AUTOMATIC FLYING WAFTY MAN

Fall 2014 CS 221 Poster Presentation

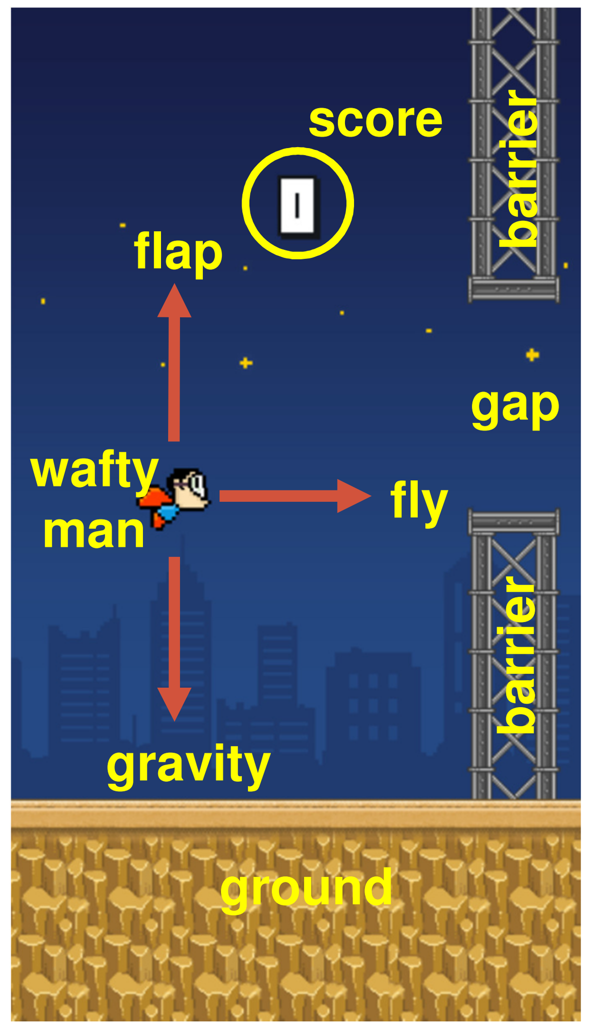
Joe Fan (joefan) and Vinod Kumar (vinodkum)

Hint: read from left-to-right, top-to-bottom

Note: Our team consists entirely of SCPD students and cannot attend the poster session in person, so per our TA's request, we created a video: https://www.youtube.com/watch?v=21SOi0oW_EI

Game Description

- Wafty Man (also known as Flappy Bird) is a popular side-scrolling game in which "wafty man" ***flies*** horizontally at a constant velocity through an endless sequence of vertical barriers with gaps.
- ***Gravity*** continually pulls wafty man down toward the ground, so the player must occasionally instruct wafty man to ***flap*** (fly momentarily upward).
- One point is awarded for each barrier that wafty man passes. The game ends when wafty man crashes into the barrier or the ground.
- To achieve high scores, the player must instruct wafty man to flap during the proper time (e.g. space, click, touch).

