Path Finding on a specific scenario

Shafali Jagan Akula

Indiana State University

sakula2@sycamores.indstate.edu

December 15, 2014

Abstract

Path finding as defined in the field of computer science is the method of finding the optimal route between any two given points in a map, and it has been the most widely used concept in many gaming applications. The path finding techniques that have been implemented so far, have conventionally made use of grid space representations. This paper, however, focuses on a new approach of establishing the underlying world based on geometric information.

1 Introduction

This paper has been published by S.D.Goodwin, S.Menon, R.G.Price. Path finding is an essential concept that many game developers use as part of their projects. It is a problem of determining the shortest path between two(start and stop) given nodes. There are many approaches to finf the optimal path between two points, but, te most widely used algorithm is the A* algorithm. Consider a given heuristic, no algorithm This paper has been published by S.D.Goodwin, S.Menon, R.G.Price. Path finding is an essential concept that many game developers use as part of their projects. It is a problem of determining the shortest path between two(start and stop) given nodes. There are many approaches to finf the optimal path between two points, but, te most widely used algorithm is the A* algorithm. Consider a given heuristic, no algorithm can find a better solution than the A* algorithm. Yet, it imposes some major resource impact in terms of time and space.

Numerous attempts to optimize this technique(A*) have been approached by users. The major components of the algorithm that are subjected to change are the heuristic cost funtion and the space representation. The heurictic cost function is the estimated distance of any particular node to the goal node in an area. This can be calculated by using various mathematical formulas such as Vancouver distance, Manhattan distance and teh Euclidean distance. The space representation is also a significant factor that contrubutes to the optimization of the alogirthm. Various space representations have been put to test such as the traditional grid representation, navigation mesh, waypoint graph, etc.

Conventionally the underlying world has been modelled as a grid in the computer field, thereby associating each node of a grid to that of an input graph, further the graph is representing by applying suitable data structures according to the level of optimization, in the algorithm. Although this approach has been practiced globally for path finding implemntations, one cannot deny that the grid representation fails to import important spatial information thereby presenting just an approximation of an optimal solution. This diminshes the path accuracy majorly.

1.1 Results

In order to see how efficient is Brep when compared to BPrep, both the Brep and BPrep were experimented. These are the results that were observed.

The paper has presented results for 4 test cases. The basic values that have affected the performance are:
/beginitemize /item Map size: The map size which is 65536 world units has been fit to 600 by 600 on screen, which means that every pixel is about 2.8 meters wide.

/item Region: The number of regions that the map is divided into.

/item Threshold: The number of times BPSolver is computed(determines the neighbor nodes) to integrate the cost function.

/item Path cost: The total cpst of the sortest path.

/item Nodes expanded: The number of nodes whose neighbors are determined.

1.1.1 BPrep

Test Case 1
Region: 8
Threshold: 32
Path Cost: 1302.4
Time: 795.66s
Nodes Expanded: 2287

Test Case 2
Region: 24
Threshold: 32
Path Cost 2057.08
Time:4579.8s
Nodes expanded: 338

1.1.2 Brep

Test Case 1
Region: 8
Threshold: 32
Path Cost: 1302.4
Time: 0.02s
Nodes Expanded: 43

Test Case 2
Region: 24
Threshold: 32
Path Cost: 2057.06
Time:0.41s

Nodes Expanded: 363

1.2 Organization

In the following pages, you will find the Definitions and Background section which will give you an idea about the basic terms that have been used in the paper and their definitions.The section Approach will give you an idea about what how the authors have worked their way through the project. Psuedocode explains the basic approach to an A* algorithm only, however, the real implementation of Brep and BPrep will require more modifictions than this usual approach.

