GLA University, mathura

Academic year 2020-21

The Title of Project

Unicorn Flappy
Bird Game

A Project Submitted
in Partial Fulfillment of the
Requirements for the Degree of
Bachelor In Technology
in
Computer science

by
Vishwa Pratap singh
(K/181500815)
Pranjal Jain(K/181500477)
Prashant Lodhi(I/181500490)
AbhishekChaudhary
(L/181500015)
Abhishek Tiwari(K/181500021)

COMPUTER SCIENCE ENGINEERING AND
TECHNOLOGY GLA UNIVERSITY, MATHURA, INDIA

October, 2020

Acknowledgement :-

I wish to express my heartfelt gratitude to the all people who have played a crucial role in this project, without their active cooperation the preparation of this project could not have been completed within the specified time limit.

I am thankful to our Respected Project supervisor ,Miss Priya Agrawal , for motivating us to complete this project with complete focus and attention.

Certification's :-

Certificate of Completion

*This is to certify that **Vishwa pratap chauhan** successfully completed 7 total hours of **Python 3 Adventures : Learn Python 3 in Fun way** online course on Nov. 25, 2020*

Ashwin Pajankar • 50,000+ Students Worldwide

Ashwin Pajankar • 50,000+ Students Worldwide, Instructor

&

Udemy

Certificate no: UC-16955f56-1283-410a-a72c-de3c527a4e72
Certificate url: ude.my/UC-16955f56-1283-410a-a72c-de3c527a4e72
Version 3

#BeAble

Certificate of Completion

*This is to certify that **Abhishek Chaudhary** successfully completed 2.5 total hours of **Python Game Development using Pygame and Python 3** online course on Nov. 25, 2020*

Attreya Bhatt

Attreya Bhatt, Instructor

&

Udemy

Certificate no: UC-398e37e8-b0ed-4b42-9284-f65df0d3ad4e
Certificate url: ude.my/UC-398e37e8-b0ed-4b42-9284-f65df0d3ad4e
Version 3

#BeAble

Certificate of Completion

*This is to certify that **Prashant Lodhi** successfully completed 2.5 total hours of **Python Game Development using Pygame and Python 3** online course on Nov. 25, 2020*

Attreya Bhatt

Attreya Bhatt, Instructor

&

Udemy

Certificate no: UC-3fdc97db-9c7b-4973-be70-86a98b270ac6
Certificate url: ude.my/UC-3fdc97db-9c7b-4973-be70-86a98b270ac6
Version 3

#BeAble

CONTENT

Introduction

Flappy bird game is developed in Python and It’s a desktop application. The game was designed and built by **Dong Nguyen** a developer who lives in vietnam. Flappy bird is a side scroller game where the player controls a bird, attempting the flying between columns of green pipes. The bird will be flying until it collisions with a pipe or it fall on ground. It’s a simple game of infinite level type. It’s a challenging game for all.

Why game as a project

Video games are not just any computer software which are made to benefit user’s daily life, games are rather made for user’s entertainment purpose, so more than anything we need to pay attention to what the user wants from the game, how to make it more entertaining, just making any game will not do, that is why it’s more challenging because I always have to carefully consider if I’m making developing it correctly to entertain users.

I also have to invest a lot of time on the proper game designing to make it visually accepted. And to add that game requires a lot of scripts. The scripts are like pieces of a puzzle which you need to put all of them together to make it work. Thus I think game is a perfect project to prove myself as a CSE student.

Project Description

Story

We choose this game for our First ever mini project. Actually the game is entertaining for anybody and in leisure time we can spend our time nicely by playing game. The flappy bird game is implemented for only desktop.

Figure 1 – is the start screen of flappy bird. The Welcome screen with msg ‘get ready’ is shown on the screen. The bird is also Displayed on the background.

Figure 1

Figure 2 -shows the screen when the game is on. The 2 pillars are displayed on the screen, and so is the score, on top of the background or the pillar.(instead of title)

