INFO1111: Computing 1A Professionalism

2023 Semester 1

Self-Learning Report

Submission number: 2

Github link: `https://github.com/zliu6203/SelfLearning`

Student name	Zhili Liu
Student ID	520451407
Topic	PyGame
Levels already achieved	X
Levels in this report	B

Contents

Instructions

Important: This section should be removed prior to submission.

You should use this LaTeX template to generate your self-learning report. Keep in mind the following key points:

- **Submissions**: There will be three opportunities during the semester to submit this report. For each submission you can attempt 1 or 2 levels. Each submission should use the same report, but amended to include new information.
- **Assessment**: In order to achieve level B, you must first have achieved level A, and so on for each level up to level D. This means that we will not assess a higher level until a lower level has been achieved (though we will review one level higher and give you feedback to help you in refining your work).
- **Minimum requirement**: Remember that in order to pass the unit, you must achieve at least level A in the self-learning (unless you achieve level B in both the skills and knowledge categories).
- **Using this template**: When completing each section you should remove the explanation text and replace it with your material.
- **Referencing**: You should also ensure that any resources you use are suitably referenced, and references are included into the reference list at the end of this document. You should use the IEEE reference style [**?**] (the reference included here shows you how this can be easily achieved).

1. Level A: Initial Understanding

1.1. Level A Demonstration

The three things that demonstrate my understanding for this topic are:

- Make a successful window for PyGame including some screen elements (sprites).
- Using sound libraries.
- Responding to user input.

1.2. Learning Approach

I approached my learning by looking for the latest tutorials online with PyGame, specifically the ones with project examples. I browsed both video and website tutorials including official documentation for many new methods and modules to build a successful game using the PyGame module of Python.

Experimentation was my main method of learning; testing all the different built-in libraries of methods and objects including sprites ("draw" module), sounds ("mixer" module), and the window itself ("display" module). I self-learnt by using previous knowledge of game development in C# and Java to my advantage - scenes, object priority, using external files etc. Additionally, I also used knowledge of window composition via WinForms and Tkinter.

1.3. Challenges and Difficulties

Due to the new syntax it was difficult to correctly load a window at all with the limited knowledge. The most difficult to learn was the while loop that the game is running on; it also contains a for loop which did not make much sense. I simply accepted it as fact instead of fully understanding it.

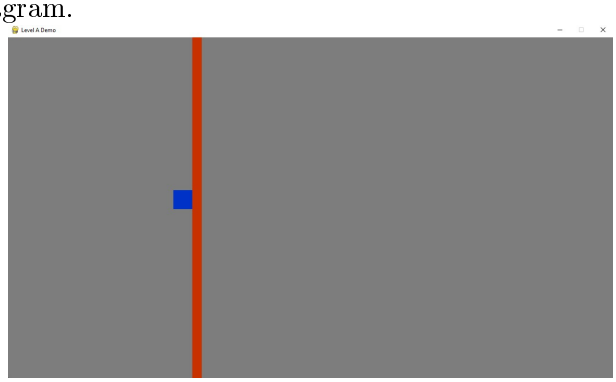
Even after the window was created, it was even more difficult to find any sprites to fill the window as opposed to Tkinter or WinForms, where sprites and screen elements were ready for use. Furthermore, PyGame did not include any built-in sound files and so sound effects & music must be found online or created locally.

1.4. Learning Sources

Learning Source	Contribution to Learning
Wiki tutorial	Understanding what each module does and its methods and objects.
Beginner video tutorial	Understanding the basics of every major aspect of PyGame.
Official documentation (mouse input)	Finding each method and the parameters necessary for these methods.
Sound help	Understanding how sound works and how to implement sound at an appropriate time.
Another video tutorial for shapes	Understanding what to do in the absence of assets with sprites.

1.5. Application artifacts

In level A, I made a very simple program in PyGame demonstrating that a successful window (1280x720) is opened with some screen elements (background, red line, blue player square). The blue square is responding to user input by following the coordinates of the mouse and not allow the mouse to go past the red line or the screen. When the cube touches the red line, a sound effect is played exactly once. This is demonstrated in this diagram.