2 Definitions and Background

2.1 A* algorithm

Following are the fundamental definitions that are concerned to the basic A* algorithm

Definition 1 (G Score). *It is the traveling cost from the start node to the current node.*

Definition 2 (H Score). *Heuristic estimate from current node to Goal Node.*

Definition 3 (F Score). *The sum of gscore and hscore.*

Definition 4 (BPSolver). *An acronym for boundary point solver, is a function used to compute the most optimum boundary point of a boundary.*

Definition 5 (Coordinate descent). *It is a non-derivative optimization algorithm which finds a local minimum of a function. It does line search along one coordinate direction at the current point in each iteration.*

3 Approach

As a starting point, consider a representation that is a hybrid of framed quadtrees and the convex polygon approach. Instead of treating the regions as nodes in the search graph, let the polygon edges be divided into a number of points based on the desired resolution. Take each of these points as nodes in the search graph. Nodes are connected if they correspond to points in the same polygon (points on the boundary between two polygons are each represented by a single node that belong to both polygons). The cost of the edges is now simply the Euclidean distance between them times the cost of the region covered by their polygon. Clearly, the properties of search using this representation are similar to that of framed quadtrees but with some additional flexibility from using convex polygons rather than quads. Such an approach would be useful for open terrain environments except that, as the resolution increases, the search cost increases dramatically. To seek an improvement to this, consider the fact that the optimal path in a given search graph passes through the boundaries of a sequence of polygons. The path crosses each boundary at a point (or possibly through multiple points on a boundary). To determine which boundary points to cross the f-score for each corresponding node is calculated and nodes are selected in the usual A* fashion.

4 Pseudocode for A* Algorithm

```
function A*(start,goal)
    closedset := the empty set    // The set of nodes already evaluated.
    openset := {start}    // The set of tentative nodes to be evaluated, initially containing the
    camefrom := the empty map    // The map of navigated nodes.

    gscore[start] := 0    // Cost from start along best known path.
    // Estimated total cost from start to goal through y.
    fscore[start] := gscore[start] + heuristiccostestimate(start, goal)

    while openset is not empty
        current := the node in openset having the lowest fscore[] value
        if current = goal
            return reconstructpath(camefrom, goal)

        remove current from openset
        add current to closedset
        for each neighbor in neighbornodes(current)
            if neighbor in closedset
                continue
            tentativegscore := gscore[current] + distbetween(current,neighbor)

            if neighbor not in openset or tentativegscore < gscore[neighbor]
                camefrom[neighbor] := current
                gscore[neighbor] := tentativegscore
                fscore[neighbor] := gscore[neighbor] + costestimate(neighbor, goal)
                if neighbor not in openset
                    add neighbor to openset

    return failure

function reconstructpath(camefrom,current)
    totalpath := [current]
    while current in camefrom:
        current := camefrom[current]
        totalpath.append(current)
    return totalpath
```

5 Future Work

Even though the results have proved that the solution or finding the shortest path on an open terrain works, there is a possibility that it may fail once the size of the map is increased. In order to combat this problem, hierarchial approaches have been adopted. Hierarchial abstraction in search, problem solving, and planning, works by replacing one state space by another that is easier to search. It would be interesting to integrate this method in a hierarchical approach. In fact, to a degree it will be absolutely necessary. This is because this approach relies on uniform cost regions. In the typical maps,even open areas are not completely uniform. A field that contains some isolated rocks and trees can be treated as a uniform cost region at a higher level of abstraction and paths around the trees and rocks can be resolved at a lower level of the hierarchy.

Acknowledgements

This study is the work done under the guidance of Professor Jeffrey Kinne, as a main project of the course Computer Science Research. It is a study of a research paper produced by S.D Goodwin, S.Menon and R.G.Price called Path Finding on Open Terrains.

References

- Path finding on open terrains
- eitemn.wikipedia.org/wiki/Pathfinding
- im ite,A. Botea, M. MÃ$\frac{1}{4}$ller and J. Schaeffer. (2004). Near optimal hierarchical path-finding. Journal of Game Development
- R.em chter and J. Pearl. (1985). Generalized bestfirst search strategies and the optimality of A*. Journal of the Association for Computing Machinery,
- A. Yahja, A. Stentz, S. Singh and B. Brumitt. (1998). Framed-quadtree path planning for mobile robots operating in sparse environments. In Proceedings, IEEE Conference on Robotics and Automation