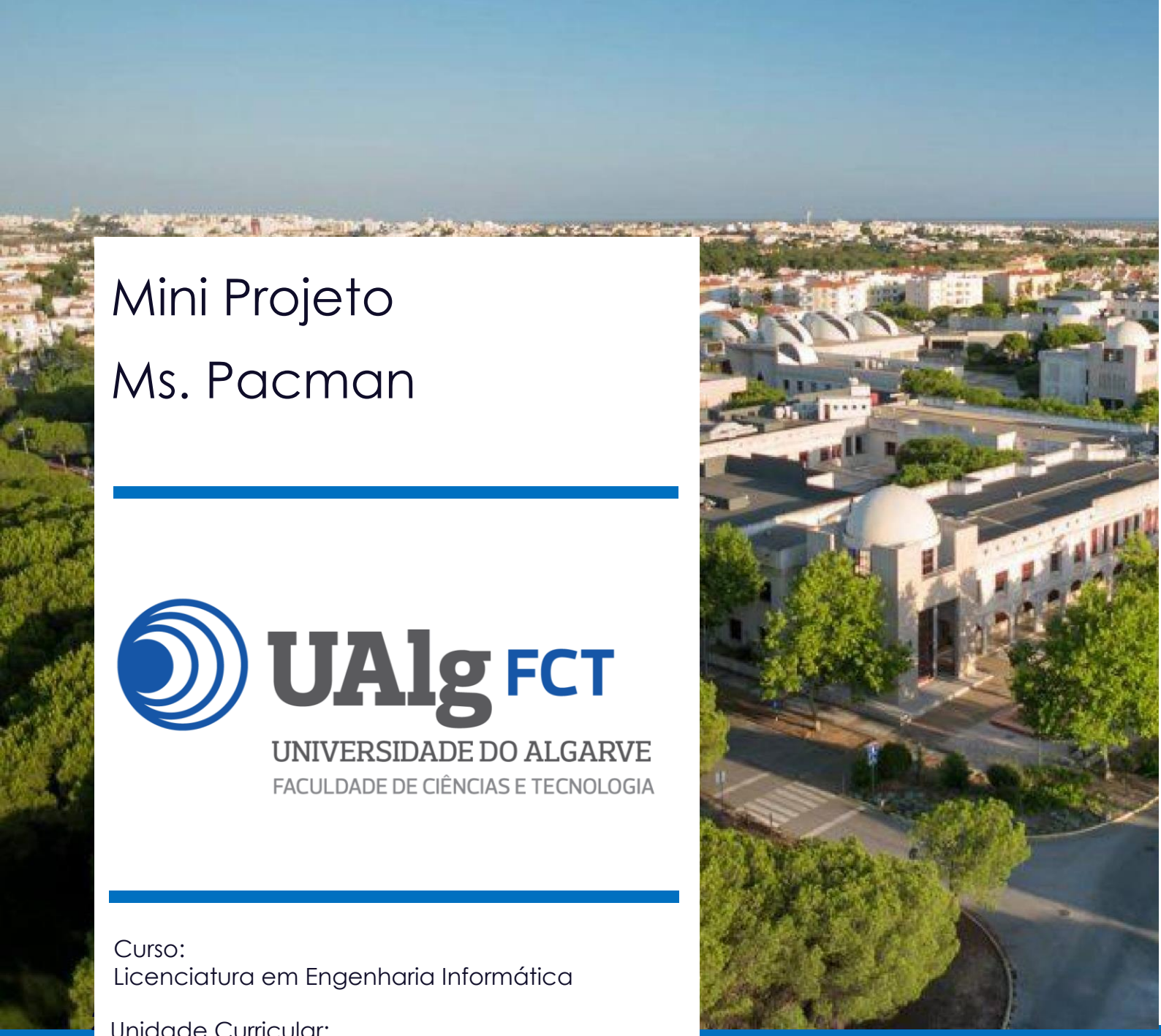

Mini Projeto

Ms. Pacman

UAlg FCT

UNIVERSIDADE DO ALGARVE

FACULDADE DE CIÊNCIAS E TECNOLOGIA

Curso:
Licenciatura em Engenharia Informática

Unidade Curricular:
Inteligência Artificial

Docente:
José Valente de Oliveira

Realizado por:
Nuno Silva a64506
Hugo Paixão a64514
Afonso Floro a64567

10/01/2022

Índice

Introdução

Neste trabalho, foi nos apresentado o desafio de implementar um agente para controlar o clássico jogo Ms. Pacman, jogo multiagente e não determinístico.

