Name: David Bijak Santoso

Class: L1BC

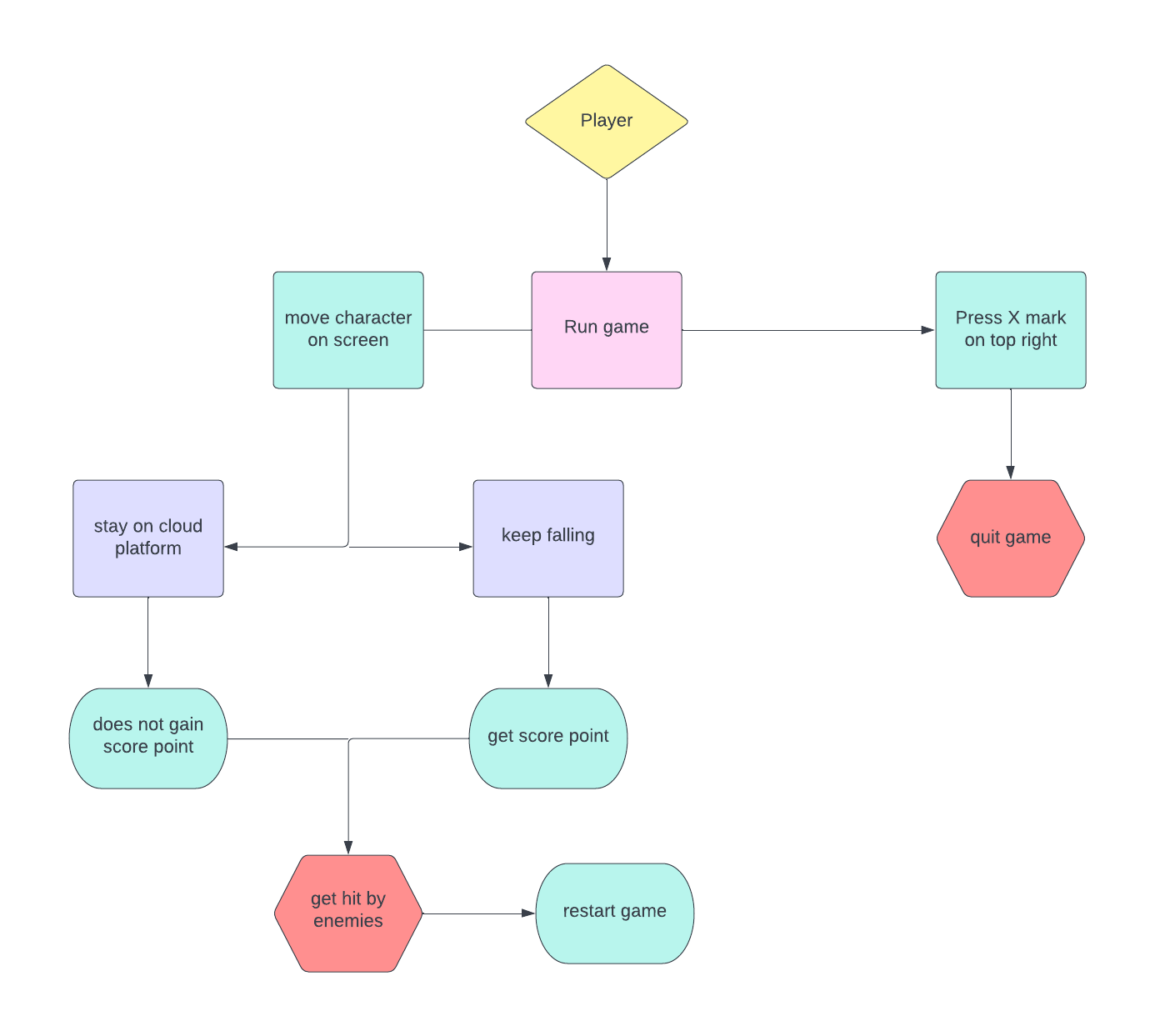
