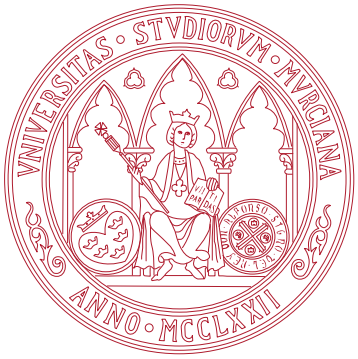

Universidad de Murcia

Facultad de Informática

Proyecto de IA para el Desarrollo Videojuegos

Profesores: Luis Daniel Hernández Molinero
Francisco Javier Marín-Blázquez Gómez

Vocal: Sergio Marín Sánchez , Grupo: 1.2
Email: sergio.marins@um.es

Jose Miguel Sánchez Fernández, Grupo: 1.2
Email: josemiguel.sanchezf@um.es

Gaspar Muñoz Cava, Grupo: 1.3
Email: gaspar.munozc@um.es

Fecha: 17 de junio de 2022

Curso: 2021/22

Índice de contenidos

Índice de figuras

1. Pathfinding táctico individual

En esta sección se explicará el funcionamiento del pathfinding táctico individual de los personajes. Para esta labor se ha hecho uso del algoritmo A* (1).

Algoritmo 1 Algoritmo A*

procedure A*(`grid`, `inicio`, `final`, h) ▷ h es la función heurística admisible
  Poner `inicio` en lista de `ABIERTOS` con $f($`inicio`$) = h($`inicio`$)$
  while lista de `ABIERTOS` no esté vacía **do**
    Obtener de la lista de `ABIERTOS` el nodo `actual` con menor $f($`nodo`$)$
    if `actual` = `final` **then** ▷ Se ha encontrado una solución
      break
    end if
    Conseguir todos los nodos `sucesor` de `actual`
    for cada `sucesor` de `actual` **do**
      Establecer `coste_sucesor` $= g($`actual`$) + w($`actual`$,$ `sucesor`$)$ ▷ $w(a, b)$ es el coste del camino entre a y b
      if `actual` está en la lista de `ABIERTOS` **then**
        if $g($`sucesor`$) \leq$ `coste_sucesor` **then**
          continue
        end if
      else if `sucesor` está en la lista de `CERRADOS` **then**
        if $g($`sucesor`$) \leq$ `coste_sucesor` **then**
          continue
        end if
        Mover `sucesor` de la lista de `CERRADOS` a la de `ABIERTOS`
      else
        Añadir `sucesor` a la lista de `ABIERTOS`
      end if
      Establecer $g($`sucesor`$) =$ `coste_sucesor`
      Establecer `actual` como nodo padre de `sucesor`
    end for
    Añadir `actual` a la lista de `CERRADOS`
  end while
  if `actual` $\neq$ `final` **then** ▷ No se ha encontrado camino
    Terminar con error.
  end if
end procedure

Este algoritmo se ha implementado casi de manera literal. Su mayor cambio viene por la parte de calcular el coste del sucesor. En este caso no sólo se ha tenido en cuenta el coste de desplazarse del nodo actual al vecino, sino que se ha tenido en cuenta el tipo de terreno así como la influencia enemiga. Por lo tanto el código implementado sería:

```
37                float newCost = current.GCost + agent.GetTerrainCost(neighbour.
    TerrainType);
```

```
38
39                  /*
40                  float influenceValue = neighbour.InfluenceValue;
41                  switch (agent.Team)
42                  {
43                      case Teams.TeamA:
44                      {
45                          newCost -= influenceValue;
46                          break;
47                      }
48                      case Teams.TeamB:
49                      {
50                          newCost += influenceValue;
51                          break;
52                      }
```