Pong V2

CS105 Final Project Report

Team Members: Joseph Tadrous, Yikang Lin, Keith Mburu

Summary:

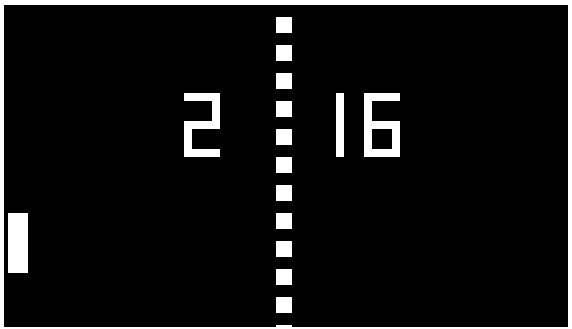
In this project, our aim is to develop the popular arcade game, Pong, using the knowledge and experience gained throughout CS105 lectures and labs. Moreover, some features are added to the ordinary pong game. Our game has two modes allowing the player to choose whether s/he wants to play a multiplayer game against one of his or her friends or to play against the AI with three different levels of difficulty. There are some other useful features to enhance the user experience such as pausing the game and returning to the main menu. To clarify:

- ***Multiplayer Mode:***
    - There are two players, and each controls one paddle
    - Player 1 uses W/S on keyboard to move the paddles vertically
    - Player 2 uses UP/DOWN on keyboard to move the paddles vertically

- ***AI Mode:***
    - One player playing against the AI.
    - There are 3 different levels of AI difficulty to play against
        - Easy
        - Normal
        - Hard

Background:

Pong game is one of the earliest arcade video games. Pong is a



two-dimensional sports game that simulates table tennis. Players control an in-game paddle by moving it vertically across the left or right side of the screen. While one player controls the paddle on the left side of the screen, the other player controls the paddle on the right side of the screen. Players use the paddles to hit a ball back and forth. The goal is for each player to reach eleven points before the opponent; points are earned when one fails to return the ball to the other. Note that we decided to make individual matches more fast paced by making the winning score 5 points instead of 11 points. Also we made changes to other aspects of the game in order to make it more accessible and interesting to play, such as making one paddle red and the other blue, making the UI clearer, and adding better sound effects.

Motivation:

During the course, we have learned a range of programming concepts. While we completed a series of labs targeting various programming skills, our team was interested in exploring how we can apply those skills in other aspects. At the same time, we all had an interest in classical computer games, a more recreational aspect of Python. Thus, we decided to write the classical computer game Pong using Python. Another reason that we chose to focus on Pong game was that it was not only a great opportunity to practice the skills learned in class-looping, if else, and so on- but also could expand our knowledge.

Main Contributions:

(To be explained in more detail in the next section)

- Object-oriented programming and its implementations
- Winsound and pygame packages
- Use of Sprite class
- Implementation of AI mode with different levels of difficulty

- Implement Pause function to pause the game and resume it.
- Display image on a specified screen and modified the display of the screen
- Detect collision between two sprites
- Make ball bounce after colliding with paddles
- Make the ball bounce on the horizontal sides of the screen
- Inheritance and override
- Control sprites(the paddles) using keyboards
- Producing certain outputs in response to certain keystrokes.

Methods, materials, data structures, packages, and algorithms:

- In our code, there are two classes: ball class and paddle class. In paddle class, we defined its properties: color, width, height, position index. Also, we defined functions called **moveup()** and **movedown()** to control the movement of paddles. In ball class, we also defined its properties: color, width, height, position index, velocity. Also, we defined functions called **update()** and **bounce()**. The **update()** function is used to update the position of the ball, and **bounce()** is used to change the velocity of the ball when a collision with paddle is detected.
- In our program, we used winsound package to play music while the game is running. The following code is an instance of using **playsound()** functions in winsound package to play music:

  ```
  winsound.PlaySound("/Users/Keith/Downloads/backtrack.wav",
  winsound.SND_ASYNC | winsound.SND_LOOP)
  ```

  We used pygame package in order to use functions embedded in the package. The following code is an instance which we used functions embedded in pygame package to display a screen:

  ```
  screen = pygame.display.set_mode((700, 500),
  pygame.FULLSCREEN)
  ```

- Ball class and paddle class inherit Sprite class. We did this because we wanted to use functions embedded in Sprite class. The following code is an instance of detecting the collision between two sprite objects using embedded function **collide_mask()** in Sprite class:

  ```
  if pygame.sprite.collide_mask(ball, paddleRed) or
  pygame.sprite.collide_mask(ball, paddleBlue):
              ball.bounce()
  ```

