

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 beginning 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

```
1  from pickle import TRUE
2  import pygame
3  import os
4  import random
5  from lib import fish
6  from lib import kb
7
8  WIDTH, HEIGHT = 1280, 720
9  WIN = pygame.display.set_mode((WIDTH, HEIGHT)) #set window size
10 pygame.display.set_caption("Klee Fishing Mania!") #set window name
11
12 FPS = 60 #fps variable
13 KLEE_MOVE_SPEED = 4 #move speed pixle/s
14 BOMB_SPEED = 5 #move speed of bomb
15 MAX_BOMBS = 3 #max number of bombs
16 MIN_FISH_SPEED = 3 #starting fish speed
17 FISH_SPEED = [MIN_FISH_SPEED]
18 BORDER_KLEE = pygame.Rect(0, HEIGHT//2 - 5, WIDTH, 10) #movement border
19 MAX_FISH = 3 #Max no of fish at one time
20 TIMELIMIT = 120000 #time limit in ticks, 1s = 1000tick
21
22 FISH_HIT = pygame.USEREVENT + 1 #user events
23 RESTART_GAME = pygame.USEREVENT + 1
24
25 pygame.font.init()
26 font1 = pygame.font.Font(os.path.join('assets', 'fonts', 'zh-cn.ttf'), 50) #load font
27 font2 = pygame.font.Font(os.path.join('assets', 'fonts', 'zh-cn.ttf'), 20)
28
29 KLEE_IMAGE = pygame.image.load(os.path.join('assets', 'klee.png')) #Import & resize Images
30 KLEE = pygame.transform.scale(KLEE_IMAGE, (100, 100))
31 BOMBREFIL_IMAGE = pygame.image.load(os.path.join('assets', 'bomb.png'))
32 BOMBREFIL = pygame.transform.scale(BOMBREFIL_IMAGE, (50, 50))
33 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.

```

73 class button(): #class for buttons
74     def __init__(self, x, y, image):
75         self.image = image
76         self.rect = self.image.get_rect()
77         self.rect.topleft = (x, y)
78         self.clicked = False
79
80     def draw(self): #draw and click detection mouse
81         action = False
82         pos = pygame.mouse.get_pos()
83         if self.rect.collidepoint(pos):
84             if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:
85                 self.clicked = True
86                 action = True
87
88             if pygame.mouse.get_pressed()[0] == 0:
89                 self.clicked = False
90
91         WIN.blit(self.image, (self.rect.x, self.rect.y))
92         return action
93
94 start_button = button(377, 540, STARTBUTTON) #button instances
95 quit_button = button(690, 540, QUITBUTTON)
96 again_button = button(177, 600, AGAINBUTTON)
97 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.

```

99 def start_screen():
100     sub = font2.render("Game By Rein", False, (255, 150, 150))
101     starting = True
102     started = False
103     clock = pygame.time.Clock()
104     pygame.display.update()
105     while starting:
106         WIN.blit(STARTSCREEN, (0,0)) #display Start screen text & buttons
107         WIN.blit(sub, (570, 690))
108         if start_button.draw():
109             starting = False
110             started = True
111         if quit_button.draw():
112             pygame.quit()
113         clock.tick(60)
114         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.