Com a realização deste relatório, pretendemos apresentar o controlador desenhado para o "Ms. Pacman", um jogo clássico, multiagente e não determinístico, considerado difícil para a maioria dos humanos. A mecânica do jogo é conceptualmente simples: o jogador controla o pacman que, posicionando-se num labirinto repleto de pastilhas, foge de 4 fantasmas que o perseguem. O objetivo do jogo passa, então, por comer todas as pastilhas sem ser alcançado pelos fantasmas, com aumento de dificuldade à medida que há progressão nos níveis. Foi-nos fornecida uma framework, a qual deveríamos estudar com vista a implementar um controlador para o comportamento do pacman, para que o próprio se comporte autonomamente, de acordo com as linhas por nós estabelecidas.

Antes de se proceder a qualquer tipo de implementação, foi feito um estudo do comportamento dos agentes do jogo, bem como do mapa e das relações que entre si se estabeleciam. À medida que se ia desenvolvendo a implementação do controlador do pacman, o seu comportamento era observado e

estudado, para ser posteriormente documentado e analisado. Foi seguida uma abordagem reativa, onde o comportamento do pacman dependia dos eventos que ocorriam no jogo, resultado do comportamento dos outros agentes (ghosts).

Após todas as alterações a que o controlador foi sujeito, obtivemos o controlador que adiante será apresentado mais aprofundadamente. O controlador irá então a competição contra outros controladores com o mesmo objetivo.

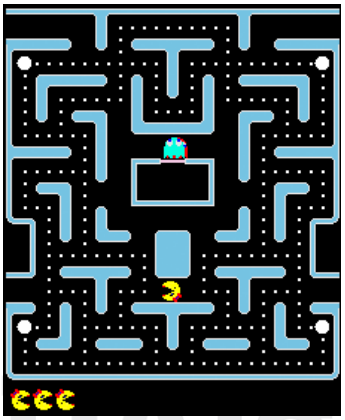

Figura 1 - Mapa do 1°nível do jogo.

Abordagem

Observou-se o comportamento do pacman e foram delineados alguns comportamentos que, de alguma forma, prolongariam o tempo de vida do pacman, aumentando assim a probabilidade de alcançar uma maior pontuação. Estabeleceram-se regras hierárquicas como:

1. Se há pílulas de poder, se o pacman está a menos de 15 nós da pílula e se o pacman chega mais cedo à pílula do que o fantasma, então vai em direção à pílula – na possibilidade do pacman ficar cercado, a única forma de escapar é comer os fantasmas, o que só é possível sob efeito da pílula de poder.
2. Se há um fantasma a menos de 25 nós, o pacman foge do fantasma – o caminho a tomar depende da proximidade do pacman a outros fantasmas ou pílulas.
3. Se o pacman acabou de comer uma pílula de poder e se há pílulas nas imediações, o pacman come essas pílulas – como o nível é ganho somente após todas as pílulas serem comidas, a inofensividade dos fantasmas não pode ser só utilizada para os comer, mas sim para comer o maior número de pílulas possível durante esse período.

4. Se o pacman acabou de comer uma pílula de poder e se há fantasmas comestíveis a menos de 20 nós, o pacman persegue o mais próximo – responsável por comer fantasmas quando tal necessidade se verificar, o que garante mais pontos do que se o pacman comesse apenas pílulas ou pílulas de poder.
5. Enquanto houver pílulas, o pacman tenta comer a pílula mais próxima – o que garante que o pacman coma todas as pílulas, o que aconteceria num plano sem fantasmas.

Após isso, estabeleceram-se outros métodos de comparação e desempate, que ajudaram a escolher, por exemplo, o melhor caminho de fuga aquando de uma perseguição defensiva, ou a melhor interseção ou cruzamento a seguir e o caminho mais proveitoso a tomar, entre outros.

