Práctica 2:
Agentes Reativos/Deliberativos

Linqi Zhu
2D
16 de mayo de 2022

UNIVERSIDAD
DE GRANADA

1. Funciones implementados

Agentes Deliberativo :

Para nivel 0 y nivel 1:

1.Bool pathFinding_Anchura(origen, un_objetivo, plan);

Función para búsqueda de plan con algoritmo de búsqueda de anchura.

Para nivel 2:

1.bool pathFinding_CosteUniforme(const estado &origen, const estado &destino, list<Action> &plan);

Función de cote uniforme para calculo de plan.

2.struct ComparaEstadosN2;

operator () de comparación para los nodo estado.

3.struct ComparaCosteBateria;

estructura para criterio de ordenar para cola con prioridad del función *pathFinding_CosteUniforme*

4.void EsK_D(estado &st);

Función para comprovar si es una casilla bikini o zapatillas.

5.int CosteBateria(nodo &n, const Action &accion)

Devolver el coste de bateria según el movimiento tomado.

Para nivel 3:

1. Void guardarVisitado(Sensores sensores)

Guardar las casillas visitado según orientación.

2. estado colocarObjetivo(const vector<vector<unsigned char>> &mapa)

Función auxiliar para colocar objetivo.

Agentes Reactivos :

Si llega a casilla K (bikini es true) o D (zapatillas es true)

```
31      if (sensores.terreno[0] == 'K')
32      {
33          actual.llevarBikini = true;
34      }
35
36      if (sensores.terreno[0] == 'D')
37      {
38          actual.llevarZapatillas = true;
39      }
```

Para los niveles 3 y 4 la actuación de sensores de orientación y guardar las casillas ya visitadas.

```
102         // Activa sensores de orientación
103         if (sensores.posF != -1 && !bien_situado)
104         {
105             fil = sensores.posF;
106             col = sensores.posC;
107             brujula = sensores.sentido;
108             bien_situado = true;
109         }
110
111         if (bien_situado)
112         {
113             // mapaResultado[fil][col] = sensores.terreno[0];
114             guardarVisitado(sensores);
115         }
```

Para nivel 3:

```
141     if (sensores.nivel == 3)
142     {
143         if (actual.llevarZapatillas == false and sensores.terreno[2] == 'B')
144         {
145             hayPlan = false;
146         }
147 
148         if (actual.llevarBikini == false and sensores.terreno[2] == 'A')
149         {
150             hayPlan = false;
151         }
152 
153         if (objetivo_n3.fila == -1 or mapaResultado[objetivo_n3.fila][objetivo_n3.columna] != '?' 
154         {
155             objetivo_n3 = colocarObjetivo(mapaResultado);
156             if (objetivo_n3.fila == -1)
157             {
158                 cout << "mapa descubierto por completo" << endl;
159                 ultimaAccion = actIDLE;
160                 return actIDLE;
161             }
162 
163             cout << "seleccionando nuevo objetivo" << endl;
164             plan1++;
165             cout << objetivo_n3.fila << endl;
166             cout << objetivo_n3.columna << endl;
167             objetivos.clear();
168             objetivos.push_back(objetivo_n3);
169             hayPlan = false;
170         }
```

Si hay agua y no lleva bikini recalcula el plan.

Si hay bosque y no lleva zapatilla recalcula el plan.

```
186     // buscar pila
187     if (encontrarX)
188     {
189         switch (ultimaAccion)
190         {
191         case actFORWARD:
192             ultimaAccion = actTURN_L;
193             encontrarX = false;
194             return actTURN_L;
195             break;
196         case actTURN_L:
197             ultimaAccion = actFORWARD;
198             return actFORWARD;
199             break;
200         case actTURN_R:
201             ultimaAccion = actFORWARD;
202             return actFORWARD;
203             break;
204         }
205     }
206     if ((sensores.sentido == 0 || sensores.sentido == 2 || sensores.sentido == 4 || sensores.sentido == 6) && sensores.bateria < 20
207     {
208         if (sensores.terreno[5] == 'X' || sensores.terreno[4] == 'X')
209         {
210             ultimaAccion = actTURN_L;
211             encontrarX = true;
212             return actTURN_L;
213         }
214         if (sensores.terreno[3] == 'X' || sensores.terreno[8] == 'X')
215         {
216             ultimaAccion = actTURN_R;
217             encontrarX = true;
218             return actTURN_R;
219         }
220         if (sensores.terreno[6] == 'X' || sensores.terreno[12] == 'X')
221         {
222             ultimaAccion = actFORWARD;
223             return actFORWARD;
224         }
225     }
```


Buscar plan y ir hasta allí.

```
172         if (sensores.terreno[0] == 'X')
173         {
174             if (sensores.bateria < 3000)
175             {
176                 sensores.bateria = sensores.bateria + 10;
177                 ultimaAccion = actIDLE;
178                 return actIDLE;
179             }
180             else
181             {
182                 hayPlan = false;
183             }
184         }
```

recargar la bateria y una vez terminado generando nuevo plan.

```
246         // colocar la casilla precipicio
247         if ((objetivo_n3.fila > 3 || objetivo_n3.columna > 3 || plan1 == 2) and !cont)
248         {
249             for (int c = 0; c < mapaResultado.size(); c++)
250             {
251                 mapaResultado[0][c] = 'P';
252                 mapaResultado[1][c] = 'P';
253                 mapaResultado[2][c] = 'P';
254                 mapaResultado[c][0] = 'P';
255                 mapaResultado[c][1] = 'P';
256                 mapaResultado[c][2] = 'P';
257             }
258 
259             for (int c = 0; c < mapaResultado.size(); c++)
260             {
261                 mapaResultado[mapaResultado.size() - 3][c] = 'P';
262                 mapaResultado[mapaResultado.size() - 2][c] = 'P';
263                 mapaResultado[mapaResultado.size() - 1][c] = 'P';
264                 mapaResultado[c][mapaResultado.size() - 3] = 'P';
265                 mapaResultado[c][mapaResultado.size() - 2] = 'P';
266                 mapaResultado[c][mapaResultado.size() - 1] = 'P';
267             }
268             cont = true;
269         }
```

Pintar casilla precipicio.