Daniel Revie

Raghav Bardwaj

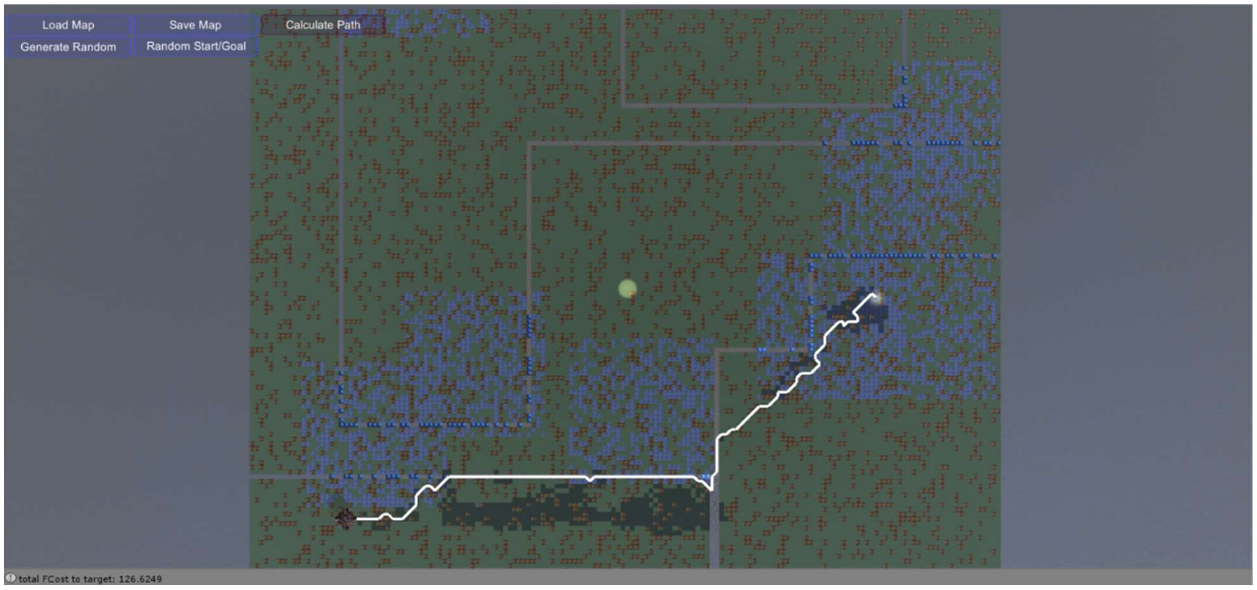
02/05/2017

Assignment 1

B) Algorithms Used

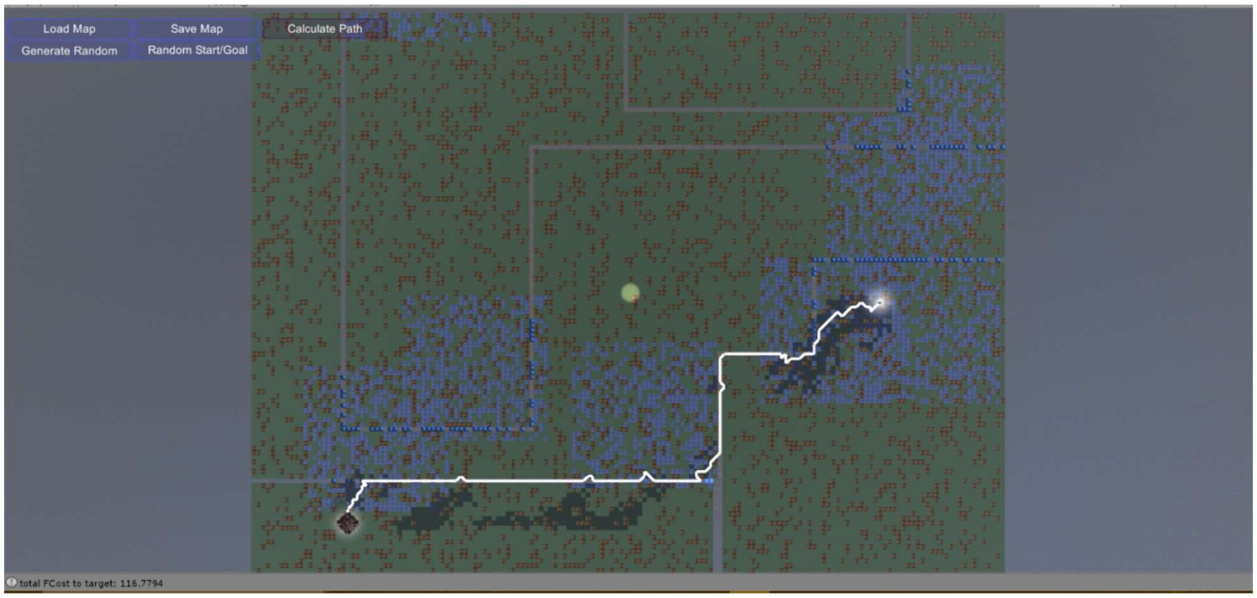
The following screenshots prove that the 3 algorithms used work to find a path between a target and a source:

A*:



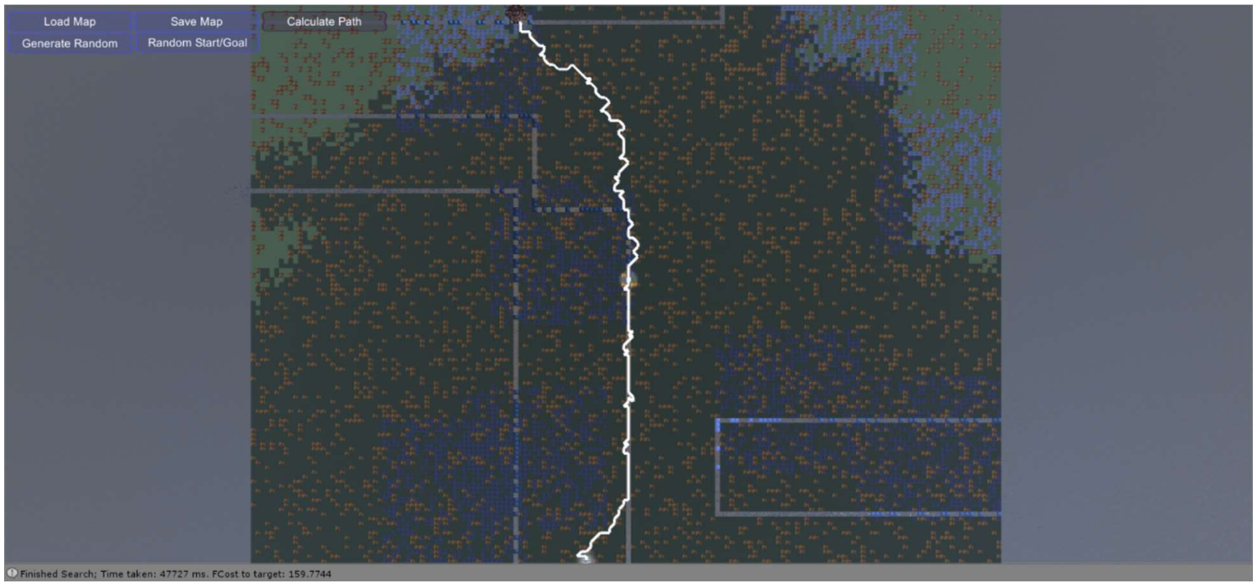
```
total FCost to target: 126.6249
```

Weighted A*



```
total FCost to target: 116.7794
```

UCS



```
Finished Search; Time taken: 47727 ms. FCost to target: 159.7744
```

Optimizations

During the construction of the project a number of steps were taken for optimization.

1. **Containers**: Containers were picked based on the positives they brought to the program
    a. HashSet: A hash set has the standard operations such as Add, Remove, etc. Because it uses a hash-based implementation the operations are O(1) which is optimal. A list used in this place would be O(n) for such operations. When considering the massive size of the grid map this saves a significant amount of time for the operations. In general, a HashSet is much faster than a list.
    b. Heap: The heap allows quick access to the highest priority item.

c. Dictionary: the dictionary structure relies on chaining to resolve collisions. A dictionary is especially efficient when considering a list of two related integers or in the case of the project nodes, which are related.

