Matematika z računalnikom: final report

Matic Pokorn

May 2024

1 Introduction

In this report I present the work for my project, which is part of the Master's course "Matematika z računalnikom". The goal of my project was to create a video game with the option for 1 or 2 players. In the case of 1 player the player plays against the computer and in the case of 2 players the players play against each other using the same keyboard. The menu and title screen can be viewed in Fig. 1.

The game is about 2 characters being able to jump around a 2D environment and aim fireballs at each other. They both have a certain amount of health and their objective is to reduce their opponent's health to zero while avoiding their fireballs. When one of the characters' health drops to zero, the game is over.

In section 2 I will cover implementation details of the game. I will present `arcade`, the framework in which I worked, the computer agent design, where I will also discuss the abandonment of reinforcement learning as the underlining approach for agent implementation. I will also discuss graphics. In section 3 I will explain how the game is played; I will describe game controls as well as different sections of the game. In section 4 I will present current issues and possible improvements and lastly, in section 5 I will give final thoughts before concluding the report.

2 Implementation

2.1 Framework

The language of implementation is Python and the game framework is `arcade` [1]. Despite `pygame` being the most popular library for game development in Python, I chose `arcade` for its relative simplicity and focus on 2D games. It is built on top of the `pyglet` multimedia library. The advantages of `arcade` over `pygame` are:

- modern OpenGL graphics
- support for Python 3 type hinting

Figure 1: Game title and menu screen. Both player can choose their preferred game options.

- better support for animated sprites
- better separation of game logic from display code
- more documentation, including complete Python game examples
- a built-in physics engine for platform games

As a beginner in Python game development I found the online tutorials exceptionally useful and I was able to quickly learn the basic principles of the framework itself. Additionally, the framework is intuitive while at the same time allowing the user to implement increasingly complex video games.

2.2 Computer agent

One of the main tasks of this project was to create a computer agent for the 1-player mode. The initial idea was to train a reinforcement learning agent, with inspiration taken from [2], where they introduced a then groundbreaking and now still one of the flagship deep reinforcement learning algorithms, Deep Q-Network. They demonstrated its performance on a number of classic Atari video games such as Space Invaders.

2.2.1 Basics of reinforcement learning

Reinforcement learning is a sub-field of machine learning designed to tackle decision-making problems. Reinforcement learning is based on the assumption

that the underlying environment can be represented as a Markov decision process, where by making decisions (but also being subject to randomness), an agent is able to travel from state to state, at each step receiving a reward (or penalty). The task of the agent is to receive the maximum cumulative reward during the environment traversal.

Reinforcement learning is tasked with finding a good policy, where agent decisions will result in a good cumulative reward. This is usually done by training the agent by letting it roam the environment and learning the optimal policy. DQN, the algorithm from [2], uses neural networks for efficient state representation; upon visiting a state, it uses past experience to find the optimal action by comparing the current state to similar states it has visited before.

2.2.2 Abandonment of reinforcement learning

As stated, I took the idea to use reinforcement learning from [2]. However, the authors there used the entire game screen for state representation and had both far superior hardware for agent training and better expertise in the field. I, however, used a sliding window of limited information about the game to represent a state, such as data about character coordinates, which way they were facing, fireball firing times and health bars.

After a number of different trials and experimenting with different reward functions and parameter changes, I failed to successfully train the agent and I decided to abandon the idea altogether. The reason why I was unable to make the algorithm learn a meaningful policy (it learned a policy, but it was by no means optimal), is unknown to me; it is possible that either the state representation was not good enough, my reward function was wrong, my neural network was too small or I had the wrong parameters.

I then decided to opt for a rule-based agent as well as focus more on game visuals.