```

116 ~ 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
117     WIN.fill((255,255,255))
118     WIN.blit(BACKGROUND_BEACH, (0, 0)) #disp background and fish stack
119 ~     if fish_caught[0] > 10:
120         WIN.blit(FISHSTACK, (-50, 500))
121 ~     if fish_caught[0] > 20:
122         WIN.blit(FISHSTACK, (0, 500))
123 ~     if fish_caught[0] > 30:
124         WIN.blit(FISHSTACK, (50, 500))
125 ~     if fish_caught[0] > 40:
126         WIN.blit(FISHSTACK, (100, 500))
127 ~     if fish_caught[0] > 50:
128         WIN.blit(FISHSTACK, (150, 500))
129 ~     if fish_caught[0] > 60:
130         WIN.blit(FISHSTACK, (200, 500))
131 ~     if fish_caught[0] > 70:
132         WIN.blit(FISHSTACK, (250, 500))
133 ~     if fish_caught[0] > 80:
134         WIN.blit(FISHSTACK, (300, 500))
135     WIN.blit(FISH, (fish_posi.x, fish_posi.y))
136 ~     for bomb in bombs: #disp bomb & fishes
137         WIN.blit(SMALLBOMB, (bomb.x, bomb.y))
138 ~     for fish in fishes1:
139         WIN.blit(FISH, (fish.x, fish.y))
140 ~     for fish2 in fishes2:
141         WIN.blit(FISH2, (fish2.x, fish2.y))
142 ~     for bomb_fill_pos in bomb_fill:
143         WIN.blit(BOMBREFIL, (bomb_fill_pos.x, bomb_fill_pos.y))
144     WIN.blit(KLEE, (klee_pos.x, klee_pos.y)) #disp klee
145     WIN.blit(BOMBBAR_EMPTY, (1180,5)) #disp empty bomb bar
146     WIN.blit(BOMBBAR_EMPTY, (1100,5))
147     WIN.blit(BOMBBAR_EMPTY, (1020,5))
148 ~     if explosion[0] > 0: #explosion
149         explosion[0] -= 1
150         exploc = explosion_pos[0]
151         WIN.blit(EXPLOSION, (exploc.x - 35, exploc.y - 35))
152 ~     if BOMBS_LEFT[0] == 3: #bomb counter
153         WIN.blit(BOMBBAR, (1180,5))
154         WIN.blit(BOMBBAR, (1100,5))
155         WIN.blit(BOMBBAR, (1020,5))
156 ~     elif BOMBS_LEFT[0] == 2:
157         WIN.blit(BOMBBAR, (1100,5))
158         WIN.blit(BOMBBAR, (1020,5))
159 ~     elif BOMBS_LEFT[0] == 1:
160         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.

```

178 def endscreen(fish_caught):
179     sub = font2.render("Game By Rein", False, (255, 150, 150))
180     clock = pygame.time.Clock()
181     fishc = "Fishes Caught: " + str(fish_caught[0])
182     caught = font1.render(fishc, False, (76, 58, 38))
