

INF1771 – 2016.1

INTELIGÊNCIA ARTIFICIAL

TRABALHO 2

MARCELO PAULON, RENAN DA FONTE, RODRIGO SILVA, GABRIEL MEDEIROS

Introdução

- ▶ Pikachu está a procura de Pokébolas escondidos em masmorras
- ▶ As masmorras podem ter buracos, inimigos, e armadilhas de teletransporte
- ▶ Pikachu deve ter bastante cuidado ao explorá-las
- ▶ Pikachu deve sair da masmorra pelo mesmo acesso que entrou

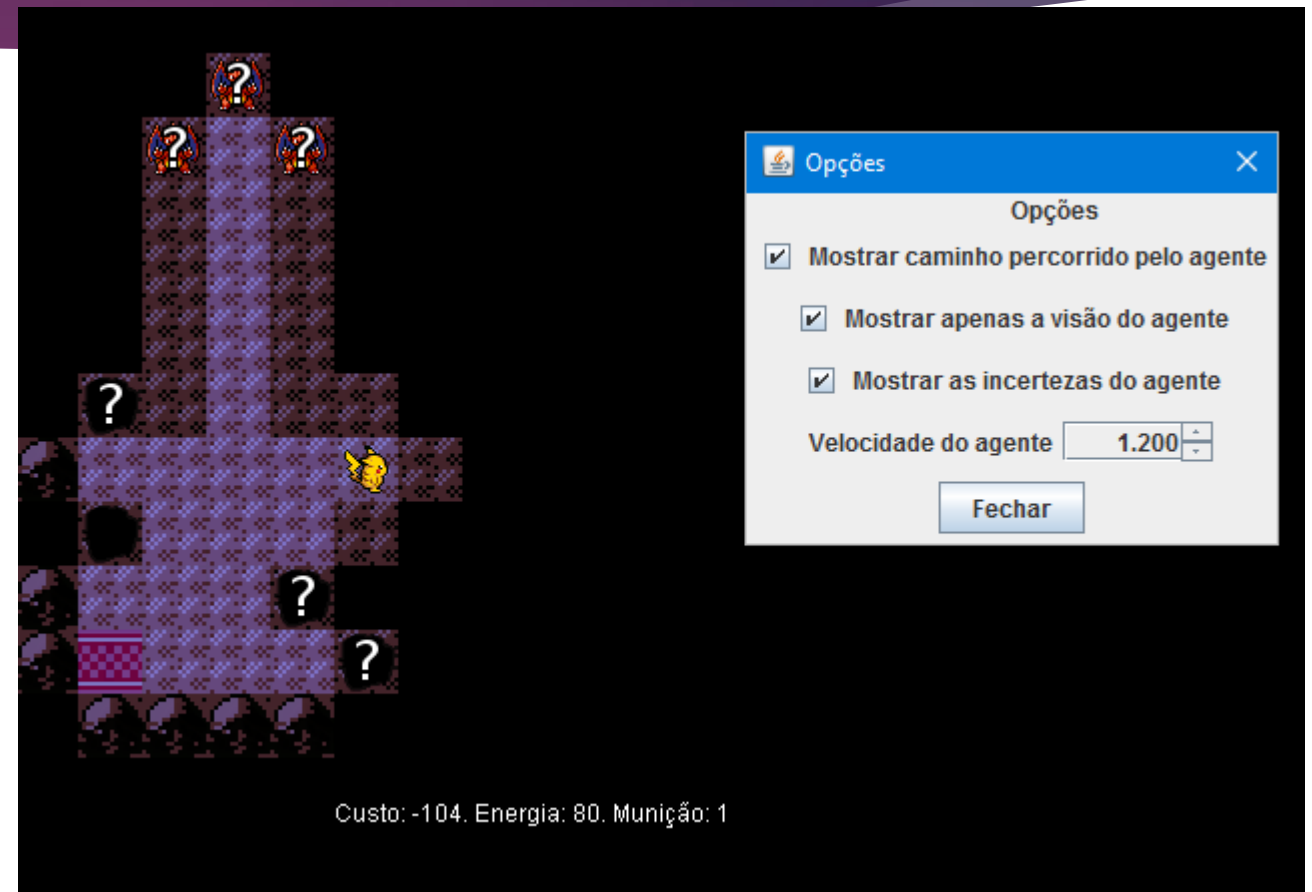


Interface e funcionalidades

- ▶ Linguagem: Java (Swing)
- ▶ Exibição do custo, energia e munição
- ▶ Geração de mapa
- ▶ Carregar/Salvar mapa
- ▶ Resetar agente
- ▶ Exibição do caminho percorrido
- ▶ Exibição da visão do agente
- ▶ Exibição das incertezas do agente
- ▶ Velocidade do agente



Interface e funcionalidades



Lógica / decisões

- ▶ Linguagem: Prolog (SWI-Prolog)
- ▶ Controle total do jogo através do Prolog
- ▶ Algoritmo de Dijkstra para encontrar powerups, caminhos abertos e a saída
- ▶ Marcação de pontos de incerteza, atualizados a cada vez que o agente se desloca no mapa
- ▶ Arriscar ser atacado quando Energia > 50 (pior dano que os inimigos podem causar)
- ▶ Sair do jogo quando não há mais caminhos seguros abertos e o custo é maior que 0
- ▶ Arriscar cair em buracos quando não há mais caminhos seguros abertos e o custo é menor que 1
- ▶ Custo final no mapa do enunciado: 581 (30 de energia, 0 de munição)




```
/* If current cell has gold, pick it up */
getNeqMove(pickGold) curPosition(X, Y) goldCell(X, Y) restrict(goldCell(X, Y), assert(floorCell(X, Y)),
    incrementGold(100) // Amount of gold picked up when entering cell and pick it up
```

```
getNextMove(pickPowerup)    curPosition(X, Y), powerupCell(X, Y), retract(powerupCell(X, Y)), assert(floorCell(X, Y),
decrementCost(1000), incrementEnergy(1) // get the primitive powerup, check if it's available and pick
```

```
getNextMove (rotate) willWalkTo(X, Y), not(isWalkable(X, Y)), rotateAgent()
```

```
getNextNode() {
    if (this.next === null) {
        return null;
    }
    this.visited = true;
    return this.next;
}
```

```
getNewPage (route) = case (condition == "all") of
    "all" -> let visited = new Set() in
        let allPages = [1..10] in
            let filteredPages = allPages \> visited in
                let newPage = filteredPages !! (length filteredPages - 1) in
                    newPage
```

```

getNetMove(ShootingActionResult)
    if (getNetMove(E) == 0)
        shoot(5)
    else
        shoot(10)
    return 0
end

```

```
curEnergy(E), E > 50, rotateAgent
```

```
getNextMove(walk)    curPosition(x, y), adjacent(x, y)
```

```
/* Ask walking to enemy/teletransport */
```

```
getTeleType(w) = perceptTeletransport(); percept
not(hasEnemy(X,Y)), nor(hasTele
not(visited(X,Y)), not(path ha
```

```
not(hasEnemy(?,?)), not(hasTeletransport(?)),  
not(path hasOpen()), timesTurned(<=0), etcAgent(),
```

```
get [nextMove, walk] = perceptHole(), willWalkTo(X, Y), costCost(C, C < 1,
not(visited(X, Y)), not(path hasOpen()), not(path hasEnemyOrTeletransport()), walkTo Basic(X, Y), !
```

... ..