Final Project
Program Design and Methods

Project Name: **Klee Fishing Mania!**

Name: **Reinhart A Tanto**
Student ID: **2502043053**
Class: **L1CC**

Project Specification

A game for people to enjoy. Concept of game comes from *GENSHIN IMPACT* where a character named *KLEE* loves to go bomb fishing. Objective of the game is to catch as many fish as possible before the time limit. Klee will have 3 bombs in the begining and after she throws those 3 bombs, a “Bomb Cache” will spawn to refill her bombs. Every 10 fish caught the game would increase in difficulty as the speed of the fish increased. The user has a time limit until night(2 mins) to catch as many fish as possible.

Input

1. Mouse position and clicks for navigating through the menu.
2. “W”, “A”, “S”, “D”, and “Space” keys for playing game

Output

1. Final amount of fish caught

Solution Design

1. Main Menu
2. Controls Screen
3. Main Game
4. End Screen

Main Menu

Consists of 2 buttons, “Start” which starts the game, and “Quit” which closes the game.

Controls Screen

First screen that is shown when “Start” button is pressed in main screen. Shows controls used to control the character in game. “W”, “A”, “S”, and “D” for movement, “Space” to throw bomb. Also shows the main objective of the game which is to catch as many fish before night time. Has the Main Game in the background and users can move the character to start.

Main Game

Scene where the game is played. Accessed by moving the character in the previous screen. User moves around and throws the bombs at the fish. After 3 bombs are thrown the counter on the top right corner of the screen is empty and user will have to pick up a “Bomb Cache” to refill it. Time of day is show on the clock symbol on top left corner of the screen. When time runs out the Scene changes to the End Screen.

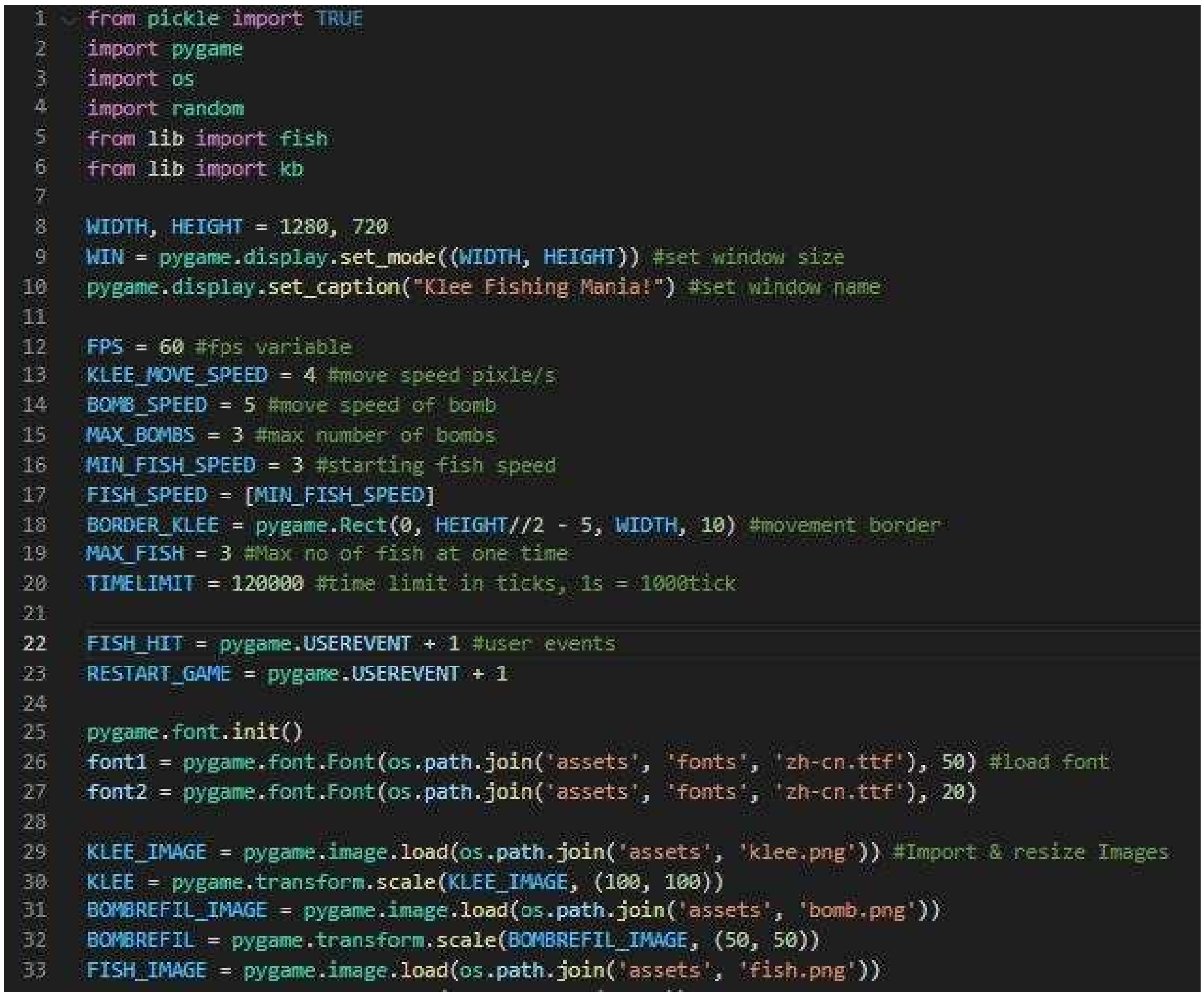
End Screen

Shows how many fish were caught and a thank you message for playing. Has 2 buttons “Quit” which closes the game, and “Again” which returns you to the main menu.

Implementation and explanation of code

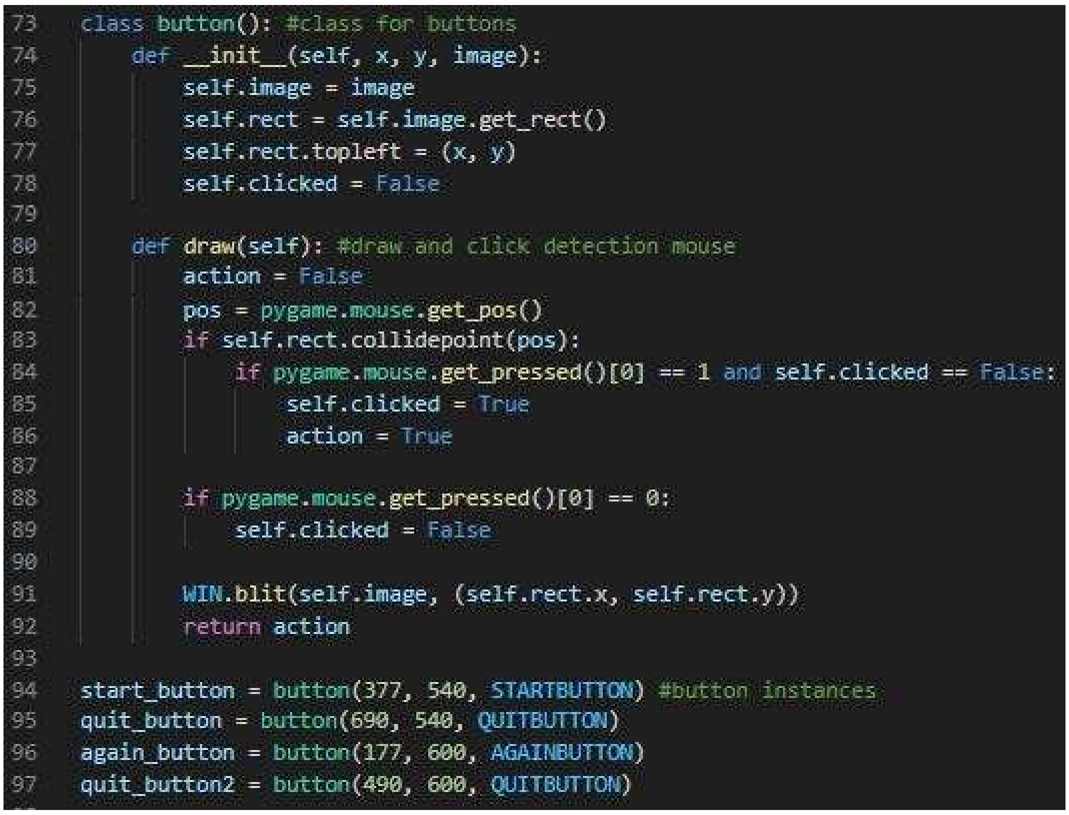
For this project I used 4 modules the gamepy, random, pickle, and os module. I also separated my code into 3 modules; runme.py(Main Module), fish.py(Fish movement and interactions), and kb.py(Character and bomb movement and interactions). I created 1 class for the buttons used in the Main Menu and End Screen.