183     ended = True
184     pygame.display.update()
185     while ended:
186         WIN.blit(BACKGROUND_END, (0,0))
187         WIN.blit(caught, (150, 360))
188         WIN.blit(sub, (570, 690))
189         if again_button.draw():
190             main()
191         if quit_button2.draw():
192             pygame.quit()
193         clock.tick(60)
194     return ended

```

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


```

196 def main():
197     klee_pos = pygame.Rect(640, 540, 100, 100) #klee position starting
198     klee_pos1 = pygame.Rect(640, 540, 100, 100)
199
200     bombs = []
201     bomb_fill = []
202     BOMBS_LEFT = [MAX_BOMBS]
203     fishes1 = []
204     fishes2 = []
205     fish_caught = [0]
206     explosion = [0]
207     explosion_pos = [0]
208     fish_last = [0]
209     FISH_SPEED = [MIN_FISH_SPEED]
210     time_started = 0
211     clock = pygame.time.Clock()
212     run = True
213     started1 = False
214     started2 = False
215     controls = False
216     show_controls = True
217
218     pygame.mixer.init()
219     pygame.mixer.music.load(os.path.join('assets', 'music', 'bgm1.mp3'))
220     pygame.mixer.music.play(loops=100)
221
222     while run: #Loop to check if user quit
223         if started1 == False:
224             if start_screen():
225                 started1 = True
226                 started2 = True
227         if started2 == False and started1 == True:
228             endscreen(fish_caught)
229             clock.tick(FPS) #set max fps
230             for event in pygame.event.get():
231                 if event.type == pygame.QUIT:
232                     run = False
233                 if event.type == pygame.KEYDOWN and controls == True:
234                     if event.key == pygame.K_SPACE and len(bombs) < MAX_BOMBS and int(BOMBS_LEFT[0]) > 0: #detect key press & check 1
235                         bomb = pygame.Rect(klee_pos.x + klee_pos.width - 50, klee_pos.y + klee_pos.height//2 - 5, 20, 20)
236                         bombs.append(bomb)
237                         BOMBS_LEFT[0] -= 1
238                         controls = True
239
240             if controls == False: #show controls screen