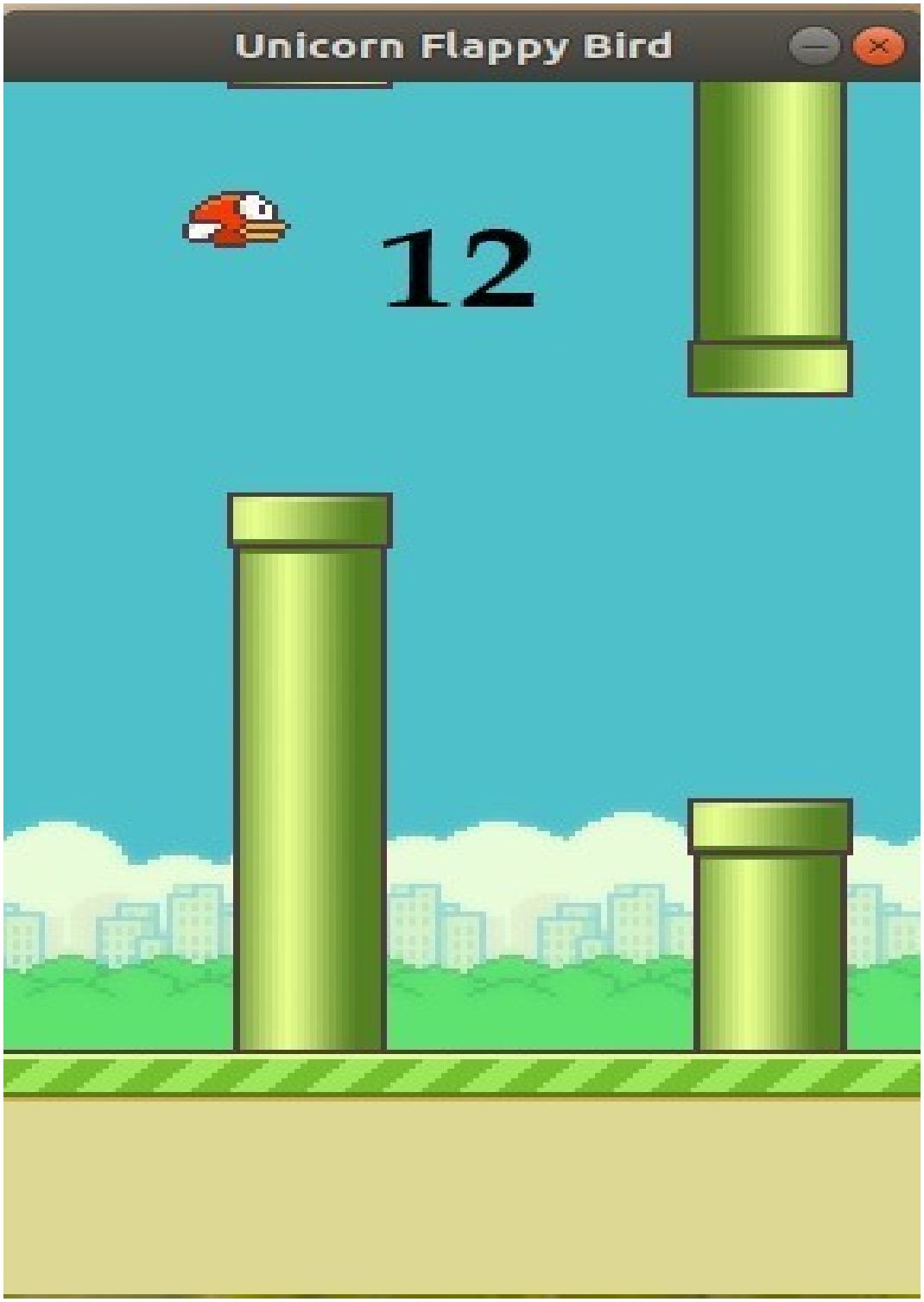


Figure 2

Requirements

A requirement is a singular documented physical or functional need that a particular design, product or process aims to safety. It can be divided into functional requirements and non-functional requirements.

- ✔ **Functional-** 2D animation , objectives selection, moving wall, collision detection, moving background etc.

- ✔ **Non-Functional –** We can keep the bird playing by pressing mouse, space bar, page up key and move it in the space of pipes.

2D Animation

Animation is a complex subject in game programming. Animation is rapid display of sequence of images which creates an illusion of movement. Python games are expected to run on multiple operating systems with different hardware specifications.

Objective selection

We create a bird object which is flying until any collision occured and the bird is flying in the wall objectives which are begin from the top and bottom of the screen.

Moving Wall

The wall is moving on and it will come randomly in size and distances.the bird is flying in the middle of the wall.

Collision Detection

When the bird touches anywhere on the wall it cause a collision. Collision detection is one of the important task of the game .If the bird touch any wall(pipes) the game will end.

Moving background

The picture used as a background image is moving on analogously. We used two same images which are coming one after another regulary.

Score counting

Score counting is the interesting for user. By the score the player knows his/her performance. If the bird cross the pipe without the collision or not fall in ground his/her score incremented by 1.

Tools

- **Language:** Python
- **IDE:** An IDE (Integrated Development Environment) contains s coe editor, a compiler or interpreter, and a debugger, accessed through a single graphical user interchange (GUI). Our IDE is **VS Code** (Visual Studio Code).

Why we are using VS code instead of any other IDE's

At its heart, Visual Studio Code features a lightning fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps us to be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings let you navigate your code with ease.

For serious coding, we'll often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code refactoring.

And when the coding gets tough, the tough get debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so we can step through source code, inspect variables, view call stacks, and execute commands in the console.

VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so we can work with source control without leaving the editor including viewing pending changes diffs.

Information

➔ **Figure A** shows welcome images on the screen and checks whether the user wants to play further more or want to quit the game.

```
main.py > ...

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)
```

Figure A

➔ **Figure B** shows the working of the sound handling in our code.

```
# 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')
```

Figure B

➔**Figure C** generates positions of two pipes(one bottom straight and one top rotated) for blitting on the screen.

```
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
```

Figure C

➔**Figure D** we declares the global variables for the game.

```
# 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'
```

Figure D

➔**Figure E** shows all the pygame imports needed to run the game.

```
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
```

Future Direction

We will add more features to the game and will change the bird and background scenery according to the user's choice and also trying to make it a multiplayer game. The status and the history will be saved and we show a graph where user can see his total performance whether it inreasing or decreasing.

Project Deliverables

- Powerpoint microsoft open XML.
- Project report
- Source code
- PDF

Summary

We choice the project to gain the proper knowledge to make desktop application. It increased our knowledge for Object Oriented Language (Python).Getting experience with python GUI. The important issues is, it’s a game application and it will be recreation for all.

Source code

```
import random
import sys
import pygame
from pygame.locals import *
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():
    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 event.type == QUIT or (event.type==KEYDOWN and event.key ==
K_ESCAPE):
            pygame.quit()
            sys.exit()
              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

    newPipe1 = getRandomPipe()
    newPipe2 = getRandomPipe()

    upperPipes = [
        {'x': SCREENWIDTH+200, 'y':newPipe1[0]['y']},
        {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newPipe2[0]['y']},
    ]
    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
```

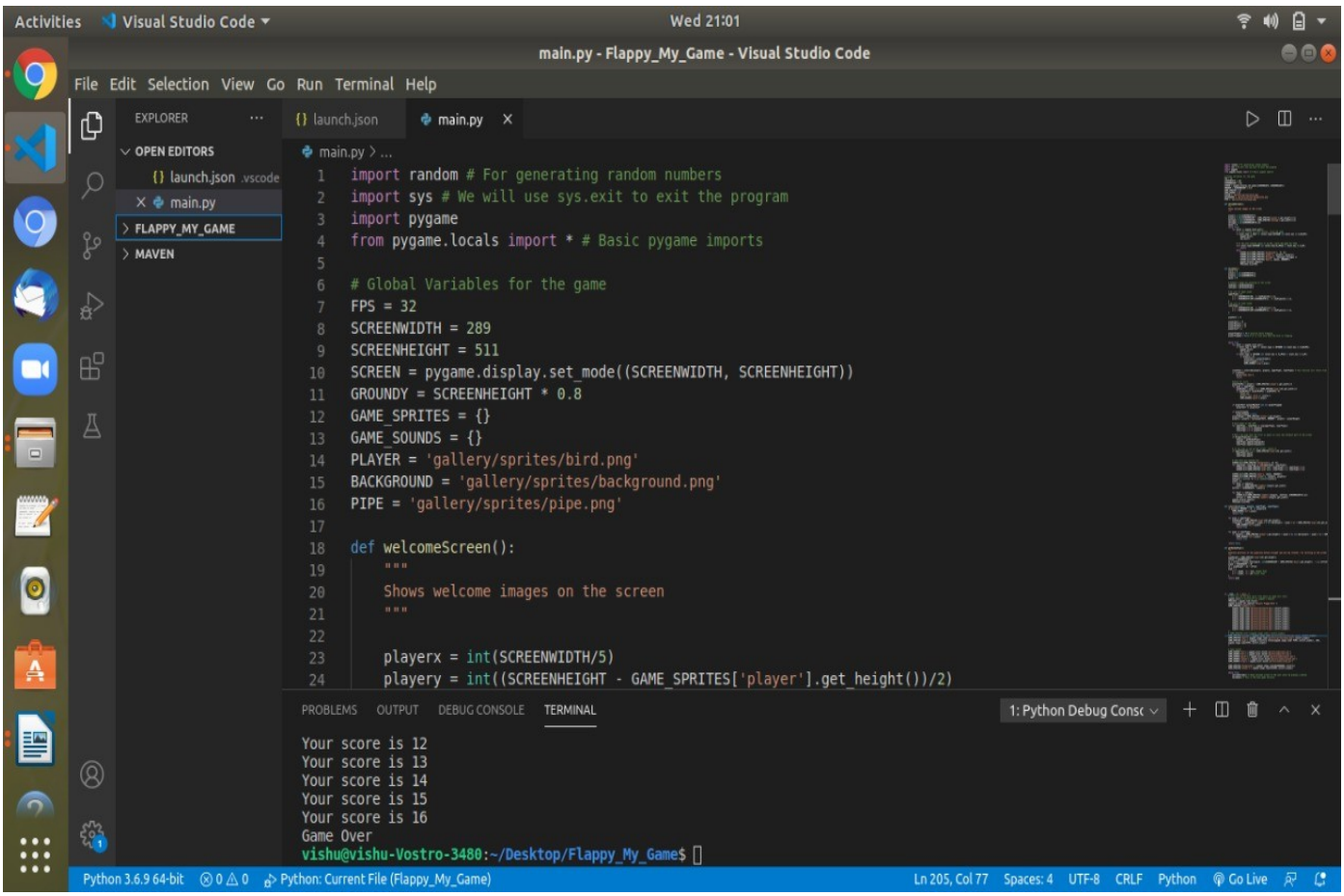
```
    playerFlapAccv = -8
    playerFlapped = False

    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

        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)
```