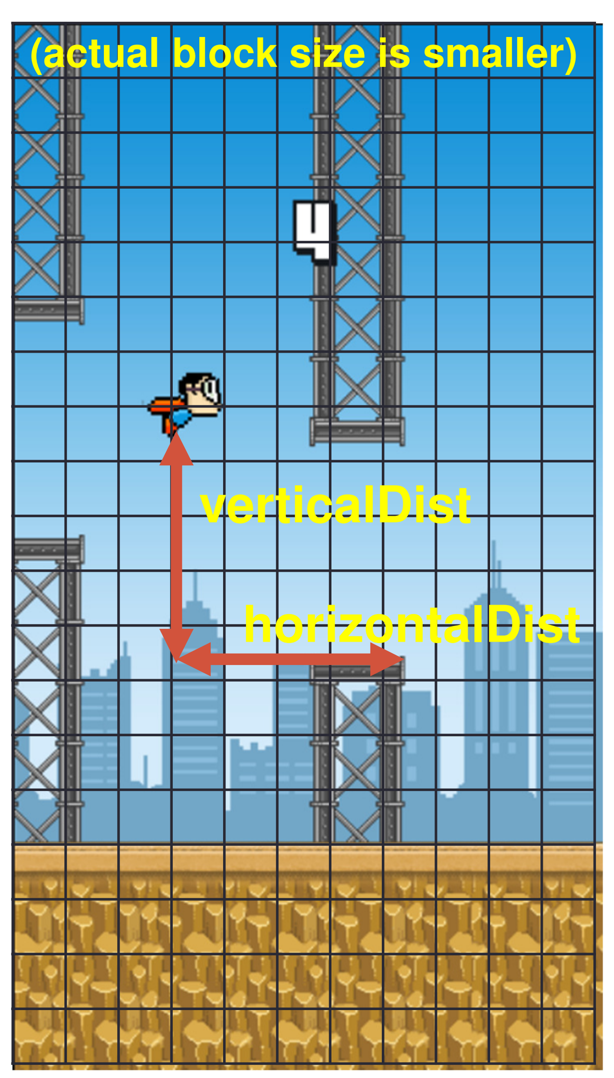


Game demo and project objective

- [demo1: Video of original game. Human player.]
- Project objective:
  - Develop an AI that learns how to play Wafty Man by reinforcement learning (Q learning)

Markov Decision Process

- Infrastructure
  - Discretize screen into blocks
  - Game loops in each "heartbeat"
  - Increase game speed by increasing "heartbeat" rate (e.g. 10x, 1000x)
  - AI implementation adds Q-Learning in game loop, which runs during each "heartbeat". No human input.
- MDP definition
  - State s = (verticalDist, horizonalDist)
  - Actions(s) = (flap, none)
  - Reward = $\begin{cases} +1 & \text{Running} \\ -1000 & \text{Dying} \end{cases}$



Q-Learning

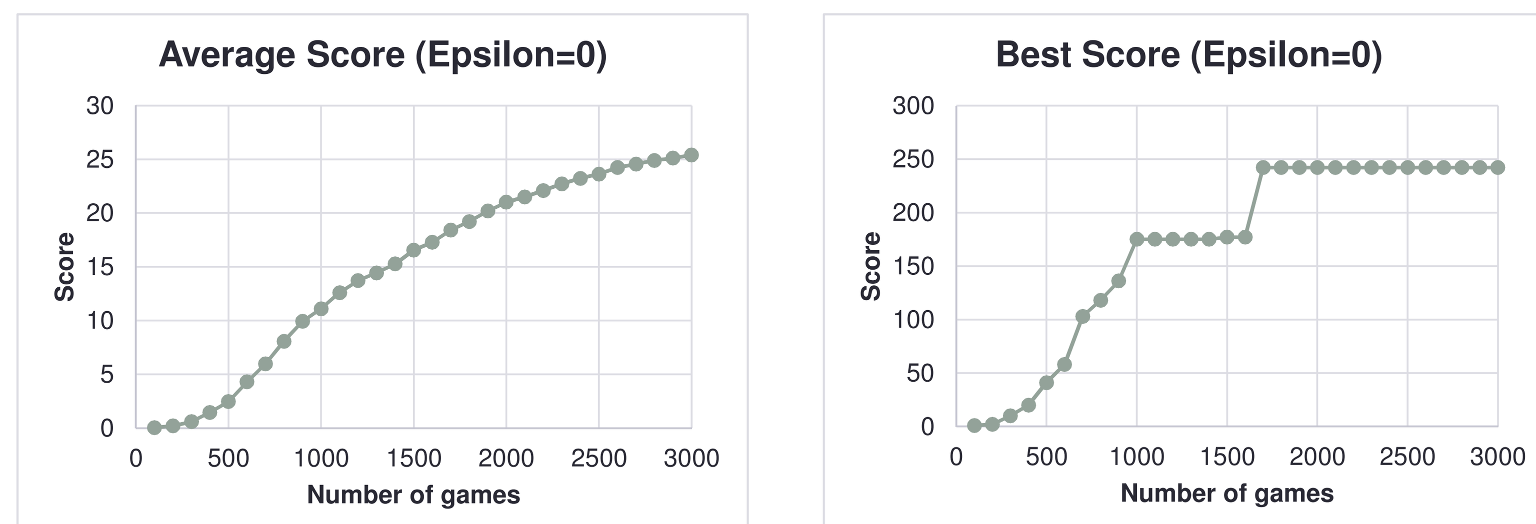
- Q-learning will select an action according to an ϵ-greedy policy
  - $\pi_{act}(s) = \begin{cases} \arg\max_{a \in Actions(s)} Q(s,a) & \text{probability } 1-\epsilon \\ \text{random from Actions}(s) & \text{probability } \epsilon \end{cases}$
- After action selection, run Q-learning to improve its estimate of Q on each (s,a,r,s'):

$$Q(s,a) = Q(s,a) - \eta\left[Q(s,a) - \left(Reward(s,a,s') + \gamma \max_{a' \in Actions(s')} Q(s',a')\right)\right]$$

- [demo2: Video of AI training. 1x speed.]
- [demo3: Video of AI training. 100x speed.]

