CS7032: AI & Agents: Ms Pac-Man vs Ghost League AI Controller Project Report

Student Name: Akshay Kulkarni **Student Number**: 16308675

Student Name: Samir Kumar **Student Number**: 16341316

Student Name: Saurabh Pathak **Student Number**: 16336962

Abstract

The following report talks about different tactics that can be used for implementing an AI agent for the MS Pac-Man vs Ghosts League. We implement a purely reactive agent with combination of offensive and defensive measures to control Ms Pac-Man actions. This primarily consists of three modules: run, kill and eat arranged according to suitable priority. The kill distance and run distance are set before hand to tell the agent when to chase, evade or ignore the ghosts. These distances are found out from well performed testing. The behaviour of the agent depends on these set parameters and can be changed to make the agent aggressive or defensive.

1. Introduction

The history of AI is filled with fantasies, possibilities, demonstrations, and promise (Buchanan 2006). (Nilsson 2010) defines artificial intelligence as "activity devoted to making machines intelligent and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment". However, only in the last decade the AI community, have been able to integrate artificial intelligence in gaming world. One such example is The Ms Pac-Man vs. Ghosts League Competition which involves artificial intelligence powered agents competing with each other to obtain the highest score. Pac-Man originally is an arcade game created by game designer Toru Iwatani, of Namco Company, in the 1980s (Srisken 2002). In Pac-Man's original version the objectives were to guide the main character (Pac-Man) around the maze eating all the Pac-dots, or pills, power pills and fruits without being eaten by the ghosts, which was quite deterministic. The Ms. Pac-Man version of the game differs in one significant way, the ghost's behaviour is non-deterministic which makes the game much more difficult because each ghost behaves differently, making it harder to play against them.

2. Problem

Aim of every game is to complete the task and win the game. In order to win Ms Pacman, there are Olsen (2014) certain rules and conditions that agent needs to go through. Staring off with elements of the game:

- Ms Pacman – Main Agent
- Ghosts – Enemy Agent
- Pills – Food/Points
- Power Pills – Bonus/Power Points/Weapon

Some of the basic rules are:

- Ms Pacman should eat all the pills in order to finish the level while surviving the collision with ghosts.
- Consuming power pills allow the Ms Pacman to defend herself and kill ghosts. These power pills only last for some time. Ghost starts running away from her after she consumes power pill.
- Ms Pacman has initially three lives to finish one stage.
- Eating every generic pill gives her 10 points. Similarly, eating power pills gives her 50 points.
- If ghost is killed it initially gives 200 points, then points get doubled for each additional ghost killed.
- Additional life can be gained by earning 10,000 points.

Normally, this is done by the user who controls the agent with the keyboard. So in AI, agent needs to strategies her moves to fulfil aim of the game. As aforementioned, aim is eating all the pills while staying within the given rules.

The basic strategy is built upon the idea that Ms Pacman need to eat all the pills as quickly as possible while keeping in mind that she moves slow while she is eating the pills. Keep check of how far the ghost is and when to run away. She should keep eating the pills till the ghost come closer and then she should try to eat the power pill. If she doesn't consume power pill and the ghosts are too close she should run away.

Approach used in order to implement the strategy is mentioned in next section.

3. Approach

In order to find out as suitable approach for the agent programming, it is important to describe agent and environment properties. (Marinescu 2016) PEAS system is used to help determine the agent properties.

PEAS:

- Agent: *Ms Pacman*
- Performance Measures: *Survive, Eat Pills, Power Pills, Earn Points, Escape from Ghost, Kill the Ghost*
- Environment: *Maze*
- Actuators: *Distance, Direction*
- Sensors: *Left Pills, Ghosts, Walls, Power Pills, Killable time*

This helps in determining the agent is purely reactive agent. In order to determine the environment properties, (Marinescu 2016) method from lectures is used as follows

Environmental Properties:

- Environment: Maze
- Observable *(Fully vs Partially)*: *Partial* as the approach will only look at certain distance in order for an agent to react.
- Deterministic vs. stochastic: *Stochastic* as at certain times agents (ghost/Ms Pacman) will move randomly.
- Episodic vs. sequential: *Episodic* as it does not keep any data from past experience.
- Static vs. dynamic: *Dynamic* as the ghost actions are not dependent on Ms Pacman.
- Discrete vs. continuous: *Continuous* as it's unsure that how many actions there will be in order to finish the game.
- Agents *(Single agent vs. Multi-agent): Multiple Agent* as there are ghosts as well.

3.1 Defining the Architecture:

Using the above exploration of environmental and agent properties, abstract architecture can be defined. As approach uses (Marinescu 2016) *purely reactive agent*, it acts based purely on the current state of the environment. Its behaviour can be modelled as:

$$\text{action}: S* \rightarrow A$$

The architecture can be defined using 4 tuple values:

$$\text{Archs} =< S, A, \textit{action}, \textit{env} >$$

Here, S represents all the possible states of environment, A is all possible actions. These actions refer to agent behaviour and at last *env* represents behaviour of environment as explained in lectures (Marinescu 2016). Where,

S = {s1, s2, ..., sn} and A = {a1, a2, ..., an}

But knowing the agent and environmental properties, we have discrete possible values for them.

s1 No killable ghosts

s2 No attacking ghosts

s3 killable ghost and attacking ghost

s4 No pills remaining

Which states S = {s1, s2, s3, s4}. The possible actions are:

a1 Go towards nearest pill.

a2 Go towards nearest killable ghost.

a3 Go away from nearest attacking ghost.

