Project Report
Flappy Bird: Horror Edition

I. Game Overview

i. **Genre:**

Flappy Bird: Horror Edition is a 2D Platformer game

ii. **Concept:**

The player as the flappy bird, flaps it's wings to avoid zombies and demonic dragons coming towards their way, while also collecting coins as a currency to purchase in the shop.

iii. **Visual:**

The art style of an 8-bit 2D Platformer game, most often seen in 1980 platformer games.

II. Inspiration

i. **Flappy Bird:**

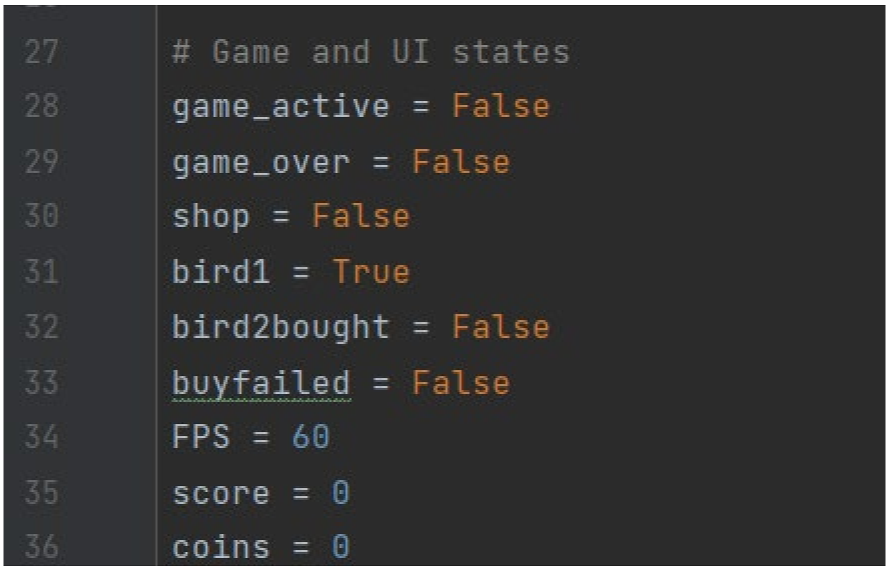
Flappy Bird is a mobile game developed by Vietnamese video game artist Dong Nguyen. The game is a side-scroller where the player controls a bird, attempting to fly between columns of green pipes without hitting them. The game was released in May 2013 but received a sudden rise in popularity in early 2014 Flappy Bird received poor reviews from some critics, who criticized its high level of difficulty and alleged plagiarism in graphics and game mechanics, while other reviewers found it addictive

ii. **Sonic.exe:**

Sonic.exe is part of a genre of online horror stories known as creepypasta.
In August 2011, user JC-the-Hyena submitted a story on the Creepypasta Wiki about a strange CD-ROM the narrator received from a friend. The CD had SONIC.EXE written on it (.exe is file extension designating an executable file). It centered around the video game franchise and character Sonic the Hedgehog. A prominent feature of the story was the look of its murderous incarnation of Sonic, who had been given black, bleeding eyeballs with glowing red pupils. The creator later explained that the creepypasta had been inspired by an edited screencap of the character. Sonic.exe grew in popularity, and a game based on Sonic.exe was created in August 2012 by MY5TCrimson, along with other tributes and parodies. The game and story both gathered tributes and parodies. It's not clear how much of the creepypasta's appeal was due to its scariness or an ironic appreciation of its amateurish writing and clichéd style, which later got it removed from the Creepypasta Wiki.

III. Gameplay Mechanics

i. Game states:



```
# Game and UI states
game_active = False
game_over = False
shop = False
bird1 = True
bird2bought = False
buyfailed = False
FPS = 60
score = 0
coins = 0
```

A list of states and variables

Game_active -> To display game UI

Game_over -> To display game over UI

Shop -> To display shop UI

Bird1 -> If bird skin 1 is equipped, if not then bird skin 2 is equipped

Bird2bought -> If bird skin 2 is bought, will display different UI, explained in IV. Interface – ii. Shop menu

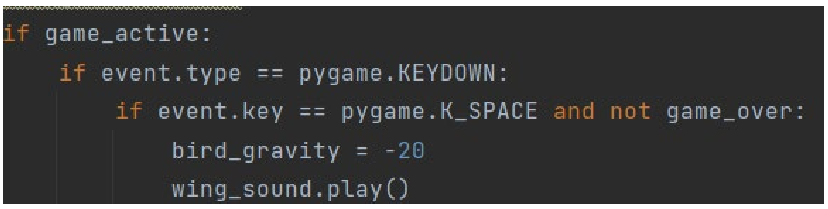
Buyfailed -> If insufficient balance to buy bird skin 2, will display “Insufficient Balance!” explained in IV. Interface – iii. Shop

FPS -> Frames Per Second, used for the last line clock.tick(FPS)

Score -> In game collected coins for each round

Coins -> Overall collected coins, displayed in the main menu and shop

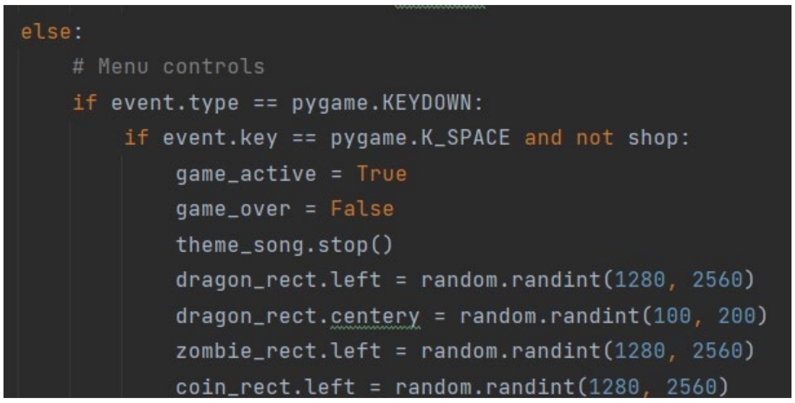
ii. Controls



```
if game_active:
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_SPACE and not game_over:
            bird_gravity = -20
            wing_sound.play()
```

Located in while True loop – for event in pygame.event.get() – if game_active

Making it able for the bird to "flap" in to a lower bird_rect.centery value if the player presses space. Only available while the game_active = True and game_over = False



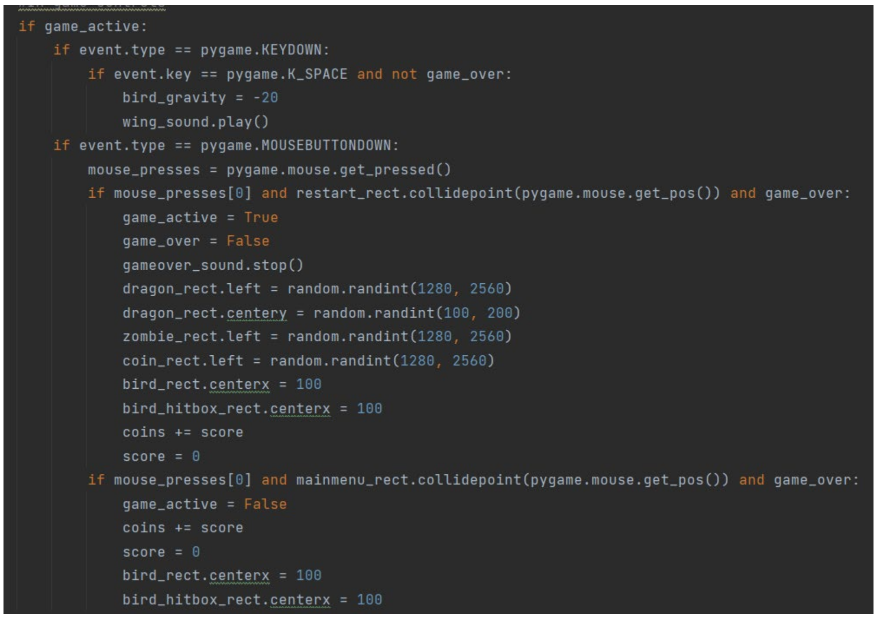
```
else:
    # Menu controls
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_SPACE and not shop:
            game_active = True
            game_over = False
            theme_song.stop()
            dragon_rect.left = random.randint(1280, 2560)
            dragon_rect.centery = random.randint(100, 200)
            zombie_rect.left = random.randint(1280, 2560)
            coin_rect.left = random.randint(1280, 2560)
```

Located in while True loop – for event in pygame.event.get() – if game_active's else statement

Making it able for the player to start the game from the main menu, by pressing space.

Only available while the game_active = False and shop = False

iii. Pressing UI surfaces



```
if game_active:
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_SPACE and not game_over:
            bird_gravity = -20
            wing_sound.play()
    if event.type == pygame.MOUSEBUTTONDOWN:
        mouse_presses = pygame.mouse.get_pressed()
        if mouse_presses[0] and restart_rect.collidepoint(pygame.mouse.get_pos()) and game_over:
            game_active = True
            game_over = False
            gameover_sound.stop()
            dragon_rect.left = random.randint(1280, 2560)
            dragon_rect.centery = random.randint(100, 200)
            zombie_rect.left = random.randint(1280, 2560)
            coin_rect.left = random.randint(1280, 2560)
            bird_rect.centerx = 100
            bird_hitbox_rect.centerx = 100
            coins += score
            score = 0
        if mouse_presses[0] and mainmenu_rect.collidepoint(pygame.mouse.get_pos()) and game_over:
            game_active = False
            coins += score
            score = 0
            bird_rect.centerx = 100
            bird_hitbox_rect.centerx = 100
```

Located in while True loop – for event in pygame.event.get()'s else statement – if game_active

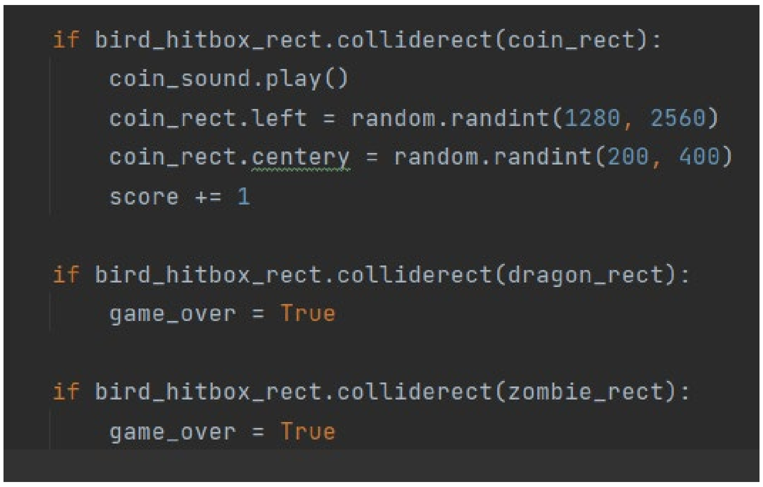
Making it able for the player to press restart, and goes straight back to the game again (game_active = True, game_over = False), having their score added to the initial coins variable, and having their score reset.

Resets the bird's death animation (explained in IV. Interface – IV. Game over UI), resets the demon dragon and zombie's randomized spawn point.

Also making it able for the player to press main menu, where it goes straight to the main menu UI (game_active = False), having their score added to their initial coins variable, and having their score reset.

Only available if the game is still running (game_active = True)

iv. Collisions



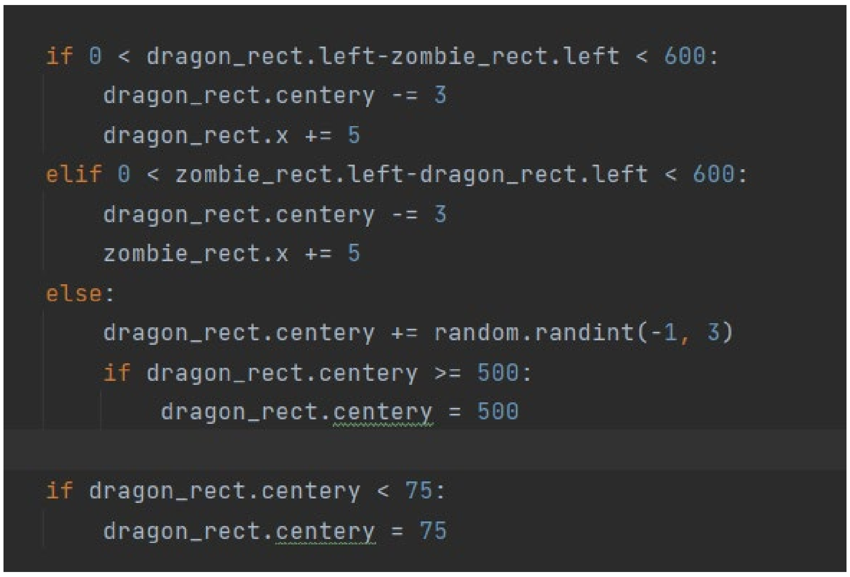
```
if bird_hitbox_rect.colliderect(coin_rect):
    coin_sound.play()
    coin_rect.left = random.randint(1280, 2560)
    coin_rect.centery = random.randint(200, 400)
    score += 1

if bird_hitbox_rect.colliderect(dragon_rect):
    game_over = True

if bird_hitbox_rect.colliderect(zombie_rect):
    game_over = True
```

If bird hitbox's rectangle collides with coin's rectangle, coin's rectangle will reposition in to a new position, giving the illusion of getting "eaten" and spawns another one. 1 will be added to the initial score value.

If bird hitbox's rectangle collides with demon dragon's rectangle, it will display game over UI.

If bird hitbox's rectangle collides with zombie's rectangle, it will display game over UI.

v. Enemy repositioning and move set algorithm

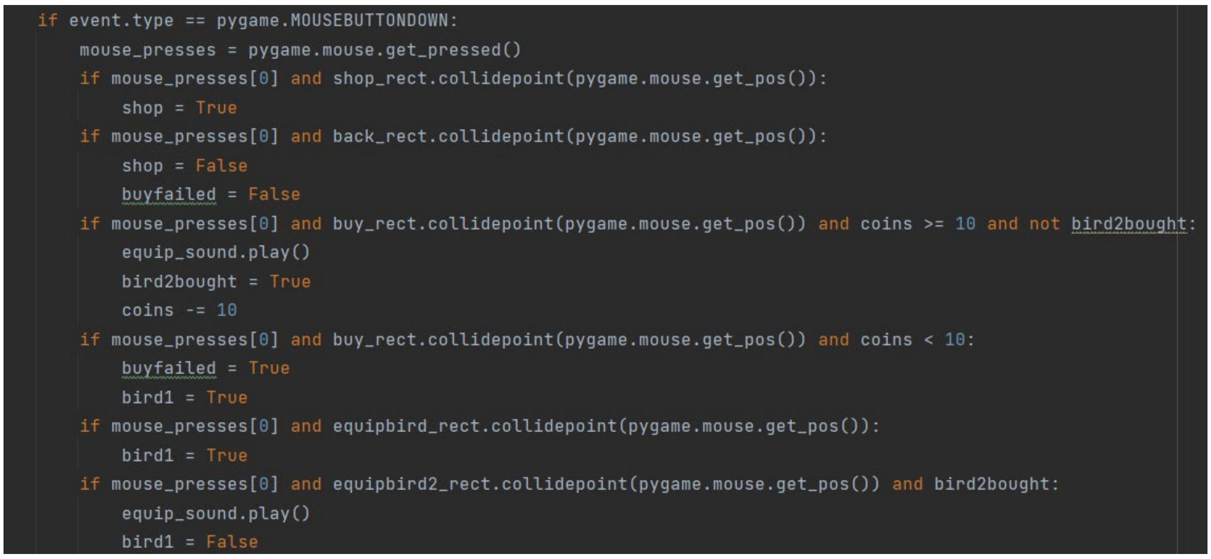


```
if 0 < dragon_rect.left-zombie_rect.left < 600:
    dragon_rect.centery -= 3
    dragon_rect.x += 5
elif 0 < zombie_rect.left-dragon_rect.left < 600:
    dragon_rect.centery -= 3
    zombie_rect.x += 5
else:
    dragon_rect.centery += random.randint(-1, 3)
    if dragon_rect.centery >= 500:
        dragon_rect.centery = 500

if dragon_rect.centery < 75:
    dragon_rect.centery = 75
```

An if else statement that repositions or changes the value of zombie_rect.x (zombie's X position), dragon_rect.centerx (dragon's X position), and dragon_rect.centery (dragon's Y position).

Does not permit the dragon to fly above the sky limit.

Adding a feel of randomized enemy move set and repositioning.


```
if event.type == pygame.MOUSEBUTTONDOWN:
    mouse_presses = pygame.mouse.get_pressed()
    if mouse_presses[0] and shop_rect.collidepoint(pygame.mouse.get_pos()):
        shop = True
    if mouse_presses[0] and back_rect.collidepoint(pygame.mouse.get_pos()):
        shop = False
        buyfailed = False
    if mouse_presses[0] and buy_rect.collidepoint(pygame.mouse.get_pos()) and coins >= 10 and not bird2bought:
        equip_sound.play()
        bird2bought = True
        coins -= 10
    if mouse_presses[0] and buy_rect.collidepoint(pygame.mouse.get_pos()) and coins < 10:
        buyfailed = True
        bird1 = True
    if mouse_presses[0] and equipbird_rect.collidepoint(pygame.mouse.get_pos()):
        bird1 = True
    if mouse_presses[0] and equipbird2_rect.collidepoint(pygame.mouse.get_pos()) and bird2bought:
        equip_sound.play()
        bird1 = False
```

Located in while True loop – for event in pygame.event.get()'s else statement – if game_active's else statement

A range of if else statements that makes the player able to visit the shop UI by pressing the shop button, go back to main menu UI by pressing the back button, and an algorithm to decide the game states.

If coins is sufficient, bird skin 2 hasn't been bought yet, and player presses buy: bird2bought = True, 10 coins deducted from balance

If coins isn't sufficient, and player pressed buy: buyfailed = True, bird1 = True

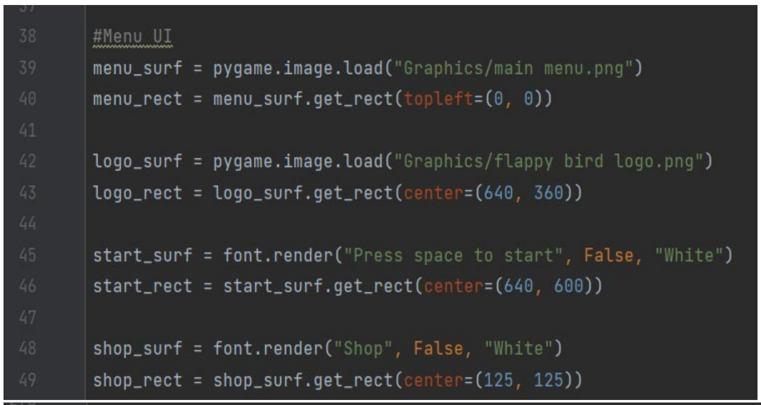
If player presses equip bird skin 1 : bird1 = True

If player pressed equip bird skin 2 and bird skin 2 has been bought: bird1 : False

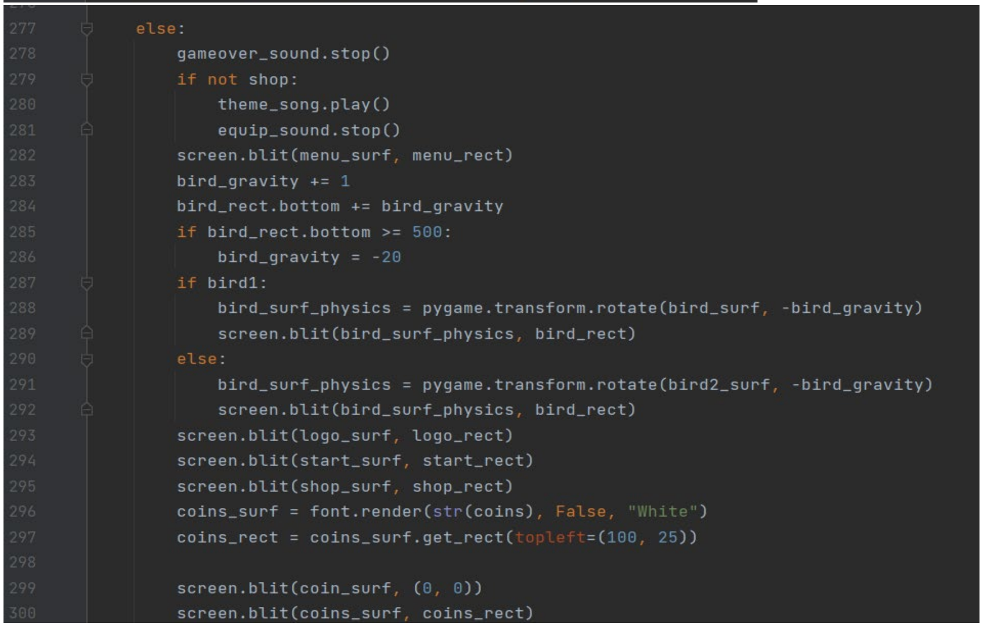
Only available if the game isn't running (game_active = False)

IV. Interface

i. Main menu:



```
#Menu UI
menu_surf = pygame.image.load("Graphics/main menu.png")
menu_rect = menu_surf.get_rect(topleft=(0, 0))

logo_surf = pygame.image.load("Graphics/flappy bird logo.png")
logo_rect = logo_surf.get_rect(center=(640, 360))

start_surf = font.render("Press space to start", False, "White")
start_rect = start_surf.get_rect(center=(640, 600))

shop_surf = font.render("Shop", False, "White")
shop_rect = shop_surf.get_rect(center=(125, 125))
```



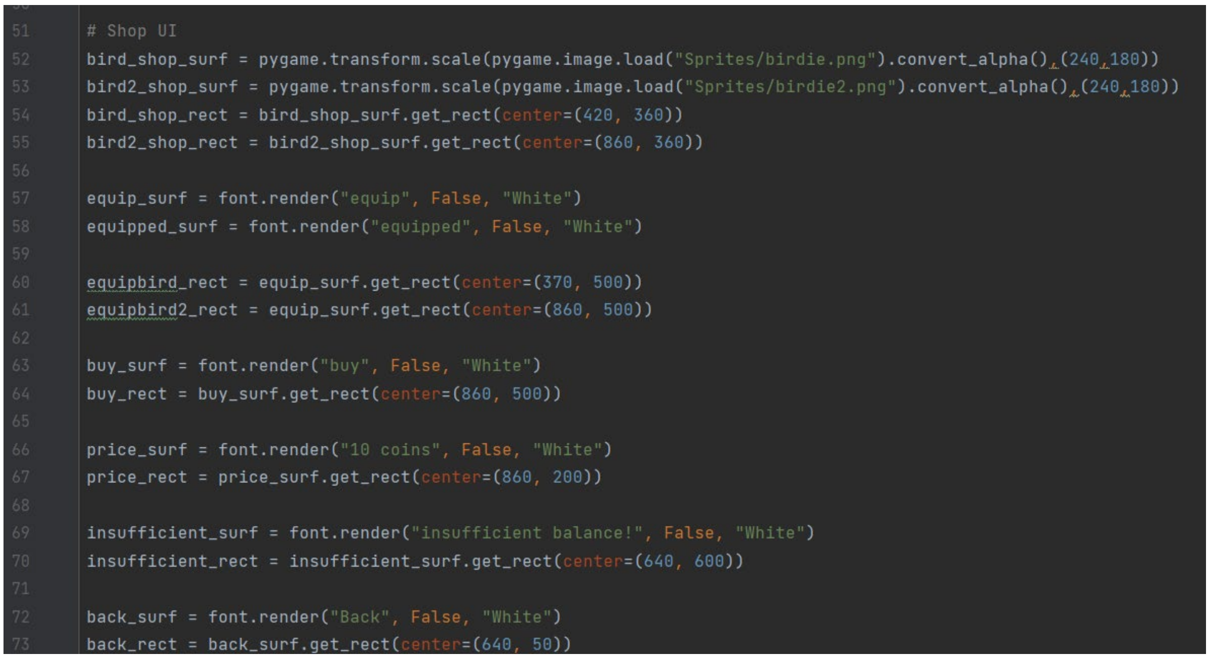
```
    else:
        gameover_sound.stop()
        if not shop:
            theme_song.play()
            equip_sound.stop()
        screen.blit(menu_surf, menu_rect)
        bird_gravity += 1
        bird_rect.bottom += bird_gravity
        if bird_rect.bottom >= 500:
            bird_gravity = -20
        if bird1:
            bird_surf_physics = pygame.transform.rotate(bird_surf, -bird_gravity)
            screen.blit(bird_surf_physics, bird_rect)
        else:
            bird_surf_physics = pygame.transform.rotate(bird2_surf, -bird_gravity)
            screen.blit(bird_surf_physics, bird_rect)
        screen.blit(logo_surf, logo_rect)
        screen.blit(start_surf, start_rect)
        screen.blit(shop_surf, shop_rect)
        coins_surf = font.render(str(coins), False, "White")
        coins_rect = coins_surf.get_rect(topleft=(100, 25))

        screen.blit(coin_surf, (0, 0))
        screen.blit(coins_surf, coins_rect)
```

Located in while True loop – if game_active's else statement

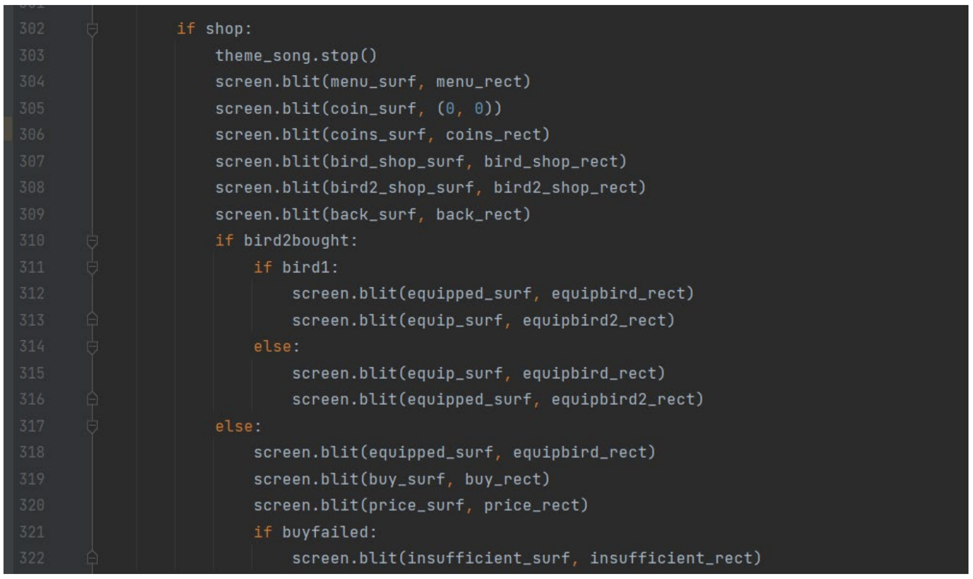
Main menu's UI, that displays menu background, Flappy Bird Logo, start text, shop button,

Coins that are attained from the game display, and the bird's flapping animation (with it's surface and rectangle written in iii. game UI).

ii. Shop menu:



```
# Shop UI
bird_shop_surf = pygame.transform.scale(pygame.image.load("Sprites/birdie.png").convert_alpha(),(240,180))
bird2_shop_surf = pygame.transform.scale(pygame.image.load("Sprites/birdie2.png").convert_alpha(),(240,180))
bird_shop_rect = bird_shop_surf.get_rect(center=(420, 360))
bird2_shop_rect = bird2_shop_surf.get_rect(center=(860, 360))

equip_surf = font.render("equip", False, "White")
equipped_surf = font.render("equipped", False, "White")

equipbird_rect = equip_surf.get_rect(center=(370, 500))
equipbird2_rect = equip_surf.get_rect(center=(860, 500))

buy_surf = font.render("buy", False, "White")
buy_rect = buy_surf.get_rect(center=(860, 500))

price_surf = font.render("10 coins", False, "White")
price_rect = price_surf.get_rect(center=(860, 200))

insufficient_surf = font.render("insufficient balance!", False, "White")
insufficient_rect = insufficient_surf.get_rect(center=(640, 600))

back_surf = font.render("Back", False, "White")
back_rect = back_surf.get_rect(center=(640, 50))
```



```
        if shop:
            theme_song.stop()
            screen.blit(menu_surf, menu_rect)
            screen.blit(coin_surf, (0, 0))
            screen.blit(coins_surf, coins_rect)
            screen.blit(bird_shop_surf, bird_shop_rect)
            screen.blit(bird2_shop_surf, bird2_shop_rect)
            screen.blit(back_surf, back_rect)
            if bird2bought:
                if bird1:
                    screen.blit(equipped_surf, equipbird_rect)
                    screen.blit(equip_surf, equipbird2_rect)
                else:
                    screen.blit(equip_surf, equipbird_rect)
                    screen.blit(equipped_surf, equipbird2_rect)
            else:
                screen.blit(equipped_surf, equipbird_rect)
                screen.blit(buy_surf, buy_rect)
                screen.blit(price_surf, price_rect)
                if buyfailed:
                    screen.blit(insufficient_surf, insufficient_rect)
```

Located in while True loop – if game_active's else statement – if shop

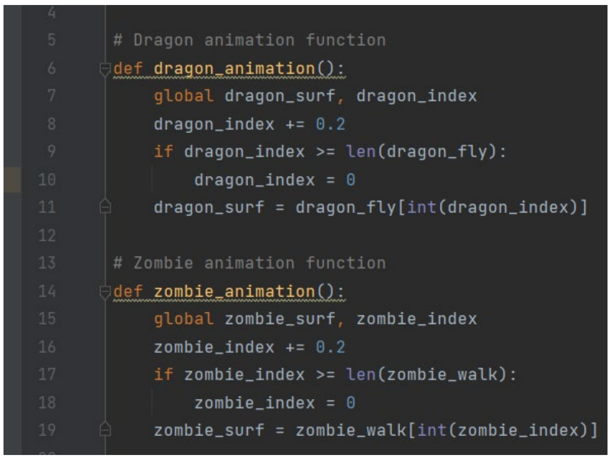
Shop menu's UI, that displays main menu background, total coin's display, back button to go back to main menu, two of the bird skins that you could buy.

Also has an if else statements that controls the display of the texts and game states:

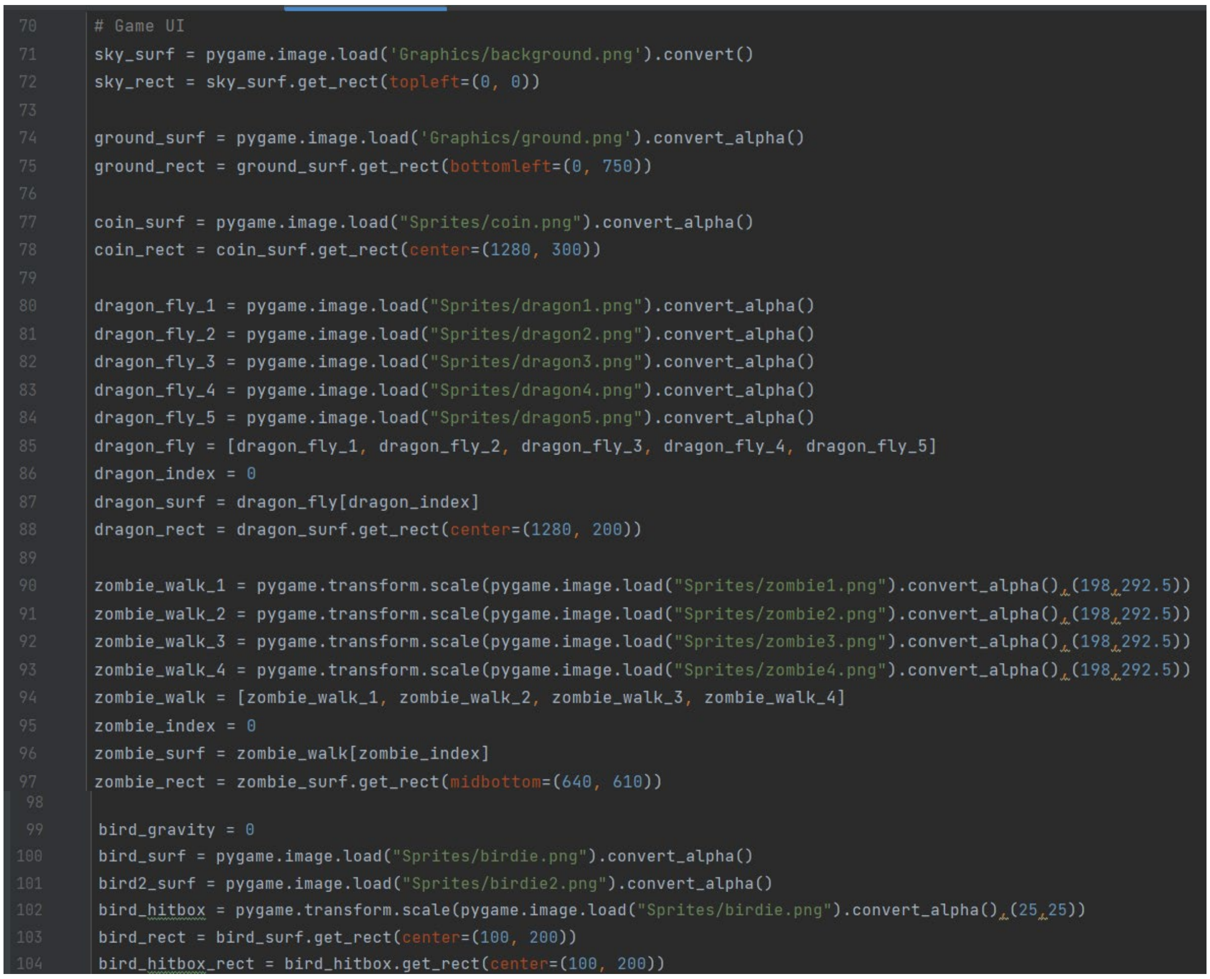
buy button and price tag displays if bird2 hasn't been bought (bird2bought = False), insufficient text displays if bird2 isn't able to be bought (buyfailed = True), equip button to equip for both the bird skins, and equipped text if equip is pressed for both the bird skins.

Bird2bought and buyfailed states are explained in Mechanics – Game states

iii. Game UI



```
# Dragon animation function
def dragon_animation():
    global dragon_surf, dragon_index
    dragon_index += 0.2
    if dragon_index >= len(dragon_fly):
        dragon_index = 0
    dragon_surf = dragon_fly[int(dragon_index)]

# Zombie animation function
def zombie_animation():
    global zombie_surf, zombie_index
    zombie_index += 0.2
    if zombie_index >= len(zombie_walk):
        zombie_index = 0
    zombie_surf = zombie_walk[int(zombie_index)]
```

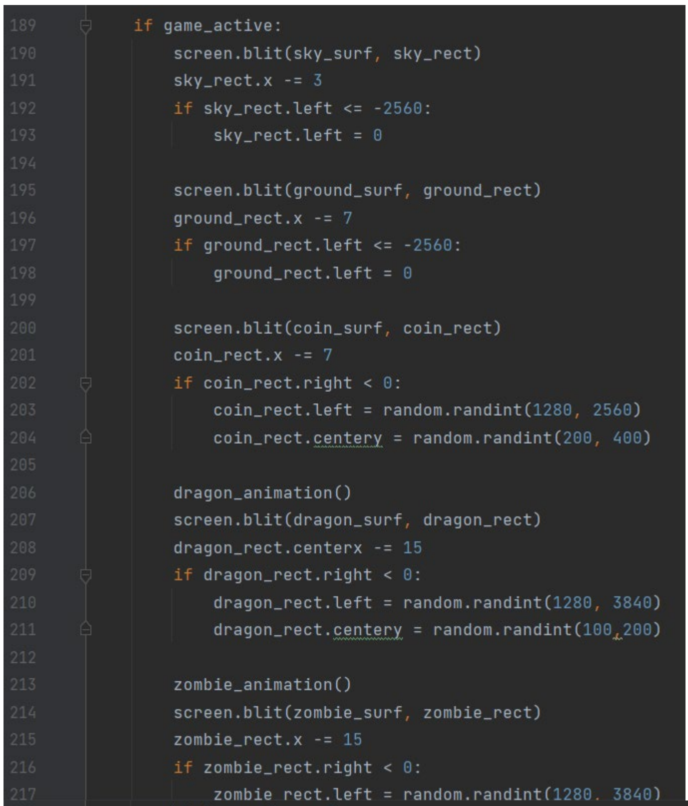


```
# Game UI
sky_surf = pygame.image.load('Graphics/background.png').convert()
sky_rect = sky_surf.get_rect(topleft=(0, 0))

ground_surf = pygame.image.load('Graphics/ground.png').convert_alpha()
ground_rect = ground_surf.get_rect(bottomleft=(0, 750))

coin_surf = pygame.image.load("Sprites/coin.png").convert_alpha()
coin_rect = coin_surf.get_rect(center=(1280, 300))

dragon_fly_1 = pygame.image.load("Sprites/dragon1.png").convert_alpha()
dragon_fly_2 = pygame.image.load("Sprites/dragon2.png").convert_alpha()
dragon_fly_3 = pygame.image.load("Sprites/dragon3.png").convert_alpha()
dragon_fly_4 = pygame.image.load("Sprites/dragon4.png").convert_alpha()
dragon_fly_5 = pygame.image.load("Sprites/dragon5.png").convert_alpha()
dragon_fly = [dragon_fly_1, dragon_fly_2, dragon_fly_3, dragon_fly_4, dragon_fly_5]
dragon_index = 0
dragon_surf = dragon_fly[dragon_index]
dragon_rect = dragon_surf.get_rect(center=(1280, 200))

zombie_walk_1 = pygame.transform.scale(pygame.image.load("Sprites/zombie1.png").convert_alpha(),(198,292.5))
zombie_walk_2 = pygame.transform.scale(pygame.image.load("Sprites/zombie2.png").convert_alpha(),(198,292.5))
zombie_walk_3 = pygame.transform.scale(pygame.image.load("Sprites/zombie3.png").convert_alpha(),(198,292.5))
zombie_walk_4 = pygame.transform.scale(pygame.image.load("Sprites/zombie4.png").convert_alpha(),(198,292.5))
zombie_walk = [zombie_walk_1, zombie_walk_2, zombie_walk_3, zombie_walk_4]
zombie_index = 0
zombie_surf = zombie_walk[zombie_index]
zombie_rect = zombie_surf.get_rect(midbottom=(640, 610))

bird_gravity = 0
bird_surf = pygame.image.load("Sprites/birdie.png").convert_alpha()
bird2_surf = pygame.image.load("Sprites/birdie2.png").convert_alpha()
bird_hitbox = pygame.transform.scale(pygame.image.load("Sprites/birdie.png").convert_alpha(),(25,25))
bird_rect = bird_surf.get_rect(center=(100, 200))
bird_hitbox_rect = bird_hitbox.get_rect(center=(100, 200))
```

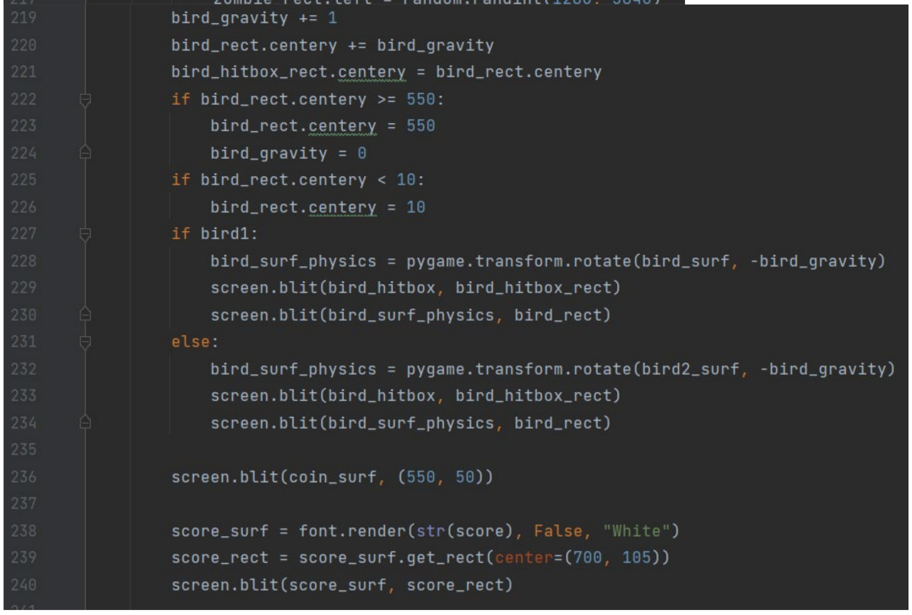
All the surfaces and rectangles for the game UI, including:

The sky, the ground, collectable coins, animated dragon, animated zombie, the bird, the bird’s hitbox.

Animated dragon and zombie controlled within def dragon_animation() and def zombie_animation()



```python
if game_active:
    screen.blit(sky_surf, sky_rect)
    sky_rect.x -= 3
    if sky_rect.left <= -2560:
        sky_rect.left = 0

    screen.blit(ground_surf, ground_rect)
    ground_rect.x -= 7
    if ground_rect.left <= -2560:
        ground_rect.left = 0

    screen.blit(coin_surf, coin_rect)
    coin_rect.x -= 7
    if coin_rect.right < 0:
        coin_rect.left = random.randint(1280, 2560)
        coin_rect.centery = random.randint(200, 400)

    dragon_animation()
    screen.blit(dragon_surf, dragon_rect)
    dragon_rect.centerx -= 15
    if dragon_rect.right < 0:
        dragon_rect.left = random.randint(1280, 3840)
        dragon_rect.centery = random.randint(100,200)

    zombie_animation()
    screen.blit(zombie_surf, zombie_rect)
    zombie_rect.x -= 15
    if zombie_rect.right < 0:
        zombie_rect.left = random.randint(1280, 3840)
```



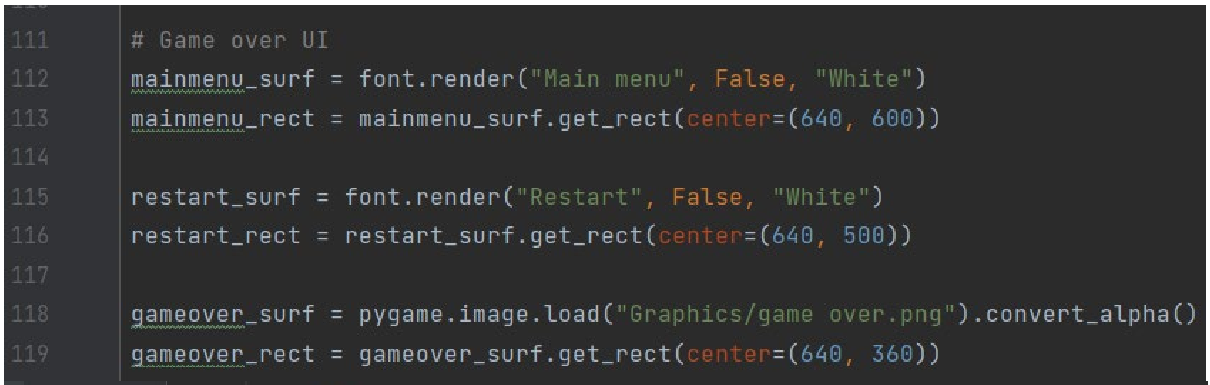
```python
    bird_gravity += 1
    bird_rect.centery += bird_gravity
    bird_hitbox_rect.centery = bird_rect.centery
    if bird_rect.centery >= 550:
        bird_rect.centery = 550
        bird_gravity = 0
    if bird_rect.centery < 10:
        bird_rect.centery = 10
    if bird1:
        bird_surf_physics = pygame.transform.rotate(bird_surf, -bird_gravity)
        screen.blit(bird_hitbox, bird_hitbox_rect)
        screen.blit(bird_surf_physics, bird_rect)
    else:
        bird_surf_physics = pygame.transform.rotate(bird2_surf, -bird_gravity)
        screen.blit(bird_hitbox, bird_hitbox_rect)
        screen.blit(bird_surf_physics, bird_rect)

    screen.blit(coin_surf, (550, 50))

    score_surf = font.render(str(score), False, "White")
    score_rect = score_surf.get_rect(center=(700, 105))
    screen.blit(score_surf, score_rect)
```

Located in while True loop - if game_active

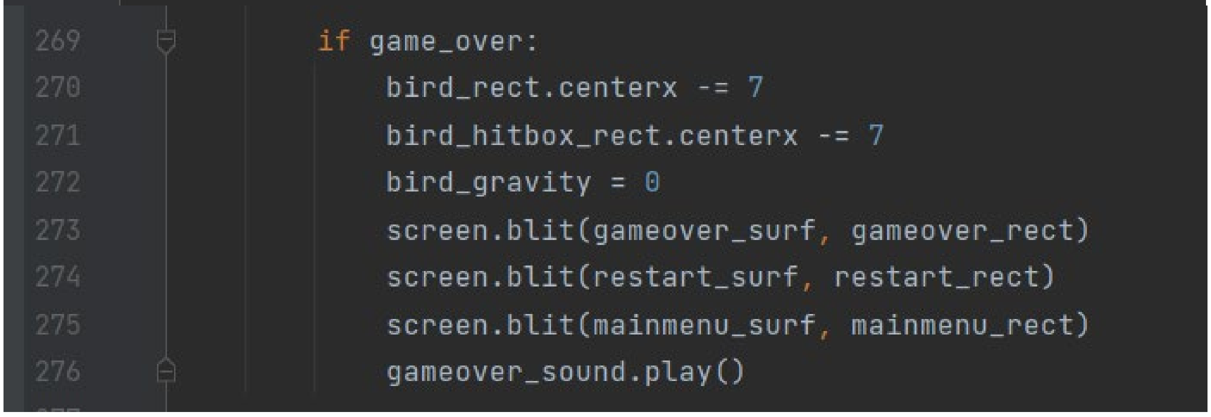
Displays the moving sky, the moving ground, the moving coins, the animated and moving dragons, the animated and moving zombies, the bird including it's overall physics, an if else statement whether bird skin 1 or bird skin 2 is equipped to display either, the coin symbol for the score display, and lastly the score display

Bird1 state is explained in Mechanics - Game states

iv. Game over UI



```python
# Game over UI
mainmenu_surf = font.render("Main menu", False, "White")
mainmenu_rect = mainmenu_surf.get_rect(center=(640, 600))

restart_surf = font.render("Restart", False, "White")
restart_rect = restart_surf.get_rect(center=(640, 500))

gameover_surf = pygame.image.load("Graphics/game over.png").convert_alpha()
gameover_rect = gameover_surf.get_rect(center=(640, 360))
```



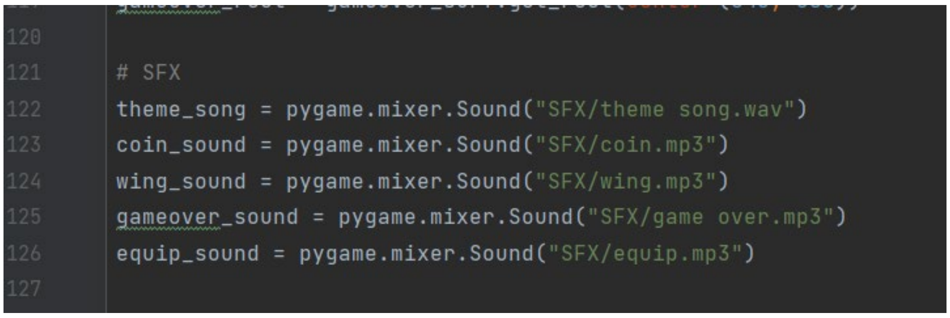
```python
269         if game_over:
270             bird_rect.centerx -= 7
271             bird_hitbox_rect.centerx -= 7
272             bird_gravity = 0
273             screen.blit(gameover_surf, gameover_rect)
274             screen.blit(restart_surf, restart_rect)
275             screen.blit(mainmenu_surf, mainmenu_rect)
276             gameover_sound.play()
```

Located in while True loop - if game_active – if game_over

Displays the game over surface, the restart button, and the main menu button.

Since the game is still active, the game still plays in the background. Though having the statement in 270 and 271 will make the bird slowly fade away to the left, as it would look like it’s being dragged down by the monster.

V. Music and SFX (Sound Effects)



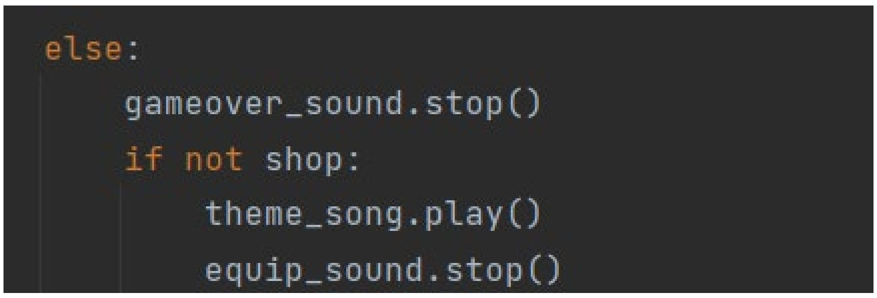
```
# SFX
theme_song = pygame.mixer.Sound("SFX/theme song.wav")
coin_sound = pygame.mixer.Sound("SFX/coin.mp3")
wing_sound = pygame.mixer.Sound("SFX/wing.mp3")
gameover_sound = pygame.mixer.Sound("SFX/game over.mp3")
equip_sound = pygame.mixer.Sound("SFX/equip.mp3")
```

Music and sound effects are taken from Sonic.exe's and Flappy Bird's soundboard

The list of sound effects that is used:

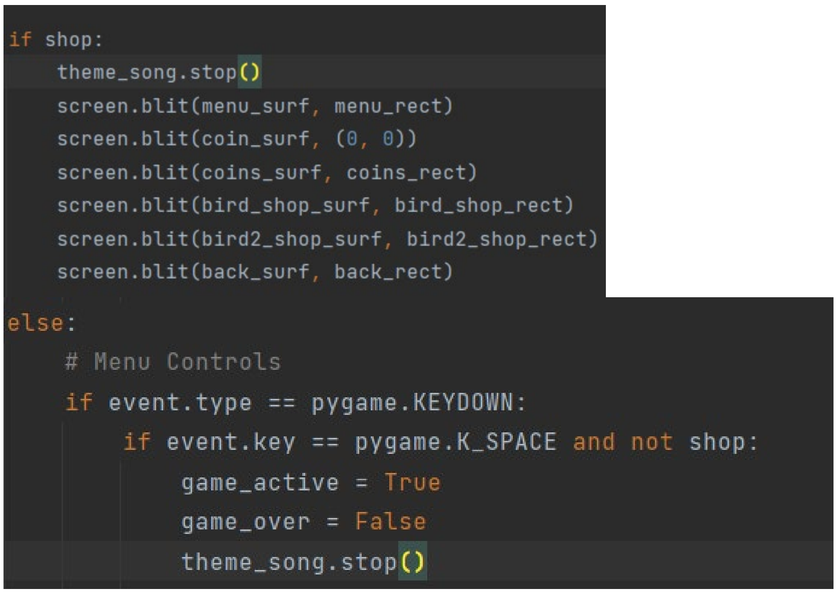
- Theme song.wav (Sonic.exe's Green Hill Theme Song)

  Plays in the main menu (game_active = False, shop = False)



```
else:
    gameover_sound.stop()
    if not shop:
        theme_song.play()
        equip_sound.stop()
```

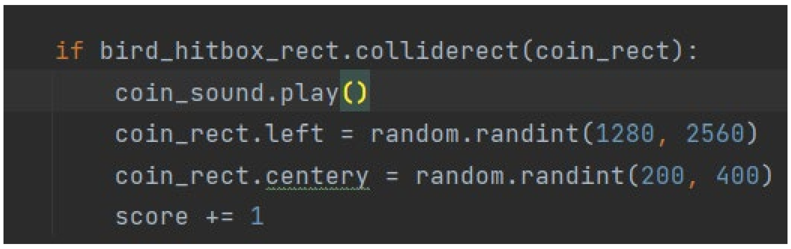
Stops when in game (game_active = True), stops when in shop UI (shop = True)



```
if shop:
    theme_song.stop()
    screen.blit(menu_surf, menu_rect)
    screen.blit(coin_surf, (0, 0))
    screen.blit(coins_surf, coins_rect)
    screen.blit(bird_shop_surf, bird_shop_rect)
    screen.blit(bird2_shop_surf, bird2_shop_rect)
    screen.blit(back_surf, back_rect)
```

```
else:
    # Menu Controls
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_SPACE and not shop:
            game_active = True
            game_over = False
            theme_song.stop()
```

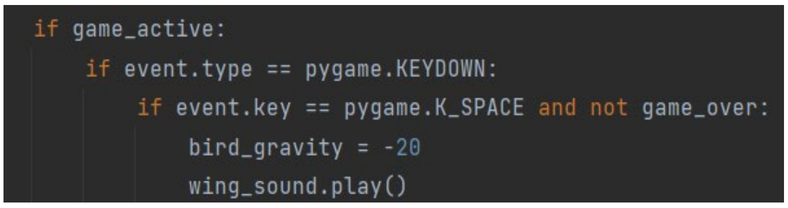
- Coin.mp3 (Original Sonic's coin collected)



```
if bird_hitbox_rect.colliderect(coin_rect):
    coin_sound.play()
    coin_rect.left = random.randint(1280, 2560)
    coin_rect.centery = random.randint(200, 400)
    score += 1
```

Plays if bird hitbox's rectangle collides with coin (game_active = True, game_over = False)

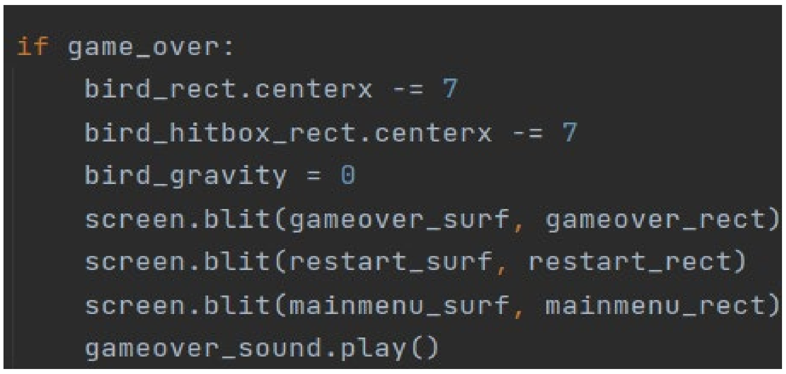
- Wing.mp3 (Flappy bird's wing sound)



```
if game_active:
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_SPACE and not game_over:
            bird_gravity = -20
            wing_sound.play()
```

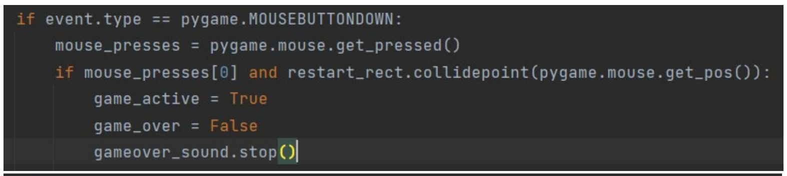
Plays if game_active = True, game_over = False, player presses space key

- Game over.mp3 (Sonic.exe's game over sound)

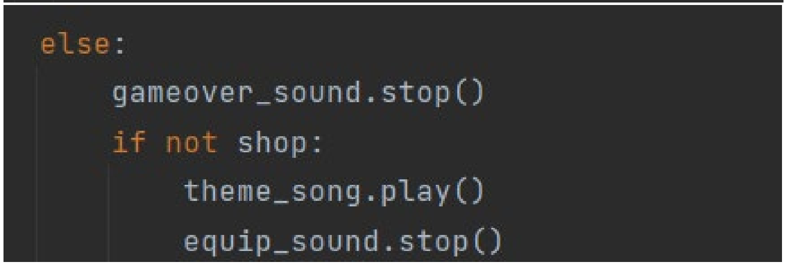


```
if game_over:
    bird_rect.centerx -= 7
    bird_hitbox_rect.centerx -= 7
    bird_gravity = 0
    screen.blit(gameover_surf, gameover_rect)
    screen.blit(restart_surf, restart_rect)
    screen.blit(mainmenu_surf, mainmenu_rect)
    gameover_sound.play()
```

Plays if game_active = True, game_over = True



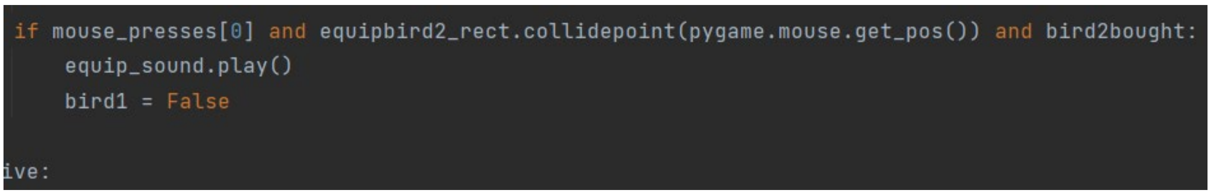
```
if event.type == pygame.MOUSEBUTTONDOWN:
    mouse_presses = pygame.mouse.get_pressed()
    if mouse_presses[0] and restart_rect.collidepoint(pygame.mouse.get_pos()):
        game_active = True
        game_over = False
        gameover_sound.stop()
```



```
else:
    gameover_sound.stop()
    if not shop:
        theme_song.play()
        equip_sound.stop()
```

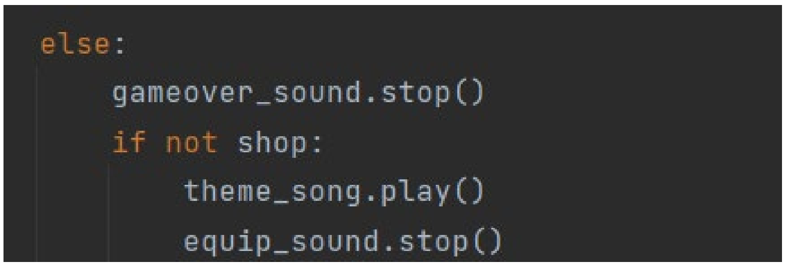
Stops if game_active = True, and game_over = False (Player press restart on game over UI)
Or if game_active = False

- Equip.mp3 (Sonic.exe's evil laugh)



```
if mouse_presses[0] and equipbird2_rect.collidepoint(pygame.mouse.get_pos()) and bird2bought:
    equip_sound.play()
    bird1 = False

ive:
```

Plays if game_active = False, Bird skin 2 is bought (bird2bought = True), and player presses equip bird skin 2 button

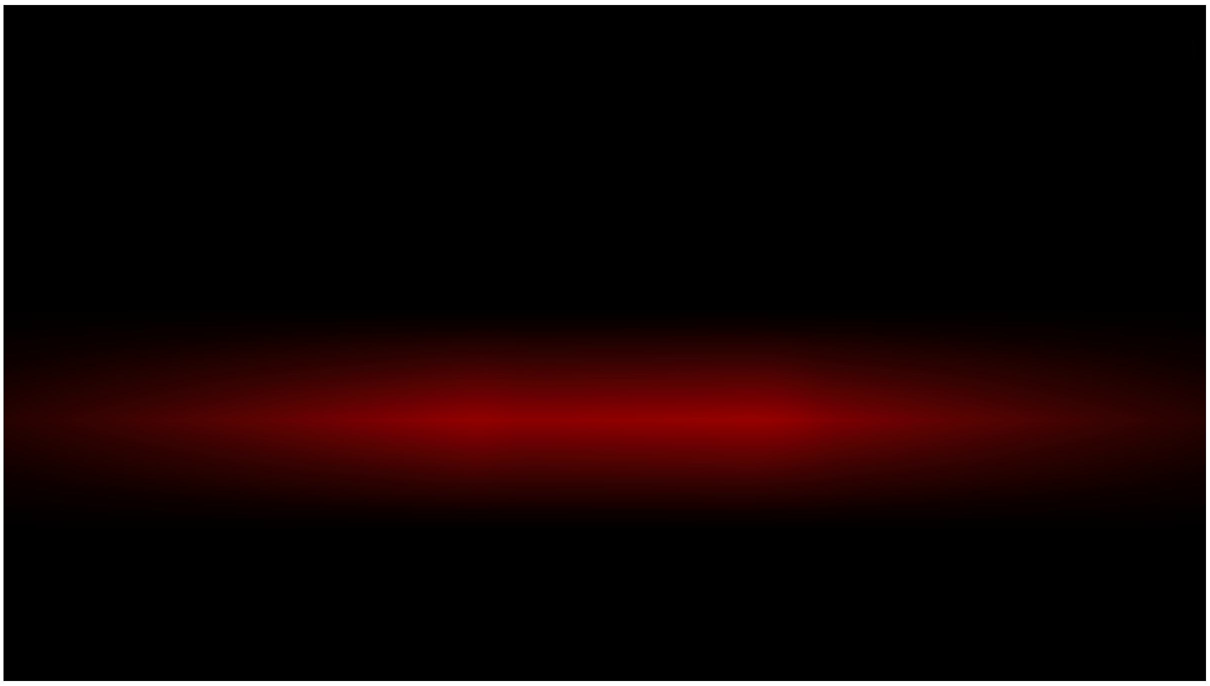


```
else:
    gameover_sound.stop()
    if not shop:
        theme_song.play()
        equip_sound.stop()
```

Stops if player goes back to main menu (game_active = False, shop = False)

VI. Sprites

i. Main menu background



ii. Flappy Bird logo

FlappyBird

iii. Game over logo

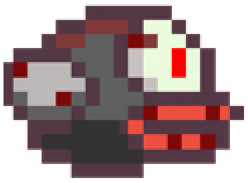
GAME OVER

iv. Game UI background

v. Game UI ground

vi. Bird (skin 1)

vii. Bird (skin 2)



viii. Demon dragon (animated)

ix. Zombie (animated)

x. Coin