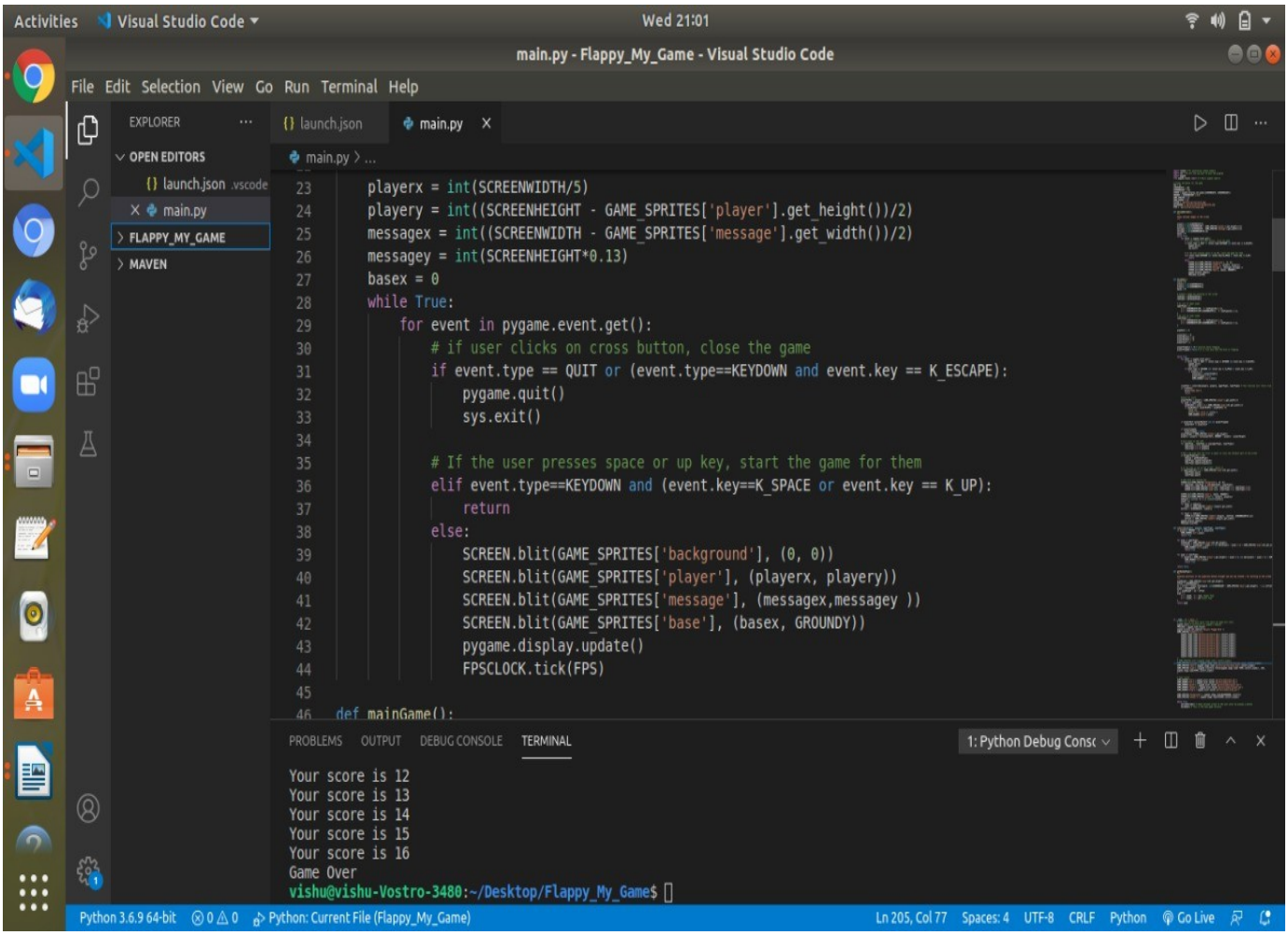
```
        for upperPipe , lowerPipe in zip(upperPipes, lowerPipes):
            upperPipe['x'] += pipeVelX
            lowerPipe['x'] += pipeVelX


        if 0<upperPipes[0]['x']<5:
            newpipe = getRandomPipe()
            upperPipes.append(newpipe[0])
            lowerPipes.append(newpipe[1])


        if upperPipes[0]['x'] < -GAME_SPRITES['pipe'][0].get_width():
            upperPipes.pop(0)
            lowerPipes.pop(0)


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

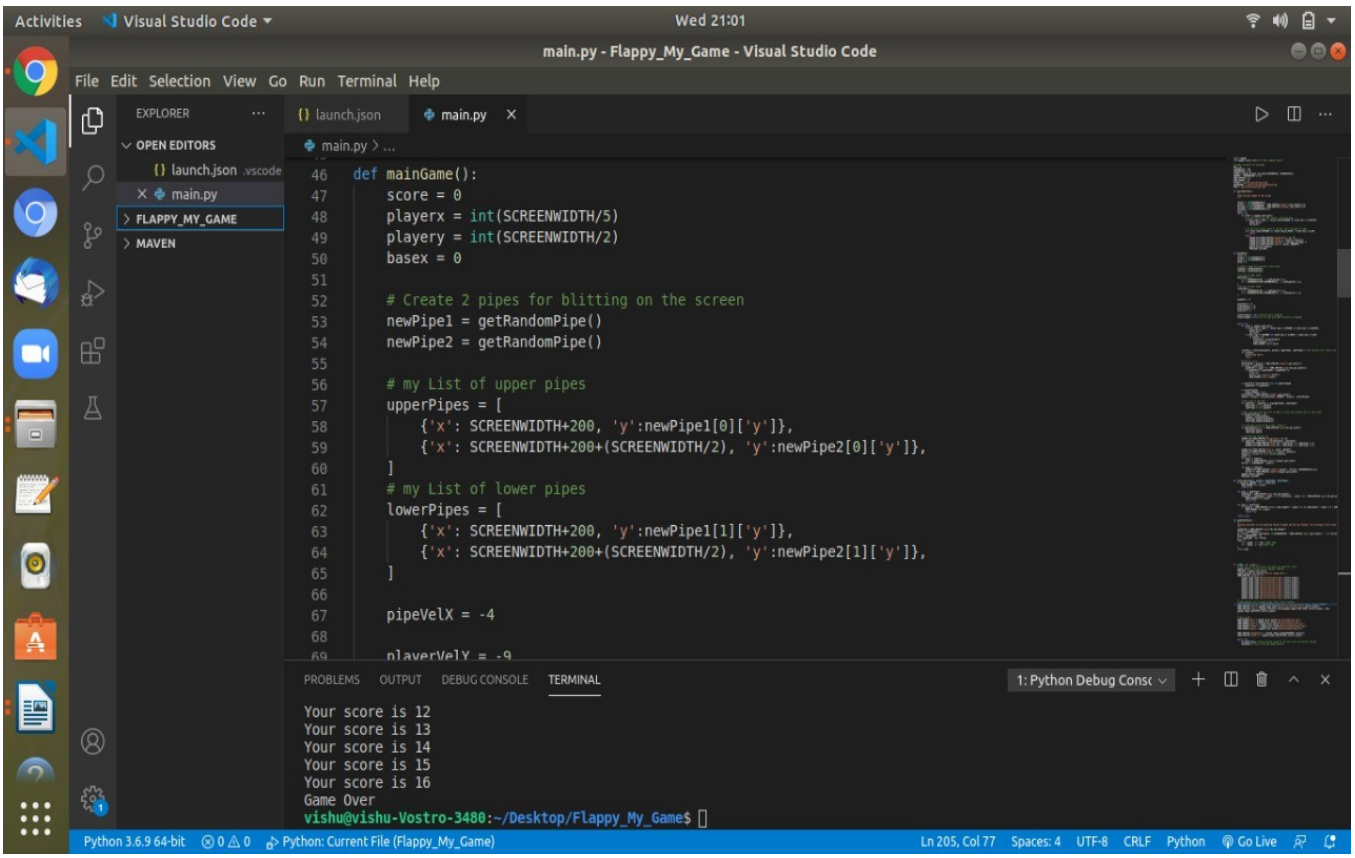
```
    for pipe in upperPipes:
        pipeHeight = GAME_SPRITES['pipe'][0].get_height()
                if(playery < pipeHeight + pipe['y'] and abs(playerx - pipe['x']) <
GAME_SPRITES['pipe'][0].get_width()):
            GAME_SOUNDS['hit'].play()
            return True

