Práctica 1. Algoritmos devoradores

Raúl Arcos Herrera
raul.arcosherrera@alum.uca.es
Teléfono: xxxxxxxx
NIF: 77175935X

14 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.

   Escriba aquí su respuesta al ejercicio 1.

   En el caso del centro de extracción de minerales se han tenido en cuenta dos factores valorar la mejor celda.

   - **Distancia respecto al centro** Se valorarán positivamente las celdas que se encuentren próximas al centro, puesto que es el punto más alejado de la aparición de los UCOS.
   - **Distancia respecto a obstáculos** Se valorará negativamente la cercanía a un obstáculo debido a que los obstáculos no protegen a las defensas de los proyectiles, por lo que es un entorpecimiento a la hora de hacer formaciones en el mapa.

   El valor final resulta de la ecuación: *1000 - Distancia al centro / Factor de distancia a obstáculo*, siendo factor de distancia a obstáculo un número del 0 al 1, donde 0 es la completa ausencia de un obstáculo mientras que 1 es distancia cero con uno.
   Esto es así debido a que ese valor depende del tamaño del mapa, es decir, no es lo mismo tener un obstáculo a 20 unidades en un mapa de 40 que en un mapa de 100.

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

   Para la factibilidad, se ha utilizado una función que devuelve un booleano que indica si una celda es factible (true o 1) o si no lo es (false o 0). La función consiste en tres filtros:

   - **Límites del mapa**, contando con el radio de la propia defensa, es decir, la suma de la posición de la celda y el radio no sobrepasa los límites del mapa.
   - **Colisión con obstáculo**, buscamos que la celda no colisione con ningún obstáculo que esté repartido por el mapa.
   - **Colisión con defensa**, A medida que vamos añadiendo defensas al tablero, es muy posible que nos topemos con celdas que colisionan con defensas, por lo que es un punto a evitar.

   Para llevar a cabo la función se ha hecho uso de las funciones auxiliares *cellCenterToPosition* y *positionToCell*, proporcionadas en el apartado de preguntas frecuentes del campus.

3. A partir de las funciones definidas en los ejercicios anteriores diseñe un algoritmo voraz que resuelva el problema para el caso del centro de extracción de minerales. Incluya a continuación el código fuente relevante.

```
Vector3 voraz(int nCellsWidth, int nCellsHeight, float mapWidth, float mapHeight, List<Object
    *> obstacles, List<Defense*> defenses, float radio, int id)
{
    float cellWidth = mapWidth / nCellsWidth;
    float cellHeight = mapHeight / nCellsHeight;
    float cellValues[nCellsWidth][nCellsHeight];
    float max = 0;
    Vector3 pos;

    for(int i = 0; i < nCellsWidth; i++){

        for(int j = 0; j < nCellsHeight; j++){

            if(factibilidad(i, j, obstacles, defenses, mapWidth, mapHeight, nCellsWidth,
                nCellsHeight, radio))
            {
                //Valor de la celda en el caso de ser la central
                if(id == 0)
                    cellValues[i][j] = cellValue(i, j, nCellsWidth, nCellsHeight, mapWidth,
                        mapHeight, obstacles, defenses,1);
                //Valor de la celda en caso de ser defensa
                else
                    cellValues[i][j] = cellValue(i, j, nCellsWidth, nCellsHeight, mapWidth,
                        mapHeight, obstacles, defenses,0);
                //Guardamos en 'max' la celda con mayor valor.
                if (cellValues[i][j] > max)
                {
                    max = cellValues[i][j];
                    pos = cellCenterToPosition(i, j, cellWidth, cellHeight);
                }
            }
        }
    }
    return pos;
}
```

4. Comente las características que lo identifican como perteneciente al esquema de los algoritmos voraces.

   Un algoritmo voraz se compone de las sigueintes características:

   - **Conjunto de candidatos**, Todas las celdas que forman el mapa.
   - **Conjunto de candidatos seleccionados**, Todas las celdas factibles.
   - **Función solución**, La función encargada de la colocación de defensas, en nuestro caso, **placeDefense**.
   - **Función de selección**, La función que selecciona la celda con un valor más alto, tanto para el centro como las defensas, en mi caso se llama **voraz**.
   - **Función de factibilidad**, Función que determina si una celda es utilizable, vista en el ejercicio 2.
   - **Función objetivo**, Para esto usamos la función CellValue, que determina un valor diferente para el centro que para las defensas con el uso de la flag *centro*.
   - **Objetivo**, Evitar durante el mayor tiempo posible que los UCOS destruyan el centro de extracción.

5. 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 resto de defensas. Suponga que el valor otorgado a una celda no puede verse afectado por la colocación de una de estas defensas en el campo de batalla. Dicho de otra forma, no es posible modificar el valor otorgado a una celda una vez que se haya colocado una de estas defensas. Evidentemente, el valor de una celda sí que puede verse afectado por la ubicación del centro de extracción de minerales.

   Una vez colocado el centro de extracción, el valor otorgado al resto de defensas irá enfocado a rodear el centro de extracción.
   En mi caso, como se comenta brevemente en el punto anterior, otorgo el valor a las celdas con la función *cellValue*, que mediante una flag determina si estamos colocando el centro de extracción o una simple defensa, por lo que una vez tenemos el centro de extracción en la posición (en mi opinión) óptima, lo rodeamos de defensas para alargar lo máximo posible el alcance de los UCOS a esta.
   La ecuación para determinar el valor es la siguiente: *value = 1000 - (distancia al centro de extracción).*
   He elegido este método debido a que al rodear al centro, hay menos huecos que no entren en el radio de ataque de las defensas por los que puedan pasar los UCOS.

6. A partir de las funciones definidas en los ejercicios anteriores diseñe un algoritmo voraz que resuelva el problema global. Este algoritmo puede estar formado por uno o dos algoritmos voraces independientes, ejecutados uno a continuación del otro. Incluya a continuación el código fuente relevante que no haya incluido ya como respuesta al ejercicio 3.

   La única función voraz con la que cuento ya fué incluida en el ejercicio 3, se puede observar que una vez pasada la factibilidad se pregunta si la id (de la defensa) es 0. // En ese mismo else, podemos ver que es el valor que se le otorgan a las celdas cuando se tiene en cuenta que son defensas, aquí hay un fragmento del código de *cellValue* que indica el valor que otorgará a las celdas cuando estamos teniendo en cuenta que son para defensas:

```
if(central)
        value = (1000 - distanceCellToCenter)/nearObstacle;
    else
    //En cambio para las defensas, buscamos que rodeen a la posici n de la central de
        extracci n.
        value = (1000 - distanceCellToDefenseZero);

    return value;
```

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 este documento confirmo que he leído la normativa de la asignatura, incluido el punto que respecta al uso de material no original.