Course: Algorithm and Programming

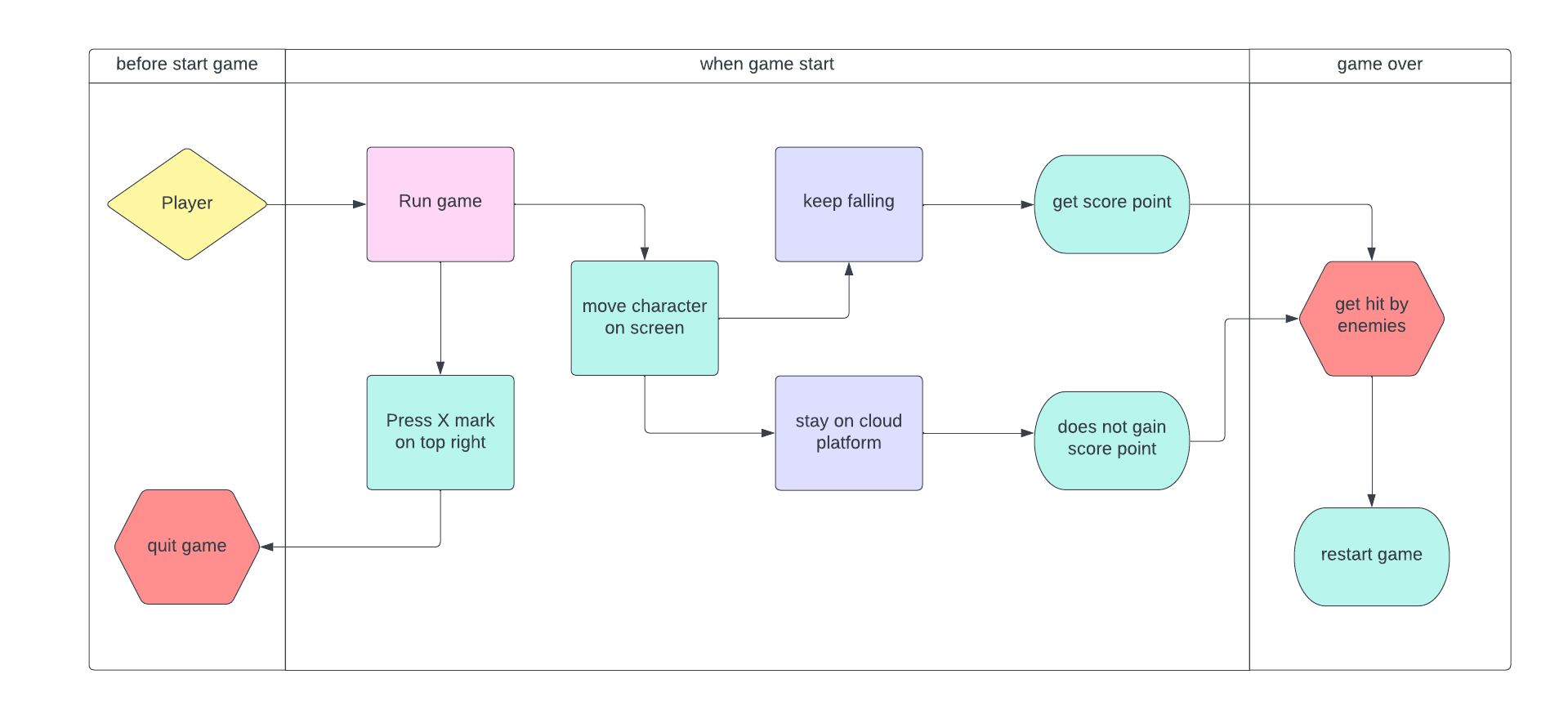