runme.py



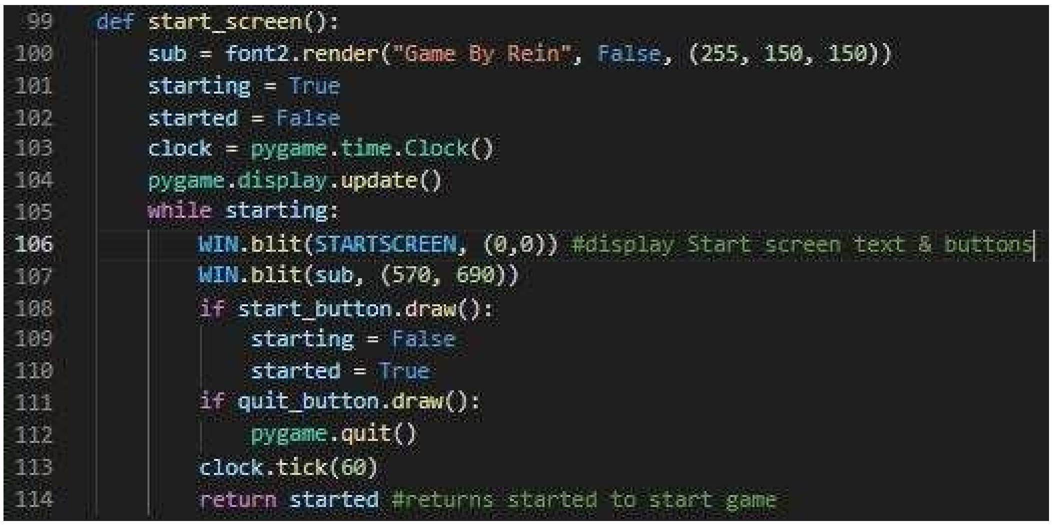
```python
from pickle import TRUE
import pygame
import os
import random
from lib import fish
from lib import kb

WIDTH, HEIGHT = 1280, 720
WIN = pygame.display.set_mode((WIDTH, HEIGHT)) #set window size
pygame.display.set_caption("Klee Fishing Mania!") #set window name

FPS = 60 #fps variable
KLEE_MOVE_SPEED = 4 #move speed pixle/s
BOMB_SPEED = 5 #move speed of bomb
MAX_BOMBS = 3 #max number of bombs
MIN_FISH_SPEED = 3 #starting fish speed
FISH_SPEED = [MIN_FISH_SPEED]
BORDER_KLEE = pygame.Rect(0, HEIGHT//2 - 5, WIDTH, 10) #movement border
MAX_FISH = 3 #Max no of fish at one time
TIMELIMIT = 120000 #time limit in ticks, 1s = 1000tick

FISH_HIT = pygame.USEREVENT + 1 #user events
RESTART_GAME = pygame.USEREVENT + 1

pygame.font.init()
font1 = pygame.font.Font(os.path.join('assets', 'fonts', 'zh-cn.ttf'), 50) #load font
font2 = pygame.font.Font(os.path.join('assets', 'fonts', 'zh-cn.ttf'), 20)

KLEE_IMAGE = pygame.image.load(os.path.join('assets', 'klee.png')) #Import & resize Images
KLEE = pygame.transform.scale(KLEE_IMAGE, (100, 100))
BOMBREFIL_IMAGE = pygame.image.load(os.path.join('assets', 'bomb.png'))
BOMBREFIL = pygame.transform.scale(BOMBREFIL_IMAGE, (50, 50))
FISH_IMAGE = pygame.image.load(os.path.join('assets', 'fish.png'))
```

The main module of the game which consists of most of the integral code. First part of code consisted of importing modules, declaring global variables, setting window size, and loading fonts & images. os module is used to find the path of the files regardless of the operating system.



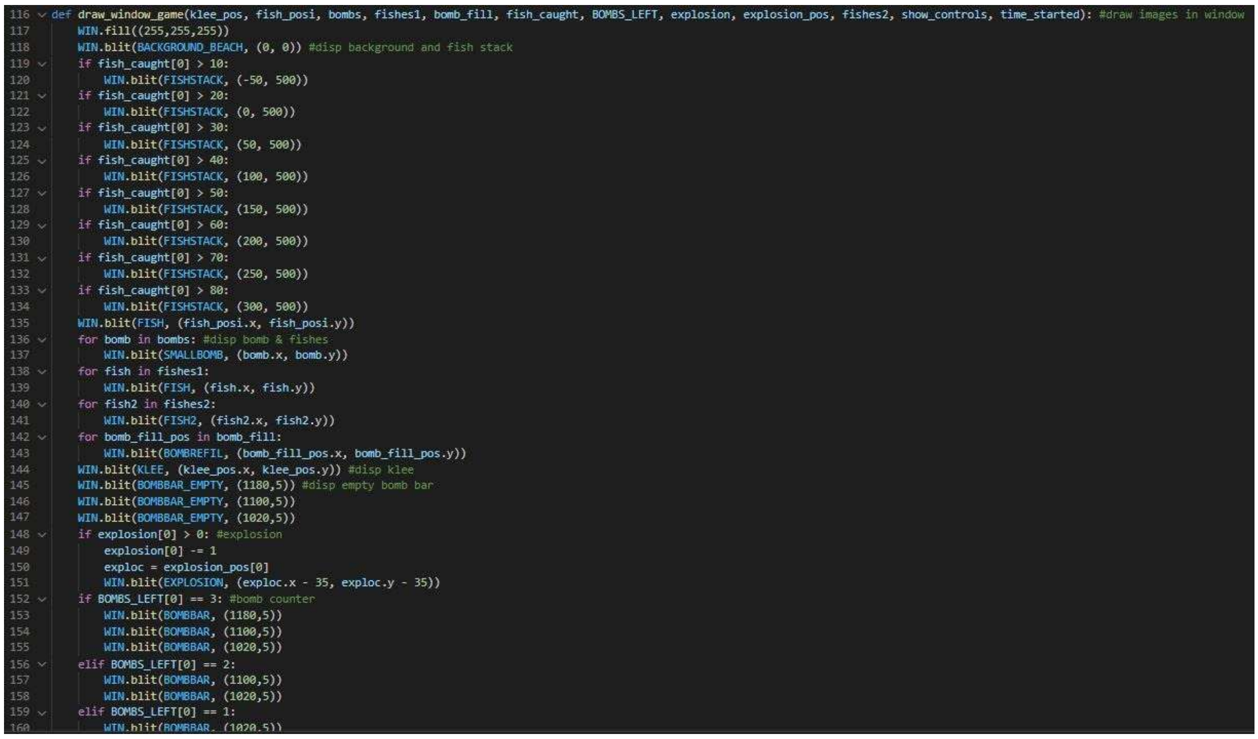
```python
class button(): #class for buttons
    def __init__(self, x, y, image):
        self.image = image
        self.rect = self.image.get_rect()
        self.rect.topleft = (x, y)
        self.clicked = False

    def draw(self): #draw and click detection mouse
        action = False
        pos = pygame.mouse.get_pos()
        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:
                self.clicked = True
                action = True

        if pygame.mouse.get_pressed()[0] == 0:
            self.clicked = False

        WIN.blit(self.image, (self.rect.x, self.rect.y))
        return action

start_button = button(377, 540, STARTBUTTON) #button instances
quit_button = button(690, 540, QUITBUTTON)
again_button = button(177, 600, AGAINBUTTON)
quit_button2 = button(490, 600, QUITBUTTON)
```

Class buttons created using module pygame for loading image and click detection. Created button instances with loaded images. Function draw used to display the button on screen and returns True value when clicked and False when mouse click is lifted.