    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 GLAian')
    GAME_SPRITES['numbers'] = (
```

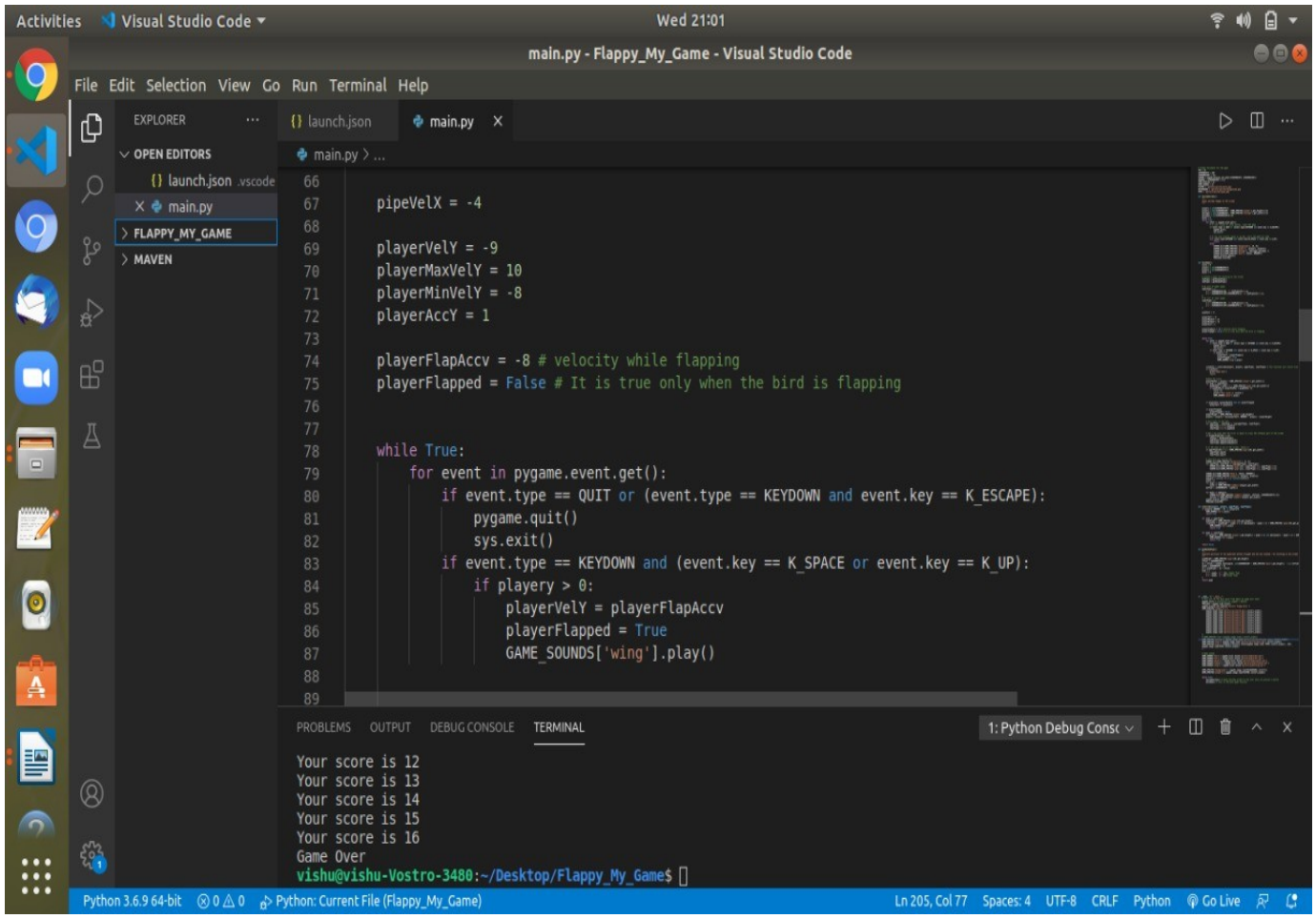
```
        pygame.image.load('gallery/sprites/0.png').convert_alpha(),
        pygame.image.load('gallery/sprites/1.png').convert_alpha(),
        pygame.image.load('gallery/sprites/2.png').convert_alpha(),
        pygame.image.load('gallery/sprites/3.png').convert_alpha(),
        pygame.image.load('gallery/sprites/4.png').convert_alpha(),
        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:
    welcome Screen()  # Shows welcome screen to the user until he presses a button
    mainGame() #This is the main game function
```

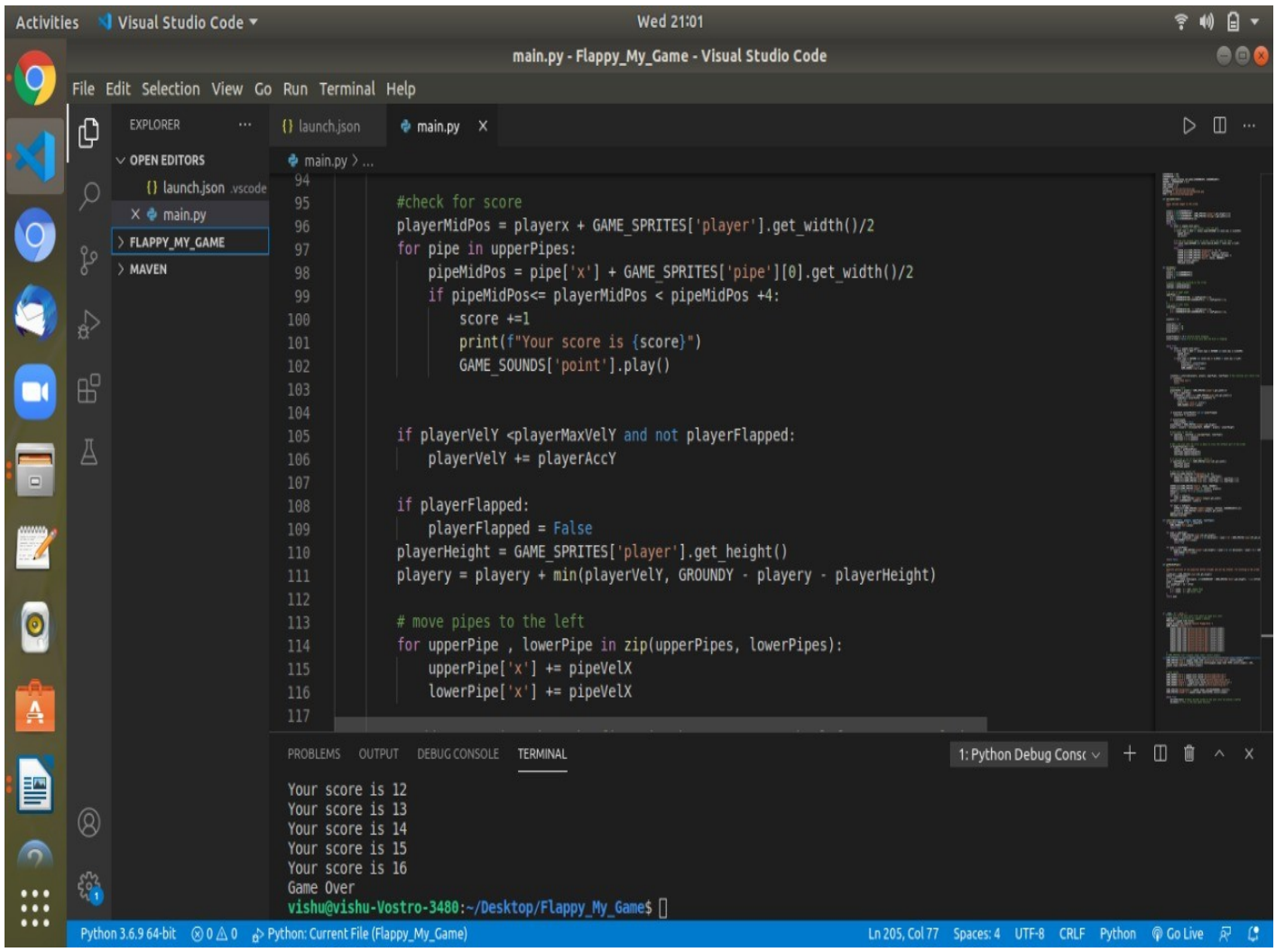
Some Screenshots

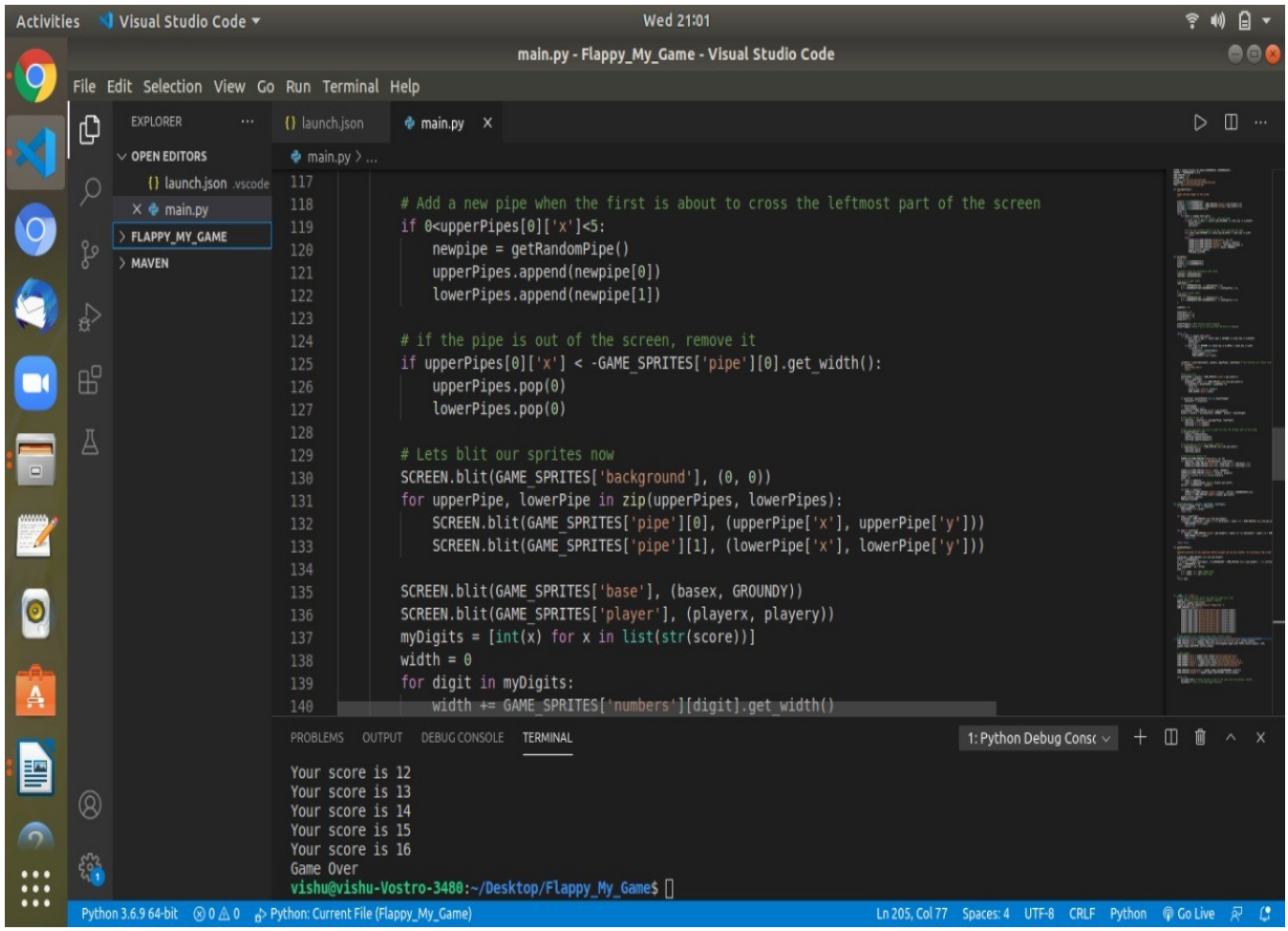


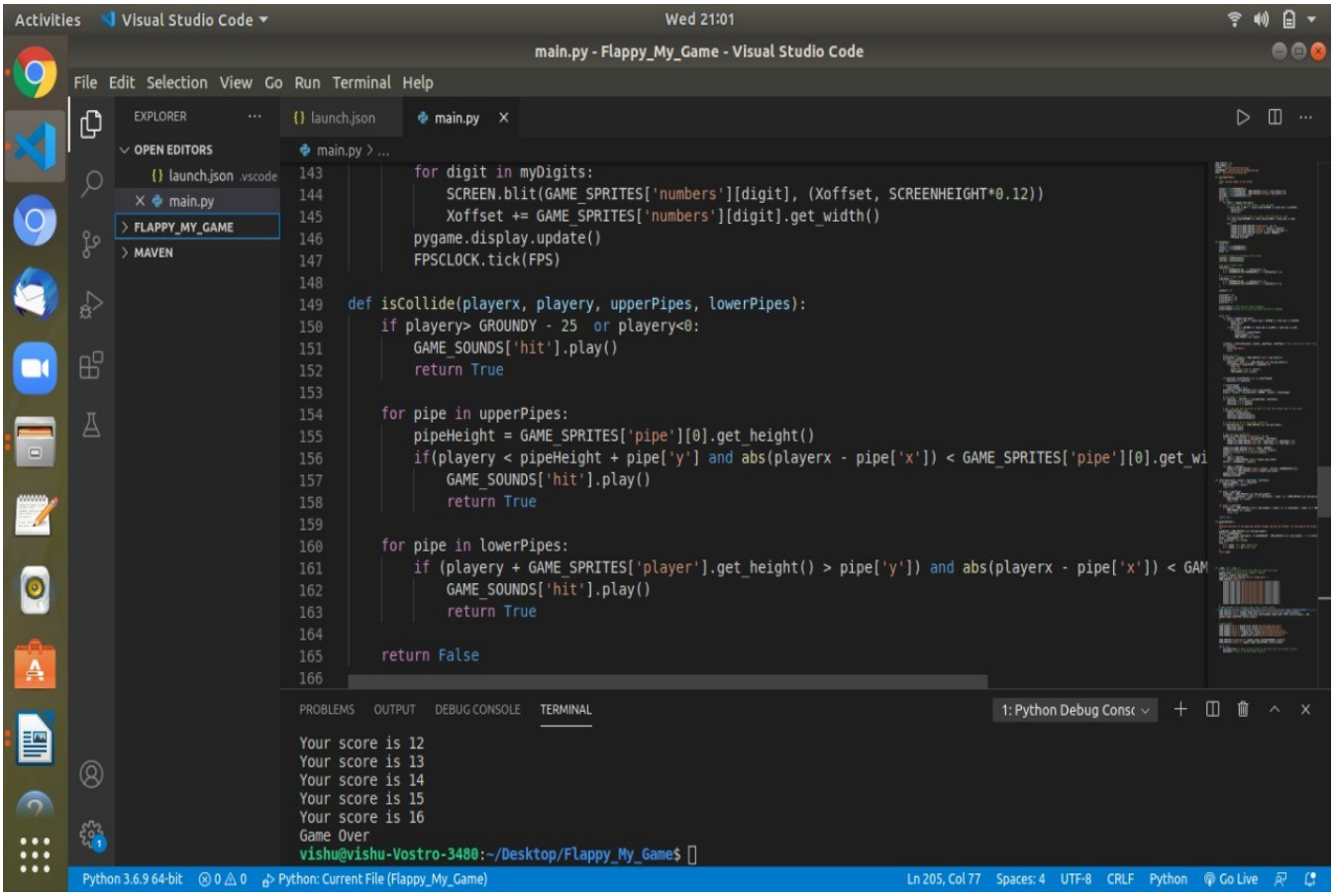

Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
main.py > ...
23 playerx = int(SCREENWIDTH/5)
24 playery = int((SCREENHEIGHT - GAME_SPRITES['player'].get_height())/2)
25 messagex = int((SCREENWIDTH - GAME_SPRITES['message'].get_width())/2)
26 messagey = int(SCREENHEIGHT*0.13)
27 basex = 0
28 while True:
29 for event in pygame.event.get():
30 # if user clicks on cross button, close the game
31 if event.type == QUIT or (event.type==KEYDOWN and event.key == K_ESCAPE):
32 pygame.quit()
33 sys.exit()
34
35 # If the user presses space or up key, start the game for them
36 elif event.type==KEYDOWN and (event.key==K_SPACE or event.key == K_UP):
37 return
38 else:
39 SCREEN.blit(GAME_SPRITES['background'], (0, 0))
40 SCREEN.blit(GAME_SPRITES['player'], (playerx, playery))
41 SCREEN.blit(GAME_SPRITES['message'], (messagex,messagey))
42 SCREEN.blit(GAME_SPRITES['base'], (basex, GROUNDY))
43 pygame.display.update()
