What are the algorithms used in this project?
->Breadth First Search & Uniform cost search & Best first search & A* algorithm

Describe the algorithms briefly based on your understanding.
->Breadth First Searchはオープンリストにキューを使って、未訪問の隣接頂点を訪問しながら、キューから頂点を繰り返し削除します。
Uniform cost searchはオープンリストにキューを使い、現在の頂点からの実際のコストが小さい順にソートする。そしてキューから取り出した頂点を訪問する。
Best first searchは任意に決めたヒューリスティック関数の値が小さい頂点を優先して探索する。
A* algorithmは現在の頂点からの実際のコストと任意に決めたヒューリスティック関数の値の合計が小さい順に訪問する。

Explain the meaning of the results obtained by running the completed program.
->Breadth First SearchはDepth First Searchよりも合計コストが少ない道を選べるが、Depth First Searchよりも探索ステップがかかることがある。
Uniform cost searchは合計コストを小さく抑えることができるが、探索ステップが長くなることがある。
Best first searchは探索ステップが他より短くなることがあるが、合計コストが大きくなりやすい。ヒューリスティック関数を使って探索をするのでヒューリスティック関数によって性能が変わりやすい。
A* algorithmは合計コストも探索ステップも小さく抑えることができるが、Best first searchの方が探索ステップが短い時もある。

Explanation of modified points (required)
TODO:1
s_queue_init(NULL);
s_queue_init(accumulate_costs);
s_queue_init(g->heuristic_values);
s_queue_init(eval_values);
それぞれのアルゴリズムに使う評価値をキューに入れる。
TODO:2
*child_vertex_state == UNVISITED
訪れていない頂点を見つけて訪問する。
TODO:3
*child_vertex_state == UNVISITED
TODO:2と同じ。
TODO:4
child_eval_value = g->heuristic_values[i] + child_accumulate_cost;
A* algorithmは現在の頂点からの実際のコストと任意に決めたヒューリスティック関数の値の合計を評価値とするのでその記述。
TODO:5
*child_vertex_state = IN_QUEUE;
s_queue_enqueue(i);
*child_vertex_state = VISITEDでも、ヒューリスティック値などによりもっと良い道が見つかることがあるので*child_vertex_state = IN_QUEUEして元に戻す。

Discussion (if needed)

Comments (if needed)