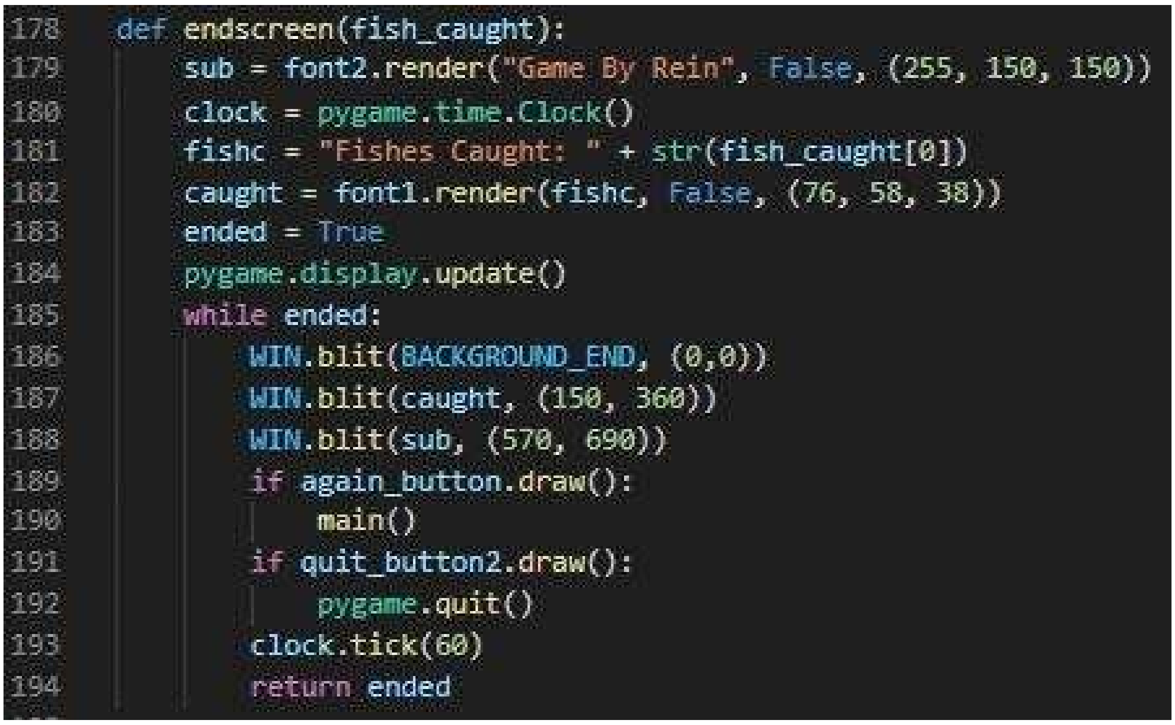
```python
def start_screen():
    sub = font2.render("Game By Rein", False, (255, 150, 150))
    starting = True
    started = False
    clock = pygame.time.Clock()
    pygame.display.update()
    while starting:
        WIN.blit(STARTSCREEN, (0,0)) #display Start screen text & buttons
        WIN.blit(sub, (570, 690))
        if start_button.draw():
            starting = False
            started = True
        if quit_button.draw():
            pygame.quit()
        clock.tick(60)
        return started #returns started to start game
```

start_screen() function for showing the STARTING SCREEN. Button class and functions from pygame module used to display images, text and buttons.



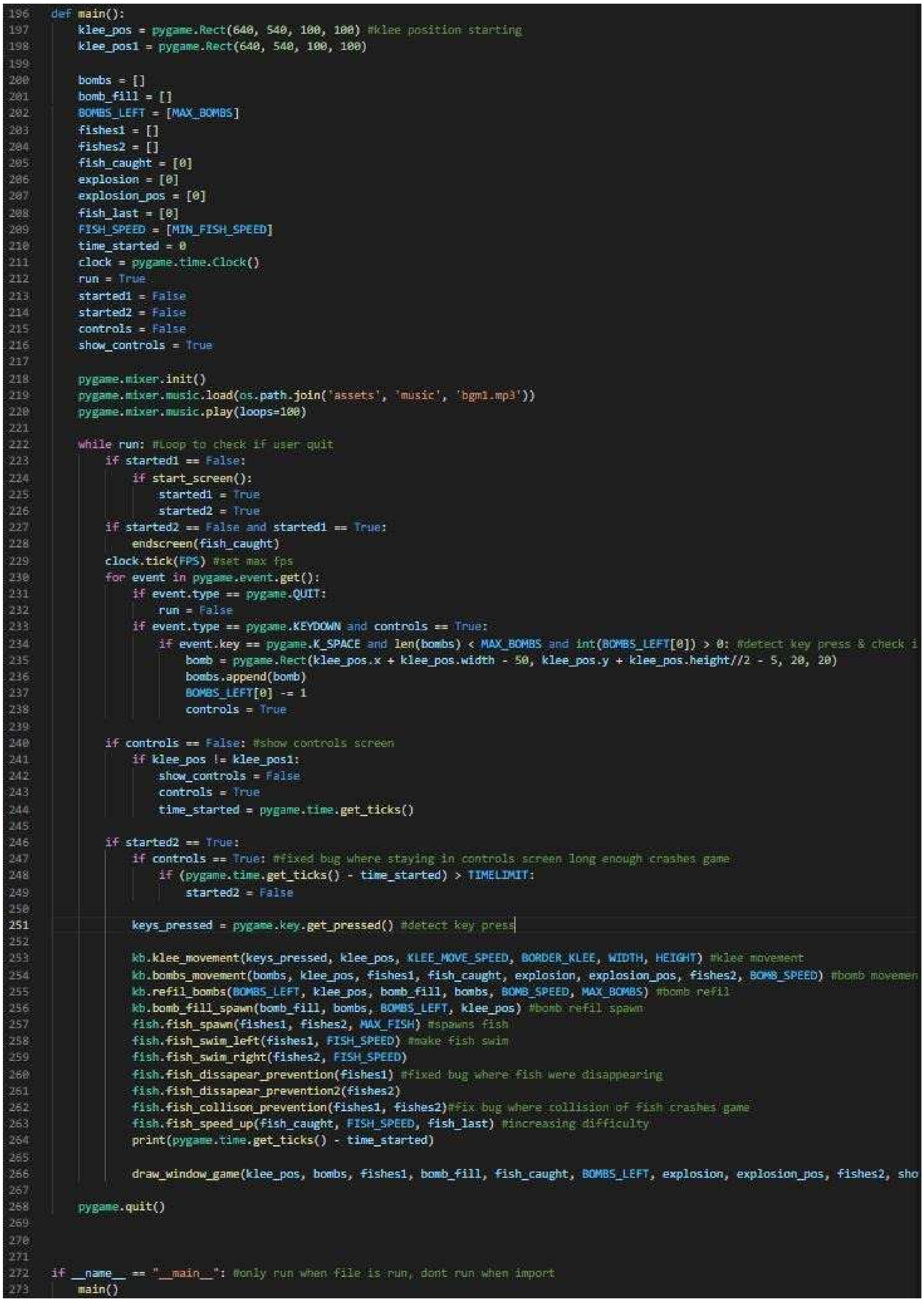
```python
def draw_window_game(klee_pos, fish_posi, bombs, fishes1, bomb_fill, fish_caught, BOMBS_LEFT, explosion, explosion_pos, fishes2, show_controls, time_started): #draw images in window
    WIN.fill((255,255,255))
    WIN.blit(BACKGROUND_BEACH, (0, 0)) #disp background and fish stack
    if fish_caught[0] > 10:
        WIN.blit(FISHSTACK, (-50, 500))
    if fish_caught[0] > 20:
        WIN.blit(FISHSTACK, (0, 500))
    if fish_caught[0] > 30:
        WIN.blit(FISHSTACK, (50, 500))
    if fish_caught[0] > 40:
        WIN.blit(FISHSTACK, (100, 500))
    if fish_caught[0] > 50:
        WIN.blit(FISHSTACK, (150, 500))
    if fish_caught[0] > 60:
        WIN.blit(FISHSTACK, (200, 500))
    if fish_caught[0] > 70:
        WIN.blit(FISHSTACK, (250, 500))
    if fish_caught[0] > 80:
        WIN.blit(FISHSTACK, (300, 500))
    WIN.blit(FISH, (fish_posi.x, fish_posi.y))
    for bomb in bombs: #disp bomb & fishes
        WIN.blit(SMALLBOMB, (bomb.x, bomb.y))
    for fish in fishes1:
        WIN.blit(FISH, (fish.x, fish.y))
    for fish2 in fishes2:
        WIN.blit(FISH2, (fish2.x, fish2.y))
    for bomb_fill_pos in bomb_fill:
        WIN.blit(BOMBREFIL, (bomb_fill_pos.x, bomb_fill_pos.y))
    WIN.blit(KLEE, (klee_pos.x, klee_pos.y)) #disp klee
    WIN.blit(BOMBBAR_EMPTY, (1180,5)) #disp empty bomb bar
    WIN.blit(BOMBBAR_EMPTY, (1100,5))
    WIN.blit(BOMBBAR_EMPTY, (1020,5))
    if explosion[0] > 0: #explosion
        explosion[0] -= 1
        exploc = explosion_pos[0]
        WIN.blit(EXPLOSION, (exploc.x - 35, exploc.y - 35))
    if BOMBS_LEFT[0] == 3: #bomb counter
        WIN.blit(BOMBBAR, (1180,5))
        WIN.blit(BOMBBAR, (1100,5))
        WIN.blit(BOMBBAR, (1020,5))
    elif BOMBS_LEFT[0] == 2:
        WIN.blit(BOMBBAR, (1100,5))
        WIN.blit(BOMBBAR, (1020,5))
    elif BOMBS_LEFT[0] == 1:
        WIN.blit(BOMBBAR, (1020,5))
```