44 FPSCLOCK.tick(FPS)
45
46 def mainGame():
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77 Spaces: 4 UTF-8 CRLF Python Go Live

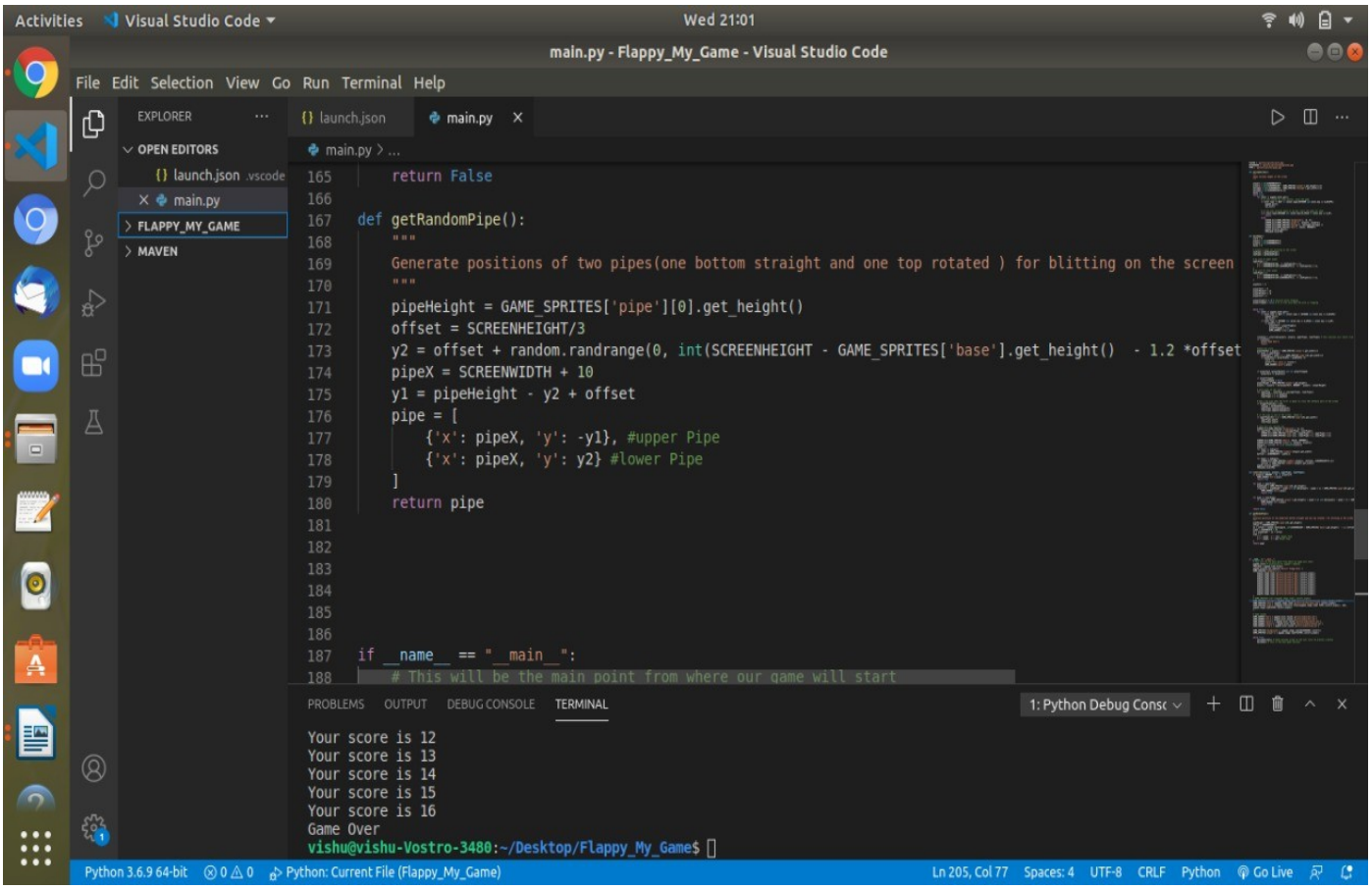

Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
def mainGame():
score = 0
playerx = int(SCREENWIDTH/5)
playery = int(SCREENWIDTH/2)
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
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77
Spaces: 4
UTF-8
CRLF
Python
Go Live

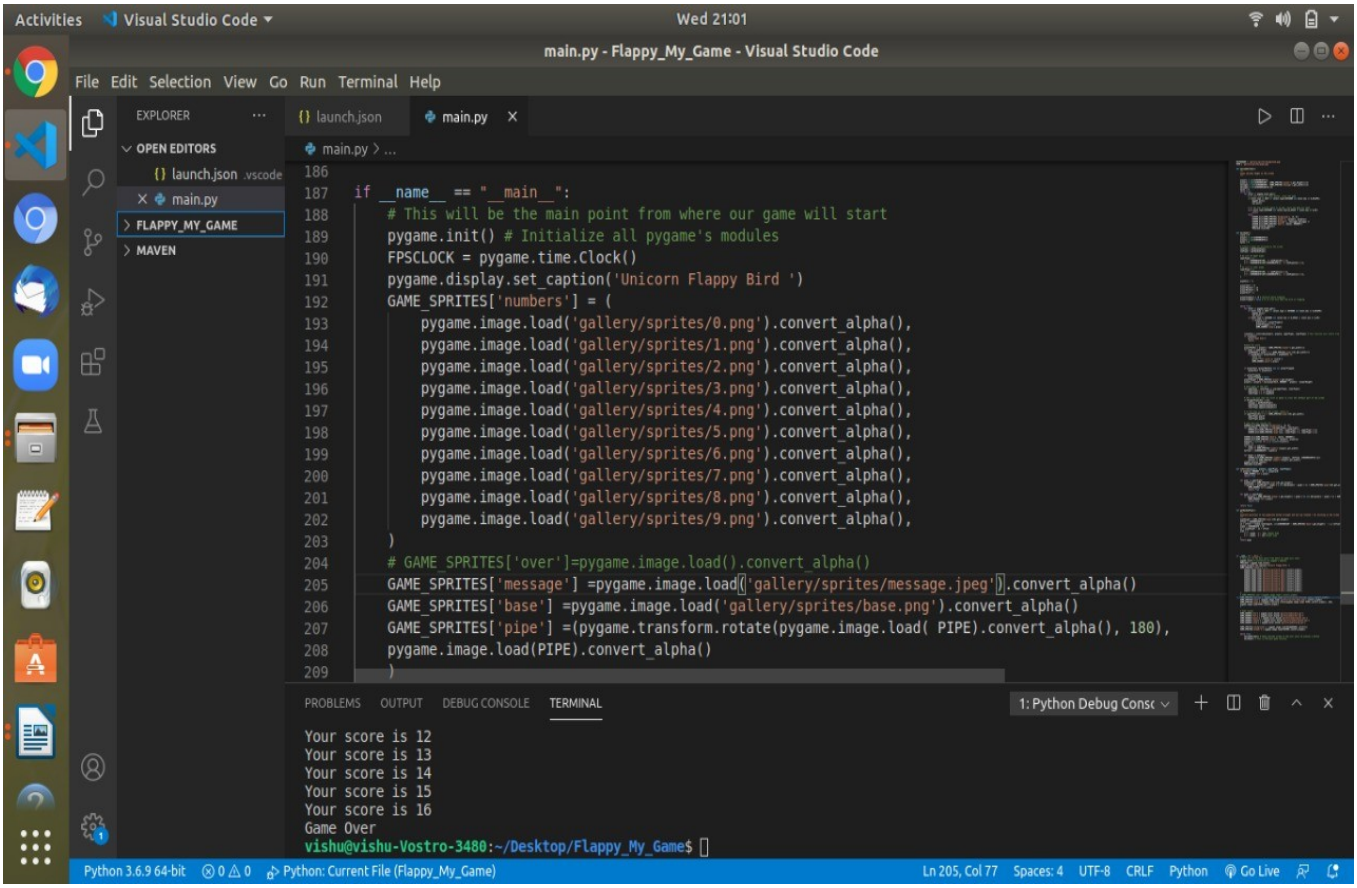

Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
main.py > ...
pipeVelX = -4
playerVelY = -9
playerMaxVelY = 10
playerMinVelY = -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()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77 Spaces: 4 UTF-8 CRLF Python Go Live


Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
main.py > ...
#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
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77 Spaces: 4 UTF-8 CRLF Python Go Live


Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
main.py > ...
117
118 # Add a new pipe when the first is about to cross the leftmost part of the screen
119 if 0<upperPipes[0]['x']<5:
120 newpipe = getRandomPipe()
121 upperPipes.append(newpipe[0])
122 lowerPipes.append(newpipe[1])
123
124 # if the pipe is out of the screen, remove it
125 if upperPipes[0]['x'] < -GAME_SPRITES['pipe'][0].get_width():
126 upperPipes.pop(0)
127 lowerPipes.pop(0)
128
129 # Lets blit our sprites now
130 SCREEN.blit(GAME_SPRITES['background'], (0, 0))
131 for upperPipe, lowerPipe in zip(upperPipes, lowerPipes):
132 SCREEN.blit(GAME_SPRITES['pipe'][0], (upperPipe['x'], upperPipe['y']))
133 SCREEN.blit(GAME_SPRITES['pipe'][1], (lowerPipe['x'], lowerPipe['y']))
134
