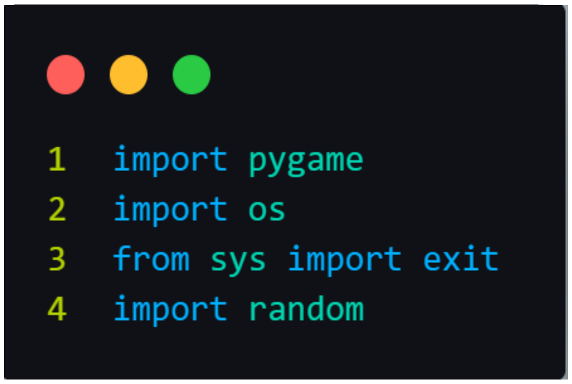
Carl Victor A. Buenafe December 10, 2023

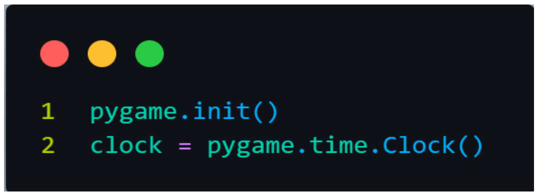
CP101 (M002) TTH: 7:30AM-9:00AM

Explaining the Code



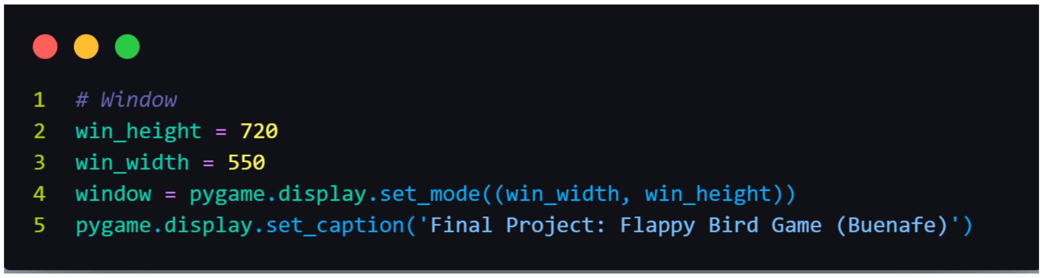
```python
import pygame
import os
from sys import exit
import random
```

- import pygame: This line imports the Pygame library, which allows the program to leverage its features to create applications such as games.
- Import os: The os module, short for operating system, provides a way to interact with the underlying operating system.
- from sys import exit: This line imports the sys module's exit function. To end the program, use the exit function.
- Import random: This line imports the random module, which contains routines for creating pseudo-random numbers. This module is widely used in games for tasks.



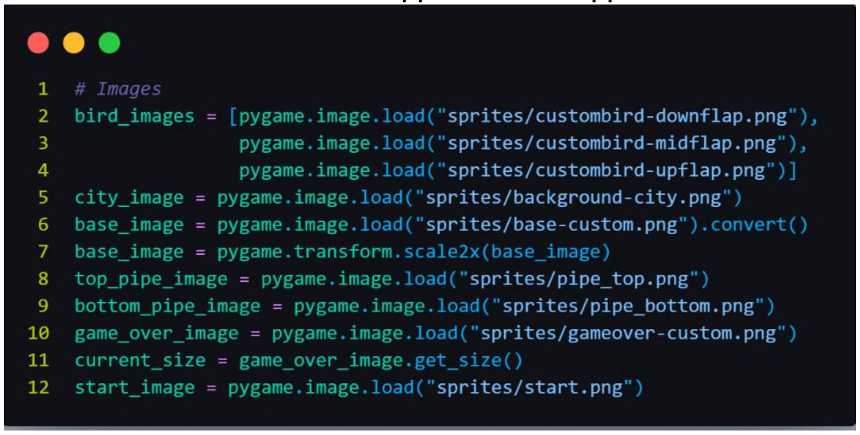
```python
pygame.init()
clock = pygame.time.Clock()
```

- The pygame.init() function is called to initialize all the Pygame modules.
- The pygame.time.Clock() function creates a new Clock object, which is used to regulate the game's frame rate.



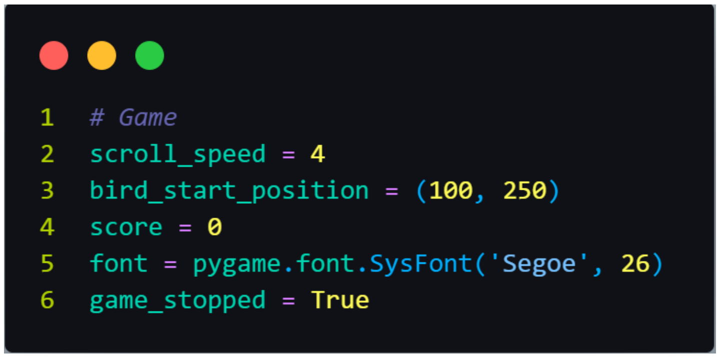
```python
# Window
win_height = 720
win_width = 550
window = pygame.display.set_mode((win_width, win_height))
pygame.display.set_caption('Final Project: Flappy Bird Game (Buenafe)')
```

- This code shows of a code for window, this code is used to function on adjusting the size of the window for the pygame when the your code runs, a window will automatically appear of how the size will appear based on the code like win_height and win_width. A variable window will be the output of appearing itself when the code runs. A pygame.display.set.caption will be the window name that will appear on the upper left of the window.



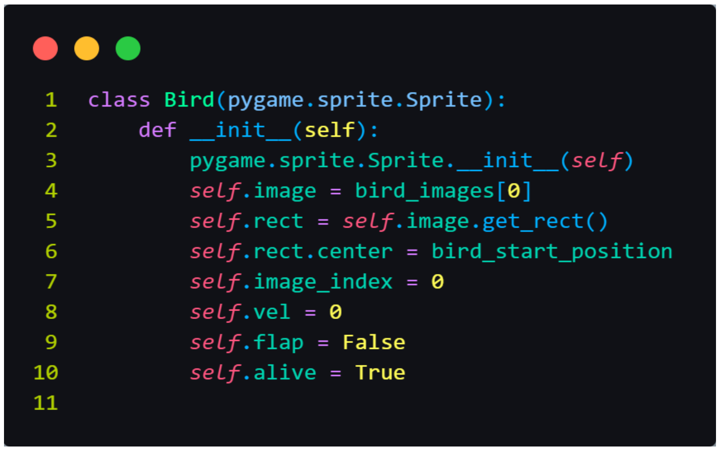
```python
# Images
bird_images = [pygame.image.load("sprites/custombird-downflap.png"),
               pygame.image.load("sprites/custombird-midflap.png"),
               pygame.image.load("sprites/custombird-upflap.png")]
city_image = pygame.image.load("sprites/background-city.png")
base_image = pygame.image.load("sprites/base-custom.png").convert()
base_image = pygame.transform.scale2x(base_image)
top_pipe_image = pygame.image.load("sprites/pipe_top.png")
bottom_pipe_image = pygame.image.load("sprites/pipe_bottom.png")
game_over_image = pygame.image.load("sprites/gameover-custom.png")
current_size = game_over_image.get_size()
start_image = pygame.image.load("sprites/start.png")
```

- This code will be the images that will insert pictures with png format for graphic purposes in the pygame, to insert the specific image, the code should put a file location or the file directory to detect the specific file to insert the image in the game.



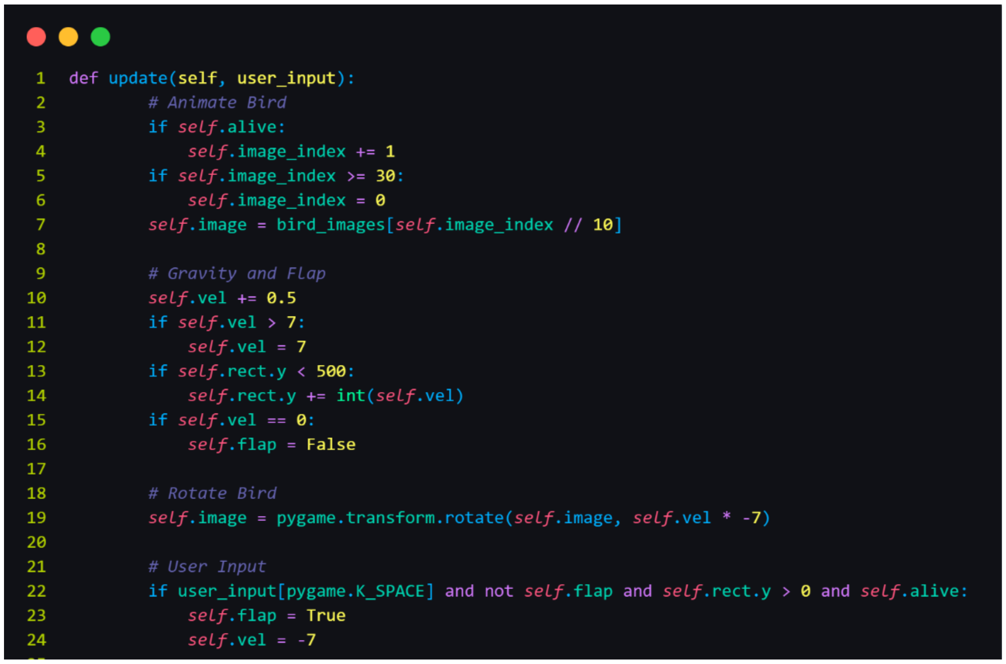
```python
# Game
scroll_speed = 4
bird_start_position = (100, 250)
score = 0
font = pygame.font.SysFont('Segoe', 26)
game_stopped = True
```

- This code functions about using the variables in the python code, using variables will work once you put a code that will work to that function to work the code by using a variable, like the bird start position, the bird or the character of the game will start at the specific starting position depending on the position of the window screen. Score variable will detect for displaying score in the screen and etc.



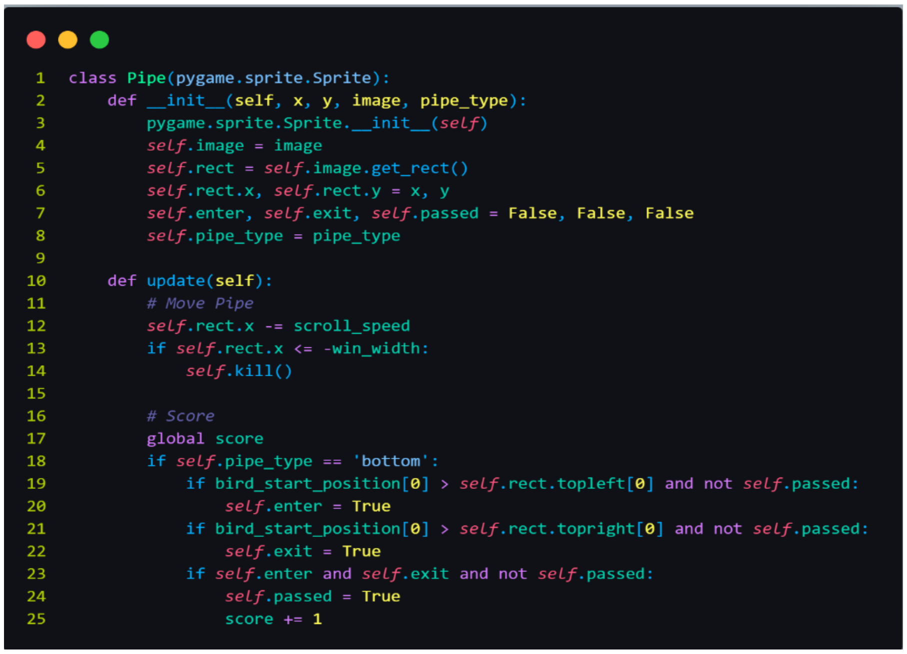
```python
class Bird(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = bird_images[0]
        self.rect = self.image.get_rect()
        self.rect.center = bird_start_position
        self.image_index = 0
        self.vel = 0
        self.flap = False
        self.alive = True

```

- The Bird class is intended for use in Pygame-based games to depict a bird figure. It has properties for controlling the bird's appearance, location, velocity, wing flapping state, and life status. This class may be used as part of a bigger game code that creates and manipulates instances of the Bird class within a game loop.



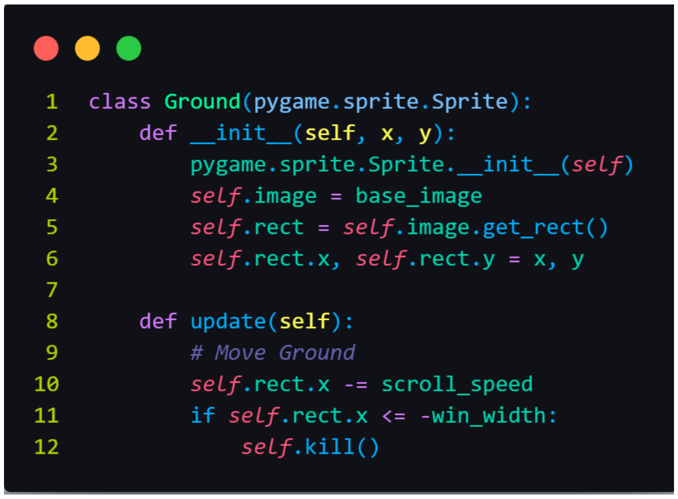
```python
def update(self, user_input):
        # Animate Bird
        if self.alive:
            self.image_index += 1
        if self.image_index >= 30:
            self.image_index = 0
        self.image = bird_images[self.image_index // 10]

        # Gravity and Flap
        self.vel += 0.5
        if self.vel > 7:
            self.vel = 7
        if self.rect.y < 500:
            self.rect.y += int(self.vel)
        if self.vel == 0:
            self.flap = False

        # Rotate Bird
        self.image = pygame.transform.rotate(self.image, self.vel * -7)

        # User Input
        if user_input[pygame.K_SPACE] and not self.flap and self.rect.y > 0 and self.alive:
            self.flap = True
            self.vel = -7
```

This code functions about the update of the entire pygame, showing of how the bird will animate during the play of the game, the gravity and the flap of the character, giving updates of how it will happen if the game is ongoing or how it will end. In summary, the code functions showing the character of a bird of how will animate by using the update function.



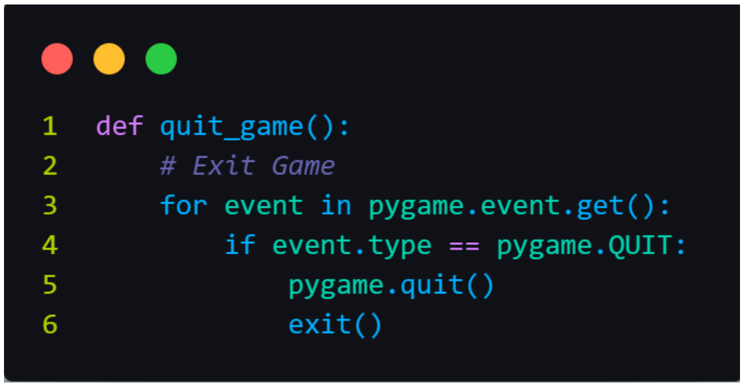
```python
class Pipe(pygame.sprite.Sprite):
    def __init__(self, x, y, image, pipe_type):
        pygame.sprite.Sprite.__init__(self)
        self.image = image
        self.rect = self.image.get_rect()
        self.rect.x, self.rect.y = x, y
        self.enter, self.exit, self.passed = False, False, False
        self.pipe_type = pipe_type

    def update(self):
        # Move Pipe
        self.rect.x -= scroll_speed
        if self.rect.x <= -win_width:
            self.kill()

        # Score
        global score
        if self.pipe_type == 'bottom':
            if bird_start_position[0] > self.rect.topleft[0] and not self.passed:
                self.enter = True
            if bird_start_position[0] > self.rect.topright[0] and not self.passed:
                self.exit = True
            if self.enter and self.exit and not self.passed:
                self.passed = True
                score += 1
```

- The Pipe class represents pipes in a side-scrolling game. The update method advances the pipes horizontally, and the scoring logic checks if the bird has gone through a bottom pipe successfully, adjusting the score appropriately.
- For the update function of the pipe class, the update function wil, show the frame by frame of a moving pipe on the screen of the game.
- And the global score will be the detection of the the pipe once a character of the game go through the pipe will detect a 1 point score in the game.



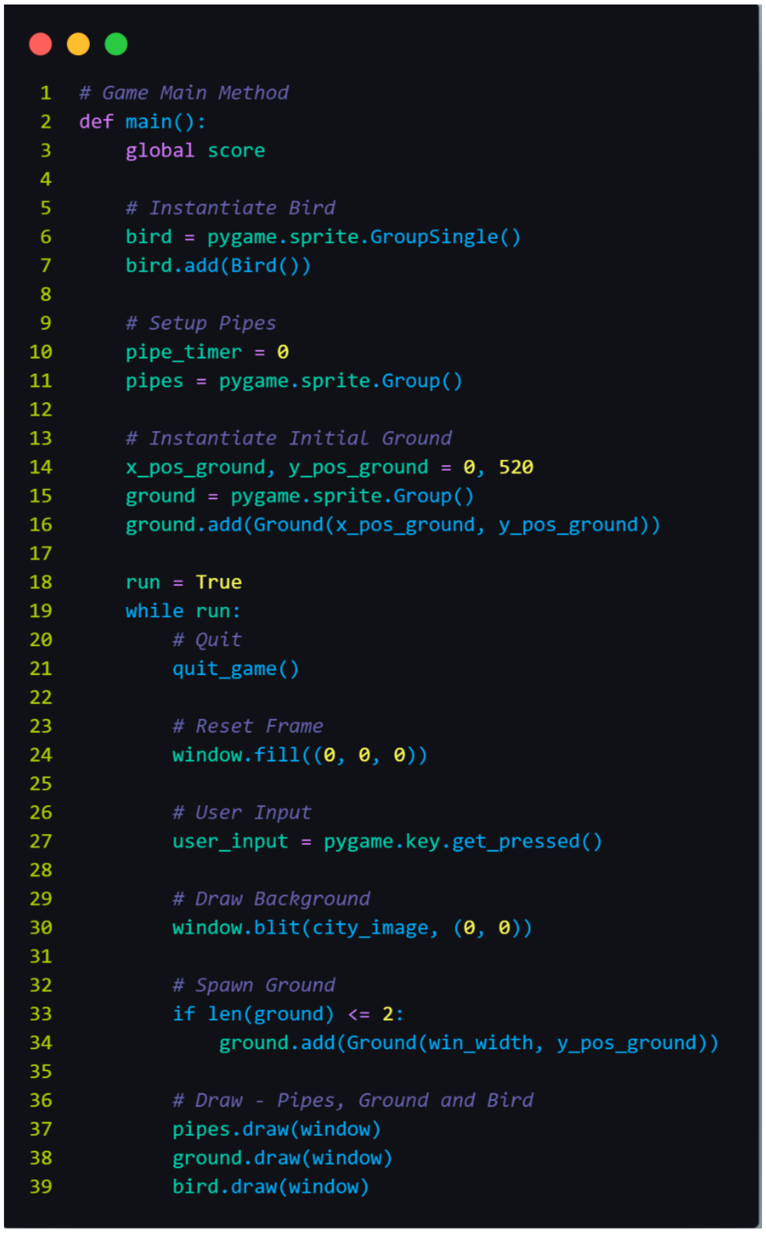
```python
class Ground(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = base_image
        self.rect = self.image.get_rect()
        self.rect.x, self.rect.y = x, y

    def update(self):
        # Move Ground
        self.rect.x -= scroll_speed
        if self.rect.x <= -win_width:
            self.kill()
```

- The ground class represents the update method that the ground image of the game will scroll or moving background in the game.
- The function of the _init_ will be the detection of the character or the bird if the bird hits the ground, the game will consider as game over.
- The update function represents the moving ground image of the game with frame by frame using the scroll_speed variable



```python
def quit_game():
    # Exit Game
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

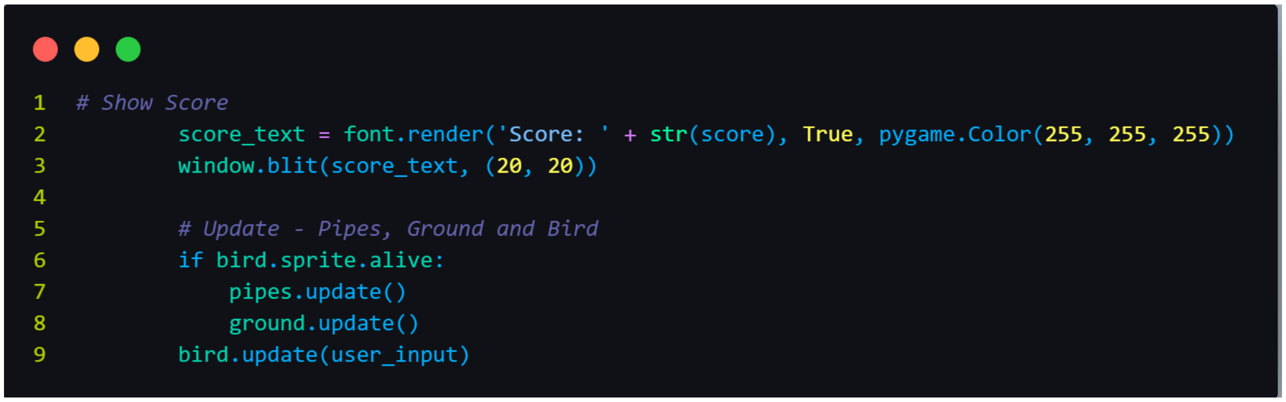
- The quit game function represents to detect to exit the game, which this code will be included in the final output of the pygame, and will be the variable if using the quit game variable in the other functions.



```python
# Game Main Method
def main():
    global score

    # Instantiate Bird
    bird = pygame.sprite.GroupSingle()
    bird.add(Bird())

    # Setup Pipes
    pipe_timer = 0
    pipes = pygame.sprite.Group()

    # Instantiate Initial Ground
    x_pos_ground, y_pos_ground = 0, 520
    ground = pygame.sprite.Group()
    ground.add(Ground(x_pos_ground, y_pos_ground))

    run = True
    while run:
        # Quit
        quit_game()

        # Reset Frame
        window.fill((0, 0, 0))

        # User Input
        user_input = pygame.key.get_pressed()

        # Draw Background
        window.blit(city_image, (0, 0))

        # Spawn Ground
        if len(ground) <= 2:
            ground.add(Ground(win_width, y_pos_ground))

        # Draw - Pipes, Ground and Bird
        pipes.draw(window)
        ground.draw(window)
        bird.draw(window)
```

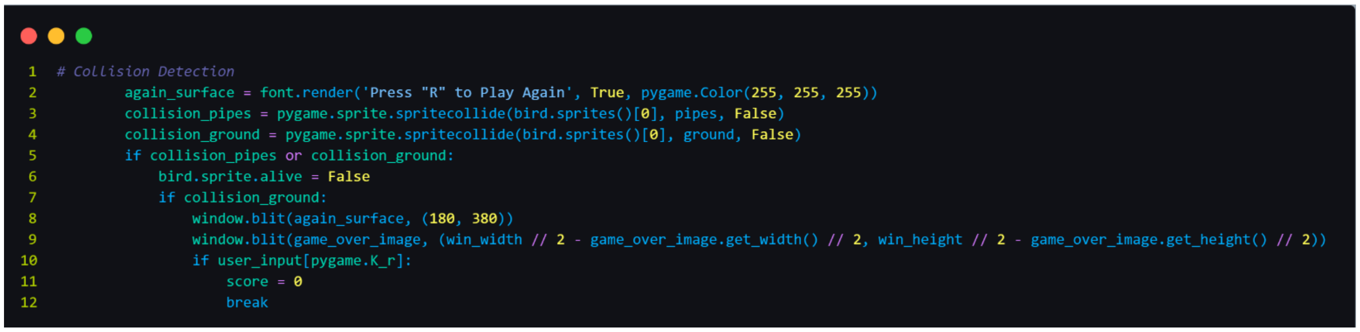
- The main function will be the overall performance in the pygame each variable will be the detection during the in game. This code shows a game's main loop, where user input is handled, game objects are updated and rendered, and the frame is refreshed on a regular basis. The game's full functionality, including collision detection, scoring, and other elements, would be implemented in other areas of the code.

- The main function will be also the update of the game, it will operate overall while the game is running for example the while run: code, we will use the other functions like the quit game when the game ends it will detect if your character detects on hitting the obstacles or the ground will considered as game over. Reset Frame will be the reset part if the game resets to its starting point. Draw background image will inserting the background image of the game, and etc. for pipes, ground and bird will the update of the game.



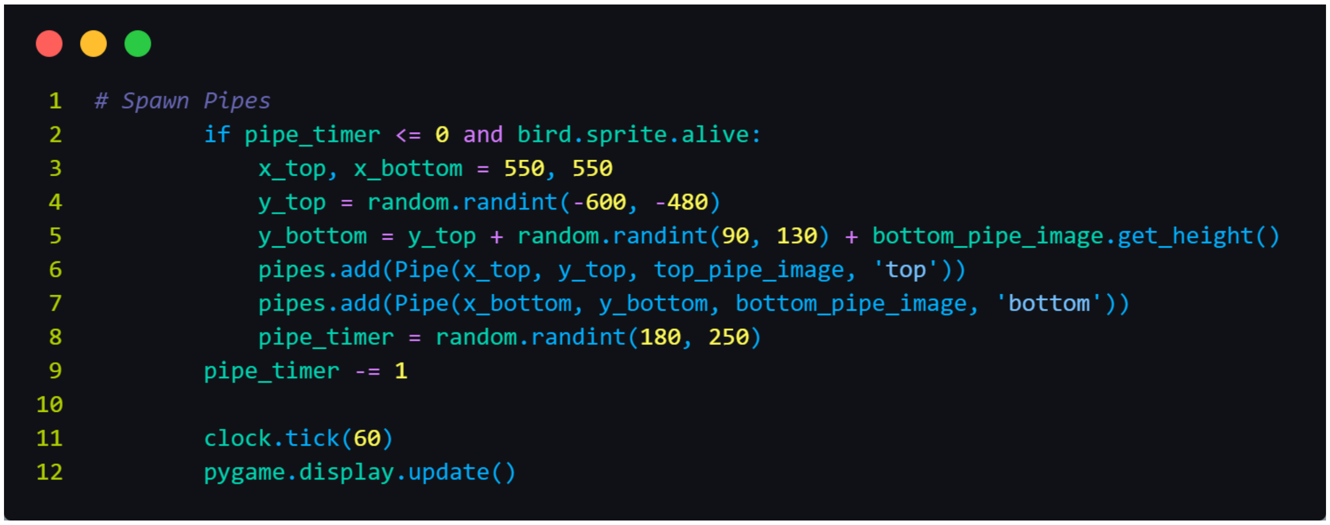
```python
# Show Score
        score_text = font.render('Score: ' + str(score), True, pygame.Color(255, 255, 255))
        window.blit(score_text, (20, 20))

        # Update - Pipes, Ground and Bird
        if bird.sprite.alive:
            pipes.update()
            ground.update()
        bird.update(user_input)
```

The show score comment will be the display of the score in the screen of the game, displaying the score text, for the pygame color will be the adjustment of the color. Window.blit will be the text position inside the window of the game, using the if statement, the score will be the detection of the pipes once the character of the bird fly through the pipes and will detect 1 point each you go through the pipe obstacles.



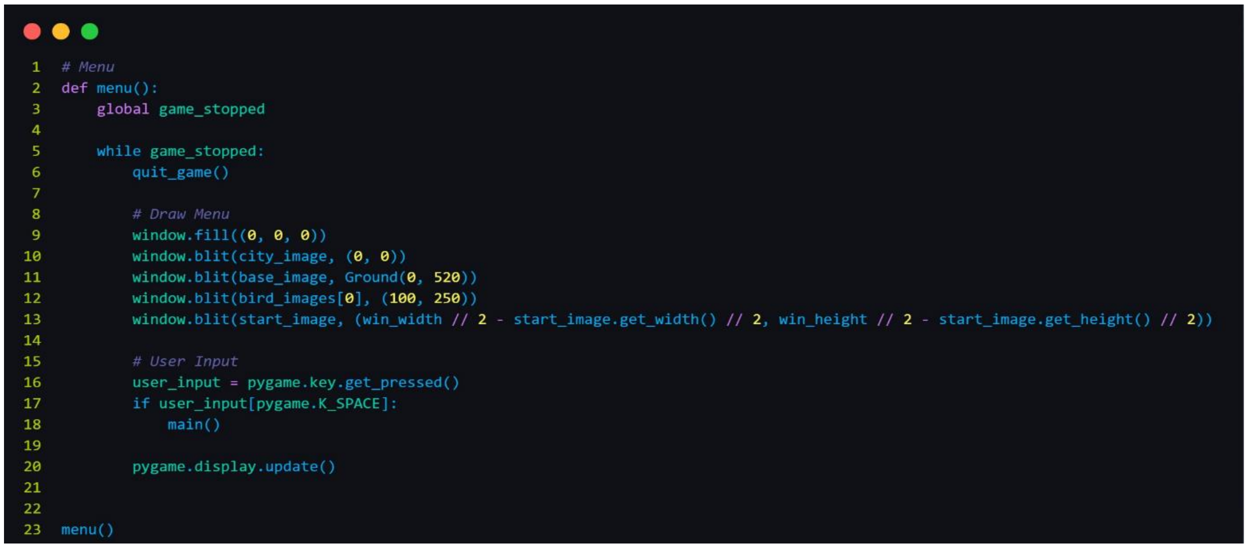
```python
# Collision Detection
        again_surface = font.render('Press "R" to Play Again', True, pygame.Color(255, 255, 255))
        collision_pipes = pygame.sprite.spritecollide(bird.sprites()[0], pipes, False)
        collision_ground = pygame.sprite.spritecollide(bird.sprites()[0], ground, False)
        if collision_pipes or collision_ground:
            bird.sprite.alive = False
            if collision_ground:
                window.blit(again_surface, (180, 380))
                window.blit(game_over_image, (win_width // 2 - game_over_image.get_width() // 2, win_height // 2 - game_over_image.get_height() // 2))
                if user_input[pygame.K_r]:
                    score = 0
                    break
```

The collision detection code represents the update of the pipe, positioning the pipes up and down by adjusting the numbers. This code segment manages what happens when a collision occurs between the bird and pipes or the ground. It sets the bird's alive status, displays a message and game over image if the collision is with the ground, and provides an option to restart the game by pressing the "R" key. The break code will be the end of the session of the game if the character considered to be as game over, pressing the R key will detect if you want to play again of the game.



```python
# Spawn Pipes
        if pipe_timer <= 0 and bird.sprite.alive:
            x_top, x_bottom = 550, 550
            y_top = random.randint(-600, -480)
            y_bottom = y_top + random.randint(90, 130) + bottom_pipe_image.get_height()
            pipes.add(Pipe(x_top, y_top, top_pipe_image, 'top'))
            pipes.add(Pipe(x_bottom, y_bottom, bottom_pipe_image, 'bottom'))
            pipe_timer = random.randint(180, 250)
        pipe_timer -= 1

        clock.tick(60)
        pygame.display.update()
```

The spawn pipes represents of pipe by pipes distance range per frame of when the pipes will appear game if the bird or the character go through the pipe obstacles. This code section is in charge of spawning a set of top and bottom pipes at regular intervals, as well as creating random y-coordinates for the pipes and resetting the timer for the next pipe spawn. It also controls the frame rate and refreshes the display.



```python
# Menu
def menu():
    global game_stopped

    while game_stopped:
        quit_game()

        # Draw Menu
        window.fill((0, 0, 0))
        window.blit(city_image, (0, 0))
        window.blit(base_image, Ground(0, 520))
        window.blit(bird_images[0], (100, 250))
        window.blit(start_image, (win_width // 2 - start_image.get_width() // 2, win_height // 2 - start_image.get_height() // 2))

        # User Input
        user_input = pygame.key.get_pressed()
        if user_input[pygame.K_SPACE]:
            main()

        pygame.display.update()


menu()
```

This code displays a basic game menu from which the user may begin playing by pressing the space bar. The menu is displayed on the screen, and the software waits for user input before proceeding to the main game loop (main()).

MY CODE (Flappy Bird)

```
import pygame

import os

from sys import exit

import random


pygame.init()

clock = pygame.time.Clock()


# Window

win_height = 720

win_width = 550

window = pygame.display.set_mode((win_width, win_height))

pygame.display.set_caption('Final Project: Flappy Bird Game (Buenafe)')


# Images

bird_images = [pygame.image.load("sprites/custombird-downflap.png"),

            pygame.image.load("sprites/custombird-midflap.png"),

            pygame.image.load("sprites/custombird-upflap.png")]

city_image = pygame.image.load("sprites/background-city.png")

base_image = pygame.image.load("sprites/base-custom.png").convert()

base_image = pygame.transform.scale2x(base_image)

top_pipe_image = pygame.image.load("sprites/pipe_top.png")

bottom_pipe_image = pygame.image.load("sprites/pipe_bottom.png")

game_over_image = pygame.image.load("sprites/gameover-custom.png")

current_size = game_over_image.get_size()

start_image = pygame.image.load("sprites/start.png")
```

```
# Game
scroll_speed = 4
bird_start_position = (100, 250)
score = 0
font = pygame.font.SysFont('Segoe', 26)
game_stopped = True


class Bird(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = bird_images[0]
        self.rect = self.image.get_rect()
        self.rect.center = bird_start_position
        self.image_index = 0
        self.vel = 0
        self.flap = False
        self.alive = True


    def update(self, user_input):
        # Animate Bird
        if self.alive:
            self.image_index += 1
        if self.image_index >= 30:
            self.image_index = 0
        self.image = bird_images[self.image_index // 10]


        # Gravity and Flap
        self.vel += 0.5
        if self.vel > 7:
            self.vel = 7
        if self.rect.y < 500:
            self.rect.y += int(self.vel)
        if self.vel == 0:
            self.flap = False

        # Rotate Bird
        self.image = pygame.transform.rotate(self.image, self.vel * -7)

        # User Input
        if user_input[pygame.K_SPACE] and not self.flap and self.rect.y > 0 and self.alive:
            self.flap = True
            self.vel = -7

class Pipe(pygame.sprite.Sprite):
    def __init__(self, x, y, image, pipe_type):
        pygame.sprite.Sprite.__init__(self)
        self.image = image
        self.rect = self.image.get_rect()
        self.rect.x, self.rect.y = x, y
        self.enter, self.exit, self.passed = False, False, False
        self.pipe_type = pipe_type

    def update(self):
        # Move Pipe
        self.rect.x -= scroll_speed
        if self.rect.x <= -win_width:
            self.kill()

        # Score
        global score
        if self.pipe_type == 'bottom':
            if bird_start_position[0] > self.rect.topleft[0] and not self.passed:
                self.enter = True
            if bird_start_position[0] > self.rect.topright[0] and not self.passed:
                self.exit = True
            if self.enter and self.exit and not self.passed:
                self.passed = True
                score += 1


class Ground(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = base_image
        self.rect = self.image.get_rect()
        self.rect.x, self.rect.y = x, y

    def update(self):
        # Move Ground
        self.rect.x -= scroll_speed
        if self.rect.x <= -win_width:
            self.kill()


def quit_game():
    # Exit Game
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()


# Game Main Method
def main():
    global score

    # Instantiate Bird
    bird = pygame.sprite.GroupSingle()
```

```
bird.add(Bird())

# Setup Pipes
pipe_timer = 0
pipes = pygame.sprite.Group()

# Instantiate Initial Ground
x_pos_ground, y_pos_ground = 0, 520
ground = pygame.sprite.Group()
ground.add(Ground(x_pos_ground, y_pos_ground))

run = True
while run:
    # Quit
    quit_game()

    # Reset Frame
    window.fill((0, 0, 0))

    # User Input
    user_input = pygame.key.get_pressed()

    # Draw Background
    window.blit(city_image, (0, 0))

    # Spawn Ground
    if len(ground) <= 2:
        ground.add(Ground(win_width, y_pos_ground))

    # Draw - Pipes, Ground and Bird
    pipes.draw(window)
    ground.draw(window)
```

```
        bird.draw(window)

        # Show Score
        score_text = font.render('Score: ' + str(score), True, pygame.Color(255, 255, 255))
        window.blit(score_text, (20, 20))

        # Update - Pipes, Ground and Bird
        if bird.sprite.alive:
            pipes.update()
            ground.update()
        bird.update(user_input)

        # Collision Detection
        again_surface = font.render('Press "R" to Play Again', True, pygame.Color(255, 255, 255))
        collision_pipes = pygame.sprite.spritecollide(bird.sprites()[0], pipes, False)
        collision_ground = pygame.sprite.spritecollide(bird.sprites()[0], ground, False)
        if collision_pipes or collision_ground:
            bird.sprite.alive = False
            if collision_ground:
                window.blit(again_surface, (180, 380))
                window.blit(game_over_image, (win_width // 2 - game_over_image.get_width() // 2, win_height // 2 -
game_over_image.get_height() // 2))
                if user_input[pygame.K_r]:
                    score = 0
                    break

        # Spawn Pipes
        if pipe_timer <= 0 and bird.sprite.alive:
            x_top, x_bottom = 550, 550
            y_top = random.randint(-600, -480)
            y_bottom = y_top + random.randint(90, 130) + bottom_pipe_image.get_height()
            pipes.add(Pipe(x_top, y_top, top_pipe_image, 'top'))
            pipes.add(Pipe(x_bottom, y_bottom, bottom_pipe_image, 'bottom'))
            pipe_timer = random.randint(180, 250)
        pipe_timer -= 1

        clock.tick(60)
        pygame.display.update()

# Menu
def menu():
    global game_stopped

    while game_stopped:
        quit_game()

        # Draw Menu
        window.fill((0, 0, 0))
        window.blit(city_image, (0, 0))
        window.blit(base_image, Ground(0, 520))
        window.blit(bird_images[0], (100, 250))
        window.blit(start_image, (win_width // 2 - start_image.get_width() // 2, win_height // 2 -
start_image.get_height() // 2))

        # User Input
        user_input = pygame.key.get_pressed()
        if user_input[pygame.K_SPACE]:
            main()

        pygame.display.update()

menu()
```