draw_window_game() function for displaying & refreshing images in the window of the Main Game screen. Takes alot of parameters for validation if the image will be displayed and the location of the image.



```python
def endscreen(fish_caught):
    sub = font2.render("Game By Rein", False, (255, 150, 150))
    clock = pygame.time.Clock()
    fishc = "Fishes Caught: " + str(fish_caught[0])
    caught = font1.render(fishc, False, (76, 58, 38))
    ended = True
    pygame.display.update()
    while ended:
        WIN.blit(BACKGROUND_END, (0,0))
        WIN.blit(caught, (150, 360))
        WIN.blit(sub, (570, 690))
        if again_button.draw():
            main()
        if quit_button2.draw():
            pygame.quit()
        clock.tick(60)
        return ended
```

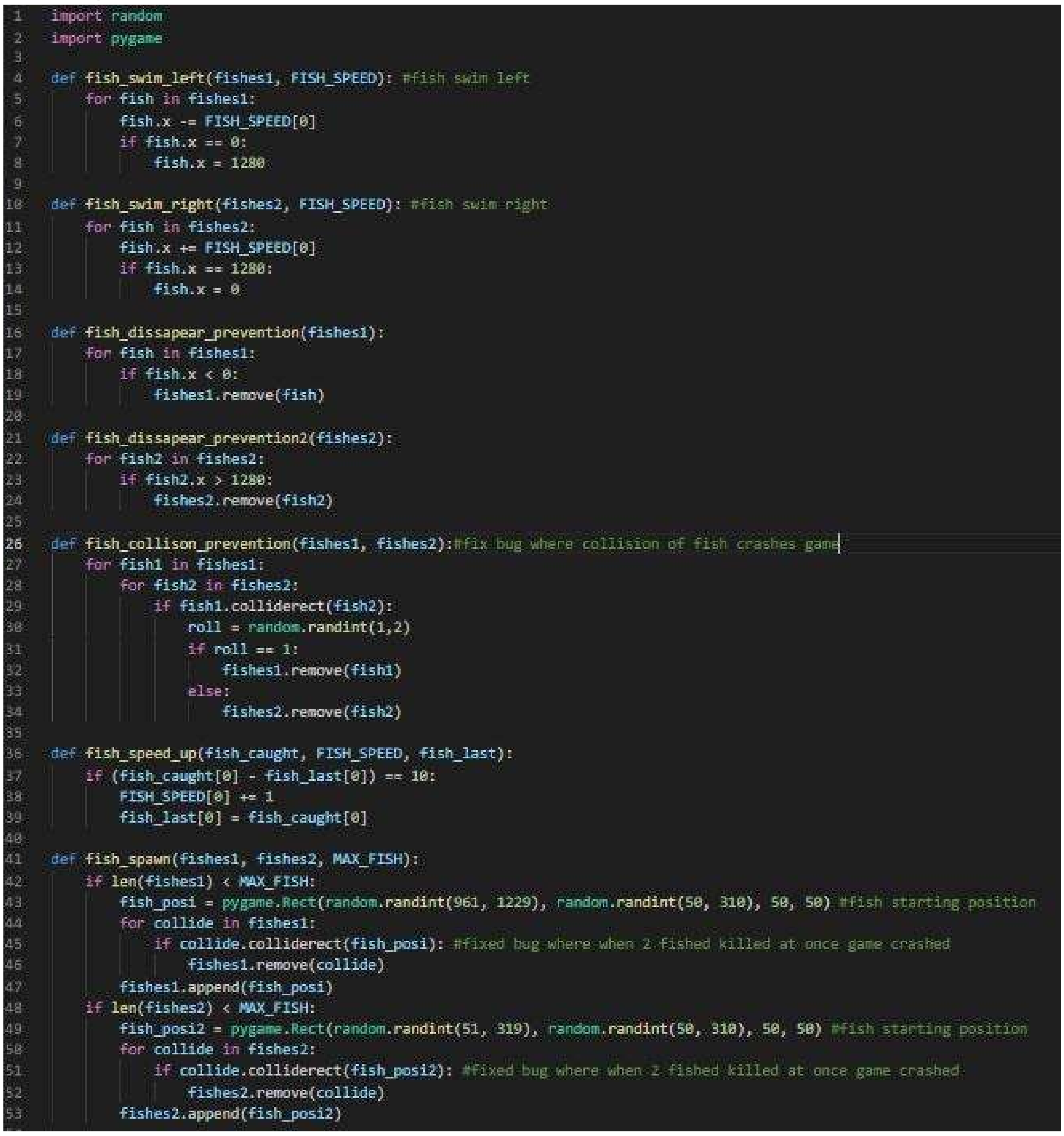
endscreen() function used to display the END SCREEN. Button class and functions from pygame module used to display images, text and buttons.



```python
def main():
    klee_pos = pygame.Rect(640, 540, 100, 100) #klee position starting
    klee_pos1 = pygame.Rect(640, 540, 100, 100)

    bombs = []
    bomb_fill = []
    BOMBS_LEFT = [MAX_BOMBS]
    fishes1 = []
    fishes2 = []
    fish_caught = [0]
    explosion = [0]
    explosion_pos = [0]
    fish_last = [0]
    FISH_SPEED = [MIN_FISH_SPEED]
    time_started = 0
    clock = pygame.time.Clock()
    run = True
    started1 = False
    started2 = False
    controls = False
    show_controls = True

    pygame.mixer.init()
    pygame.mixer.music.load(os.path.join('assets', 'music', 'bgm1.mp3'))
    pygame.mixer.music.play(loops=100)

    while run: #Loop to check if user quit
        if started1 == False:
            if start_screen():
                started1 = True
                started2 = True
        if started2 == False and started1 == True:
            endscreen(fish_caught)
        clock.tick(FPS) #set max fps
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
            if event.type == pygame.KEYDOWN and controls == True:
                if event.key == pygame.K_SPACE and len(bombs) < MAX_BOMBS and int(BOMBS_LEFT[0]) > 0: #detect key press & check i
                    bomb = pygame.Rect(klee_pos.x + klee_pos.width - 50, klee_pos.y + klee_pos.height//2 - 5, 20, 20)
                    bombs.append(bomb)
                    BOMBS_LEFT[0] -= 1
                    controls = True

        if controls == False: #show controls screen
            if klee_pos != klee_pos1:
                show_controls = False
                controls = True
                time_started = pygame.time.get_ticks()

        if started2 == True:
            if controls == True: #fixed bug where staying in controls screen long enough crashes game
                if (pygame.time.get_ticks() - time_started) > TIMELIMIT:
                    started2 = False

            keys_pressed = pygame.key.get_pressed() #detect key press

            kb.klee_movement(keys_pressed, klee_pos, KLEE_MOVE_SPEED, BORDER_KLEE, WIDTH, HEIGHT) #klee movement
            kb.bombs_movement(bombs, klee_pos, fishes1, fish_caught, explosion, explosion_pos, fishes2, BOMB_SPEED) #bomb movemen
            kb.refil_bombs(BOMBS_LEFT, klee_pos, bomb_fill, bombs, BOMB_SPEED, MAX_BOMBS) #bomb refil
            kb.bomb_fill_spawn(bomb_fill, bombs, BOMBS_LEFT, klee_pos) #bomb refil spawn
            fish.fish_spawn(fishes1, fishes2, MAX_FISH) #spawns fish
            fish.fish_swim_left(fishes1, FISH_SPEED) #make fish swim
            fish.fish_swim_right(fishes2, FISH_SPEED)
            fish.fish_dissapear_prevention(fishes1) #fixed bug where fish were disappearing
            fish.fish_dissapear_prevention2(fishes2)
            fish.fish_collison_prevention(fishes1, fishes2)#fix bug where collision of fish crashes game
            fish.fish_speed_up(fish_caught, FISH_SPEED, fish_last) #increasing difficulty
            print(pygame.time.get_ticks() - time_started)

            draw_window_game(klee_pos, bombs, fishes1, bomb_fill, fish_caught, BOMBS_LEFT, explosion, explosion_pos, fishes2, sho

    pygame.quit()



if __name__ == "__main__": #only run when file is run, dont run when import
    main()
```