241                 if klee_pos != klee_pos1:
242                     show_controls = False
243                     controls = True
244                     time_started = pygame.time.get_ticks()
245
246             if started2 == True:
247                 if controls == True: #fixed bug where staying in controls screen long enough crashes game
248                     if (pygame.time.get_ticks() - time_started) > TIMELIMIT:
249                         started2 = False
250
251             keys_pressed = pygame.key.get_pressed() #detect key press
252
253             kb.klee_movement(keys_pressed, klee_pos, KLEE_MOVE_SPEED, BORDER_KLEE, WIDTH, HEIGHT) #klee movement
254             kb.bombs_movement(bombs, klee_pos, fishes1, fish_caught, explosion, explosion_pos, fishes2, BOMB_SPEED) #bomb movement
255             kb.refil_bombs(BOMBS_LEFT, klee_pos, bomb_fill, bombs, BOMB_SPEED, MAX_BOMBS) #bomb refil
256             kb.bomb_fill_spawn(bomb_fill, bombs, BOMBS_LEFT, klee_pos) #bomb refil spawn
257             fish.fish_spawn(fishes1, fishes2, MAX_FISH) #spawns fish
258             fish.fish_swim_left(fishes1, FISH_SPEED) #make fish swim
259             fish.fish_swim_right(fishes2, FISH_SPEED)
260             fish.fish_dissapear_prevention(fishes1) #fixed bug where fish were disappearing
261             fish.fish_dissapear_prevention2(fishes2)
262             fish.fish_collison_prevention(fishes1, fishes2) #fix bug where collision of fish crashes game
263             fish.fish_speed_up(fish_caught, FISH_SPEED, fish_last) #increasing difficulty
264             print(pygame.time.get_ticks() - time_started)
265
266             draw_window_game(klee_pos, bombs, fishes1, bomb_fill, fish_caught, BOMBS_LEFT, explosion, explosion_pos, fishes2, sho
267
268     pygame.quit()
269
270
271
272 if __name__ == "__main__": #only run when file is run, dont run when import
273     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

```
1  import random
2  import pygame
3
4  def fish_swim_left(fishes1, FISH_SPEED): #fish swim left
5      for fish in fishes1:
6          fish.x -= FISH_SPEED[0]
7          if fish.x == 0:
8              fish.x = 1280
9