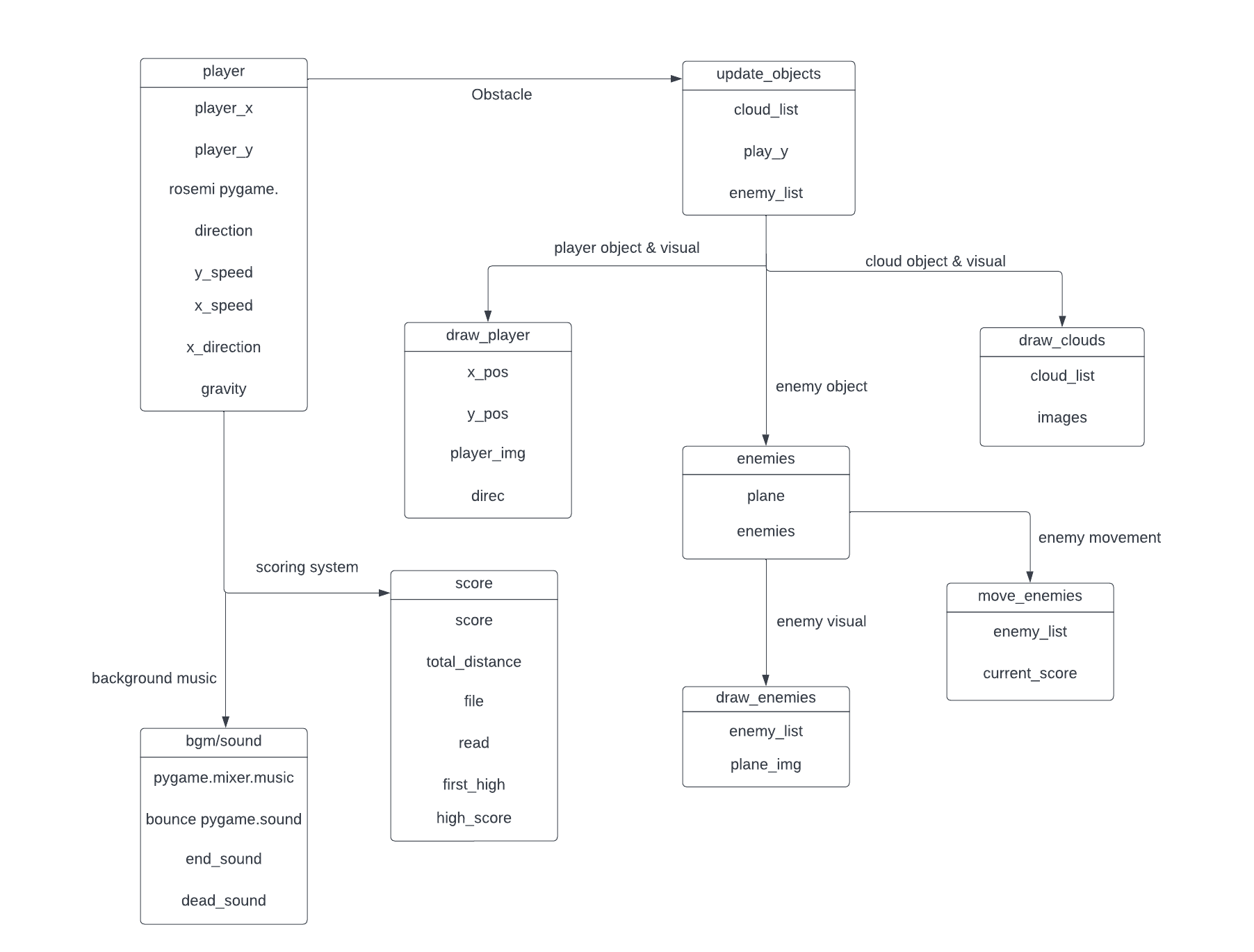
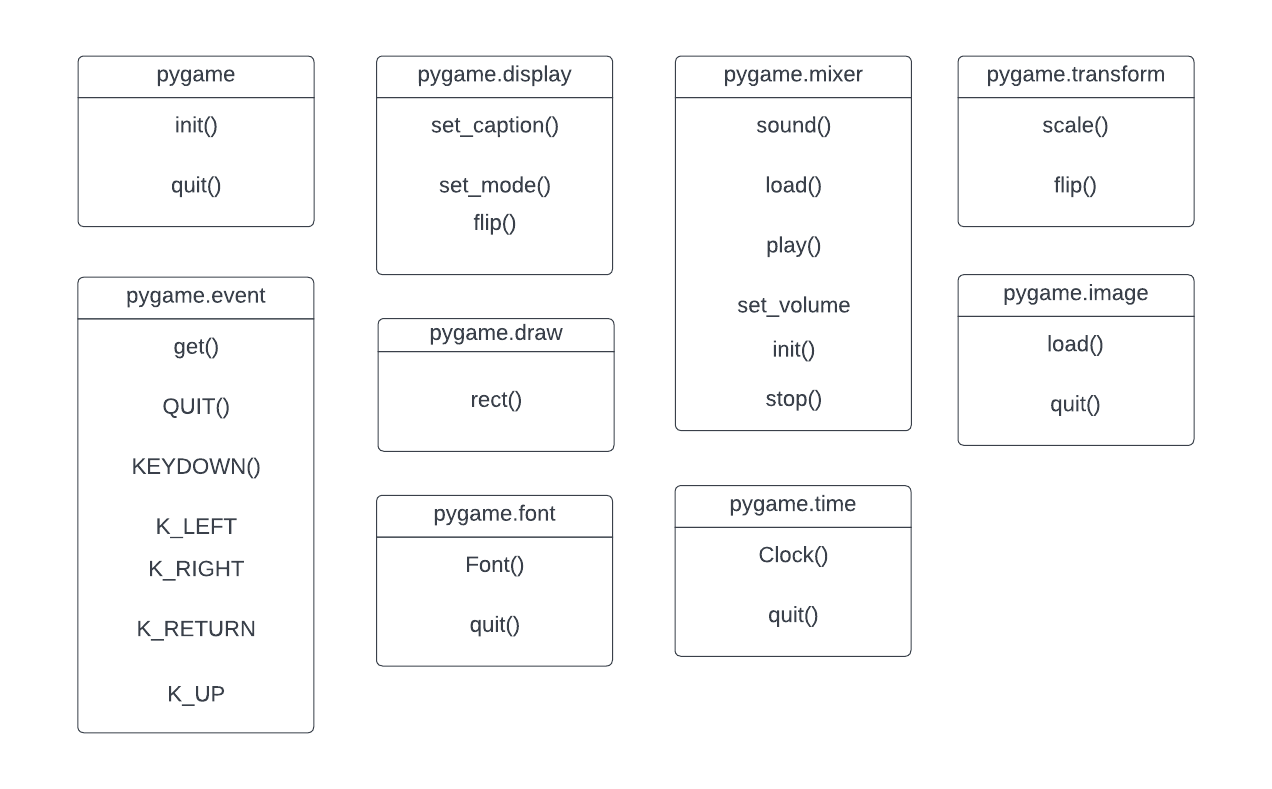
**Project Final Game**