Apesar da efetividade desta abordagem, é sabido que um agente preditivo poderia evitar com maior efetividade as emboscadas causadas pelos fantasmas, o que prolongaria o tempo de vida do pacman, aumentando assim a probabilidade de alcançar uma maior pontuação. Tentou utilizar-se métodos como o algoritmo de Monte Carlo para a procura da melhor jogada possível, todavia, tal não foi implementado com sucesso. Assim, o comportamento do pacman, segue uma abordagem totalmente reativa, onde vê o seu comportamento condicionado pelas ações dos restantes agentes do jogo e pelas circunstâncias prevalecentes após cada movimento.

UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Unit Tests

Não se utilizaram testes unitários durante o desenvolvimento do código.

Design

1. Alterámos a classe **Game** onde estão os métodos que permitem a execução do jogo:

1.1. Selecionámos a equipa de fantasmas, entre *Rand*, *Pincer* ou *Legacy*, (l.33-35) consoante a necessidade no imediato.

1.2. Alterámos a quantidade de "vidas" do pacman para 4 (l.66).

2. Criámos a classe **IA2122P2G07** para a implementação do controlador:

2.1. closestPill: retorna a pílula mais próxima de um determinado nó. Se não existir, retorna null;

2.2. closestPower: retorna a pílula do poder, ainda não comida, mais próxima do pacman. Se não existir retorna null;

2.3. closestEdibleGhost: retorna o fantasma comestível mais próximo do pacman. Se não existir retorna null;

2.4. closestKillerGhost: retorna o fantasma não-comestível mais próximo do pacman. Se não existir retorna null;

2.5. pillsNearPower: retorna a pílula em melhor posição, relativamente à pílula de poder mais próxima e ao pacman. Se não existir retorna null;

2.6. bestAdj: retorna o nó adjacente mais próximo ao nó objetivo;

2.7. fantasmaInTheWay: verifica se existe algum fantasma no caminho (entre o pacman e o nó objetivo);

2.8. seeAround: retorna o nó que, não tendo um fantasma pelo caminho, está a pelo menos 20 nós do pacman e está mais perto do pacman do que de todos os fantasmas. Não pode ser nem o nó atual do pacman, nem o nó atual dos fantasmas;

2.9. seeAroundInter: retorna a interseção (nó com 3 ou 4 nós adjacentes) que, não tendo um fantasma pelo caminho, está a pelo menos 20 nós do pacman e está mais perto do pacman do que de todos os fantasmas. Não pode ser nem o nó atual do pacman, nem o nó atual dos fantasmas;

2.10. cornered: retorna a quantidade de fantasmas a menos de 20 nós do pacman;

2.11. atePower: retorna a pílula do poder, já comida, mais próxima do pacman. Utilizado para limpar as pílulas mais próximas às zonas onde previamente existiam pílulas do poder. Se não existir retorna null;

2.12. edibleGhosts: retorna true se existe pelo menos um fantasma comestível, false se não;

2.13. lastSecond: impede que o pacman seja comido por algum fantasma, no momento exato da transição de estado de comestível para não-comestível;

2.14. ghostInterClose: retorna true se existe pelo menos um fantasma a menos de 20 nós do pacman, false se não;

2.15. survive: analisa o mapa e retorna o melhor nó a seguir;

2.16. action: método principal, contém as chamadas aos restantes métodos.

A classe **IA2122P2G07** teve como base a classe **SimplePillEater**, por ser o controlador, por defeito, do pacman. O primeiro método desenvolvido foi closestPower e muitos dos outros métodos implementados tiveram-no como base, dadas as semelhanças de sintaxe. Certas funções como closestEdibleGhost e closestKillerGhost resultaram do desdobramento de uma função geral, closestGhost, e permaneceram intactas desde uma fase inicial da implementação, por se referirem a regras elementares para a obtenção do comportamento desejado, por parte do pacman. Os últimos métodos desenvolvidos foram seeAround e seeAroundInter por terem um grau de complexidade superior à maioria dos restantes e por neles estarem presentes várias chamadas a outros métodos.

A metodologia geral para o desenvolvimento dos métodos foi "tentativa-erro", visto que era o comportamento do pacman, bem como a sua relação com os outros agentes e o próprio mapa, que definiam sempre o nosso próximo passo. Assim, à medida que cada método era inserido, o jogo era executado com vista a testar a validade e efetividade do método. A apresentação dos métodos é clara e limpa, o que confere maior simplicidade e mais fácil compreensão ao código.