10 def fish_swim_right(fishes2, FISH_SPEED): #fish swim right
11     for fish in fishes2:
12         fish.x += FISH_SPEED[0]
13         if fish.x == 1280:
14             fish.x = 0
15
16 def fish_dissapear_prevention(fishes1):
17     for fish in fishes1:
18         if fish.x < 0:
19             fishes1.remove(fish)
20
21 def fish_dissapear_prevention2(fishes2):
22     for fish2 in fishes2:
23         if fish2.x > 1280:
24             fishes2.remove(fish2)
25
26 def fish_collison_prevention(fishes1, fishes2):#fix bug where collision of fish crashes game
27     for fish1 in fishes1:
28         for fish2 in fishes2:
29             if fish1.colliderect(fish2):
30                 roll = random.randint(1,2)
31                 if roll == 1:
32                     fishes1.remove(fish1)
33                 else:
34                     fishes2.remove(fish2)
35
36 def fish_speed_up(fish_caught, FISH_SPEED, fish_last):
37     if (fish_caught[0] - fish_last[0]) == 10:
38         FISH_SPEED[0] += 1
39         fish_last[0] = fish_caught[0]
40
41 def fish_spawn(fishes1, fishes2, MAX_FISH):
42     if len(fishes1) < MAX_FISH:
43         fish_pos1 = pygame.Rect(random.randint(961, 1229), random.randint(50, 310), 50, 50) #fish starting position
44         for collide in fishes1:
45             if collide.colliderect(fish_pos1): #fixed bug where when 2 fish killed at once game crashed
46                 fishes1.remove(collide)
47         fishes1.append(fish_pos1)
48     if len(fishes2) < MAX_FISH:
49         fish_posi2 = pygame.Rect(random.randint(51, 319), random.randint(50, 310), 50, 50) #fish starting position
50         for collide in fishes2:
51             if collide.colliderect(fish_posi2): #fixed bug where when 2 fish killed at once game crashed
52                 fishes2.remove(collide)
53         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_collison_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_collison_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

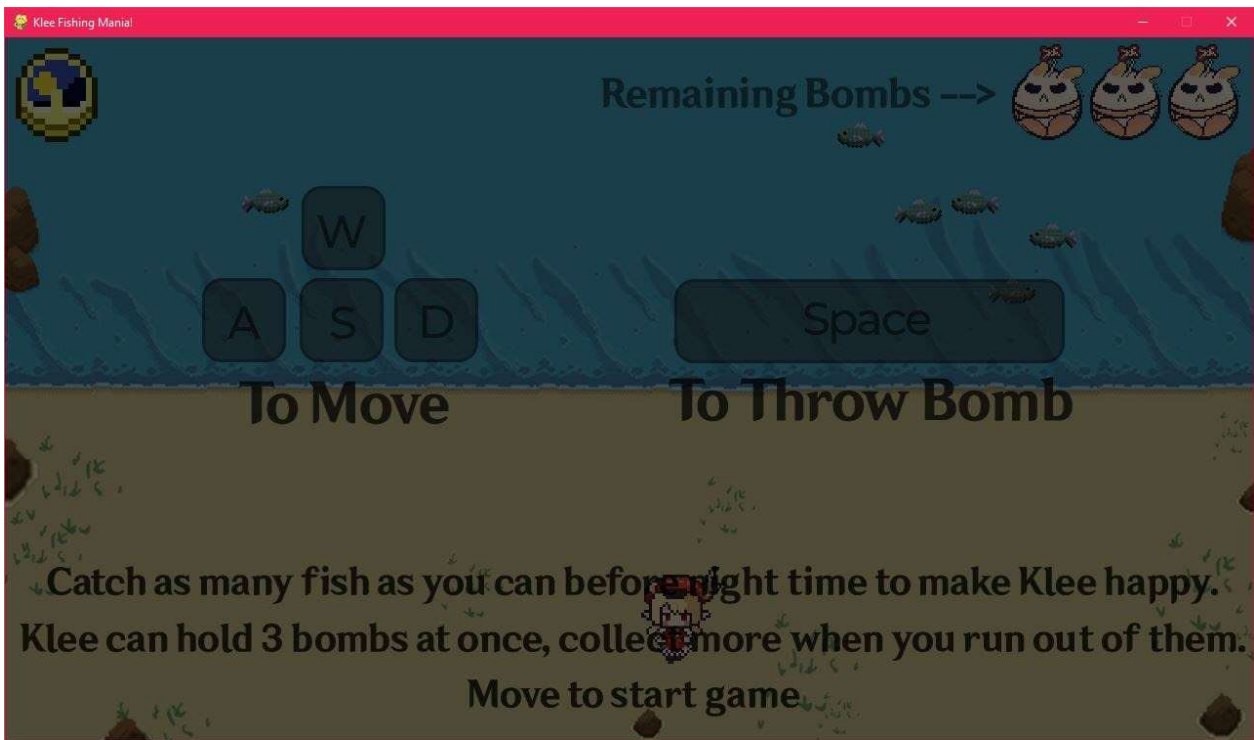
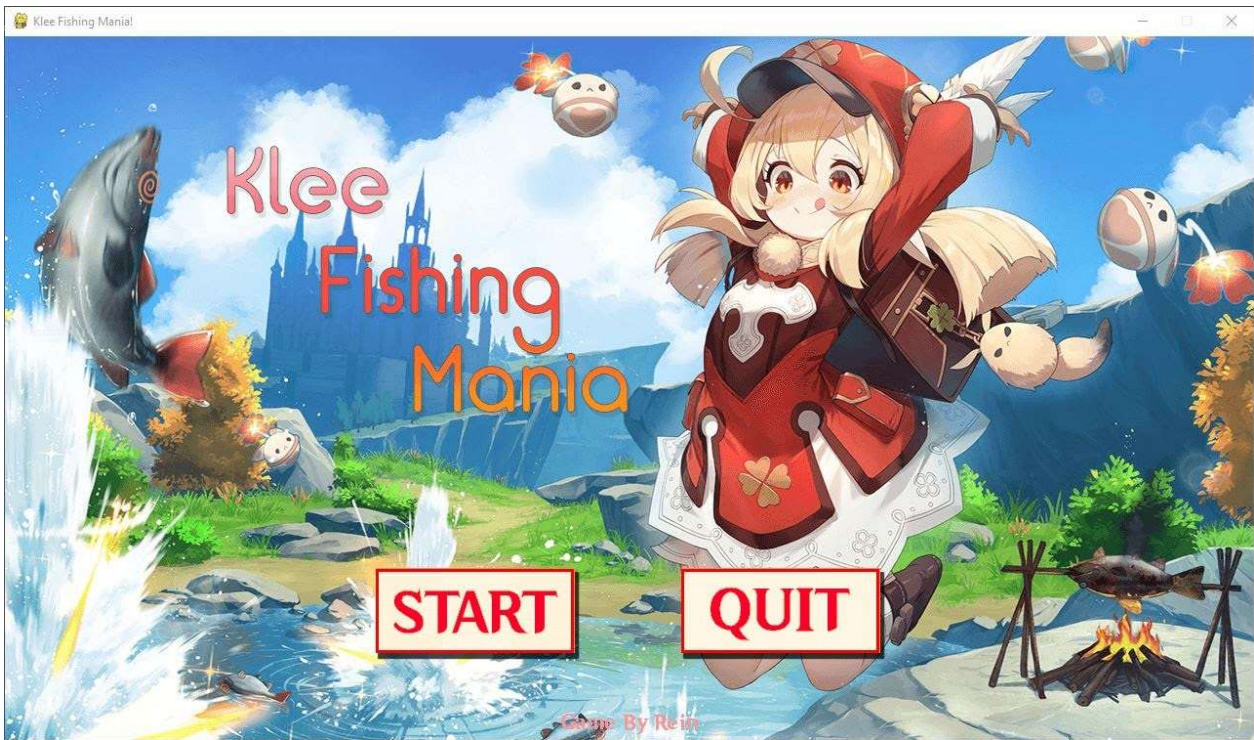
on the left and right area of the screen depending on which direction they will swim. It also has a part built in to prevent fish spawning together.