1. **Project Documentation**
2. Brief Description

This project was made for Algorithm and Programming course final semester result. The project is to make a functional game using only python programming language. The game was based on a tutorial video and the concept was about free falling and dodging enemies. The title is called “Rose Fall!” and the game is based on a Nijisanji character Rosemi Lovelock. Game was designed for a fan made vtuber style video game that is simple and easy to understand on the objective. It uses various assets that suits the vibe for the gameplay, for example pink colour for the background and clouds. Also, the player controls chibi pudding shape Rosemi to add the fun for the gameplay. The game mechanic is simple, players fall and dodges enemies. There are cloud platforms to prevent getting hit by enemies and players may uses it in various ways depends on their gameplay.

1. Use-case Diagram



1. Activity Diagram
2. Class Diagram
3. Modules
4. Essential Algorithms

update\_objects = This is a function to create an infinite loop that can display in the gameplay for the core mechanics.

def update\_objects(cloud\_list, play\_y, enemy\_list):🡨variables to use

    lowest\_cloud = 0 🡨lowest cloud that could randomly spawn in display

    update\_speed = 10 🡨speed to increase difficulty and cloud random generate

    if play\_y > 200: 🡨vertical position to generate display if speed is high

        play\_y -= update\_speed

        for q in range(len(enemy\_list)): 🡨enemy display

            enemy\_list[q][1] -= update\_speed

        for j in range(len(cloud\_list)): 🡨cloud display

            cloud\_list[j][1] -= update\_speed

            if cloud\_list[j][1] > lowest\_cloud: 🡨for lowest cloud

                lowest\_cloud = cloud\_list[j][1]

        if lowest\_cloud < 750: 🡨for generating clouds

            num\_clouds = random.randint(1,2)

            if num\_clouds == 1:

                x\_pos = random.randint(0, WIDTH - 70) 🡨x position for cloud

                y\_pos = random.randint(HEIGHT + 100, HEIGHT + 300)🡨y position

                cloud\_type = random.randint (1, 3) 🡨type of cloud assets

                cloud\_list.append([x\_pos, y\_pos, cloud\_type]) 🡨display clouds

            else:

                x\_pos = random.randint(0, WIDTH/2 - 70) 🡨x position for cloud

                y\_pos = random.randint(HEIGHT + 100, HEIGHT + 300)

                cloud\_type = random.randint (1, 3)🡨type of cloud assets

                x\_pos2 = random.randint(WIDTH/2 + 70, WIDTH - 70)🡨x position

                y\_pos2 = random.randint(HEIGHT + 100, HEIGHT + 300) 🡨for y

                cloud\_type2 = random.randint (1, 3) 🡨type of cloud assets

                cloud\_list.append([x\_pos, y\_pos, cloud\_type]) 🡨display clouds

                cloud\_list.append([x\_pos2, y\_pos2, cloud\_type2])🡨display ^^

    return play\_y, cloud\_list, enemy\_list 🡨return list variable

draw\_clouds = This is a function to create clouds on display by using assets and use as platform on gameplay player’s perspective.

