Multimedia Project Report

Flappy Bird game using pygame

Group : 8

Team Member :

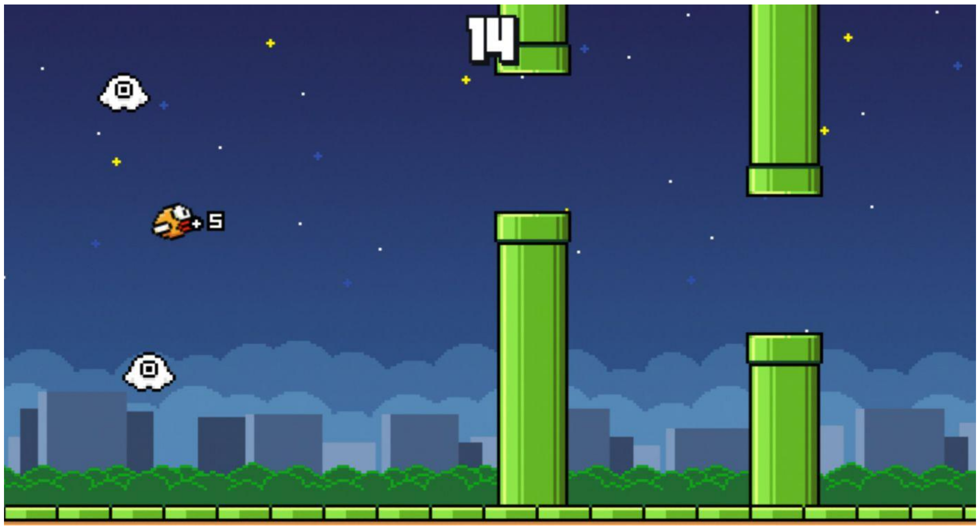
Vedant Dhoble - S20190010197
Laurel Verma - S20190010109
Pawankumar Jaiswal - S20190010069
Yash Gupta - S20190010197

Overview :

In this project we have created a Flappy Bird game using pygame. Pygame is a cross-platform set of Python modules designed for **writing video games**. It includes computer graphics and sound libraries designed to be used with the Python programming language.

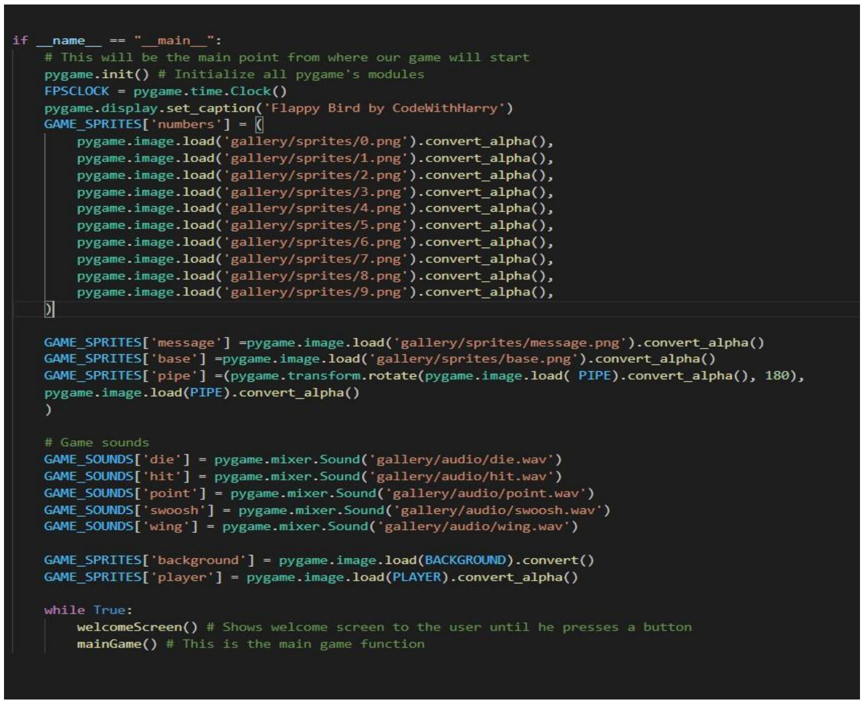
Flappy Bird is an arcade-style game in which the player **controls the bird Faby,**

which moves persistently to the right. The player is tasked with navigating Faby through pairs of pipes that have equally sized gaps placed at random heights.



Code :

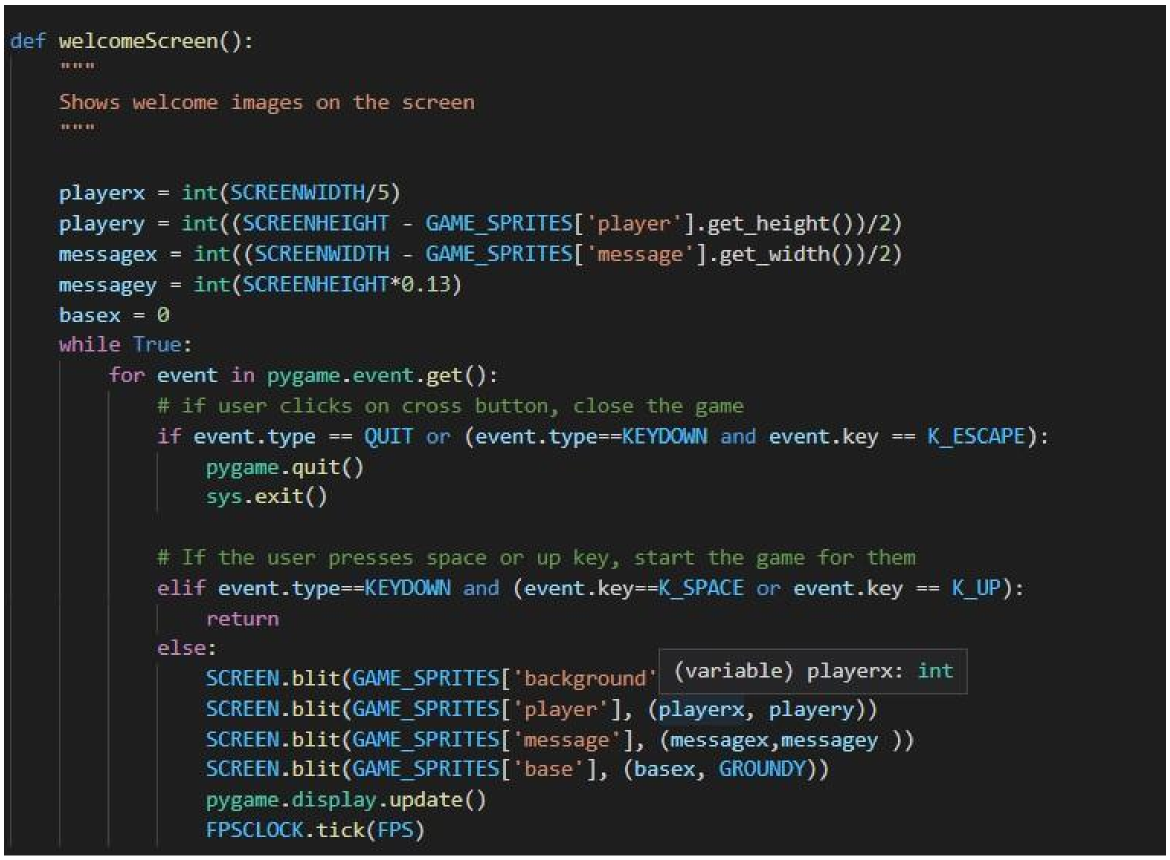
1. Main body :



```python
if __name__ == "__main__":
    # This will be the main point from where our game will start
    pygame.init() # Initialize all pygame's modules
    FPSCLOCK = pygame.time.Clock()
    pygame.display.set_caption('Flappy Bird by CodeWithHarry')
    GAME_SPRITES['numbers'] = (
        pygame.image.load('gallery/sprites/0.png').convert_alpha(),
        pygame.image.load('gallery/sprites/1.png').convert_alpha(),
        pygame.image.load('gallery/sprites/2.png').convert_alpha(),
        pygame.image.load('gallery/sprites/3.png').convert_alpha(),
        pygame.image.load('gallery/sprites/4.png').convert_alpha(),
        pygame.image.load('gallery/sprites/5.png').convert_alpha(),
        pygame.image.load('gallery/sprites/6.png').convert_alpha(),
        pygame.image.load('gallery/sprites/7.png').convert_alpha(),
        pygame.image.load('gallery/sprites/8.png').convert_alpha(),
        pygame.image.load('gallery/sprites/9.png').convert_alpha(),
    )

    GAME_SPRITES['message'] =pygame.image.load('gallery/sprites/message.png').convert_alpha()
    GAME_SPRITES['base'] =pygame.image.load('gallery/sprites/base.png').convert_alpha()
    GAME_SPRITES['pipe'] =(pygame.transform.rotate(pygame.image.load( PIPE).convert_alpha(), 180),
    pygame.image.load(PIPE).convert_alpha()
    )

    # Game sounds
    GAME_SOUNDS['die'] = pygame.mixer.Sound('gallery/audio/die.wav')
    GAME_SOUNDS['hit'] = pygame.mixer.Sound('gallery/audio/hit.wav')
    GAME_SOUNDS['point'] = pygame.mixer.Sound('gallery/audio/point.wav')
    GAME_SOUNDS['swoosh'] = pygame.mixer.Sound('gallery/audio/swoosh.wav')
    GAME_SOUNDS['wing'] = pygame.mixer.Sound('gallery/audio/wing.wav')

    GAME_SPRITES['background'] = pygame.image.load(BACKGROUND).convert()
    GAME_SPRITES['player'] = pygame.image.load(PLAYER).convert_alpha()

    while True:
        welcomeScreen() # Shows welcome screen to the user until he presses a button
        mainGame() # This is the main game function
```

A Main function, to load all different components present in the game i.e background image etc as well as to call the user defined functions. Each function has its own unique task. The code also contains some predefined global variables such as screen width, screen height, fps, images url etc.

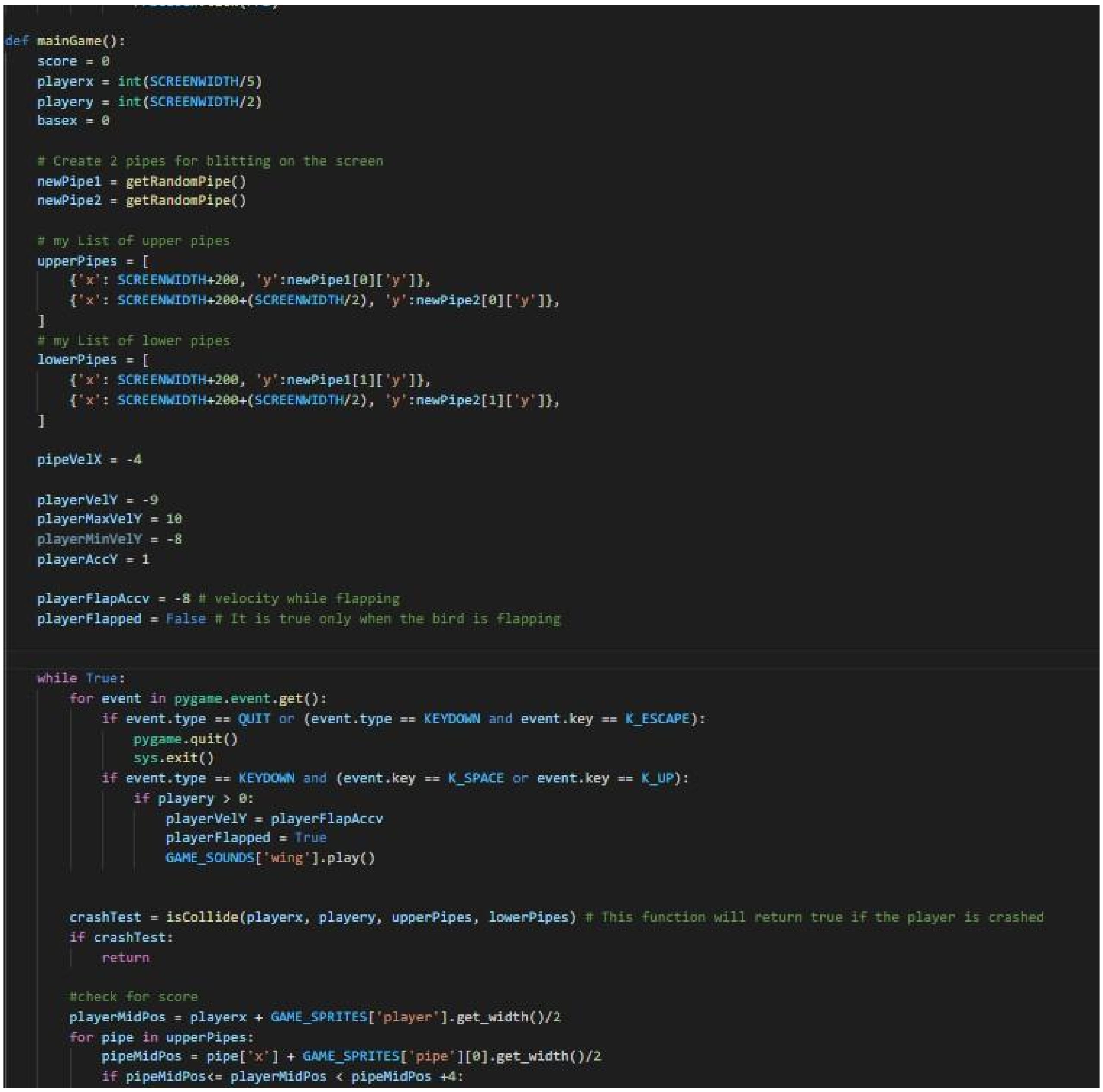
2. Welcome function :



```python
def welcomeScreen():
    """
    Shows welcome images on the screen
    """

    playerx = int(SCREENWIDTH/5)
    playery = int((SCREENHEIGHT - GAME_SPRITES['player'].get_height())/2)
    messagex = int((SCREENWIDTH - GAME_SPRITES['message'].get_width())/2)
    messagey = int(SCREENHEIGHT*0.13)
    basex = 0
    while True:
        for event in pygame.event.get():
            # if user clicks on cross button, close the game
            if event.type == QUIT or (event.type==KEYDOWN and event.key == K_ESCAPE):
                pygame.quit()
                sys.exit()

            # If the user presses space or up key, start the game for them
            elif event.type==KEYDOWN and (event.key==K_SPACE or event.key == K_UP):
                return
            else:
                SCREEN.blit(GAME_SPRITES['background' (variable) playerx: int
                SCREEN.blit(GAME_SPRITES['player'], (playerx, playery))
                SCREEN.blit(GAME_SPRITES['message'], (messagex,messagey ))
                SCREEN.blit(GAME_SPRITES['base'], (basex, GROUNDY))
                pygame.display.update()
                FPSCLOCK.tick(FPS)
```

This function is called first, whenever the user runs the code. It loads a basic screen that contains an image in background and a bird in foreground. When a user **presses space or up arrow key** the game starts and a new function is called that is the mainGame().

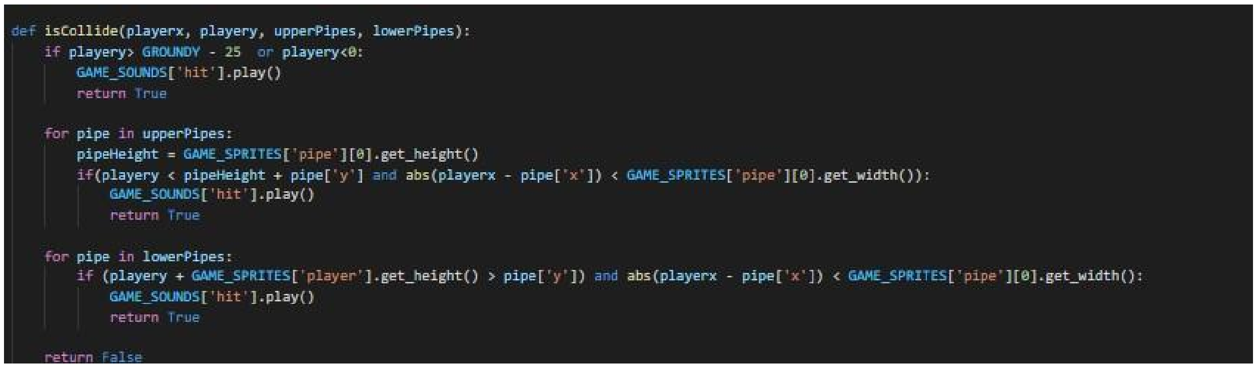
3. mainGame function :



```python
def mainGame():
    score = 0
    playerx = int(SCREENWIDTH/5)
    playery = int(SCREENWIDTH/2)
    basex = 0

    # Create 2 pipes for blitting on the screen
    newPipe1 = getRandomPipe()
    newPipe2 = getRandomPipe()

    # my List of upper pipes
    upperPipes = [
        {'x': SCREENWIDTH+200, 'y':newPipe1[0]['y']},
        {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newPipe2[0]['y']},
    ]
    # my List of lower pipes
    lowerPipes = [
        {'x': SCREENWIDTH+200, 'y':newPipe1[1]['y']},
        {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newPipe2[1]['y']},
    ]

    pipeVelX = -4

    playerVelY = -9
    playerMaxVelY = 10
    playerMinVelY = -8
    playerAccY = 1

    playerFlapAccv = -8 # velocity while flapping
    playerFlapped = False # It is true only when the bird is flapping


    while True:
        for event in pygame.event.get():
            if event.type == QUIT or (event.type == KEYDOWN and event.key == K_ESCAPE):
                pygame.quit()
                sys.exit()
            if event.type == KEYDOWN and (event.key == K_SPACE or event.key == K_UP):
                if playery > 0:
                    playerVelY = playerFlapAccv
                    playerFlapped = True
                    GAME_SOUNDS['wing'].play()


        crashTest = isCollide(playerx, playery, upperPipes, lowerPipes) # This function will return true if the player is crashed
        if crashTest:
            return

        #check for score
        playerMidPos = playerx + GAME_SPRITES['player'].get_width()/2
        for pipe in upperPipes:
            pipeMidPos = pipe['x'] + GAME_SPRITES['pipe'][0].get_width()/2
            if pipeMidPos<= playerMidPos < pipeMidPos +4:
```

This function contains all the basic operations such as generating random pipes from upside and downside, increasing the speed, counter to increment the score, calling the functions iscollide(), generaterandompipes() etc.

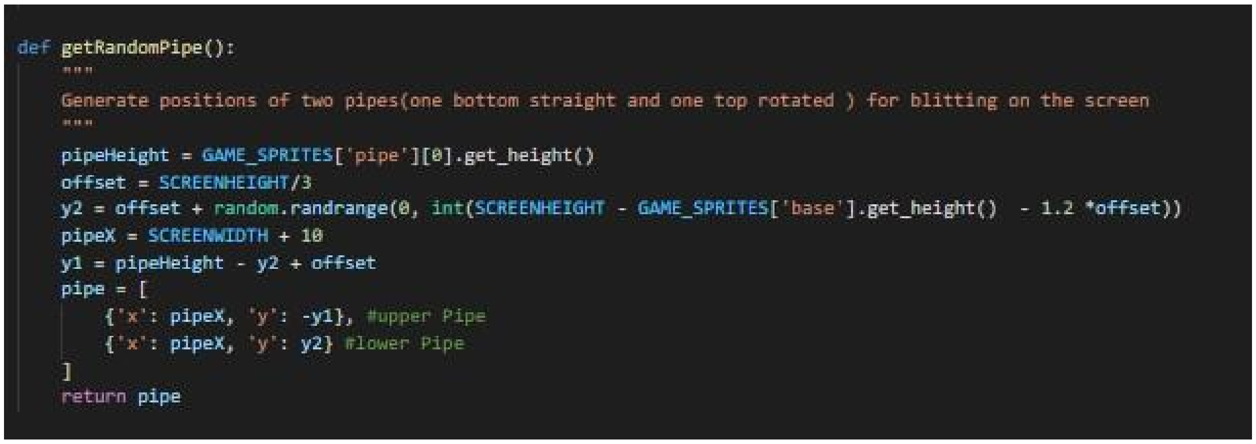
4. iscollide function :



```
def isCollide(playerx, playery, upperPipes, lowerPipes):
    if playery> GROUNDY - 25  or playery<0:
        GAME_SOUNDS['hit'].play()
        return True

    for pipe in upperPipes:
        pipeHeight = GAME_SPRITES['pipe'][0].get_height()
        if(playery < pipeHeight + pipe['y'] and abs(playerx - pipe['x']) < GAME_SPRITES['pipe'][0].get_width()):
            GAME_SOUNDS['hit'].play()
            return True

    for pipe in lowerPipes:
        if (playery + GAME_SPRITES['player'].get_height() > pipe['y']) and abs(playerx - pipe['x']) < GAME_SPRITES['pipe'][0].get_width():
            GAME_SOUNDS['hit'].play()
            return True

    return False
```

This function checks whether the bird is colliding with the upper pipe or lower pipe after every input if yes, then the function returns the value false else returns the value true.

5. getRandompipe function :



```
def getRandomPipe():
    """
    Generate positions of two pipes(one bottom straight and one top rotated ) for blitting on the screen
    """
    pipeHeight = GAME_SPRITES['pipe'][0].get_height()
    offset = SCREENHEIGHT/3
    y2 = offset + random.randrange(0, int(SCREENHEIGHT - GAME_SPRITES['base'].get_height()  - 1.2 *offset))
    pipeX = SCREENWIDTH + 10
    y1 = pipeHeight - y2 + offset
    pipe = [
        {'x': pipeX, 'y': -y1}, #upper Pipe
        {'x': pipeX, 'y': y2} #lower Pipe
    ]
    return pipe
```

This function generates random pipes from upside as well as from down the side of the screen with different heights but equal gap between them and returns the pipe.

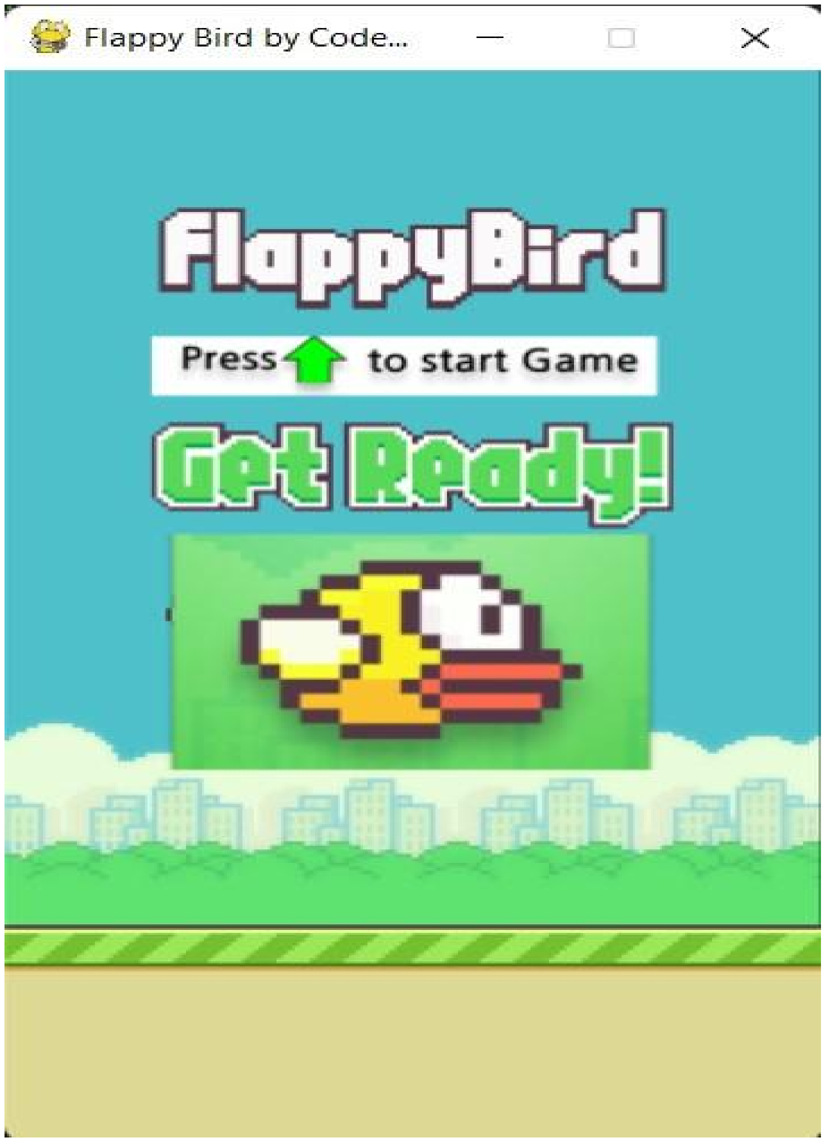
FULL CODE :

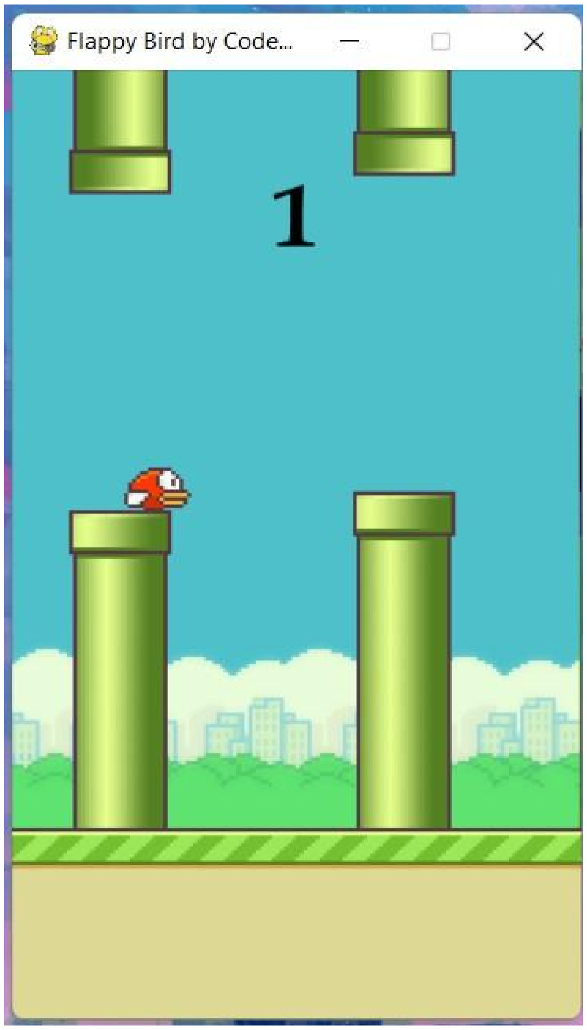
```
import random # For generating random numbers
import sys # We will use sys.exit to exit the program
import pygame
from pygame.locals import * # Basic pygame imports

# Global Variables for the game
FPS = 32
SCREENWIDTH = 289
SCREENHEIGHT = 511
SCREEN = pygame.display.set_mode((SCREENWIDTH, SCREENHEIGHT))
GROUNDY = SCREENHEIGHT * 0.8
GAME_SPRITES = {}
GAME_SOUNDS = {}
PLAYER = 'gallery/sprites/bird.png'
BACKGROUND = 'gallery/sprites/background.png'
PIPE = 'gallery/sprites/pipe.png'

def welcomeScreen():
    """
    Shows welcome images on the screen
    """

    playerx = int(SCREENWIDTH/5)
    playery = int((SCREENHEIGHT - GAME_SPRITES['player'].get_height())/2)
    messagex = int((SCREENWIDTH - GAME_SPRITES['message'].get_width())/2)
    messagey = int(SCREENHEIGHT*0.13)
    basex = 0
    while True:
        for event in pygame.event.get():
            # if user clicks on cross button, close the game
            if event.type == QUIT or (event.type==KEYDOWN and event.key ==
K_ESCAPE):
                pygame.quit()
                sys.exit()

            # If the user presses space or up key, start the game for them
            elif event.type==KEYDOWN and (event.key==K_SPACE or event.key
== K_UP):
```

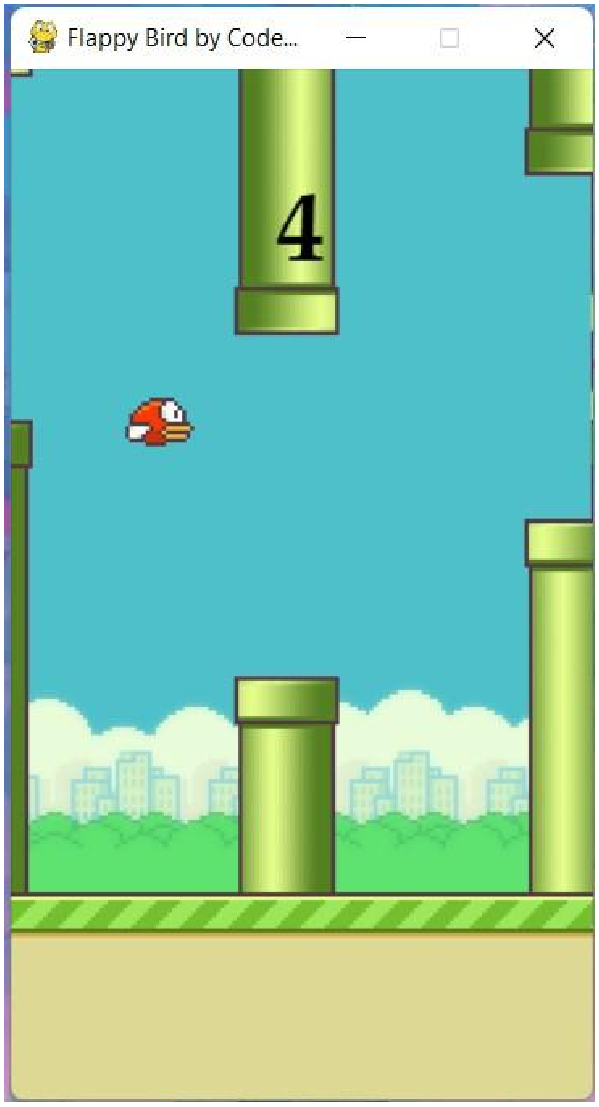
```
                return
            else:
                SCREEN.blit(GAME_SPRITES['background'], (0, 0))
                SCREEN.blit(GAME_SPRITES['player'], (playerx, playery))
                SCREEN.blit(GAME_SPRITES['message'], (messagex,messagey ))
                SCREEN.blit(GAME_SPRITES['base'], (basex, GROUNDY))
                pygame.display.update()
                FPSCLOCK.tick(FPS)

def mainGame():
    score = 0
    playerx = int(SCREENWIDTH/5)
    playery = int(SCREENWIDTH/2)
    basex = 0

    # Create 2 pipes for blitting on the screen
    newPipe1 = getRandomPipe()
    newPipe2 = getRandomPipe()

    # my List of upper pipes
    upperPipes = [
        {'x': SCREENWIDTH+200, 'y':newPipe1[0]['y']},
        {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newPipe2[0]['y']},
    ]
    # my List of lower pipes
    lowerPipes = [
        {'x': SCREENWIDTH+200, 'y':newPipe1[1]['y']},
        {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newPipe2[1]['y']},
    ]

    pipeVelX = -4

    playerVelY = -9
    playerMaxVelY = 10
    playerMinVelY = -8
    playerAccY = 1

    playerFlapAccv = -8 # velocity while flapping
    playerFlapped = False # It is true only when the bird is flapping
    while True:
        for event in pygame.event.get():
            if event.type == QUIT or (event.type == KEYDOWN and event.key
== K_ESCAPE):
                pygame.quit()
                sys.exit()
            if event.type == KEYDOWN and (event.key == K_SPACE or
event.key == K_UP):
                if playery > 0:
                    playerVelY = playerFlapAccv
                    playerFlapped = True
                    GAME_SOUNDS['wing'].play()


        crashTest = isCollide(playerx, playery, upperPipes, lowerPipes) #
This function will return true if the player is crashed
        if crashTest:
            return

        #check for score
        playerMidPos = playerx + GAME_SPRITES['player'].get_width()/2
        for pipe in upperPipes:
            pipeMidPos = pipe['x'] + GAME_SPRITES['pipe'][0].get_width()/2
            if pipeMidPos<= playerMidPos < pipeMidPos +4:
                score +=1
                print(f"Your score is {score}")
                GAME_SOUNDS['point'].play()



        if playerVelY <playerMaxVelY and not playerFlapped:
            playerVelY += playerAccY

        if playerFlapped:
            playerFlapped = False
        playerHeight = GAME_SPRITES['player'].get_height()
        playery = playery + min(playerVelY, GROUNDY - playery -
playerHeight)

        # move pipes to the left
        for upperPipe , lowerPipe in zip(upperPipes, lowerPipes):
            upperPipe['x'] += pipeVelX
            lowerPipe['x'] += pipeVelX

        # Add a new pipe when the first is about to cross the leftmost
part of the screen
        if 0<upperPipes[0]['x']<5:
            newpipe = getRandomPipe()
            upperPipes.append(newpipe[0])
            lowerPipes.append(newpipe[1])

        # if the pipe is out of the screen, remove it
        if upperPipes[0]['x'] < -GAME_SPRITES['pipe'][0].get_width():
            upperPipes.pop(0)
            lowerPipes.pop(0)

        # Lets blit our sprites now
        SCREEN.blit(GAME_SPRITES['background'], (0, 0))
        for upperPipe, lowerPipe in zip(upperPipes, lowerPipes):
            SCREEN.blit(GAME_SPRITES['pipe'][0], (upperPipe['x'],
upperPipe['y']))
            SCREEN.blit(GAME_SPRITES['pipe'][1], (lowerPipe['x'],
lowerPipe['y']))

        SCREEN.blit(GAME_SPRITES['base'], (basex, GROUNDY))
        SCREEN.blit(GAME_SPRITES['player'], (playerx, playery))
        myDigits = [int(x) for x in list(str(score))]
        width = 0
        for digit in myDigits:
            width += GAME_SPRITES['numbers'][digit].get_width()
        Xoffset = (SCREENWIDTH - width)/2

        for digit in myDigits:
            SCREEN.blit(GAME_SPRITES['numbers'][digit], (Xoffset,
SCREENHEIGHT*0.12))
            Xoffset += GAME_SPRITES['numbers'][digit].get_width()
        pygame.display.update()
        FPSCLOCK.tick(FPS)

def isCollide(playerx, playery, upperPipes, lowerPipes):
```

```python
    if playery> GROUNDY - 25  or playery<0:
        GAME_SOUNDS['hit'].play()
        return True

    for pipe in upperPipes:
        pipeHeight = GAME_SPRITES['pipe'][0].get_height()
        if(playery < pipeHeight + pipe['y'] and abs(playerx - pipe['x']) <
GAME_SPRITES['pipe'][0].get_width()):
            GAME_SOUNDS['hit'].play()
            return True

    for pipe in lowerPipes:
        if (playery + GAME_SPRITES['player'].get_height() > pipe['y']) and
abs(playerx - pipe['x']) < GAME_SPRITES['pipe'][0].get_width():
            GAME_SOUNDS['hit'].play()
            return True

    return False

def getRandomPipe():
    """
    Generate positions of two pipes(one bottom straight and one top
rotated ) for blitting on the screen
    """
    pipeHeight = GAME_SPRITES['pipe'][0].get_height()
    offset = SCREENHEIGHT/3
    y2 = offset + random.randrange(0, int(SCREENHEIGHT -
GAME_SPRITES['base'].get_height()  - 1.2 *offset))
    pipeX = SCREENWIDTH + 10
    y1 = pipeHeight - y2 + offset
    pipe = [
        {'x': pipeX, 'y': -y1}, #upper Pipe
        {'x': pipeX, 'y': y2} #lower Pipe
    ]
    return pipe

if __name__ == "__main__":
    # This will be the main point from where our game will start
    pygame.init() # Initialize all pygame's modules
    FPSCLOCK = pygame.time.Clock()
    pygame.display.set_caption('Flappy Bird by Group 8')
    GAME_SPRITES['numbers'] = (
        pygame.image.load('gallery/sprites/0.png').convert_alpha(),
        pygame.image.load('gallery/sprites/1.png').convert_alpha(),
        pygame.image.load('gallery/sprites/2.png').convert_alpha(),
        pygame.image.load('gallery/sprites/3.png').convert_alpha(),
        pygame.image.load('gallery/sprites/4.png').convert_alpha(),
        pygame.image.load('gallery/sprites/5.png').convert_alpha(),
        pygame.image.load('gallery/sprites/6.png').convert_alpha(),
        pygame.image.load('gallery/sprites/7.png').convert_alpha(),
        pygame.image.load('gallery/sprites/8.png').convert_alpha(),
        pygame.image.load('gallery/sprites/9.png').convert_alpha(),
    )

    GAME_SPRITES['message']
=pygame.image.load('gallery/sprites/message.png').convert_alpha()
    GAME_SPRITES['base']
=pygame.image.load('gallery/sprites/base.png').convert_alpha()
    GAME_SPRITES['pipe'] =(pygame.transform.rotate(pygame.image.load(
PIPE).convert_alpha(), 180),
    pygame.image.load(PIPE).convert_alpha()
    )

    # Game sounds
    GAME_SOUNDS['die'] = pygame.mixer.Sound('gallery/audio/die.wav')
    GAME_SOUNDS['hit'] = pygame.mixer.Sound('gallery/audio/hit.wav')
    GAME_SOUNDS['point'] = pygame.mixer.Sound('gallery/audio/point.wav')
    GAME_SOUNDS['swoosh'] = pygame.mixer.Sound('gallery/audio/swoosh.wav')
    GAME_SOUNDS['wing'] = pygame.mixer.Sound('gallery/audio/wing.wav')

    GAME_SPRITES['background'] = pygame.image.load(BACKGROUND).convert()
    GAME_SPRITES['player'] = pygame.image.load(PLAYER).convert_alpha()

    while True:
        welcomeScreen() # Shows welcome screen to the user until he
presses a button
        mainGame() # This is the main game function
```

ScreenShots :







THANK YOU