- We implemented an AI mode in which one player would face off against the computer. To do this, we made AI's paddle move either up or down depending on the position of the ball in relation to it. If the ball was above the paddle, the paddle would move up, and vice versa. However, we didn't want the AI to be unbeatable, so we added some error while moving vertically, making it choose a random pixel(position for the AI paddle) from the interval 2 * (-1, Difficulty * 2). The -1 causes the paddle of the AI to be pulled towards the wrong direction at times. The following code shows how we implement the AI mode:

  ```
   if ball.rect.y > paddleRed.rect.y:
          paddleRed.rect.y = paddleRed.rect.y + 2 *
  random.randint(-1, Difficulty * 2)
   if ball.rect.y < paddleRed.rect.y:
          paddleRed.rect.y = paddleRed.rect.y - 2 * random.randint(-1,
  Difficulty * 2)
  ```

- To enhance the user experience, Pause() function was implemented. The user can press T to pause or resume the game. A flag was used to indicate that the game is paused. If the flag is True, the Pause() function loop will run. If the flag is False, then the Pause() loop breaks and the program's main loop will continue running. The following code is the implementation of **Pause()** function

```
def Pause():
  paused = True
  while paused:
     for event in pygame.event.get():
        if event.type == pygame.QUIT:
           pygame.quit()
           quit()
        if event.type == pygame.KEYDOWN:
           if event.key == pygame.K_t:
              paused = False
           if event.key == pygame.K_ESCAPE:
              pygame.quit()
              quit()
```

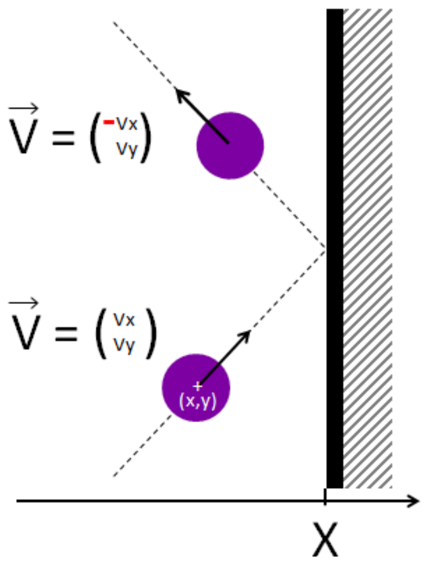
- We were able to clear the screen by filling it with a certain color using the **surface.fill()** pygame function, whose parameter is a color, specified by its level of Red, Green, and Blue. For example, we often used **screen.fill(BLACK)** to clear the screen after defining BLACK to be (0,0,0). In addition, we displayed an image on the screen by assigning **pygame.image.load()** to image variable with the file path of the image as the parameter, then using **screen.blit()** with image variable and the intended position of the image as parameters of the function. The following is an instance of using these two functions:

```
image = pygame.image.load("/Users/Keith/Downloads/Pygamer credits.png")
 screen.blit(image, (0, 0))
pygame.display.flip()
```

- In our code, we succeeded at writing a formula to detect the collision between the ball and paddles, and we were able to change the velocity of the ball after collision. As we have mentioned, we used **collide_mask()** function embedded in Sprite class to detect the collision between two sprite objects. The following is an instance of using such function

  ```
  if pygame.sprite.collide_mask(ball, paddleRed) or
  pygame.sprite.collide_mask(ball, paddleBlue):
          ball.bounce()
  ```

  Also, we were able to change the velocity of the ball after its collision with a paddle using **bounce()** function. The function works by reversing the sign for x-component velocity and randomly assigning a new y-component of velocity with different magnitude without changing its direction. By letting the program choose a random y-component velocity in the interval (-10, 10), we ensured that the ball's velocity would always be unpredictable. The following picture illustrates the mechanism of how the velocity of a ball changes before and after collision

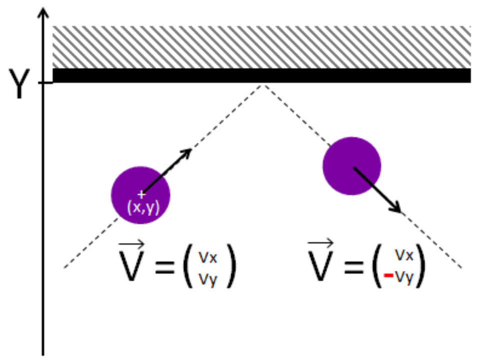


$\vec{V} = \begin{pmatrix} -V_x \\ V_y \end{pmatrix}$
$\vec{V} = \begin{pmatrix} V_x \\ V_y \end{pmatrix}$
(x,y)
X