main() function is the main part of this module. Starts with declaring all the local variables that will be used. Then music is played with the pygame module using the os module to find the path to the file. After that a loop starts which encases the remaining part of the function, uses the local variable “run” to decide whether the game is still running and quits the game if it isn’t running anymore or when the window is closed. Within this loop it calls start_screen() function which returns a boolean value. When the returned value is True it will start the “Main Game” but before that there will be a “Controls Screen” that is overlaid above the “Main Game” until the character is moved. pygame.event.get() is used to detect the keypresses and user interactions with the window. When a spacebar input is detected it appends the value of “bomb” to the array “bombs” which is used to tell the location of the bomb fired and move it. When the main game is started a local variable is used to store the tick at which it started so that after the time limit it calls the endscreen() function and this value is also used for the clock displayed in the “Main Game” screen. Another variable is used to store the keys pressed. After that the functions from the 2 sub-modules are called, and the draw_window_game() function is called to display the images on the window. In the end there is an if statement that only allows the main() function to run when it is launched directly.

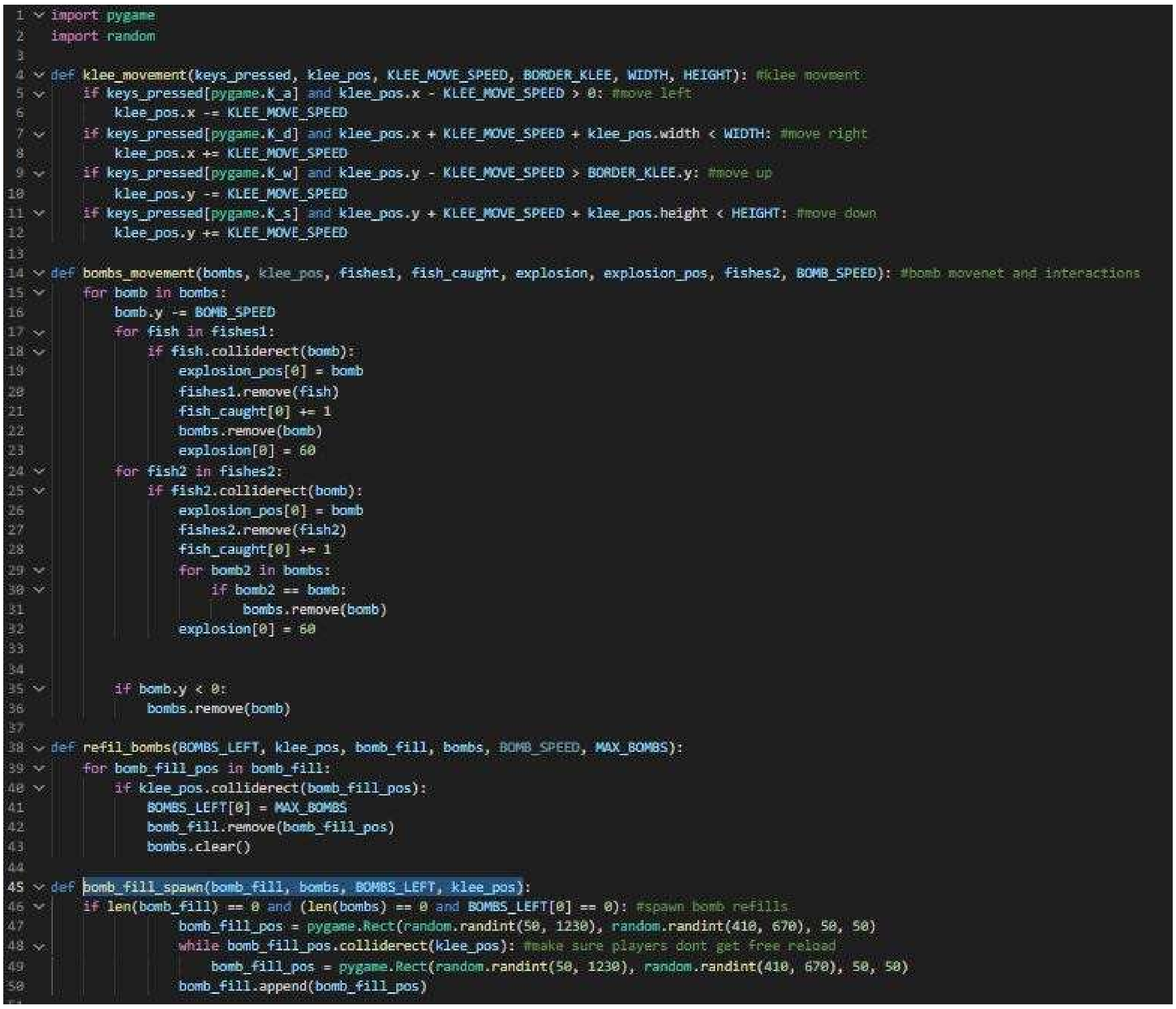
fish.py



```python
import random
import pygame

def fish_swim_left(fishes1, FISH_SPEED): #fish swim left
    for fish in fishes1:
        fish.x -= FISH_SPEED[0]
        if fish.x == 0:
            fish.x = 1280

def fish_swim_right(fishes2, FISH_SPEED): #fish swim right
    for fish in fishes2:
        fish.x += FISH_SPEED[0]
        if fish.x == 1280:
            fish.x = 0

def fish_dissapear_prevention(fishes1):
    for fish in fishes1:
        if fish.x < 0:
            fishes1.remove(fish)

def fish_dissapear_prevention2(fishes2):
    for fish2 in fishes2:
        if fish2.x > 1280:
            fishes2.remove(fish2)

def fish_collison_prevention(fishes1, fishes2):#fix bug where collision of fish crashes game
    for fish1 in fishes1:
        for fish2 in fishes2:
            if fish1.colliderect(fish2):
                roll = random.randint(1,2)
                if roll == 1:
                    fishes1.remove(fish1)
                else:
                    fishes2.remove(fish2)

def fish_speed_up(fish_caught, FISH_SPEED, fish_last):
    if (fish_caught[0] - fish_last[0]) == 10:
        FISH_SPEED[0] += 1
        fish_last[0] = fish_caught[0]

def fish_spawn(fishes1, fishes2, MAX_FISH):
    if len(fishes1) < MAX_FISH:
        fish_posi = pygame.Rect(random.randint(961, 1229), random.randint(50, 310), 50, 50) #fish starting position
        for collide in fishes1:
            if collide.colliderect(fish_posi): #fixed bug where when 2 fished killed at once game crashed
                fishes1.remove(collide)
        fishes1.append(fish_posi)
    if len(fishes2) < MAX_FISH:
        fish_posi2 = pygame.Rect(random.randint(51, 319), random.randint(50, 310), 50, 50) #fish starting position
        for collide in fishes2:
            if collide.colliderect(fish_posi2): #fixed bug where when 2 fished killed at once game crashed
                fishes2.remove(collide)
        fishes2.append(fish_posi2)
```