def draw\_clouds(cloud\_list, images): 🡨variables to use

    platforms = [] <--to return cloud platforms and make collusion

    for j in range(len(cloud\_list)):

        image = images[cloud\_list[j][2] - 1] 🡨assets to use for cloud

        platform = pygame.rect.Rect((cloud\_list[j][0] + 5, cloud\_list[j][1] + 70), (100, 10)) 🡨hitmarker for clouds

        screen.blit(image, (cloud\_list[j][0], cloud\_list[j][1])) 🡨display clouds on screen

        pygame.draw.rect(screen, 'pink', [cloud\_list[j][0] + 5, cloud\_list[j][1] + 70, 100, 3]) 🡨use only for displaying hitmarker

        platforms.append(platform) 🡨to show it/visible on screen

    return platforms 🡨return variable

draw\_player = This is a function to create player on display by using assets and to control the character for the player to move.

def draw\_player(x\_pos, y\_pos, player\_img, direc): 🡨variables to use

    if direc == -1: 🡨if player bounce up character flip body 180

        player\_img = pygame.transform.flip(player\_img, False, True) 🡨flip

    screen.blit(player\_img, (x\_pos, y\_pos)) 🡨display player on screen

    player\_rect = pygame.rect.Rect((x\_pos + 7, y\_pos + 32), (36, 10))🡨hitmark

    #pygame.draw.rect(screen, 'green', player\_rect, 3)🡨use to display hitmark

    return player\_rect 🡨return variable

draw\_enemies = This is a function to create enemies (plane) on display by using assets and to use as an obstacle for player to avoid.

def draw\_enemies(enemy\_list, plane\_img): 🡨variables to use

    enemy\_rects = [] 🡨to make enemy hitmark

    for j in range(len(enemy\_list)):

        enemy\_rect = pygame.rect.Rect((enemy\_list[j][0] - 1, enemy\_list[j][1] + 80), (300, 80)) 🡨hitmarker for enemy

        #pygame.draw.rect(screen, 'orange', enemy\_rect, 3)🡨to display hitmark

        enemy\_rects.append(enemy\_rect) 🡨to show hitmark visible

        if enemy\_list[j][2] == 1: 🡨display shark on right position

            screen.blit(plane\_img, (enemy\_list[j][0], enemy\_list[j][1]))

        elif enemy\_list[j][2] == -1: 🡨display shark on left position

            screen.blit(pygame.transform.flip(plane\_img, 1, 0), (enemy\_list[j][0], enemy\_list[j][1])) 🡨flipping enemy viewpoint

    return enemy\_rects 🡨return variable

move\_enemies = This is a function to create enem(ies)y movement on display to make gameplay immersive, fun, and difficult.

def move\_enemies(enemy\_list, current\_score): 🡨variables to use

    enemy\_speed = 2 + current\_score//15 🡨enemies become hard/difficulty

    for j in range(len(enemy\_list)): 🡨move enemy from left and right

        if enemy\_list[j][2] == 1:

            if enemy\_list[j][0] < WIDTH: 🡨code for enemy right to left

                enemy\_list[j][0] += enemy\_speed 🡨enemy speed

            else:

                enemy\_list[j][2] = -1 🡨enemy starting position

        elif enemy\_list[j][2] == -1:

            if enemy\_list[j][0] > -320: 🡨code for enemy left to right

                enemy\_list[j][0] -= enemy\_speed 🡨enemy speed

            else:

                enemy\_list[j][2] = 1 🡨enemy starting position

        if enemy\_list[j][1] < -100:

            enemy\_list[j][1] = random.randint(HEIGHT, HEIGHT + 500) 🡨area enemies spawn

    return enemy\_list 🡨return variable

1. Screenshot for Application



1. Lesson Learned/Reflection

What I have learned is that using pygame takes a lot of fail and retry. I may know the keywords, variables, and correct variables to create lines of codes, but I still haven’t try to diverse and understand fully the concept of them. I may try do experiments and testing, although I’ll wait until the second semester to start the course so I can follow the materials without forgetting them since I always do. On the other hand, I had fun in this project and it motivates me wanting to try more if the times come though.

