Práctica 4. Exploración de grafos

Manuel Diaz Gil
manuel.diazgil@alum.uca.es
Teléfono: 667361517
NIF: 45382945N

29 de enero de 2018

1. Comente el funcionamiento del algoritmo y describa las estructuras necesarias para llevar a cabo su implementación.

   Algoritmo usado A* que calcula el mejor camino para el uco segun sus posibilidades, recoriendo los nodos y expandiendose segun su valor.

   Estructuras de datos usadas monticulo para ordenar el mejor nodo y set para guardar los nodos abiertos y usados, y asi poder comprobar cuales se han recorrido ya y cuales quedan por recorrer.

2. Incluya a continuación el código fuente relevante del algoritmo.

```
std::priority_queue<AStarNode*> opened;
std::set<AStarNode*> closed,open;

originNode->G = 0;
originNode->H = abs((originNode)->position.length()
- targetNode->position.length());

originNode->F = (originNode)->G + (originNode)->H
+  additionalCost[(int) ((originNode)->position.y / cellsHeight)]
[(int) ((originNode)->position.x / cellsWidth)];

opened.push(originNode);
int maxIter = 100;
AStarNode* current = originNode;

while(current != targetNode && maxIter > 0 and !opened.empty()) {
    current = opened.top();
    opened.pop();
    float min = INF_F;
    if(current != targetNode){
        for (List<AStarNode*>::iterator it=current->adjacents.begin();
         it != current->adjacents.end(); ++it){

            if(closed.count(*it) == 0){
                if(open.count(*it) == 0){
                    (*it)->parent = current;
                    (*it)->G = current->G + abs((*it)->position.length()
                     - current->position.length());

                    (*it)->H = abs((*it)->position.length()
                     - targetNode->position.length());

                    (*it)->F = (*it)->G + (*it)->H
                    + additionalCost[(int) ((*it)->position.y / cellsHeight)]
```

```
                        [(int) ((*it)->position.x / cellsWidth)];

                    opened.push((*it));
                    open.insert((*it));
                }else{
                    min = abs((*it)->position.length() - current->position.length());
                    if((*it)->G > (current->G + min)){
                        (*it)->parent = current;
                        (*it)->G = current->G + min;
                        (*it)->F = (*it)->G + (*it)->H
                        + additionalCost[(int) ((*it)->position.y / cellsHeight)]
                        [(int) ((*it)->position.x / cellsWidth)];

                    }
                }
            }
        }

    }
    closed.insert(current);
}
current = targetNode;
path.push_front(current->position);
while(current->parent != originNode){
    current = current->parent;
    path.push_front(current->position);
}
--maxIter;
```

Todo el material incluido en esta memoria y en los ficheros asociados es de mi autoría o ha sido facilitado por los profesores de la asignatura. Haciendo entrega de esta práctica confirmo que he leído la normativa de la asignatura, incluido el punto que respecta al uso de material no original.