Práctica 1. Algoritmos devoradores

Alvaro Santos Romero
alvaro.santosromero@alum.uca.es
Teléfono: xxxxxxxx
NIF: xxxxxxxxm

8 de noviembre de 2021

1. Describa a continuación la función diseñada para otorgar un determinado valor a cada una de las celdas del terreno de batalla para el caso del centro de extracción de minerales.

   Para asignar un valor a la mejor celda para la colocación del Centro de Extracción de Minerales, primero se ha dividido el tamaño del tablero en 2 (tanto de alto como de ancho), para evitar que este sea colocado cerca de los límites del mapa.

   Posteriormente, se han ido multiplicando los valores de las celdas dependiendo de la posición de la celda. ej: celda[2][2] tendrá menor valor que celda[3][3] por estar más cerca de los límites del mapa.

2. Diseñe una función de factibilidad explicita y descríbala a continuación.

   La función de factibilidad diseñada tiene como objetivo comprobar si la colocación de las defensas es válida.

   La colocación de una defensa es válida si:

   1-. El radio de la defensa mas el centro de la celda donde está ubicada la defensa no supera los límites del mapa.

   2-. Distancia entre el centro de la celda y la posición del obstáculo es mayor que la suma de los radios de la defensa y del obstáculo.

   3-. Distancia entre el centro de la celda y la posición de la defensa actual es mayor que la suma de los radios de la defensa actual y las que están ya colocadas.

   bool factible(List¡Defense *¿ defenses, const Defense currentDef, List¡Object *¿ obstacles, float mapHeight, float cellWidth, float cellHeight, float mapWidth, int row, int col, bool **freeCells) bool fact = true;

   Vector3 posicion = cellCenterToPosition(row, col, cellWidth, cellHeight);

   /*AQUI SE COMPRUEBA QUE LAS DEFENSAS NO SE COLOCAN FUERA DEL MAPA*/

```
/*Funcion para dar valor a la mejor celda del mapa, donde se colocará el centro de extracción*/
float cellValueCEM(int row, int col, bool **freeCells, int nCellsWidth, int nCellsHeight,
float mapWidth, float mapHeight, List<Object *> obstacles)
{
    float cellWidth = mapWidth / nCellsWidth;
    float cellHeight = mapHeight / nCellsHeight;
    int bestRow = nCellsHeight / 2, bestCol = nCellsWidth / 2;
    Vector3 position = cellCenterToPosition(row, col, cellWidth, cellHeight);
    Vector3 bestPosition = cellCenterToPosition(bestRow, bestCol, cellWidth, cellHeight);
    float value;

    value = bestPosition.length() - _distance(position, bestPosition);

    if (row < 3 || row > nCellsHeight - 3 || col > nCellsWidth - 3 || col < 3)
        value = value * 0.01;

    return value;
}
```

Figura 1: Estrategia devoradora para la mina