1. **Program/Source Code**

#Rose Fall! (Rosemi-sama)

#game inspiration from https://www.youtube.com/watch?v=VhX1hicQo00&t=2349s

#game description: Game where player main task is to fall while dodging enemies and moving from platform to another

import pygame 🡨import this is a must for pygames

import random 🡨for assets

#all this for storing variables

pygame.init() 🡨random generator

pygame.mixer.init() 🡨background music

WIDTH = 500 🡨screen width size

HEIGHT = 800 🡨screen height size

fps = 60 🡨fps

timer = pygame.time.Clock() 🡨timer tell how many times it tick

huge\_font = pygame.font.Font('assets/font.ttf', 42)🡨Main title

font = pygame.font.Font('assets/font.ttf', 24) 🡨font

pygame.display.set\_caption('Rose Fall!')🡨caption title

screen = pygame.display.set\_mode([WIDTH, HEIGHT]) 🡨screen display

bg = (255, 174, 185) 🡨color background

game\_over = False 🡨game over scenario

clouds = [[200, 100, 1], [50, 330, 2], [350, 330, 3], [200, 670, 1]] 🡨clouds x,y coordinates

cloud\_images = [] 🡨cloud images

for i in range(1, 4): 🡨command to create cloud images

    img = pygame.image.load(f'assets/clouds/cloud{i}.png')

    cloud\_images.append(img)

#player variables

player\_x = 240 🡨x position

player\_y = 40 🡨y position

rosemi = pygame.transform.scale(pygame.image.load('assets/rosemilovelock.png'), (50, 50)) 🡨assets

direction = -1 🡨going down

y\_speed = 0 🡨vertical speed

gravity = 0.3 🡨gravity yes physics does exist in pygame smh

x\_speed = 3 🡨horizontal speed

x\_direction = 0 🡨direction

#score variables

score = 0 🡨score

total\_distance = 0 🡨distance for the scoring system

file = open('high\_scores.txt', 'r') 🡨to save high score

read = file.readlines() 🡨to put high score system

first\_high = int(read[0])

high\_score = first\_high

#enemies variables

plane = pygame.transform.scale(pygame.image.load('assets/plane.png'), (300, 200)) 🡨enemy assets

enemies = [[-234, random.randint(400, HEIGHT - 100), 1]] 🡨plane x y coordinates

#sounds and bgm

pygame.mixer.music.load('assets/theme.mp3') 🡨bgm

bounce = pygame.mixer.Sound('assets/bounce.mp3') 🡨bounce noise

end\_sound = pygame.mixer.Sound('assets/game\_over.mp3') 🡨end theme/game over

end\_sound.set\_volume(0.2) 🡨volume sound for game over

dead\_sound = pygame.mixer.Sound('assets/wee.mp3') 🡨death noise

pygame.mixer.music.play() 🡨play when game start

pygame.mixer.music.play(-1) 🡨for looping

pygame.mixer.music.set\_volume(0.2) 🡨volume sound for bgm

#function to create and draw clouds in display

#VVVVVVVVV

def draw\_clouds(cloud\_list, images): 🡨variables to use

    platforms = [] 🡨to return cloud platforms and make collusion

    for j in range(len(cloud\_list)):

        image = images[cloud\_list[j][2] - 1] 🡨assets to use for cloud

        platform = pygame.rect.Rect((cloud\_list[j][0] + 5, cloud\_list[j][1] + 70), (100, 10)) 🡨hitmarker for clouds

        screen.blit(image, (cloud\_list[j][0], cloud\_list[j][1])) 🡨display clouds on screen

        pygame.draw.rect(screen, 'pink', [cloud\_list[j][0] + 5, cloud\_list[j][1] + 70, 100, 3]) 🡨use only for displaying hitmarker