Experiment 1

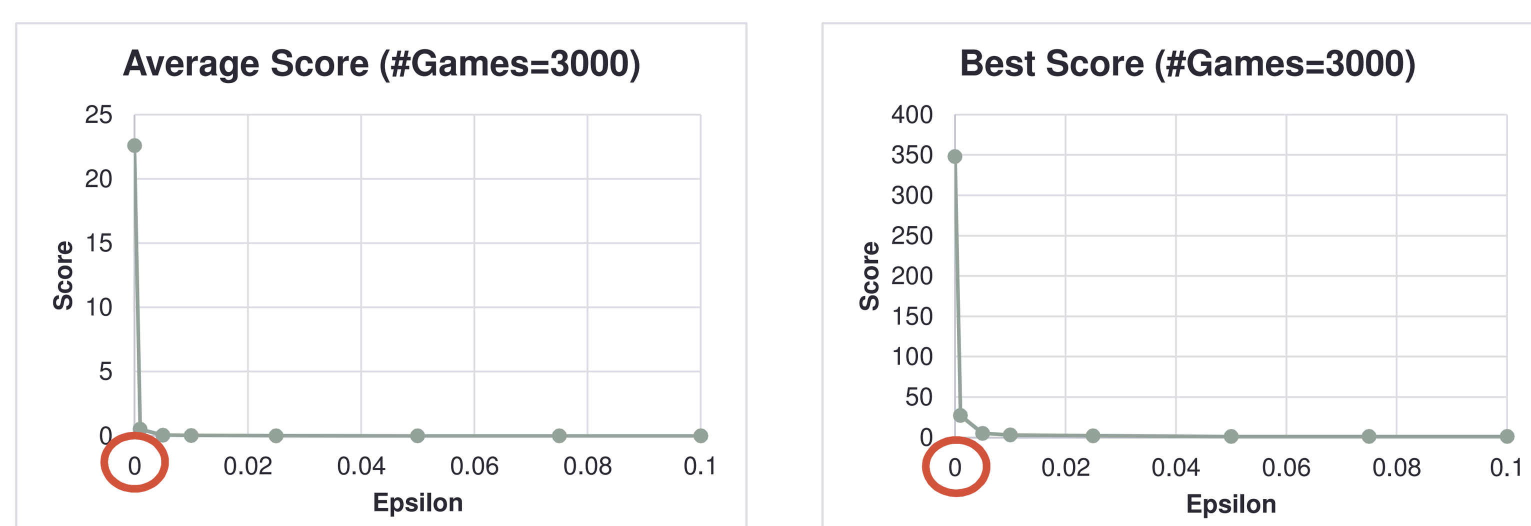
- Suppose we use the "best" Q-Learning hyper-parameters. How does the score change as AI Wafty Man plays more games?



- Analysis: Increasingly higher average and best score indicates that AI Wafty Man is actually learning

Experiment 2a

- How does the score differ for various Q Learning ϵ?



- Analysis: AI Wafty Man does not benefit from exploration. The game physics is exact, and the geometry is repetitive.

Experiment 2b

- How does the score differ for various Q Learning η?

- Analysis: Best step size is $\eta = 0.6$. Higher step sizes ($\eta \geq 0.8$) suffered from excessive $Q(s,a)$ volatility.

AI plays Wafty Man

- After training Q-Learning, the AI can play Wafty Man confidently
- [demo4: Video of AI playing Wafty Man.]
- How do humans perform?
  - No official scoreboard, but people online boast scores in the 100's range

Experiment 3

- Error analysis
  - Lack of generalization results in rote memorization and slow training time.
  - Challenging game scenario: The gap for the second lower barrier is lower, so if wafty man flaps too late for the first barrier, there is not enough time for gravity to pull wafty man down to pass the second gap, so wafty man smacks the second upper barrier.
  - Does the game instantiate barriers outside the screen?
    - If so, can AI Wafty Man use them in Q-learning?
- Proposed solution (planned for final report)
  - Enhance feature vector $\phi(s,a)$. Incorporate domain knowledge to improve generalization, and learn their weights.

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\left[Q(s,a) - \left(Reward(s,a,s') + \gamma \max_{a' \in Actions(s')} Q(s',a')\right)\right]\phi(s,a)$$