[6]

Java Code

O código desenvolvido (com javadocs) está em anexo na pasta "Código".

Javadoc

Package screenpac.controllers

Class IA2122P2G07

java.lang.Object
    screenpac.controllers.IA2122P2G07

All Implemented Interfaces:

`screenpac.controllers.AgentInterface, screenpac.extract.Constants`

```
public class IA2122P2G07
extends Object
implements screenpac.controllers.AgentInterface, screenpac.extract.Constants
```

Field Summary

Fields

Modifier and Type	Field	Description
`Stack<screenpac.model.Node>`	`positions`	

Fields inherited from interface screenpac.extract.Constants

`agentOverlapDistance, BG, blinky, CENTRE, DIV, DOWN, dx, dy, edible, edibleColor, edibleGhostStartingScore, ghostColors, initGhostDelay, inky, LEFT, MAG, NEUTRAL, nGhosts, nLives, pacMan, pill, pillOverlapDistance, pillScore, pinky, powerScore, rand, RIGHT, sue, UP`

Constructor Summary

Constructors

Constructor	Description
`IA2122P2G07()`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`int`	`action (screenpac.model.GameStateInterface gs)`	Method analysis the map state and returns the best direction to ms.pacman go to
`screenpac.model.Node`	`atePower (screenpac.model.GameStateInterface gs)`	Method returns the closest Power to Ms.pacman that was already ate if closest is null returns null
`screenpac.model.Node`	`bestAdj (screenpac.model.GameStateInterface gs, ArrayList<screenpac.model.Node> nodes, screenpac.model.Node node)`	Method returns the closest adjacent node to the target

screenpac.model.Node	closestEdibleGhost (screenpac.model.GameStateInterface gs)	Method returns the closest ghost to Ms.pacman in an edible state if closest is null returns null
screenpac.model.Node	closestKillerGhost (screenpac.model.GameStateInterface gs)	Method returns the closest ghost to Ms.pacman that's not in an edible or returning state
screenpac.model.Node	closestPill (screenpac.model.GameStateInterface gs, screenpac.model.Node node)	Method returns the closest Pill to a given Node if closest is null return null
screenpac.model.Node	closestPower (screenpac.model.GameStateInterface gs)	Method returns the closest Power to Ms.pacman position if closest is null return null
int	cornered (screenpac.model.GameStateInterface gs, screenpac.model.Node node)	Method returns how many ghosts are less than 30 nodes of distance
boolean	edibleGhosts (screenpac.model.GameStateInterface gs)	Method returns true if at least 1 ghost is in a edible state else returns false
boolean	ghostInterClose (screenpac.model.GameStateInterface gs, screenpac.model.Node var)	Method returns if a ghost is a 20 nodes close to Ms.pacman else return false
boolean	ghostInTheWay (screenpac.model.GameStateInterface gs, ArrayList<screenpac.model.Node> nodes, screenpac.model.Node target)	Method returns true if at least 1 ghost is in the way of Ms.pacman to reach the target else returns false
boolean	lastSecond (screenpac.model.GameStateInterface gs)	Method returns false if at least 1 ghost as an edible time less that 8 else returns true
screenpac.model.Node	pillsNearPower (screenpac.model.GameStateInterface gs)	Method returns the Pill that is in better position in relation to the closest ate Power and to Ms.pacman if closest is null returns null
screenpac.model.Node	seeAround (screenpac.model.GameStateInterface gs)	Method returns the closest node, that doesn't have a ghost in the way, that is at least 20 nodes far way from Ms.pacman and is closer to Ms.pacman than to a ghost that node can't be the same that Ms.pacman or ghosts are at that moment else returns null
screenpac.model.Node	seeAroundInter (screenpac.model.GameStateInterface gs)	Method returns the closest node, that doesn't have a ghost in the way, that is at least 20 nodes intersection (with 3 or 4 adjacent nodes) far way from Ms.pacman and is closer to Ms.pacman than to a ghost that node can't be the same that Ms.pacman or ghosts are at that moment else returns seeAround
screenpac.model.Node	survive (screenpac.model.GameStateInterface gs)	Method analysis the the map and returns the best Node where ms.pacman showed run to