Making A = {a1, a2, a3}.

So using the above simplification and knowing the possible actions and states, formation of the rules is as follow:

- if ghost is far away and a2 if ghost is too close to the agent, otherwise a0: **—s0 → a0 ∨ a2: a0**
- if ghost is too far away and a1 if close enough, otherwise a0: **—s1 → a0 ∨ a1: a0**
- if attacking and eatable ghosts are too far away, a1 if edible ghost is close enough and hostile ghost is far enough away and a2 if the hostile ghost is too close, otherwise a0: **—s2 → a0 ∨ a1 ∨ a2: a0**
- as no pills marks the start of the next round, otherwise a0**—s3 → a0**

These are simple conditions when referred to the earlier mentioned states and actions.

3.2 Subsumption architecture

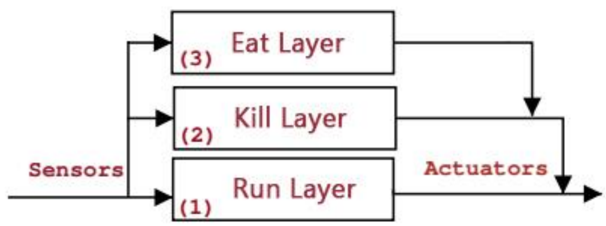
(Marinescu 2016) Moving from abstract architecture to concrete architecture is a step where we define a general architecture in to specific architecture by further specification of actions and deciding a form of representation. One way of specifying the action function is *Layered*. "*Layered* is a combination of logic based or BDI and reactive decision strategies."

Architecture supporting this specification is Subsuming architecture. Total ordering of possible behaviours is used to represent Brooks (1986) Subsumptional Behaviour. Such as

$$R \subseteq Beh : \prec \subseteq R \times R.$$

Possible actions of going towards the nearest pill and killable ghost or going away from attacking ghost, require the vital element which is distance. From the set of rules, it can be diverged in to mainly two categories of killing distance and running distance. Three possible layers that can be determined of each possible action and they are set in an order:

$$\mathbf{(1) \prec (2) \prec (3)}$$



(1) *Run layer*: It is responsible for agent to get away from the attacking ghost if distance between them if lesser than running distance. Otherwise agent keeps on consuming pills.

(2) *Kill Layer*: After consuming power pill, it provides with a move which lead to killing the killable ghost if distance between them is lesser than killing distance. Otherwise agent keeps consuming pills.

(3) *Eat Layer:* This layer moves the agent to nearest pill and keeps on doing this.

They are set in this order of priority suitable for winning and getting the highest score. Running from the ghost is vital as if one loses life there are no chances of achieving high score. Killing enemy ghost gives much higher points than pills as they keep on getting double with each killed ghost. So, killing is a better approach for higher scores than eating pills.

4. Evaluation

The following section talks about evaluation of different strategies and results.

4.1 Best Combination of distances.

From above implementation, it raises the issue of deciding what value of distances will be used for an efficient agent. As aforementioned, there are two distances kill distance and run distance. With the realisation that run distance should be fairly less because agent should not keep running to avoid the ghosts. It should eat pills and if the ghosts come closer then only it should run away. Whereas, if Ms Pacman has eaten the power pill and is able to kill ghosts then it should look for larger distance to cover because she does not want to miss the chance of increasing the score in high number by killing ghosts.

In order to find out best combination of both kill distance and run distance, test is performed by running the game with 160 combinations of both distances. Value for kill distance was ranging from 0 to 80 because very high values will cover the whole screen and till it reaches the far away ghost its killable time might finish resulting in loss. Similarly, range for run distance values is 0 to 45, as aforementioned it should be fairly low. Both values were given the increment of 5 in order to get better approximation of distances.

The values from the test were record in excel sheet as shown in Table no. 1. Highest score comes by having distances around 5 and 75 for run distance and kill distance respectively.

4.2 Comparison with other controller:

In order to find out that approach implemented in this report is better than other controllers, average and high score from the test performed on layered approach of this report were taken in record as shown in Table 1.

Another test was performed using nearest pill eater controller. Game again was tested 160 times against the same ghost controller. The high score and average score has been recorded from 160 test values as shown in Table 2.

From implemented approach high score and average score was 7190 and 2773. Whereas using nearest pill controller gave 4560 and 2321 as high and average score respectively. It is clear that implemented approach provides with better solution.

5. Contributions of each team member:

Mostly whole team has been involved throughout the process. Eventually, tasks were divided. Samir did the background research and designed the algorithms. He decided the architecture suitable for the AI controller for Ms Pacman. All the documentation and records were maintained by Samir. Then report writing was also his responsibility.