- We made the ball bounce when it collides with the bottom(y-pixel490) and top(y-pixel=85) sides of the screen. When the ball collides with the top or bottom side of the screen, we invert the direction of the y-component velocity without changing its magnitude. At the same time, we will check whether the x-pixel of the ball is within the horizontal bounds of the screen(0<=x<= 690). The following code is the implementation of how to change the velocity of the ball when it touches the top or bottom side of the screen:

```
if ball.rect.x <= 690 and ball.rect.x >= 0:
    if ball.rect.y < 85:
        ball.velocity[1] = -ball.velocity[1]
    if ball.rect.y > 490:
        ball.velocity[1] = -ball.velocity[1]
```

The following figure illustrates the mechanism of how the x-component and y-component velocity of the ball change when the ball collides with the bottom or top side of the screen.


Y
(x,y)
$\vec{V} = \begin{pmatrix} V_x \\ V_y \end{pmatrix}$
$\vec{V} = \begin{pmatrix} V_x \\ -V_y \end{pmatrix}$

- As we used Object Oriented Programming in our program, we had to understand the concept of Inheritance. Inheritance allows us to define a class that inherits all the methods and properties from another class. Parent class is the class being inherited from. Child class is the class that inherits from another class. In our program, the parent class is the Sprite Class, while the child classes are both the Ball class and the Paddles class. For instance, we created the Ball class and inherits some of its properties from the parent class, Sprite. The following is an instance of Ball class inheriting Sprite class

```
class Ball(pygame.sprite.Sprite):

# This class represents a ball.  It derives from the "Sprite" class in
Pygame.

  def __init__(self, color, width, height):

    super().__init__()

self.image = pygame.Surface([width, height])

self.image.fill(BLACK)

self.image.set_colorkey(BLACK)
```

Also, we overrode the **update()** function in the Ball class. The following code is the implementation.

```
class Ball(pygame.sprite.Sprite):

       ...

       def update(self):

               self.rect.x += self.velocity[0]

               self.rect.y += self.velocity[1]
```

- A major part of our program is to control the movement of the paddles. We defined keys as **keys = pygame.key.get_pressed()** to get input from the user when certain keys are pressed. Then we call the methods written in Paddle class: **moveUp()** and **moveDown()**. The following code shows when users press certain keys, paddles will move up or down.

```
if keys[pygame.K_w]:
  paddleRed.moveUp(6)
if keys[pygame.K_s]:
  paddleRed.moveDown(6)
if keys[pygame.K_UP]:
  paddleBlue.moveUp(6)
if keys[pygame.K_DOWN]:
  paddleBlue.moveDown(6)
```

- An essential part in our program is to produce certain outputs in response to certain keystrokes. In other words, we programmed that users can change the display of the screen during Pong game using keyboard. For example, pressing ENTER starts the game, pressing P starts the multiplayer mode, pressing C starts the AI mode, pressing E/N/H chooses the difficulty, pressing T pauses the game, pressing R returns to main menu, pressing M minimizes the screen, and pressing ESCAPE quits the game. To implement these we used the built-in function **event.key** and **pygame.K**. The following code shows when certain key is pressed, the corresponding if-section code will run.

```
if event.key == pygame.K_t:
  Pause()
if event.key == pygame.K_ESCAPE:
  Continue = False
if event.key == pygame.K_m:
  screen = pygame.display.set_mode((700, 500))
```

Skills:

- Object Oriented Programming; Classes, Methods
- Pygame Library
- Variables, Assignment, Rebinding, Scope of variables,
- Conditionals, if-else, nested if-else, chain of conditionals,
- Loops (while, for): nested loops,
- Functions, arguments, default arguments

Challenges:

- The playsound package we used made sound choppy initially
- Difficulty getting the game to restart after someone wins
- Paddles not responding to keystrokes at certain points
- Implementing a beatable AI mode with different difficulties
- Making AI difficulties as accurate as possible
- Getting the game to display in fullscreen mode
- Resetting the scores, ball and paddle positions after each game
- Arriving at a more efficient velocity formula for the ball
- Making the ball always move horizontally when each round of the game starts

Conclusion & Future Work:

To sum up, we succeeded in developing the Pong game with two modes (Multiplayer & against AI) using Python and Pygame Library. Some additional features were added to the game to make it more interesting and to enhance the user experience. Many technical and non-technical lessons were learnt throughout the project.