In the code, I used simple python syntax with some new PyGame methods and elements. In the follwing screengrabs, I showed this by using the main function and setting initial variables.

```python
import pygame
import os

pygame.init()
pygame.mixer.init()

# initialise some values
WIDTH, HEIGHT = 1280, 720
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Level A Demo")
FPS = 60

BG_COLOUR = (125, 125, 125)

def draw(timer):
    # Draws the scene frame by frame

    # get the mouse position
    MOUSE_X, MOUSE_Y = pygame.mouse.get_pos()

    if MOUSE_X > 370:
        MOUSE_X = 370
        SOUND1 = pygame.mixer.Sound(os.path.join('sound_effect1.wav'))
        if timer == 0:
            pygame.mixer.Sound.play(SOUND1)
    # if the mouse passes the red line, trigger sound effect
```

(a) Code showing variables

```python
def main():
    clock = pygame.time.Clock()
    running = True
    k = 0
    # this allows the program to run continuously
    while running:
        clock.tick(FPS)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

        # for the timing of the sound effect
        if pygame.mouse.get_pos()[0] > 370:
            k += 1
        else:
            k = 9999999999999
        draw(k % 10000000000000)

    # once the program stops running, quit program
    pygame.quit()

if __name__ == "__main__":
    main()
```

(b) Code showing main function

Additionally, I used a simple sound effect from the web (sound_effect1.wav) to play whenever the player hits the red line. I also used pyinstaller to make the .py into a .exe which is also included in the Git repository.

2. Level B: Basic Application

Whilst level A is about doing something simple with the topic to just show that you have started to be able to use the tool or technology, level B is about doing something practical that might actually be useful.

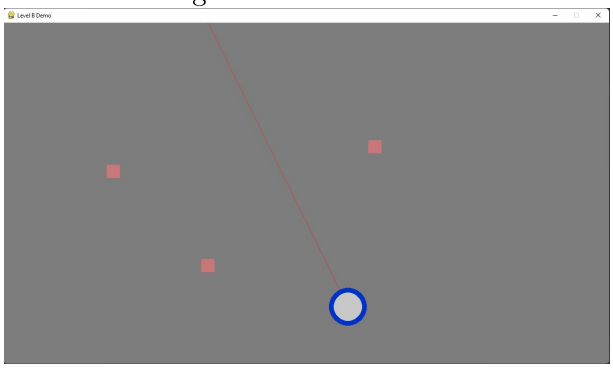
2.1. Level B Demonstration

The application I have developed is a simple game in PyGame. This involves the three points in the level B proposal: events such as scenes, making a simple game with an end goal (win/lose), and program simple enemies. I have demonstrated the first point by making different levels in the game, satisfied the second point by creating a game with a goal (to finish all 5 levels), and also programmed simple enemies with random movement and bouncing back from the screen if it goes offscreen.

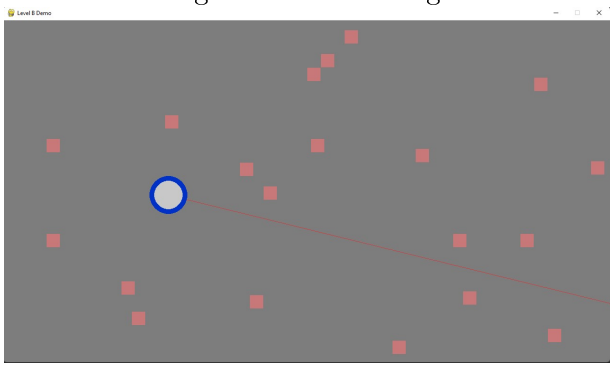
2.2. Application artifacts

I have created a simple topdown shooter game using built-in assets, modules, and tools only. The game I have made is simple: shoot all enemies using a periodic laser which can destroy all enemies collided with the laser. I used classes and small aspects of object oriented programming to achieve this.

In the following screengrab, the player can shoot a laser which only fires when the left mouse button is pressed and the cooldown for the laser is done. This laser collides with all enemies within its vicinity and destroys them. The player in blue and white can be controlled using WASD and the laser is shot in the direction of the cursor.