Implementing designed algorithms in an efficient way that it suits the architectural approach was responsibility of Saurabh. He made sure that code clean and written in academic format. Code was kept clean and understandable by him.

All the testing was performed by Akshay. Test code was also written by him. He made sure that all the steps were implemented properly. He was responsible for rest of the refactoring of the code. Essential formatting of report was also done by him.

These responsibilities were handed over by considering the skills and efficiency among the team members.

6. Conclusion

Building an intelligent and autonomous agent is quite a complex task. The main difficulties we encountered were, mainly setting up the initial platform for the Ms Pacman. Though abstract architecture provides required definition about the agent and environment behaviour, it takes some effort to make it work as expected in practice. After getting it running it was all about designing efficient algorithms and their implementation in coding. The report describes the construction of purely reactive agent based upon subsumption layer architecture and assessment of our strategy with respect to various parameter values. The assessment results have shown that with our strategy there is higher possibility of getting a high score than in other strategy. The present strategy can be improved by adding machine learning aspects to it so that MS. Pac-man can react more accurately in processed scenarios.

7. References

Andrei Marinescu. (2016) Agents: Definition and Formal Architectures. Trinity College Dublin. Retrieved from https://www.scss.tcd.ie/~amarines/teaching/cs7032/Lecture%202/lecture_2_full_abstractarch.pdf on 26 December 2016

Andrei Marinescu. (2016) Reactive agents & Simulation. Trinity College Dublin. Retrieved from https://www.scss.tcd.ie/~amarines/teaching/cs7032/Lecture%204/reactive.pdf on 26 December 2016

Brooks, R. (1986). A robust layered control system for a mobile robot. IEEE journal on robotics and automation, 2(1):14–23

Buchanan B.G (2006) A (Very) Brief History of Artificial Intelligence AI Magazine Volume 26 Number 4: 1

Kurt Olsen (2004) Lynx Tips & Tricks. Retrieved from http://amigan.1emu.net/kolsen/instructions/ms.pacman.html on 24 December 2016

Nilsson N.J (2010) The Quest for Artificial Intelligence: A History of Ideas and Achievements Cambridge University Press

Risken S. (2002) PacMania: How Pac-Man and Friends Became Pop Culture Icons Stanford University

8. Appendix

	Killing Distance															
		0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75
Run Distance	0	2020	2040	2040	2020	2070	2910	2120	2920	4360	1840	4600	3890	1810	4640	3300	3960
	5	3810	3580	2860	3720	5860	2470	2500	3260	6810	4660	6780	6270	4450	3400	**7190**	7130
	10	2850	2010	5120	3030	3030	2540	2620	2470	3280	3420	3100	3330	4050	5110	6000	3130
	15	2850	3450	1580	4190	2090	3640	2550	2540	3720	3370	1660	1550	5410	4520	4220	3280
	20	1510	1990	1400	3150	2630	2570	3440	3520	3600	3530	2670	3850	2990	3440	2240	2360
	25	1410	1340	2810	1320	1670	1310	1310	2700	1300	1340	3620	1340	2350	2840	4020	5100
	30	1280	1270	2860	1300	2200	1330	1280	2990	2720	2260	2660	2650	1630	2690	2290	6520
	35	1540	1530	1390	1540	580	1400	2350	1290	1310	1270	1320	2320	600	980	2720	4050
	40	1530	1300	1300	1280	1350	1280	1410	1780	1260	1350	1560	3780	2310	5450	2270	2910
	45	2720	560	3130	2770	3070	2690	2760	2730	2720	580	2710	580	5310	2580	2600	2370

Highest Score :	7190
Average Score :	2773.688

Table 1: The above table shows the results for highest score obtained for our strategy

Tested By Running Game 160 Times															
2300	2170	2140	1990	1350	2040	3670	2020	2000	2010	2020	2860	2120	2280	2020	1970
3190	2120	2020	3280	2240	2010	2220	2010	3120	2950	3260	2020	2020	1420	2020	2660
2050	2180	2020	2030	2090	2180	2040	2920	1970	2020	2010	2120	2010	2260	2070	2000
2860	2200	2860	2200	2010	2000	1990	2660	1370	2860	3190	3060	2010	2660	2860	2860
2010	2920	3290	2000	2020	3260	2020	2900	2020	2020	1990	2030	2020	2010	2240	2020
3390	1950	2860	2040	2860	2840	2000	2000	2170	2860	**4560**	2660	2020	2860	3520	2020
3100	2020	3060	2040	2860	2040	1350	3090	2060	2050	2660	2040	3490	2020	1980	2170
2020	2060	2920	2860	2170	2140	2000	2390	2880	2070	2020	2920	2950	2210	2040	2040
2020	2060	2530	2020	2150	2050	2050	2020	2010	2010	2120	1490	2860	2010	2020	2860
2070	2020	2010	2660	2240	2090	2040	2020	2000	2020	2020	2120	2100	2010	3520	2040

Highest Score:	**4560**
Average Score:	**2321.875**

Table 2: The above table shows the results for highest score obtained by nearest pill strategy