2.2.3 Rule-based agent

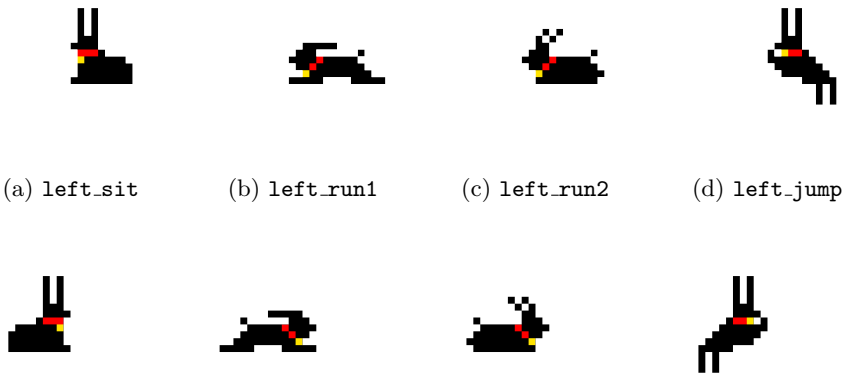
The rule based agent takes actions, which depend on positions of both characters as well as fireball positions. The rules are a bit more complex, but the general behaviour is described as follows:

- If you are on the left side of the screen, face right and shoot fireballs in equally spaced intervals. Same goes for the right side.
- If there is a fireball in proximity, jump in the air to avoid it.
- If your opponent crosses to your half of the screen, move to the other half until you are 10% of the screen width away from the respective edge.

2.3 Graphics

There are a number of custom graphics elements within the game. I decided to choose a pixel art theme and designed all elements using an online pixel sprite

design tool `www.piskelapp.com/`. The characters (bunnies) are animated to sit, run and jump and the fireballs also have a small animation. I also designed the grass blocks. In Fig. 2 all bunny frame representations can be seen and in Fig. 3 all fireball frame representations are available for viewing.



(a) `left_sit` (b) `left_run1` (c) `left_run2` (d) `left_jump`

(e) `right_sit` (f) `right_run1` (g) `right_run2` (h) `right_jump`

Figure 2: Character animation frames for the red bunny. There are four frames for movements to the left and four frames for movements to the right. The second player plays the same character, but with a blue collar.

(a) `left_fireball1` (b) `left_fireball2` (c) `left_fireball1` (d) `left_fireball2`

Figure 3: Fireball animation frames.

3 Gameplay

In this section I will describe game controls and the gameplay itself.

3.1 Running the game

The game is available on github: `https://github.com/maticpokorn/mat_rac`. After cloning the repository, it can simply be run by entering the following command in the command line: `python game.py`.

3.2 Controls

There are 8 keyboard keys used in the game:

- **Player 1:** `A` and `D` for moving left and right, `W` for jumping and `Q` for attack.
- **Player 2:** `LEFT` and `RIGHT` for moving left and right, `UP` for jumping and `M` for attack.

3.3 Game menu

The game menu screen is the first thing the players see after entering the game. Both players can select their desired mode ("player" or "computer"). They can toggle between the two options by pressing `W` or `S` for player 1 and `UP` or `DOWN` for player 2. When they decide the mode, they select it with `Q` and `M` for player 1 and player 2, respectively. After they have both selected their desired mode, the game begins.

3.4 Playing the game

At the start of the game, each of the characters is positioned at their half of the screen with each character having a health bar of `100`. They can shoot fireballs at each other and if hit, the health of the character drops by `10`. A snapshot of the gameplay can be seen in Fig. 4.

3.5 Game over

The game ends when the health bar of one of the players reaches zero. When this happens, the game returns to the main screen and the players can again begin the game.

4 Issues and future work

There is one particular issue with the game. The characters sometimes get stuck on the ground and cannot move unless they jump in the air. This issue is probably connected to the placement of the grass blocks and needs fixing.

Figure 4: Gameplay snapshot.

Going on, the game is currently still a bit boring. I plan to change that by adding a power up, which replenishes a little bit of your health, and giving you a boost, where your fireballs fly faster and deal more damage. I addition to that, I plan to add a few more platforms to make the game more interesting.

5 Conclusion

In this report I presented the work I did for my project as part of the Master's course "Matematika z računalnikom".

References

[1] Paul Vincent Craven. Easy to use python library for creating 2d arcade games. `https://github.com/pythonarcade/arcade`.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.