This screengrab shows the huge number of enemies in level 4 of this game:



This code snippet shows the object oriented programming and the hierarchy of classes and objects. The Entity class includes both the Player and Enemy classes as its children. The enemy class also contains code for random movement (see more in GitHub).

```python
class Entity:

    def __init__(self, x, y, speed):
        self.x = x
        self.y = y
        self.speed = speed

    def move_up(self):
        self.y -= self.speed

    def move_down(self):
        self.y += self.speed

    def move_left(self):
        self.x -= self.speed

    def move_right(self):
        self.x += self.speed


class Player(Entity):

    def display(self):
        pygame.draw.circle(WIN, (0, 50, 200), (self.x, self.y), 40)
        pygame.draw.circle(WIN, (200, 200, 200), (self.x, self.y), 30)


class Enemy(Entity):

    def __init__(self, x, y):
        super().__init__(x, y, 1)
        self.direction = "down"

    def display(self):
```

These two code snippets shows the Game class being the main class which handles all the events.

```python
class Game:

    LASER_COLOUR = 125, 125, 125
    def __init__(self, level):
        self.enemies = []
        for coord in level:
            self.enemies.append(Enemy(coord[0], coord[1]))
        self.player = Player(640, 360, 4)
        self.firing = False
        self.LASER_COLOUR = 125, 125, 125
        self.timer = 0


    def draw(self):
        # Draws the scene frame by frame

        WIN.fill(BG_COLOUR)
        MOUSE_X, MOUSE_Y = pygame.mouse.get_pos()

        is_left = pygame.mouse.get_pressed()[0] # is left click

        self.firing = is_left and self.timer % 40 in range(0, 14)

        if self.firing:
            LASER_COLOUR = 210, 60, 60
        else:
            LASER_COLOUR = 125, 125, 125

        pygame.draw.line(WIN, LASER_COLOUR, (self.player.x, self.player.y),
                         (512 * (MOUSE_X - self.player.x),
                          512 * (MOUSE_Y - self.player.y)))


        self.player.display()
```

(a) Code showing Game class

```python
        laser_coords = [(20/i * (MOUSE_X - self.player.x) + self.player.x,
                         20/i * (MOUSE_Y - self.player.y) + self.player.y) for i in range(1, 200)]


        for e in self.enemies:
            if self.firing:
                for coords in laser_coords:
                    if abs(coords[0] - e.x) < 19 and abs(coords[1] - e.y) < 19:
                        try:
                            self.enemies.remove(e)
                        except ValueError:
                            print("Error handler")

            if self.timer % 50 == 0 or e.x < 1 or e.y < 1 or e.x > 1251 or e.y > 691:
                e.change_dir()
            e.display()


        ke = pygame.key.get_pressed()
        if ke[pygame.K_a]:
            self.player.move_left()
        if ke[pygame.K_w]:
            self.player.move_up()
        if ke[pygame.K_d]:
            self.player.move_right()
        if ke[pygame.K_s]:
            self.player.move_down()

        self.timer += 1

        pygame.display.update()
        print(len(self.enemies))
        return len(self.enemies)
```

(b) Code showing how enemies work

3. Level C: Deeper Understanding

Level C focuses on showing that you have actually understood the tool or technology at a relatively advanced level. You will need to compare it to alternatives, identifying key strengths and weaknesses, and the areas where this tool is most effective.

3.1. Strengths

What are the key strengths of the item you have learnt? (50-100 words)

3.2. Weaknesses

What are the key weaknesses of the item you have learnt? (50-100 words)

3.3. Usefulness

Describe one scenario under which you believe the topic you have learnt could be useful. (50-100 words)

3.4. Key Question 1

Note: This question is in the table in the 'Self Learning: List of Topics' page on Canvas. (50-100 words)

3.5. Key Question 2

Note: This question is in the table in the 'Self Learning: List of Topics' page on Canvas. (50-100 words)

4. Level D: Evolution of skills

4.1. Level D Demonstration

This is a short description of the application that you have developed. (50-100 words). **IMPORTANT:** *You might wish to submit this as part of an earlier submission in order to obtain feedback as to whether this is likely to be acceptable for level D.*

4.2. Application artifacts

Include here a description of what you actually created (what does it do? How does it work? How did you create it?). Include any code or other related artefacts that you created (these should also be included in your github repository).

If you do include screengrabs to show what you have done then these should be annotated to explain what it is showing and what the application does.

4.3. Alternative tools/technologies

Identify 2 alternative tools/technologies that can be used instead of the one you studied for your topic. (e.g. if your topic was Python, then you might identify Java and Golang)

4.4. Comparative Analysis

Describe situations in which both your topic and each of the identified alternatives would be preferred over the others (100-200 words).