Pong Game

Libraries Used:

Pygame:

- For building the game, it provides various rich features to handle games.

Numpy:

- To handle matrices and vector operations.

Keras:

- For model building, training, saving and loading the weights.
- To Load the model from pickle file.

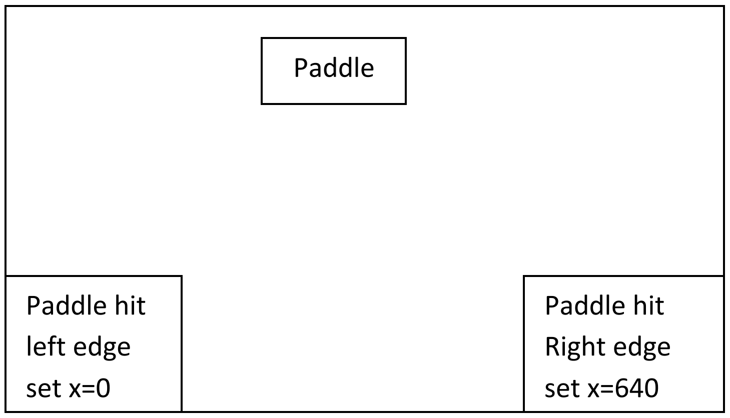
Pong Game Building (Environment):

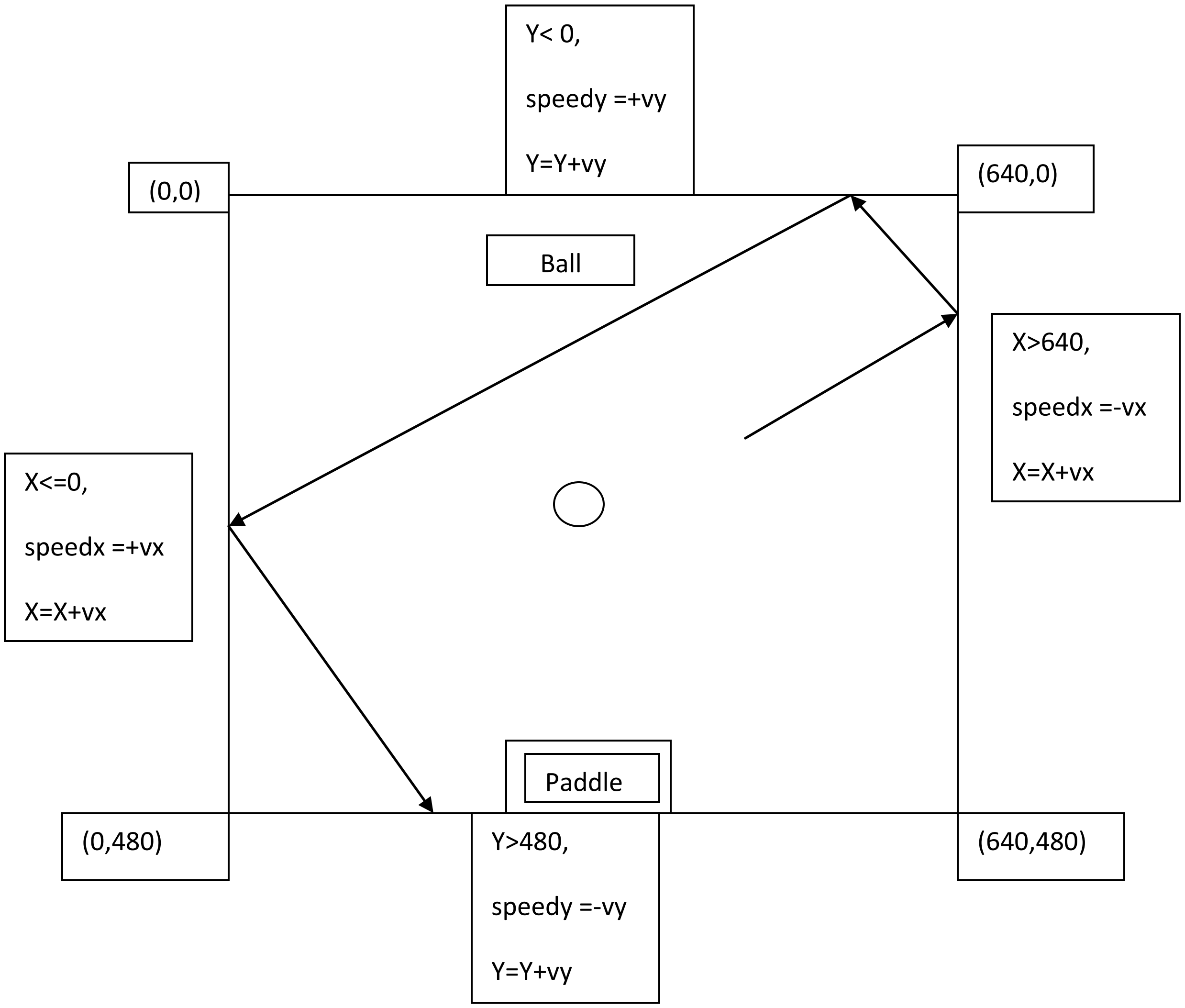
Pong Class:

- Showing the graphics of ball using pygame.
- Movement of ball in X and Y Co-ordinates i.e. speed in X and Y co-ordinate System.
- Updating the ball position using update method.
- Setting the direction of the pong when hit with walls.
- Collision Detection and scoring points.

Paddle Class:

- Showing the graphics of paddle using pygame.
- Movement of paddle in X- axis i.e. speed and direction in X-axis.
- Updating the paddle position depending upon the input i.e. direction
- Setting the position of the paddle when it hit the walls.





Agent Building (DQN):

- Initializing hyper parameters for model.
- Building model object using keras.
- Remembering the game states, action, next states and rewards.
- Act on the next step.
- Replay the game update rewards and fitting on the state.

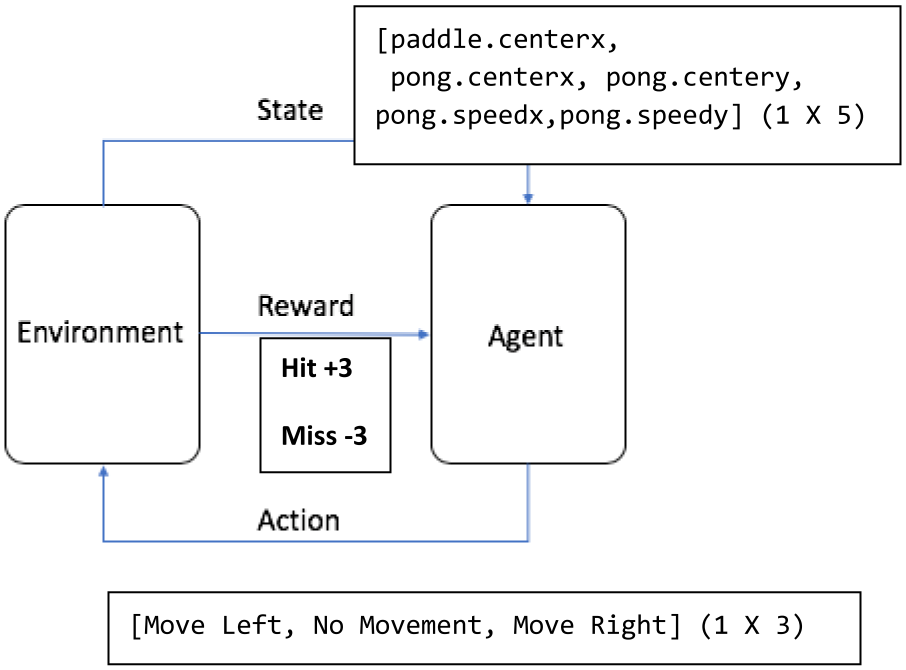
PongGame (Environment):

- Combines all pong and paddle functionalities.
- Initializing Scoring, pong, paddle and rewards.
- Reset the pong and paddle position after an episode is done.
- Act on the next step i.e. paddle movement upon the input from action.
- Updating the screen with paddle and pong positions and scores.

Training Process (Run):

- Interaction with Environment and Agent.
- Initializing DQN with state shape and PongGame.
- Input (5 units), Model is 2 hidden layer (64 units), output (3 units), Adam as optimizer, “relu” as activation for hidden layers and output activation is “linear”, MSE as loss.
- Running on number of episodes internally divided into iterations.
- Each iteration goes through action of agent depending upon the action we get a reward and next step.
- Agent remembers the states, action, and reward.
- Agent replays to fit in the model.
- Model weights are saved.

Model Architecture:


[paddle.centerx, pong.centerx, pong.centery, pong.speedx,pong.speedy] (1 X 5)
State
Environment
Reward
Agent
Hit +3
Miss -3
Action
[Move Left, No Movement, Move Right] (1 X 3)

Bonus Questions:

1. Discuss two machine learning approaches that you did not use in your pong implementation, but that would also be able to perform the task. Explain how these methods could be applied instead of your chosen method.

   Use CNN instead of neural network. Input should be difference in the image pixels at (n+1)-(n) instance be passed as state vector.

   Instead of neural network try RNN, with the state vector same as presented in the solution.

7. Make one or two suggestions for how you would implement dynamic difficulty adjustment during game play, based on a human player's performance. (This question relates to a bonus features in option B, but does not require that you chose that option, nor that you actually implemented the bonus feature).

As the Human player score increases the velocity of the ball by a multiplication factor.

Make the velocity of the ball unpredictable, just by increase or decrease velocity of ball by random function.