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Details

positions

```
public Stack<screenpac.model.Node> positions
```

Constructor Details

IA2122P2G07

```
public IA2122P2G07()
```

Method Details

closestPill

```
public screenpac.model.Node closestPill(screenpac.model.GameStateInterface gs,
                                        screenpac.model.Node node)
```

Method returns the closest Pill to a given Node if closest is null return null

Parameters:

gs - - GameStateInterface

node - - Node

Returns:

closest - Node

closestPower

```
public screenpac.model.Node closestPower(screenpac.model.GameStateInterface gs)
```

Method returns the closest Power to Ms.pacman position if closest is null return null

Parameters:

gs - - GameStateInterface

Returns:

closest - Node

closestEdibleGhost

```
public screenpac.model.Node closestEdibleGhost(screenpac.model.GameStateInterface gs)
```

Method returns the closest ghost to Ms.pacman in an edible state if closest is null returns null

Parameters:

gs - - GameStateInterface

Returns:

closest - Node

closestKillerGhost

```
public screenpac.model.Node closestKillerGhost(screenpac.model.GameStateInterface gs)
```

Method returns the closest ghost to Ms.pacman that's not in an edible or returning state

Parameters:

gs - - GameStateInterface

Returns:

closest - Node

edibleGhosts

```
public boolean edibleGhosts(screenpac.model.GameStateInterface gs)
```

Method returns true if at least 1 ghost is in a edible state else returns false

Parameters:

gs - - GameStateInterface

Returns:

true/false - boolean

atePower

```
public screenpac.model.Node atePower(screenpac.model.GameStateInterface gs)
```

Method returns the closest Power to Ms.pacman that was already ate if closest is null returns null

Parameters:

gs - - GameStateInterface

Returns:

closest - Node

pillsNearPower

```
public screenpac.model.Node pillsNearPower(screenpac.model.GameStateInterface gs)
```

Method returns the Pill that is in better position in relation to the closest ate Power and to Ms.pacman if closest is null returns null

Parameters:

gs - - GameStateInterface

Returns:

closest - Node

bestAdj

```
public screenpac.model.Node bestAdj(screenpac.model.GameStateInterface gs,
                                    ArrayList<screenpac.model.Node> nodes,
                                    screenpac.model.Node node)
```

Method returns the closest adjacent node to the target

Parameters:

gs - - GameStateInterface

nodes - - ArrayList

node - - Node

Returns:

closest - Node

ghostInTheWay

```
public boolean ghostInTheWay(screenpac.model.GameStateInterface gs,
                             ArrayList<screenpac.model.Node> nodes,
                             screenpac.model.Node target)
```

Method returns true if at least 1 ghost is in the way of Ms.pacman to reach the target else returns false

Parameters:

gs - - GameStateInterface

nodes - - ArrayList

target - - Node

Returns:

true/false - boolean

seeAround

```
public screenpac.model.Node seeAround(screenpac.model.GameStateInterface gs)
```

Method returns the closest node, that doesn't have a ghost in the way, that is at least 20 nodes far way from Ms.pacman and is closer to Ms.pacman than to a ghost that node can't be the same that Ms.pacman or ghosts are at that moment else returns null

Parameters:

gs - - GameStateInterface

Returns:

closest - Node

seeAroundInter

```
public screenpac.model.Node seeAroundInter(screenpac.model.GameStateInterface gs)
```

Method returns the closest node, that doesn't have a ghost in the way, that is at least 20 nodes intersection (with 3 or 4 adjacent nodes) far way from Ms.pacman and is closer to Ms.pacman than to a ghost that node can't be the same that Ms.pacman or ghosts are at that moment else returns seeAround

Parameters:

gs - - GameStateInterface

Returns:

closest - Node

cornered

```
public int cornered(screenpac.model.GameStateInterface gs,
                    screenpac.model.Node node)
```

Method returns how many ghosts are less than 30 nodes of distance

Parameters:

gs - - GameStateInterface

node - - Node

Returns:

count - int

survive

```
public screenpac.model.Node survive(screenpac.model.GameStateInterface gs)
```

Method analysis the the map and returns the best Node where ms.pacman showed run to

Parameters:

gs - - GameStateInterface

Returns:

target - Node

lastSecond

```
public boolean lastSecond(screenpac.model.GameStateInterface gs)
```