        platforms.append(platform) 🡨to show it/visible on screen

    return platforms 🡨return variable

#function to create and draw player in display

#VVVVVVV

def draw\_player(x\_pos, y\_pos, player\_img, direc): 🡨variables to use

    if direc == -1: 🡨if player bounce up character flip body 180

        player\_img = pygame.transform.flip(player\_img, False, True) 🡨flip

    screen.blit(player\_img, (x\_pos, y\_pos)) 🡨display player on screen

    player\_rect = pygame.rect.Rect((x\_pos + 7, y\_pos + 32), (36, 10))🡨hitmark

    #pygame.draw.rect(screen, 'green', player\_rect, 3)🡨use to display hitmark

    return player\_rect 🡨return variable

#function to create and draw enemy in display

#VVVVVVVV

def draw\_enemies(enemy\_list, plane\_img): 🡨variables to use

    enemy\_rects = [] 🡨to make enemy hitmark

    for j in range(len(enemy\_list)):

        enemy\_rect = pygame.rect.Rect((enemy\_list[j][0] - 1, enemy\_list[j][1] + 80), (300, 80)) 🡨hitmarker for enemy

        #pygame.draw.rect(screen, 'orange', enemy\_rect, 3)🡨to display hitmark

        enemy\_rects.append(enemy\_rect) 🡨to show hitmark visible

        if enemy\_list[j][2] == 1: 🡨display shark on right position

            screen.blit(plane\_img, (enemy\_list[j][0], enemy\_list[j][1]))

        elif enemy\_list[j][2] == -1: 🡨display shark on left position

            screen.blit(pygame.transform.flip(plane\_img, 1, 0), (enemy\_list[j][0], enemy\_list[j][1])) 🡨flipping enemy viewpoint

    return enemy\_rects 🡨return variable

#function to create enemy movement

#VVVVVV

def move\_enemies(enemy\_list, current\_score): 🡨variables to use

    enemy\_speed = 2 + current\_score//15 🡨enemies become hard/difficulty

    for j in range(len(enemy\_list)): 🡨move enemy from left and right

        if enemy\_list[j][2] == 1:

            if enemy\_list[j][0] < WIDTH: 🡨code for enemy right to left

                enemy\_list[j][0] += enemy\_speed 🡨enemy speed

            else:

                enemy\_list[j][2] = -1 🡨enemy starting position

        elif enemy\_list[j][2] == -1:

            if enemy\_list[j][0] > -320: 🡨code for enemy left to right

                enemy\_list[j][0] -= enemy\_speed 🡨enemy speed

            else:

                enemy\_list[j][2] = 1 🡨enemy starting position

        if enemy\_list[j][1] < -100:

            enemy\_list[j][1] = random.randint(HEIGHT, HEIGHT + 500) 🡨area enemies spawn

    return enemy\_list 🡨return variable

#function to create infinite loop

#VVVVVV

def update\_objects(cloud\_list, play\_y, enemy\_list):🡨variables to use

    lowest\_cloud = 0 🡨lowest cloud that could randomly spawn in display

    update\_speed = 10 🡨speed to increase difficulty and cloud random generate

    if play\_y > 200: 🡨vertical position to generate display if speed is high

        play\_y -= update\_speed

        for q in range(len(enemy\_list)): 🡨enemy display

            enemy\_list[q][1] -= update\_speed

        for j in range(len(cloud\_list)): 🡨cloud display

            cloud\_list[j][1] -= update\_speed

            if cloud\_list[j][1] > lowest\_cloud: 🡨for lowest cloud

                lowest\_cloud = cloud\_list[j][1]

        if lowest\_cloud < 750: 🡨for generating clouds

            num\_clouds = random.randint(1,2)

            if num\_clouds == 1:

                x\_pos = random.randint(0, WIDTH - 70) 🡨x position for cloud

                y\_pos = random.randint(HEIGHT + 100, HEIGHT + 300)🡨y position

                cloud\_type = random.randint (1, 3) 🡨type of cloud assets

                cloud\_list.append([x\_pos, y\_pos, cloud\_type]) 🡨display clouds

            else:

                x\_pos = random.randint(0, WIDTH/2 - 70) 🡨x position for cloud

                y\_pos = random.randint(HEIGHT + 100, HEIGHT + 300)