A sub-module which uses the random and pygame module. It has 7 functions which all relate to the fish's movement, interactions with each other, and spawning. fish_swim_left() and fish_swim_right() are used to display the direction at which the fish swim. fish_dissapear_prevention(), fish_dissapear_prevention2(), and fish_collision_prevention() are all functions to fix a bug I encountered where sometimes the fish would not be respawned if they left the screen at the same time, and collision prevention is for a bug where the game would crash when 2 fish are removed at the same time. fish_collision_prevention() has a random generator which randomises the fish to be removed so that the player cannot predict which one will disappear when 2 fish going opposite directions collide. fish_speed_up() was added to increase the difficulty of the game as the player progressed. fish_spawn() is for spawning the fish in random locations on the left and right area of the screen depending on which direction they will swim. It also has a part built in to prevent fish spawning together.

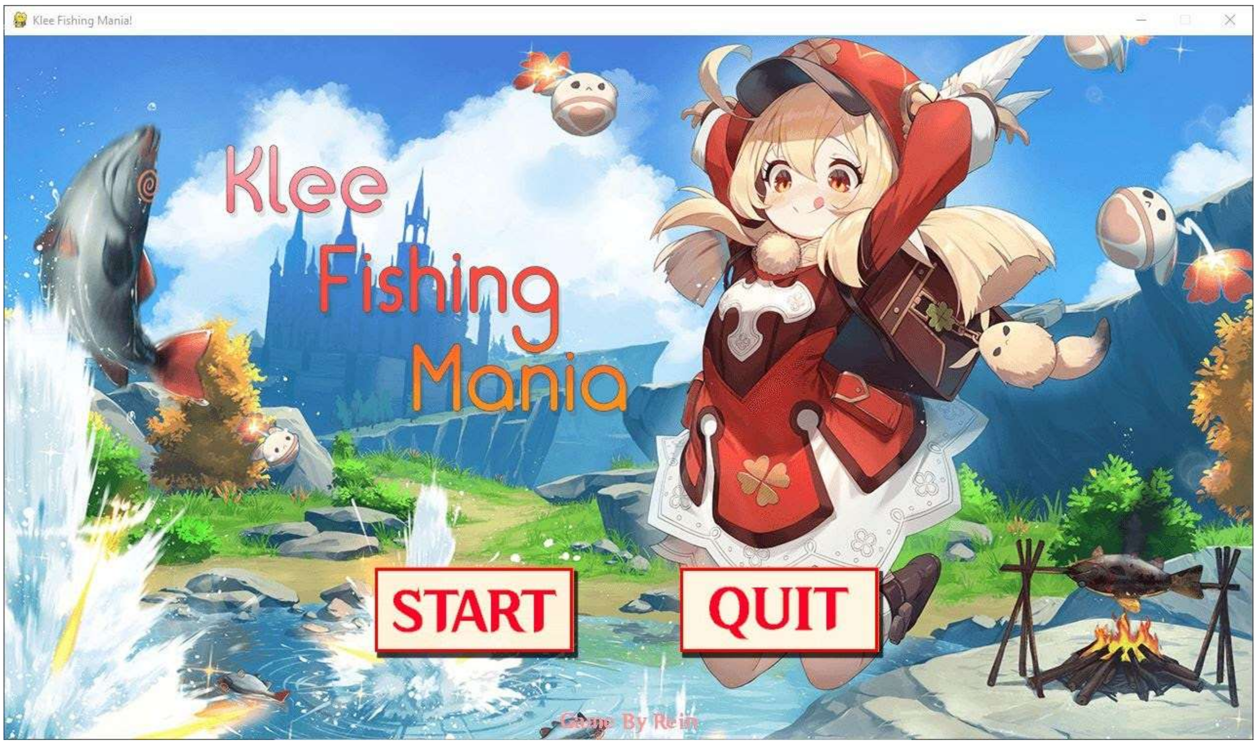
kb.py

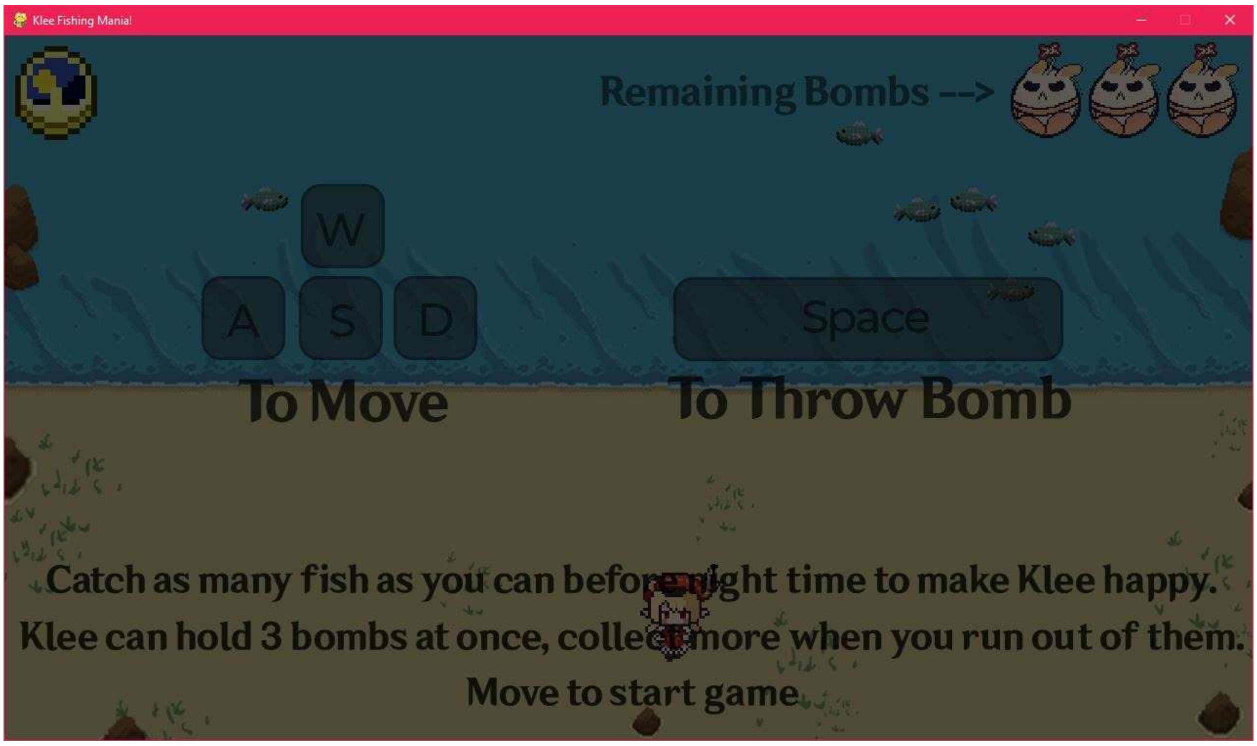


```python
import pygame
import random

def klee_movement(keys_pressed, klee_pos, KLEE_MOVE_SPEED, BORDER_KLEE, WIDTH, HEIGHT): #klee movment
    if keys_pressed[pygame.K_a] and klee_pos.x - KLEE_MOVE_SPEED > 0: #move left
        klee_pos.x -= KLEE_MOVE_SPEED
    if keys_pressed[pygame.K_d] and klee_pos.x + KLEE_MOVE_SPEED + klee_pos.width < WIDTH: #move right
        klee_pos.x += KLEE_MOVE_SPEED
    if keys_pressed[pygame.K_w] and klee_pos.y - KLEE_MOVE_SPEED > BORDER_KLEE.y: #move up
        klee_pos.y -= KLEE_MOVE_SPEED
    if keys_pressed[pygame.K_s] and klee_pos.y + KLEE_MOVE_SPEED + klee_pos.height < HEIGHT: #move down
        klee_pos.y += KLEE_MOVE_SPEED

def bombs_movement(bombs, klee_pos, fishes1, fish_caught, explosion, explosion_pos, fishes2, BOMB_SPEED): #bomb movenet and interactions
    for bomb in bombs:
        bomb.y -= BOMB_SPEED
        for fish in fishes1:
            if fish.colliderect(bomb):
                explosion_pos[0] = bomb
                fishes1.remove(fish)
                fish_caught[0] += 1
                bombs.remove(bomb)
                explosion[0] = 60
        for fish2 in fishes2:
            if fish2.colliderect(bomb):
