ネットワークプログラミング レポート

Ec5 40 番 若月 耕紀

1 はじめに

本レポートでは，ネットワークプログラミングのロボットプログラムについての動作を記す．

2 プログラムの動作

作成したロボットプログラムの動作について以下に示す．

2.1 自身やエネルギータンクの情報取得

目標とするエネルギータンクを決めるために，自身の位置情報と，フィールドのエネルギータンクの情報を取得する．取得する情報は，自身の座標，各エネルギータンクの座標と得点とする．自身の情報を取得するコードとエネルギータンクの情報を取得するコードをリスト 1，2 に示す．

リスト 1　自身の情報を取得するコード (一部抜粋)

```
String[] str_ship = line.split("␣");
if(str_ship[0].equals(name)){
        ship_x = Integer.parseInt(str_ship[1]);
        ship_y = Integer.parseInt(str_ship[2]);
}
```

リスト 2　エネルギータンクの情報を取得するコード (一部抜粋)

```
String[] str_energy = line.split("␣");
energyX[energyVal] = Integer.parseInt(str_energy[0]);
energyY[energyVal] = Integer.parseInt(str_energy[1]);
energyP[energyVal] = Integer.parseInt(str_energy[2]);
```

2.2 目標の決定

今回作成したプログラムでは，基本的には自身に最も近いエネルギータンクを目標とする．ただし，2 点間の距離を算出する際に，エネルギータンクのポイントに応じて距離を短くするように減算を行う．また，境界を越えると反対側に出ることを考慮して計算する．

目標とするエネルギータンクの座標を取得するコードをリスト 3 に示す．

リスト 3　目標とするエネルギータンクの座標を取得するコード (一部抜粋)

```
if(energyVal > 1){
        x = 128 - Math.abs(128 - Math.abs(ship_x - energyX[0]));
        y = 128 - Math.abs(128 - Math.abs(ship_y - energyY[0]));
        r1 = (x * x) + (y * y) - (energyP[0] * energyP[0] * weight);
        for(i = 1, j = 0; i < energyVal; i++){
```

```
6                   x = 128 - Math.abs(128 - Math.abs(ship_x - energyX[i]));
7                   y = 128 - Math.abs(128 - Math.abs(ship_y - energyY[i]));
8                   r2 = (x * x) + (y * y) - (energyP[i] * energyP[i] * weight);
9                   if(r1 > r2){
10                          r1 = r2;
11                          j = i;
12                  }
13          }
14  }else{
15          j = 0;
16  }
17  TargetEnergyX = energyX[j];
18  TargetEnergyY = energyY[j];
```

2.3 移動について

自身の座標とエネルギータンクの座標を長方形 (正方形) の対角の頂点として考える．この時，長方形に近い形であれば，上下または左右に大きく移動させ，正方形に近い形であれば，斜めに移動させる．

移動する方向を決める際には，目標を決定する際と同様に，境界を越えたら反対側に出ることを考慮する．移動方向を決める条件分岐のコードをリスト 4 に示す．

リスト 4　移動方向を決める条件分岐のコード (一部抜粋)

```
1  x = ship_x - TargetEnergyX;
2  y = ship_y - TargetEnergyY;
3  if(Math.abs(Math.abs(x) - Math.abs(y)) >= shapeError){/長方形に近い場合
4          if((Math.abs(x) - Math.abs(y)) >= 0){/横に長い場合
5                   if(Math.abs(x) <= 128){/順方向
6                           if((x) <= 0){/右側
7                   }
8          }else{/縦に長い場合
9                   if(Math.abs(y) <= 128){/順方向
10                          if((y) <= 0){/上側
11         }
12 }else{/正方形に近い場合
13         /左右判定
14         if(Math.abs(x) <= 128){/順方向
15                  if((x) <= 0){/右側
16         /上下判定
17         if(Math.abs(y) <= 128){/順方向
18                  if((y) <= 0){/上側
19 }
```