- We improved our team-work and collaboration skills. We were able to divide the work into three sections and combine our work which is much more efficient than working alone.
- We enhanced our time management and organization skills. This project gave us the opportunity to design and follow a schedule. Making a schedule helped us complete our work in time and understand how to design a feasible schedule.
- We gained experience in debugging a code that was combined using sections of codes completed by different group members. In our program, Keith worked on the main code, Joseph worked on the paddle class, and Yikang worked on the ball class. Combing our work helped us understand how to debug as a team.
- We had a good understanding of how to approach a new area of knowledge in Computer Science. For our project, we need to use Pygame intensively which is a package that we were unfamiliar with. We viewed online tutorial, sample codes, pygame documentations to help us write our own Pong game.

If we had more time, we would have:

- Come up with a more efficient velocity formula for ball
- Modify the difficulty level in AI mode to make it as accurate as possible
- Add instructions for playing the game on the screen
- Increase number of players and balls
- Enable users to make selections of game mode using the mouse instead of keyboard

References:

https://www.101computing.net/pong-tutorial-using-pygame-adding-a-scoring-system/

https://nick.sarbicki.com/blog/learn-pygame-with-pong/

https://github.com/hamdyaea/Daylight-Pong-python3

http://trevorappleton.blogspot.com/2014/04/writing-pong-using-python-and-pygame.html

https://codereview.stackexchange.com/questions/24167/pong-game-in-python

https://www.youtube.com/watch?v=e4VRgY3tkh0

https://stackoverflow.com/questions/33426238/how-to-make-paddles-move-in-pygame-pong

https://stackoverflow.com/questions/38110911/pygame-font-not-working-and-not-showing-any-errors-also

https://stackoverflow.com/questions/15993287/what-is-the-correct-syntax-to-enable-fullscreen-toggle-in-pygame

https://github.com/nomelif/Audionodes/issues/10

https://stackoverflow.com/questions/13984066/pygame-restart

https://www.reddit.com/r/learnpython/comments/94wb54/how_to_reset_screen_in_pygame/

https://www.pygame.org/docs/ref/display.html

https://stackoverflow.com/questions/36636508/segmentation-fault-11-when-i-run-pygame

https://www.geeksforgeeks.org/python-display-images-with-pygame/

https://stackoverflow.com/questions/30744237/how-to-create-a-pause-button-in-pygame

https://www.pygame.org/docs/ref/color.html

https://archives.seul.org/pygame/users/Jan-2009/msg00128.html

https://stackoverflow.com/questions/32038240/pygame-screen-blit-not-working

https://stackoverflow.com/questions/16044229/how-to-get-keyboard-input-in-pygame

https://gamedev.stackexchange.com/questions/138888/user-input-text-in-pygame

https://www.pygame.org/docs/ref/draw.html

https://nerdparadise.com/programming/pygame/part5

https://stackoverflow.com/questions/43678493/pygame-is-not-responding-after-playing-an-audio-file

https://www.pygame.org/wiki/GettingStarted#Pygame%20Installation

Work Timeline:

Member's Name/Week	**Joseph Tadrous**	**Yikang Lin**	**Keith Mburu**
Nov. 21-23	- Be familiar with the built-in functions of libraries of pygame and random- 2 hour - Learn about the ideas of class, constructor and their implementations- 2 hours - Start thinking about the logic of pong game, especially Ball Class- 4 hours	- Be familiar with the built-in functions of libraries of pygame and random- 2 hour - Learn about the ideas of class, constructor and their implementations- 2 hours - Start thinking about the logic of pong game, especially Ball Class- 4 hours	- Be familiar with the built-in functions of libraries of pygame and random- 2 hour - Learn about the ideas of class, constructor and their implementations- 2 hours - Start thinking about the logic of pong game, especially Ball Class- 4 hours

Nov. 23-30	Implement the Ball Class: - Writing constructor- 1 hour Implement the Player Class: - Writing function to update the position of players (paddle)- 2 hours Start thinking about the main code-2 hours	Implement the Ball Class: - Writing function to update the position of the ball- 2 hours - Writing function for bounce- 1 hour Start thinking about the main code-2 hours	Implement the Ball Class: - Writing function to reset the ball- 2 hours Implement the Player Class: - Writing constructor- 2 hours Start thinking about the main code-2 hours
Dec. 1-9	- Implement the main code-12 hours - Debugging and finalizing-3 hours - Rehearse for the final	- Implement the main code-12 hours - Debugging and finalizing-3 hours - Rehearse for the final	- Implement the main code-12 hours - Debugging and finalizing-3 hours - Rehearse for the final

	presentation -2~3 hours	presentation -2~3 hours	presentation -2~3 hours