                explosion_pos[0] = bomb
                fishes2.remove(fish2)
                fish_caught[0] += 1
                for bomb2 in bombs:
                    if bomb2 == bomb:
                        bombs.remove(bomb)
                explosion[0] = 60


        if bomb.y < 0:
            bombs.remove(bomb)

def refil_bombs(BOMBS_LEFT, klee_pos, bomb_fill, bombs, BOMB_SPEED, MAX_BOMBS):
    for bomb_fill_pos in bomb_fill:
        if klee_pos.colliderect(bomb_fill_pos):
            BOMBS_LEFT[0] = MAX_BOMBS
            bomb_fill.remove(bomb_fill_pos)
            bombs.clear()

def bomb_fill_spawn(bomb_fill, bombs, BOMBS_LEFT, klee_pos):
    if len(bomb_fill) == 0 and (len(bombs) == 0 and BOMBS_LEFT[0] == 0): #spawn bomb refills
                bomb_fill_pos = pygame.Rect(random.randint(50, 1230), random.randint(410, 670), 50, 50)
                while bomb_fill_pos.colliderect(klee_pos): #make sure players dont get free reload
                    bomb_fill_pos = pygame.Rect(random.randint(50, 1230), random.randint(410, 670), 50, 50)
                bomb_fill.append(bomb_fill_pos)
```

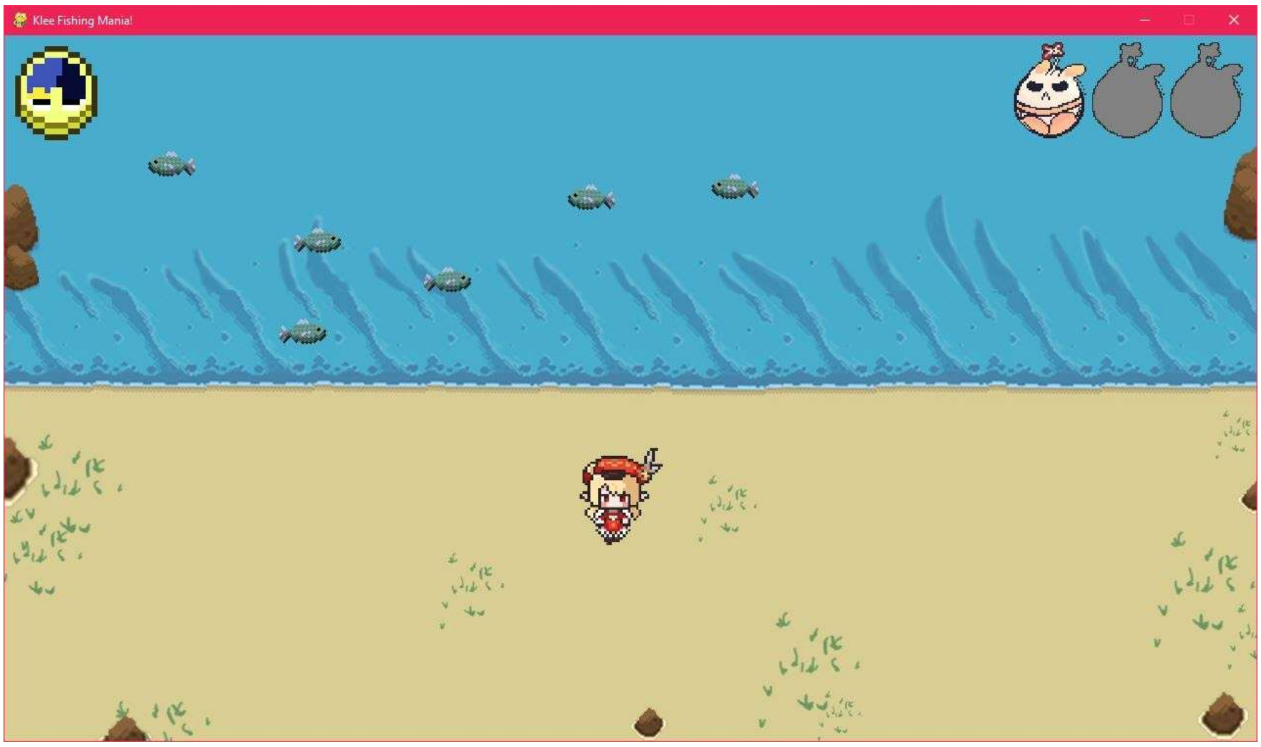
A submodule which uses the random and pygame module. It contains 4 functions. klee_movement() is for the character movement and bomb_movement() is for the movement of the bomb. refil_bombs() and bomb_fill_spawn() are for checking if Klee has any more bombs and for spawning the “Bomb Cache” when she runs out of bombs. klee_movement() uses the variable keys_pressed from the main() function in runme.py to change the variable of klee_pos which is used to determine the position of Klee. bomb_movement() moves the bombs fired with the speed BOMB_SPEED and removes both the bomb and fish when they collide. A part was also added for when the bomb collides with 2 fish at once. When the bomb hits the upper border of the window it is also removed. refil_bombs() checks if Klee has any more bombs and clears the bombs array if she has fired 3 regardless if they hit or missed, this part was added as I encountered bugs where when 2 bombs hit the border at once, only 1 was removed from the array. bomb_fill_spawn() spawns a “Bomb Cache” when Klee has fired 3 bombs and the bombs array is empty. It spawns a bomb in a random location in the beach and makes sure it doesn’t spawn directly on Klee so players dont get a free reload.

