planning algorithms give the distance as the main criterion. Also, as opposed to traditional cases of path planning, there are paths which do not have any obstacles but still cannot be accessed due to limitations in the jump height. Also the obstacles are of different categories as well and additional items are present like platforms and ladders and jumppads. Due to these changes this problem differs heavily from all the traditional path planning algorithms that currently exist.

VI. Future Work

A lot of work can be done in this, mostly by using a combination of C Spaces and A* or Dijkstra's algorithms.

Conclusion

This task differed heavily from the conventional path-planning algorithms that exist. Thus understanding what this problem means and how and to what extent it differs I had to read up a lot on different path planning algorithms which increased my knowledge on this topic considerably. Path planning can be highly convoluted and this task was a challenging application of the basics of path planning.