Method returns false if at least 1 ghost as an edible time less that 8 else returns true

Parameters:

gs - - GameStateInterface

Returns:

true/false - boolena

ghostInterClose

```
public boolean ghostInterClose(screenpac.model.GameStateInterface gs,
                               screenpac.model.Node var)
```

Method returns if a ghost is a 20 nodes close to Ms.pacman else return false

Parameters:

gs - - GameStateInterface

var - - Node

Returns:

true/false - boolean

action

```
public int action(screenpac.model.GameStateInterface gs)
```

Method analysis the map state and returns the best direction to ms.pacman go to

Specified by:

action in interface screenpac.controllers.AgentInterface

Parameters:

gs - - GameStateInterface

Returns:

dir - int

UML

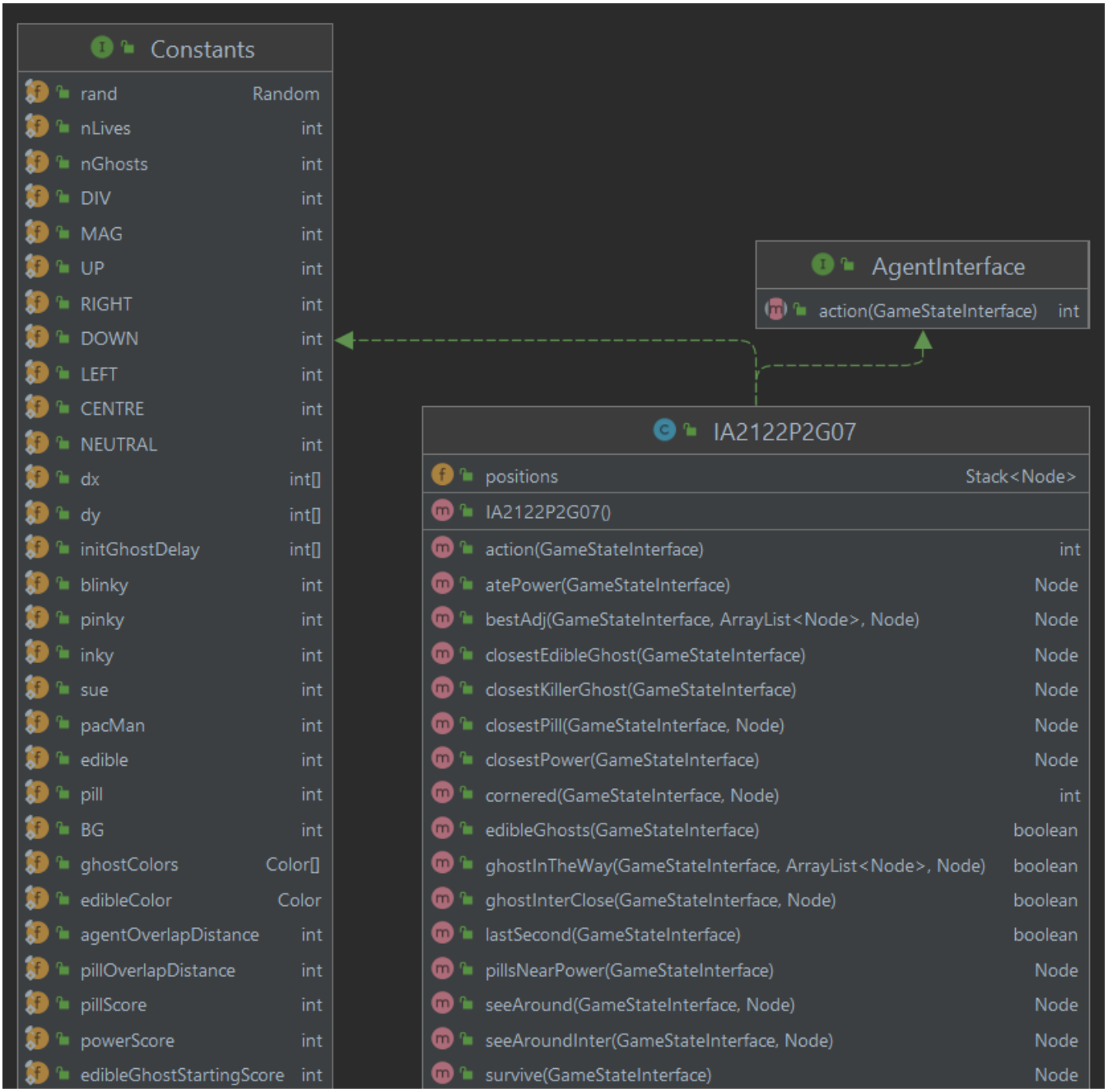

Figura 2 - Diagrama UML Class Implementada

Resultados

#	RandTeam		PincerTeam		LegacyTeam	
	Pontos	Nível	Pontos	Nível	Pontos	Nível
1	62040	23	24870	7	4390	1
2	28460	10	11890	3	2360	1
3	71860	28	12330	3	11750	2
4	46740	18	17710	6	3590	1
5	37960	13	13930	4	11680	2
6	29900	11	22800	7	5120	1
7	23490	9	7360	2	8670	2
8	28580	11	9670	3	8130	2
9	45940	18	24730	7	10590	2
10	78810	29	10140	3	6100	1
11	65320	24	12350	4	10930	2
12	48080	18	18900	5	11340	2
13	34610	13	10290	3	9610	2
14	31010	11	21600	6	9040	2
15	37890	13	16760	6	8720	2
16	67020	26	12180	4	10040	2
17	84600	33	10670	3	4720	1
18	36620	14	12140	4	10460	2
19	13830	5	18320	6	5230	2
20	36800	14	14780	5	3480	1
Média	45478	17,05	15171	4,55	7797	1,65

Como era expectável, quão mais difícil é a oposição (fantasmas), piores são os resultados obtidos, o que é corroborado pela descida dos valores médios de pontos obtidos e nível atingido, à medida que se altera a equipa de fantasmas.

Por não termos conseguido implementar um algoritmo de procura que transformasse a abordagem reativa numa abordagem preditiva, não temos como comparar os resultados, contudo acreditamos que ter-se-iam obtido melhores resultados globais recorrendo a uma abordagem preditiva, em detrimento de uma abordagem reativa. Ainda assim, à medida que se aumenta