135 SCREEN.blit(GAME_SPRITES['base'], (basex, GROUNDY))
136 SCREEN.blit(GAME_SPRITES['player'], (playerx, playery))
137 myDigits = [int(x) for x in list(str(score))]
138 width = 0
139 for digit in myDigits:
140 width += GAME_SPRITES['numbers'][digit].get_width()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77 Spaces: 4 UTF-8 CRLF Python Go Live


Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
main.py > ...
143 for digit in myDigits:
144 SCREEN.blit(GAME_SPRITES['numbers'][digit], (Xoffset, SCREENHEIGHT*0.12))
145 Xoffset += GAME_SPRITES['numbers'][digit].get_width()
146 pygame.display.update()
147 FPSCLOCK.tick(FPS)
148
149 def isCollide(playerx, playery, upperPipes, lowerPipes):
150 if playery> GROUNDY - 25 or playery<0:
151 GAME_SOUNDS['hit'].play()
152 return True
153
154 for pipe in upperPipes:
155 pipeHeight = GAME_SPRITES['pipe'][0].get_height()
156 if(playery < pipeHeight + pipe['y'] and abs(playerx - pipe['x']) < GAME_SPRITES['pipe'][0].get_wi
157 GAME_SOUNDS['hit'].play()
158 return True
159
160 for pipe in lowerPipes:
161 if (playery + GAME_SPRITES['player'].get_height() > pipe['y']) and abs(playerx - pipe['x']) < GAM
162 GAME_SOUNDS['hit'].play()
163 return True
164
165 return False
166
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Conso
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77 Spaces: 4 UTF-8 CRLF Python Go Live


Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
main.py > ...
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
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77
Spaces: 4
UTF-8
CRLF
Python
Go Live


Activities
Visual Studio Code
Wed 21:01
main.py - Flappy_My_Game - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
launch.json .vscode
main.py
FLAPPY_MY_GAME
MAVEN
launch.json
main.py
main.py > ...
186
187 if __name__ == "__main__":
188 # This will be the main point from where our game will start
189 pygame.init() # Initialize all pygame's modules
190 FPSCLOCK = pygame.time.Clock()
191 pygame.display.set_caption('Unicorn Flappy Bird ')
192 GAME_SPRITES['numbers'] = (
193 pygame.image.load('gallery/sprites/0.png').convert_alpha(),
194 pygame.image.load('gallery/sprites/1.png').convert_alpha(),
195 pygame.image.load('gallery/sprites/2.png').convert_alpha(),
196 pygame.image.load('gallery/sprites/3.png').convert_alpha(),
197 pygame.image.load('gallery/sprites/4.png').convert_alpha(),
198 pygame.image.load('gallery/sprites/5.png').convert_alpha(),
199 pygame.image.load('gallery/sprites/6.png').convert_alpha(),
200 pygame.image.load('gallery/sprites/7.png').convert_alpha(),
201 pygame.image.load('gallery/sprites/8.png').convert_alpha(),
202 pygame.image.load('gallery/sprites/9.png').convert_alpha(),
203)
204 # GAME_SPRITES['over']=pygame.image.load().convert_alpha()
205 GAME_SPRITES['message'] =pygame.image.load('gallery/sprites/message.jpeg').convert_alpha()
206 GAME_SPRITES['base'] =pygame.image.load('gallery/sprites/base.png').convert_alpha()
207 GAME_SPRITES['pipe'] =(pygame.transform.rotate(pygame.image.load(PIPE).convert_alpha(), 180),
208 pygame.image.load(PIPE).convert_alpha()
209)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc
Your score is 12
Your score is 13
Your score is 14
Your score is 15
Your score is 16
Game Over
vishu@vishu-Vostro-3480:~/Desktop/Flappy_My_Game$
Python 3.6.9 64-bit
Python: Current File (Flappy_My_Game)
Ln 205, Col 77 Spaces: 4 UTF-8 CRLF Python Go Live

References

- ✗ **https://www.pygame.org/news**
- ✗ **https://inventwithpython.com/pygame/**

End of Report.