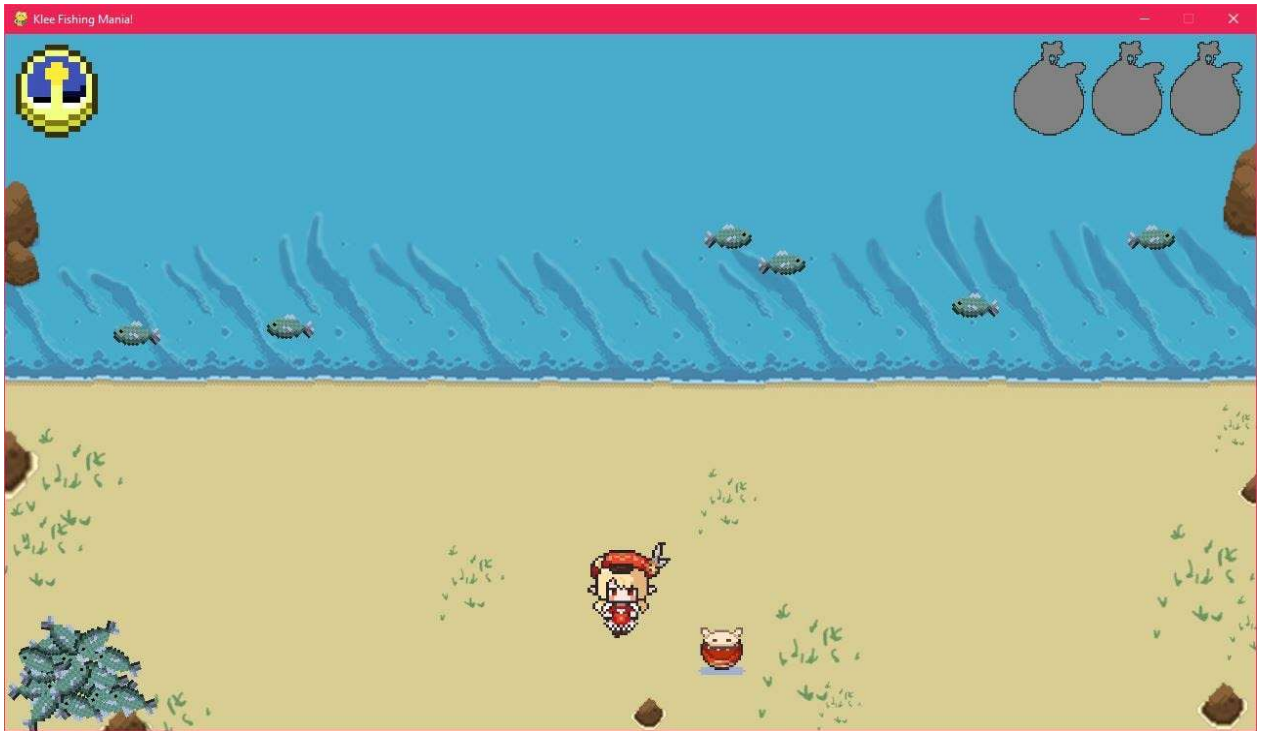
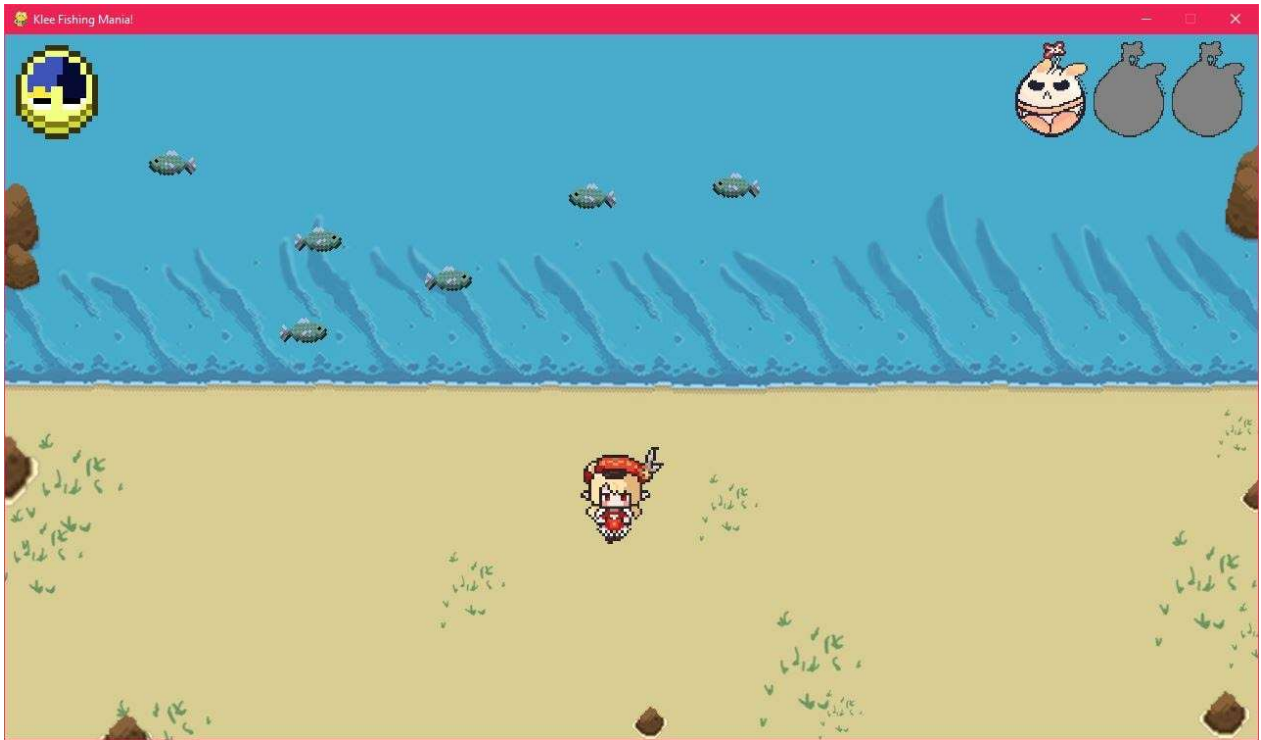
kb.py

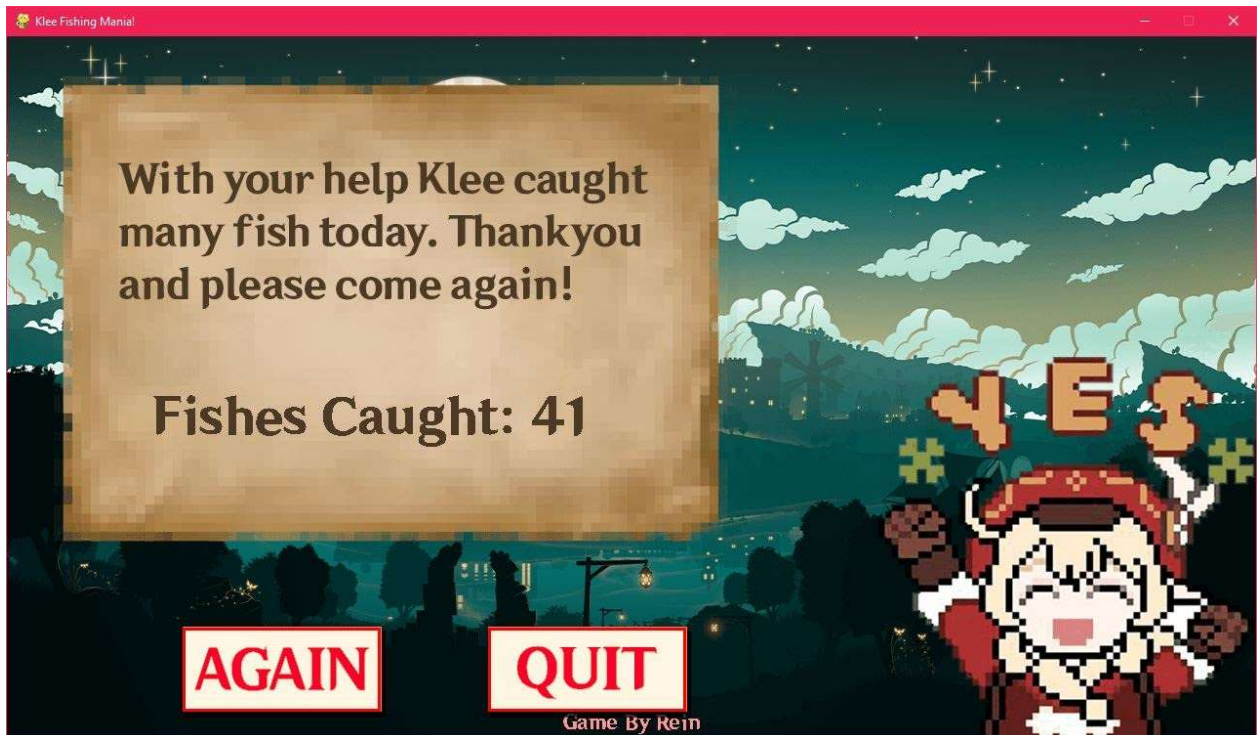
```
1 import pygame
2 import random
3
4 def klee_movement(keys_pressed, klee_pos, KLEE_MOVE_SPEED, BORDER_KLEE, WIDTH, HEIGHT): #klee movement
5     if keys_pressed[pygame.K_a] and klee_pos.x - KLEE_MOVE_SPEED > 0: #move left
6         klee_pos.x -= KLEE_MOVE_SPEED
7     if keys_pressed[pygame.K_d] and klee_pos.x + KLEE_MOVE_SPEED + klee_pos.width < WIDTH: #move right
8         klee_pos.x += KLEE_MOVE_SPEED
9     if keys_pressed[pygame.K_w] and klee_pos.y - KLEE_MOVE_SPEED > BORDER_KLEE.y: #move up
10        klee_pos.y -= KLEE_MOVE_SPEED
11    if keys_pressed[pygame.K_s] and klee_pos.y + KLEE_MOVE_SPEED + klee_pos.height < HEIGHT: #move down
12        klee_pos.y += KLEE_MOVE_SPEED
13
