Pong Game

By Pranav A. Bhounsule, pranav@uic.edu

1 Goal

The overall goal is to create a single-player pong game. The project has two stages:

- **Stage 1:** You will complete the provided code get a basic version of the pong game to work. This involves programming the ball to bounce off the walls and the paddle. This is needed for the second stage.
- **Stage 2:** You will add more features into the game to make an advanced version of the pong game. This is mostly an open-ended exercise where you are required to add at least three innovations that make the game more interesting to play.

Here is some exceptional work by a student from Fall 2024. You dont have to do something as elaborate as this: https://youtu.be/GXdUYpLQw8U

2 Understanding the game provided to you

Please download the game *basic_pong.py* from Github using this link: https://github.com/pab47/robotics/blob/main/f24/basic_pong.py. To run the game, you will need to install the package *pygame*. Once you open Microsoft Visual Studio Code, in the terminal prompt found on the bottom, type *python3 -m pip install pygame* (press return) and wait for installation to complete. Close and reopen Microsoft Visual Studio Code and run *basic_pong.py*.

If pygame was successfully installated then it will produce an animation that looks like the one shown in Fig. 1. You will notice that the red ball penetrates the paddle and the game stops when the red ball reaches the bottom left corner of the screen. Now run the code again but use the left/right arrow keys to move the paddle. You will see that the ball is not detected by the paddle and the game ends.

3 Stage 1: Programming the basic pong game

Your task is to use the hints in the code to program the game.

To make it easy for you, I have marked the two places in the code. Program them sequentially as follows

1. Write code in the block that says "Basic pong edit #1". Write code here to enable the ball to detect the wall and/or paddle and to reverse its speed. After you program this part, you should check if the ball bounces of the paddle and the wall.
2. Write code in the block that says "Basic pong edit #2". Write code here to stop the paddle from leaving the screen on the left or right wall. After you program this block the paddle will not be prevented from moving too far on the left or too far on the right so as to disappear from the screen.

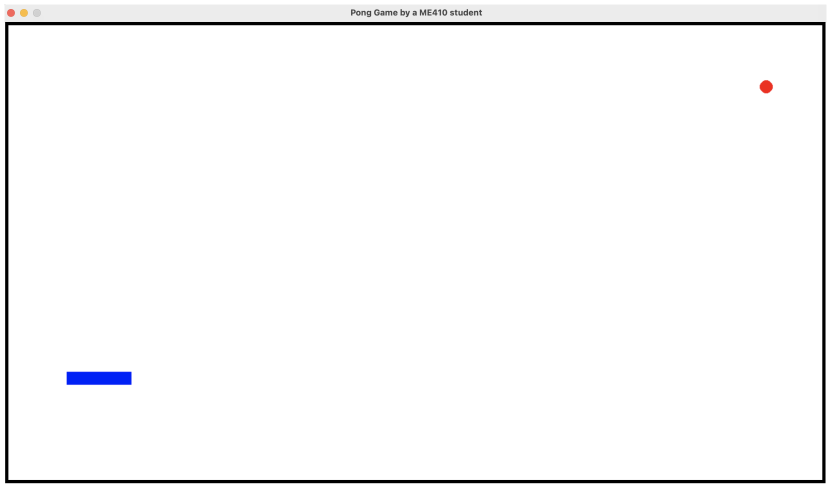

Figure 1: After running *basic_pong.py* in Python you will see a figure that looks like this.

Hints to help you get started: There are a series of constants that are defined (e.g., x_max, vx, dt). These should be self-explanatory from their names. If not, you will have to understand their meaning by skimming through the code and/or varying the constants and running the code. There are variables that are set by the physics of the ball and paddle (e.g., ball_pos, paddle_pos). You will use these variables along with the constants to code the game.

- Figure 2 gives some hints on the pong board including the origin, axis, boundaries for the walls, and paddle location.
- Basic pong edit #1: You will have to detect if the ball has hit the obstacle, the left wall or the right wall or the top wall or the paddle. After detection, you need to impose conditions on the velocity of the ball after it hits the particular wall/paddle. More specifically, The velocity of the ball before it hits the wall/paddle is known. Let us call this: $\vec{v}^- = v_x^- \hat{\imath} + v_y^- \hat{\jmath}$ (known) where $\hat{\imath}$ and $\hat{\jmath}$ are unit vectors in the x- and y-direction respectively. The velocity after the ball leaves the wall/paddle is unknown. Let us call this: $\vec{v}^+ = v_x^+ \hat{\imath} + v_y^+ \hat{\jmath}$ (unknown). Using physics, the unknown velocity may be found from the known velocity as follows.

  1. If the ball hits the left or right wall, the velocities after collision are: $v_y^+ = v_y^-$ and $v_x^+ = -e_x v_x^-$.
  2. If the ball hits the top wall or the paddle, the velocities after collision are: $v_x^+ = v_x^-$ and $v_y^+ = -e_y v_y^-$.

  where e_x and e_y are numbers that you can choose to make the game interesting. Initially, it is suggested that you put $e_x = e_y = 1$ to get the game to work. Other suggestions are: you could keep $e_x = 1$ and change $e_y = 1 + 0.1 \times$ np.random.rand() (where rand() is a random number generator function available once you import numpy as np) so that the ball accelerates if it hits the paddle or top wall, you could also reverse the logic to $e_y = 1$ and $e_x = 1 + 0.1 \times$ np.random.rand(). Feel free to choose other variants.

- Basic pong edit #2: You need to ensure that the paddle stops when it touches the left or right wall.

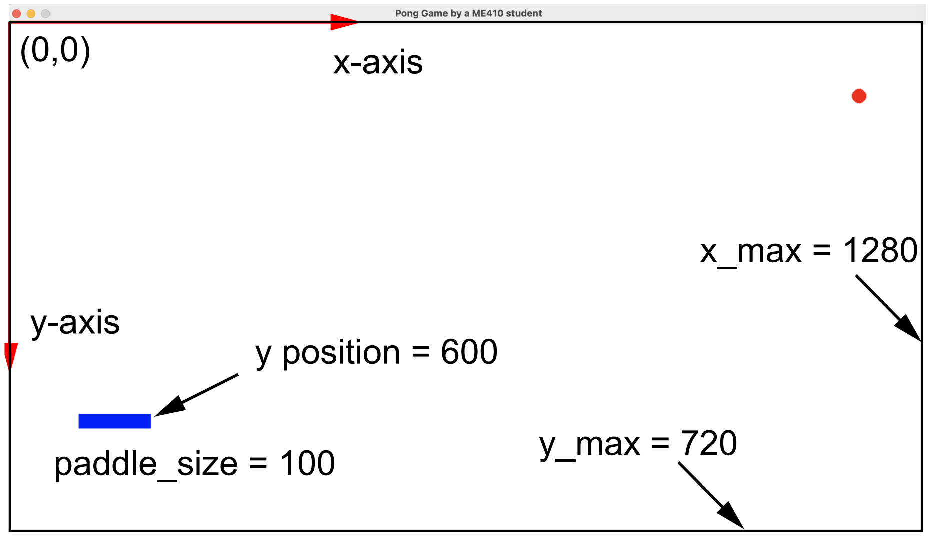

Figure 2: The pong figure with some key variables in the code.

4 Stage 2: Programming the advanced pong game

The advanced version of the game may be programmed once you are done with the basic version above. You will add *at least* three innovations to the game. One of them has to be adding a new visual element that interacts with the paddle/ball in some way (see point 1 below) and other two totally up to you. Here are some ideas but you are free to come up with your own.

1. It is mandatory to add at least *ONE* visual element that interacts with the ball or the paddle. For example, (1) you could add random stationary balls on the board, which garner extra points when hit; (2) you could add a stationary/moving horizontal bar on the board that would garner extra point if the ball hits the board; (3) you could create a two-ball pong game but note that it is difficult to play such a game unless you clearly think on how to play such a game to make it interesting (perhaps one slows down one the other is hit).

2. You could add have a scoring criteria and show it explicitly as the person plays the game. You could keep a track of the highest score and display it. For the latter, you will have to write the score to a file on your drive (search for the command "fopen", "fclose", "fread", "fwrite").

3. You could add up/down movement to the paddle in addition to the left/right movement.

NOTE: I do not recommend making the innovations too complex (e.g., creating a two-player pong) so that you are able to finish the project on time. Also, please put your name using "set_caption" (see code for a snippet). Don't forget to increase the initial speed of the ball else it will be too slow to be interesting. For *pygame* API see here https://www.pygame.org/docs/.