a dificuldade, os resultados serão sempre consecutivamente piores em termos absolutos. Assim, concluímos que quão mais complexo e refinado é o comportamento dos fantasmas, mais baixas são as pontuações obtidas, menos eficaz se torna a abordagem reativa e mais se justificava a implementação de uma abordagem preditiva.

Conclusão

Com a realização deste relatório, pretendíamos, como já havia sido previamente frisado, apresentar o controlador desenhado para o "Ms. Pacman". Ora, não só apresentámos o controlador, como providenciámos todo um conjunto de informações relevantes acerca do mesmo, tais como o raciocínio utilizado, o código propriamente dito, os javadocs. a implantação do diagrama em UML correspondente ao código desenvolvido e uma análise aos resultados obtidos.

Não obstante, a abordagem seguida não foi ao encontro do pretendido, tendo em conta que, em vez de uma abordagem preditiva, apenas conseguimos seguir uma abordagem reativa. Uma abordagem preditiva aumentaria, com certeza, a probabilidade de obter melhores resultados, o que, infelizmente, não conseguimos alcançar. Acreditamos que o sucesso de uma abordagem preditiva seria, efetivamente, o objetivo principal deste mini projeto e entristece-nos que não o tenhamos alcançado, independentemente do quão efetiva seja a implementação por nós obtida. Ainda assim, independentemente do sucesso – ou não – em adotar a abordagem pretendida, acreditamos que, no presente relatório, constam todos os dados e informações requeridos e necessários a uma compreensão geral, mais ou menos aprofundada, daquilo que foi o trabalho que desenvolvemos.

Assim, podemos tomar o objetivo deste mini projeto como como parcialmente bem-sucedido, por não termos conseguido implementar o controlador seguindo a abordagem mais proveitosa (preditiva).

Bibliografia/Webgrafia

Coleção de PDF's fornecida pelo professor José Valente de Oliveira (via Tutoria Eletrónica)

http://www.pacman-vs-ghosts.net/

https://en.wikipedia.org/wiki/Ms._Pac-Man

https://sapientia.ualg.pt/handle/10400.1/3245

UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA