KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:
IFF-0/3 gr. studentė
Simona Ragauskaitė
2022 m. kovo 14 d.
Priėmė:
Lekt. Evaldas Guogis
Doc. Svajūnas Sajavičius

KAUNAS 2022

TURINYS

1. Scala (LD2)

1.1. Darbo užduotis

Naudojant programavimo įrankį / žaidimo kūrimo imitatorių „Scalatron“ sukurti programą / bot‘ą. Šiam kuriamam bot‘ui reikia:

- Panaudoti bent kels mater boto išleidžiamus botų padėjėjų tipus (pvz.: minos, raketos į priešus, „kamikadzės“, rinkikai, masalas ir pan.)
- Panaudoti bent kurį vieną iš kelio radimo algoritmų (DFS, BFS, A*, Greedy, Dijkstra)

1.2. Programos tekstas

Visas programos kodas yra paremtas Scalatron „Reference“ botu.
Pakeistas vietas kodo, kurias parašiau pateikiu žemiau.

FindPath objektas, laiko statines kelio radimo funkcijas. Yra panaudotas BFS (paieškos į plotį) algoritmas:

```
object FindPath {
    // BFS

    def apply(bot: Bot, view: View) : XY = {
        var visited = Set[XY]()          // visited places
        val queue = new Queue[XY]()      // places waiting to be visited
        var map = Map[XY,XY]()           // the path taken
        val viewRange = view.size / 2            // check if neighbour within view

        queue.enqueue(XY(0,0))           // queue the starting point

        while(!queue.isEmpty){
            val currentXY = queue.dequeue()              // grab the first element
in queue

            for ( newXY <- getNeighbours(currentXY, viewRange))
            {
                val cell = view(newXY) // get the cell value

                if (cell != 'W' && cell != 'b' && cell != 'p' && cell != 's' &&
cell != 'm'){
                    if ( !visited.contains(newXY)){
                        queue.enqueue(newXY)
                        visited += newXY
                        map += (newXY -> currentXY)

                        // logic for stopping and reconstructing the path
                        if(cell == 'P' || cell == 'B'){
                            var result = newXY
                            var logger = result + ""

                            while (  map(result) != XY(0,0)){
                                result = map(result)
                                logger += " " + result
                            }

                            bot.log(logger.dropRight(1))
                            return result
```

```
                        }
                    }
                }
            }
        }
        return XY(0,0)
    }

    def getNeighbours(start: XY, range: Int) : Set[XY] =
    {
        var neighbours = Set[XY]()

        for (x <- -1 to 1; y <- -1 to 1){           // everyithing in 3x4
            if (!(x == 0 && y == 0)){               // excluding middle
                val newXY = start + XY(x,y)
                // excluding out of range
                if (newXY.x > -range && newXY.x < range && newXY.y > -range &&
newXY.y < range){
                    neighbours += newXY
                }
            }
        }

        neighbours
    }
}
```

Taip pat sukūriau naują boto tipą, kuris vadinasi LandMine. Jis padedamas prie boto ir sprogsta. Jeigu šalia yra priešų ir nėra mano boto:

```
def reactAsLandmine(bot: MiniBot) {
        var masterClose = false
        var explode = false

        for (newXY <- FindPath.getNeighbours(XY(0,0), 5); newXY2 <-
FindPath.getNeighbours(newXY, 5))
        {
            bot.log(bot.view(newXY2)+"")
            bot.view(newXY2) match {
                case 'M' => masterClose = true
                case 'm' => explode = true
                case 'b' => explode = true
                case 's' => explode = true
                case  _ =>
            }
        }

        if (explode && !masterClose) bot.explode(3)
    }
```

Šio boto paleidimas bei pagrindinio boto valdymas (... - išskirtas kodas, kuris buvo identiškas Reference):

```
    def forMaster(bot: Bot) {
```

```
        …

        val bfs = FindPath(bot, bot.view)
        if(XY(0,0) != bfs){
            bot.move(bfs)
        }
        else
        {
            // determine movement direction
            directionValue(lastDirection) += 10 // try to break ties by favoring
the last direction
            val bestDirection45 = directionValue.zipWithIndex.maxBy(_._1)._2
            val direction = XY.fromDirection45(bestDirection45)

            bot.move(direction)
            bot.set("lastDirection" -> bestDirection45)
        }


        if(bot.energy > 1000 && nextToMine < bot.time){//spawn mines
            bot.spawn(XY.Down, "mood"->"LandMine")
            bot.set("nextToMine" -> (bot.time + 20))
        }
      …

    def forSlave(bot: MiniBot) {
        bot.inputOrElse("mood", "Lurking") match {
            case "Aggressive" => reactAsAggressiveMissile(bot)
            case "Defensive" => reactAsDefensiveMissile(bot)
            case "LandMine" => reactAsLandmine(bot)
            case s: String => bot.log("unknown mood: " + s)
        }
    }
```

1.3. Programos veikimo ekrano kopija