Flappy Bird : Learning to Fly

Kunal Vyas
University of Massachusetts, Lowell
kunal_vyas@student.uml.edu

Vinayak Panchal
University of Massachusetts, Lowell
vinayak_panchal@student.uml.edu

ABSTRACT
In this paper we have attempted to teach Flappy Bird to fly. This game is a good application of Artificial Intelligence, where we can make it learn how to play the game by itself. For this problem, we have employed Q-Learning, which doesn't need a knowledge of the model and learns and gets better by experience as it tries and fails.

Author Keywords
qlearning
flappy bird
reinforcement learning

INTRODUCTION
Flappy Bird is one the most popular mobile games recently. Also, it is extremely frustrating because it is tough to score double digits unless the person has spent some hours on it. We aim to make the agent learn by making mistakes and score higher by using Q-Learning to learn the best action to be performed at the right position.

PROJECT DESCRIPTION
We have used *Markov Decision Processes* to solve the problem. To solve this MDP, we have used *Q-Learning*.

The state space is discretised over the horizontal and vertical distance from the next lower pipe. The Action is to either jump or not. There is a reward for staying alive (1) and a high negative reward (-1000) for dying. The learning rate, alpha is 0.7. The discount factor is 1, because we don't need to penalize anything here. We used the game code from **here**[1]. This game has been written using Python programming language and uses the pygame library.

Also, the code for the Q-learning algorithm which can be found in the file `learning.py` has been used from **here**. Firstly, to find the distance from nearest pipe, Kunal wrote a function, which finds the distance of the next pipe ahead of the bird. At any given time, the program can have upto three pipes. So, we are discarding the distance difference if the x-coordinate difference between the bird and the pipe is negative, i.e when the bird has crossed the pipe. And then taking the element with the lowest value of x-coordinate difference.

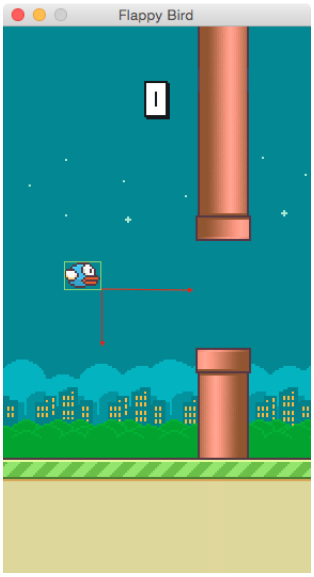

Figure 1. Screenshot of the game. The bird has to pass through the pipes to score higher

$$\underbrace{Q_t(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t,a_t)}_{\text{learning rate}} \times \Big[\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t,a_t)}_{\text{old value}} \Big]$$

Figure 2. The Q-Learning Algorithm

After calculating the distance, we have our states defined, and the action is implemented via a static boolean variable, which defines the actions performed by the bird.

As the bird learns by performing actions randomly, it learns the q-values for the states it visits. After sufficient learning, based on the exploration rate, it decides what action is to be performed at a particular state.

In this project, Kunal did the AI part, and Vinayak worked on the GUI.

ANALYSIS OF RESULTS
If the agent(bird) learns to play without dying, we will know that the learning has been successful. We have attempted to produce a desktop app using python which displays the learnt agent successfully scoring points and learns from mis-

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s is terminal

Figure 3. Algorithm

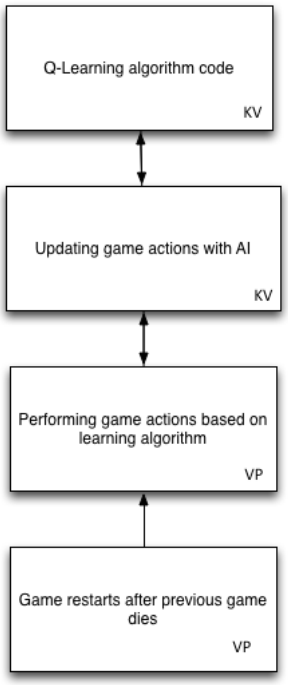

Figure 4. Block Diagram

takes. We are able to get the bird's position and the algorithm is learning the q-values.

DISCUSSION

Initially selecting a topic was very difficult for us as there are many possibilities and many areas where artificial intelligence could be used. Since this is our first project in the field of artificial intelligence, we thought of starting it with something basic which would help us understand the courses foundation concepts. So, we chose to work on Flappy Bird and thought of developing an AI agent which would encourage flappy bird to learn and take the best decisions based on the data acquired during learning episodes. Over the years, we have realized that the biggest challenge in the field of software engineering is the knowledge of which algorithm to apply to which problem. For our problem, A* search algorithm would have given similar results as the q- learning algorithm would give. A well written heuristic would have given good results but it would lead the game to be static. Means, any changes in the game environment would force us to change the heuristic as well. But the real beauty of AI is revealed when the agent knows nothing about the environment. After episodes of learning, the agent formulates the best set of state-action pairs which may even underrate the heuristic written for A* search. The hardest time in our development process was to figure out how to save the state action pair as the game is very dynamic. In the game, which is designed to be governed by the laws of physics and gravity, it was very difficult to capture previous actions and reward them appropriately. An example would be, suppose the current state of the bird is (40, 50) and it performs a jump and hits the pipe at state (20, 30). So by the game dynamics, the previous state was (22,32) and reward would be set for that state. A solution for this problem was to decrease the jump velocity so that there is not much difference between the state where the action was performed and the recorded state. In this way, we found the game went on smoothly and we obtained precise state-action pairs. At last, we learnt that in dynamic environments its difficult to obtain accurate results but we can still rely on the precise results and observe the agent working absolutely fine.

CONCLUSION

Overall, the project was a success and the bird could learn and fly by its own. It was a decent implementation of the concepts learnt during the course. But, as an independent work, we would like to extend the game so that there will be some in-game rewards like bonus points. Moreover, it would be interesting to see if we put a random agent which will try to thwart flappy birds plans. Developing an AI agent for such complex environments would uncover more about the field and help us learn more.

ACKNOWLEDGMENTS

The work described in this paper was conducted as part of a Fall 2014 Artificial Intelligence course, taught in the Computer Science department of the University of Massachusetts Lowell by Prof. Fred Martin.
We would like to thank Sourabh Verma for making the Flappy Bird game[1] and keeping it open source for people to use.

REFERENCES

1. Flappy Bird Clone.
   https://github.com/sourabhv/FlappyBirdClone
2. Q-Learning on Wikipedia.
   https://en.wikipedia.org/wiki/Q-learning