                cloud\_type = random.randint (1, 3)🡨type of cloud assets

                x\_pos2 = random.randint(WIDTH/2 + 70, WIDTH - 70)🡨x position

                y\_pos2 = random.randint(HEIGHT + 100, HEIGHT + 300) 🡨for y

                cloud\_type2 = random.randint (1, 3) 🡨type of cloud assets

                cloud\_list.append([x\_pos, y\_pos, cloud\_type]) 🡨display clouds

                cloud\_list.append([x\_pos2, y\_pos2, cloud\_type2])🡨display ^^

    return play\_y, cloud\_list, enemy\_list 🡨return list variable

#main game to run and loop the infinite gameplay

run = True 🡨loop running

while run: 🡨for stuff you want to happen when loop

    screen.fill(bg) 🡨make background

    timer.tick(fps) 🡨fps for gameplay

    cloud\_platforms = draw\_clouds(clouds, cloud\_images) 🡨clouds

    player = draw\_player(player\_x, player\_y, rosemi, direction) 🡨player

    enemy\_boxes = draw\_enemies(enemies, plane) 🡨enemy

    enemies = move\_enemies(enemies, score) 🡨enemy movement

    player\_y, clouds, enemies = update\_objects(clouds, player\_y, enemies)

    if game\_over: 🡨game over

        end\_text = huge\_font.render('Rose Fall!', True, 'red') 🡨end text

        end\_text2 = font.render('Game Over: Press Enter to Restart', True, 'black') 🡨end text again

        screen.blit(end\_text, (70,20)) 🡨show text on screen

        screen.blit(end\_text2, (60,80)) 🡨show text on screen

        player\_y = - 300

        y\_speed = 0

    for i in range(len(cloud\_platforms)): 🡨bounce effect

        if direction == -1 and player.colliderect(cloud\_platforms[i]):

            y\_speed \*= -1 🡨player going up and down physics

            if y\_speed > -2:

                y\_speed = -2

            bounce.play() 🡨player hit clouds

    for event in pygame.event.get(): 🡨to quit while loop

        if event.type == pygame.QUIT: 🡨the x mark top right to quit

            run = False 🡨to quit game

        if event.type == pygame.KEYDOWN: 🡨up and down physics

            if event.key == pygame.K\_LEFT:

                x\_direction = -1

            elif event.key == pygame.K\_RIGHT:

                x\_direction = 1

            if event.key == pygame.K\_RETURN and game\_over: 🡨to restart game if game over

                game\_over = False 🡨all of these are assets if game restart

                player\_x = 240 <VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV

                player\_y = 40

                direction = -1

                y\_speed = 0

                x\_direction = 0

                score = 0

                total\_distance = 0

                enemies = [[-234, random.randint(400, HEIGHT - 100), 1]]

                clouds = [[200, 100, 1], [50, 330, 2], [350, 330, 3], [200, 670, 1]]

                end\_sound.stop()

                pygame.mixer.music.play()

                pygame.mixer.music.play(-1)

        if event.type == pygame.KEYUP: 🡨player movement <>

            if event.key == pygame.K\_LEFT: 🡨to move left

                x\_direction = 0

            elif event.key == pygame.K\_RIGHT: 🡨to move right

                x\_direction = 0

    if y\_speed < 10 and not game\_over:

        y\_speed += gravity 🡨moving collusion player

    player\_y += y\_speed

    if y\_speed < 0:

        direction = 1

    else:

        direction = -1

    player\_x += x\_speed \* x\_direction

    if player\_x > WIDTH: 🡨phasing through screen left and right

        player\_x = -30

    elif player\_x < -50:

        player\_x = WIDTH - 20

    for i in range(len(enemy\_boxes)): 🡨when die score gets record

        if player.colliderect(enemy\_boxes[i]) and not game\_over: 🡨when die

            end\_sound.play()

            dead\_sound.play()

            end\_sound.play(-1)

            pygame.mixer.music.stop()

            game\_over = True

            if score > first\_high: 🡨if get high score

                file = open('high\_scores.txt', 'w')

                write\_score = str(score)

                file.write(write\_score)

                file.close()

                first\_high = score

    total\_distance += y\_speed 🡨display scores

    score = round(total\_distance / 100)

    score\_text = font.render(f'Score: {score}', True, 'black')

    screen.blit(score\_text, (10, HEIGHT - 70))

    if score > high\_score:🡨if score is more than high score it will highlight

        high\_score = score 🡨replace new high score with old

    score\_text2 = font.render(f'High Score: {high\_score}', True, 'black')

    screen.blit(score\_text2, (10, HEIGHT - 40))

    pygame.display.flip() 🡨to get out loop

pygame.quit() 🡨quit

1. **Presentation video**

****