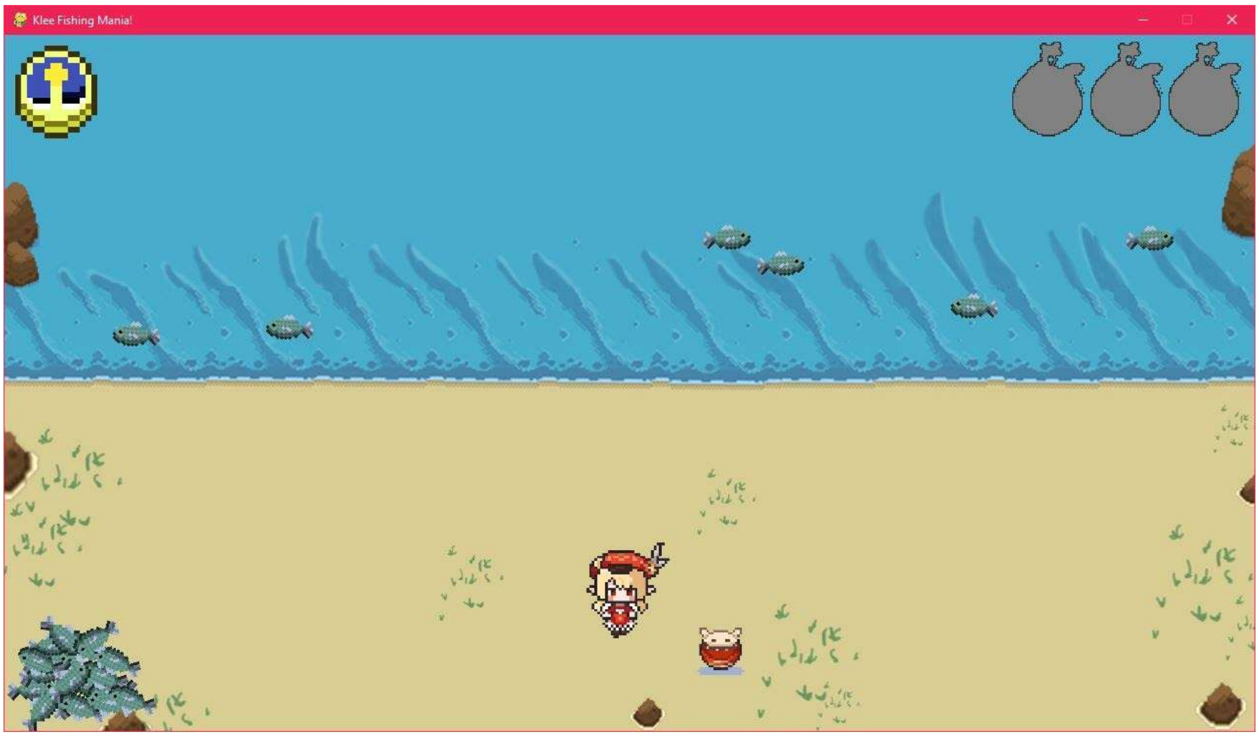
Proof Of Working Program

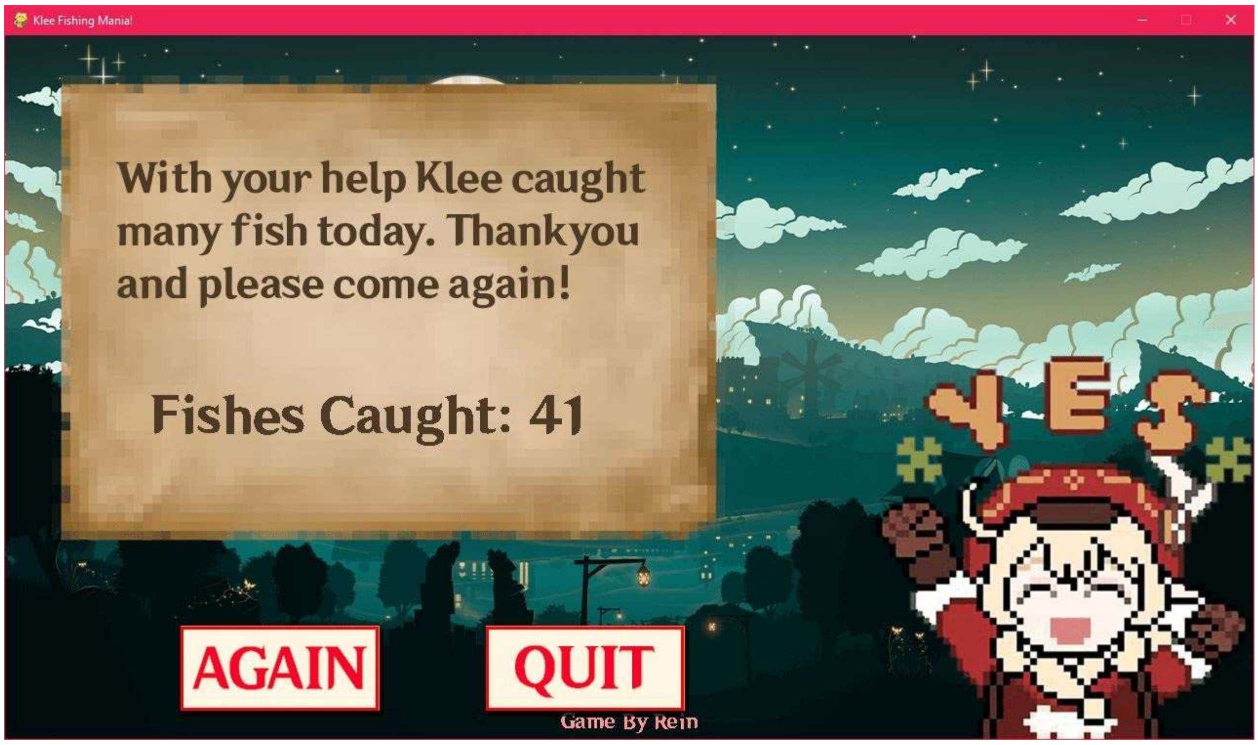




Catch as many fish as you can before night time to make Klee happy.
Klee can hold 3 bombs at once, collect more when you run out of them.
Move to start game.


Klee Fishing Mania!


Klee Fishing Mania!



With your help Klee caught many fish today. Thankyou and please come again!

Fishes Caught: 41

Reflection and Experience

I really enjoyed the experience of making this game as this was the first time I was making a game in python. I learned how to use the pygame and os library, as this was the first time I used it. At first it was easy to figure out how to do the basic things but as I tested the game further I ran into alot of bugs, for example when the fish would hit the border of the window at the same time only 1 would be removed from the array and the other one would continue to move off the window, or when I first made the class for the buttons they were not made so that when you released your mouse it reset so I couldn't press the start button, or when the bombs array was not cleared properly when 2 bombs left the screen at the same time so the "Bomb Cache" wouldn't spawn.
Overall, I was able to gain a lot of experience from making this game that will help me with making my next game. In the end I was able to produce a game that I am proud of.

References

https://www.pygame.org/
https://docs.python.org/3/library/os.html
All the assets were taken off google images or Genshin Impact game files or made by me.