14 def bombs_movement(bombs, klee_pos, fishes1, fish_caught, explosion, explosion_pos, fishes2, BOMB_SPEED): #bomb movement and interactions
15     for bomb in bombs:
16         bomb.y -= BOMB_SPEED
17         for fish in fishes1:
18             if fish.colliderect(bomb):
19                 explosion_pos[0] = bomb
20                 fishes1.remove(fish)
21                 fish_caught[0] += 1
22                 bombs.remove(bomb)
23                 explosion[0] = 60
24         for fish2 in fishes2:
25             if fish2.colliderect(bomb):
26                 explosion_pos[0] = bomb
27                 fishes2.remove(fish2)
28                 fish_caught[0] += 1
29         for bomb2 in bombs:
30             if bomb2 == bomb:
31                 bombs.remove(bomb)
32                 explosion[0] = 60
33
34     if bomb.y < 0:
35         bombs.remove(bomb)
36
37
38 def refill_bombs(BOMBS_LEFT, klee_pos, bomb_fill, bombs, BOMB_SPEED, MAX_BOMBS):
39     for bomb_fill_pos in bomb_fill:
40         if klee_pos.colliderect(bomb_fill_pos):
41             BOMBS_LEFT[0] = MAX_BOMBS
42             bomb_fill.remove(bomb_fill_pos)
43             bombs.clear()
44
45 def bomb_fill_spawn(bomb_fill, bombs, BOMBS_LEFT, klee_pos):
46     if len(bomb_fill) == 0 and (len(bombs) == 0 and BOMBS_LEFT[0] == 0): #spawn bomb refills
47         bomb_fill_pos = pygame.Rect(random.randint(50, 1230), random.randint(410, 670), 50, 50)
48         while bomb_fill_pos.colliderect(klee_pos): #make sure players dont get free reload
49             bomb_fill_pos = pygame.Rect(random.randint(50, 1230), random.randint(410, 670), 50, 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. refill_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. refill_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







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.