Heuristics Used

It is possible to implement alternative heuristics to optimize the search. Listed below are five heuristics used.

1. **Manhattan:** often considered the standard heuristic for a square grid. This function computes the minimum cost for moving to an adjacent space. Here we can choose D to the lowest cost between adjacent squares.
$$h(n) = D * (|n.x - target.x| + |n.y - target.y|)$$
2. **Diagonal Distance UC:** this heuristic is useful for 8-way movement when the cost of diagonal movement is equivalent to that of the non-diagonal movement. This heuristic becomes quite inaccurate if the costs of diagonal and non-diagonal are not the same.
$$h(n) = c * max\ (|n.x - target.x|, |\ n.y - target.y|)\ , where\ c = cost\ of\ movement$$
3. **Diagonal Distance:** this heuristic is similar to the uniform cost diagonal cost heuristic but in this case the diagonal and the non-diagonal costs do not have to be the same.
$$h_{diagonal}(n) = min\ (|n.x - target.x|, |n.y - target.y|)$$
$$h_{straight}(n) = |n.x - target.x| + \lceil n.y - target.y \rceil$$
$$h(n) = (Weight * 2 * h_{diagonal}) + Weight * (h_{straight} - 2 * h_{diagonal})$$
4. **Euclidean Distance:** in the case of non-grid movement the Euclidean distance heuristic becomes very efficient. However, in this case there is grid movement. This heuristic calculates the straight line distance to the target. It however will be slower than the Manhattan or diagonal distance, but will yield the shortest path.
$$h(n) = \sqrt{(n.x - target.x)^2 + (n.y - target.y)^2}$$
5. **Euclidean Distance Squared:** for this heuristic the square root is not taken and h(n) is left squared. To make the g(n) the same relative to h(n) for the calculation of f(n) the g(n) value must be squared as well.
$$h(n)^2 = (n.x - target.x\ )^2 + (n.y - target.y)^2, make\ sure\ g(n)\ is\ squared$$
6. **Chebyshev Distance:** all eight adjacent cells are considered in this heuristic and are reachable by an equivalent cost.
$$h(n) = max(\ |n.x - target.x\ |, |n.y - target.y|)$$

Evaluation of Benchmarks

50 Benchmark grids were created with different start/end points. They were tested against different heuristics and a table of the data created. Since the data content was huge, we decided to compute averages of the benchmarks and display them instead. The averages of the benchmarks are provided below:

Average A* Results:

	Manhattan	Diagonal Shortcut	Euclidean	Euclidean NoSQRT	Chebyshev
Time(ms)	6182.348	6549.308	12985.54	2826	27025.62
FCost	129.8195	129.7386	129.924	164.4343	153.6759
NodeExpansion	712	712	2727	158	15237

Average Weighted A* Results:

	Manhattan	Diagonal Shortcut	Euclidean	Euclidean NoSQRT	Chebyshev
Time(ms)	6380.111	7244	14161.2	6334.6	25346
FCost	132.0261	141.5506	135.2366	140.441	141.7918
NodeExpansion	556	550	620	131	3975

Uniform Cost Search

Average FCost: 174.066

Average Time: 50312.5 ms

Average Node Expansion: 15193

Results

After analyzing the results of the benchmarks it became easier for us to determine how the performance and efficiency varied among algorithms. We also tested multiple heuristics within the weighted and unweighted A star algorithms. We found that the Manhattan and Diagonal heuristics produced the best results for the weighted A Star and the unweighted A Star; they were quire reliable as well, finding a suitable path nearly every time. The Euclidean heuristic was the next best following Manhattan and Diagonal. Euclidean was not as efficient and optimal as the previous two. The Euclidean with no square root was able to quickly identify a path but ignored the penalty of movement and generated an inefficient path from start to goal. The Chebyshev heuristic was slightly more efficient that Euclidean no square root but often got lost during its search and consumed a large amount of time. The weighted a star algorithm outperformed the regular A star algorithm. It often found a route just as efficient or more efficient in less time than regular A star.