VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590018, Karnataka

Python Application Programming

A mini project report on

Flappy Bird Game

Submitted By:

Amratya Singh **(1GA17CS011)**

Anuraj Deep **(1GA17CS017)**

Under the Guidance of

Mrs Sushmitha S

Associate Professor

Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Global Academy of Technology

Rajarajeshwarinagar, Bangalore – 560098

2019-2020

Global Academy of Technology

Department of Computer Science and Engineering

Certificate

This is to certify that the project entitled **"Flappy Bird Game"** is a bonafide work carried out by **Anuraj Deep(1GA17CS017), Amratya Singh(1GA17CS011)** as a part of assignment in Python Applications Programming in Computer Science and Engineering during the year 2019-2020.

Max. marks	Marks obtained	Faculty Name and Signature
07		

ABSTRACT

Flappy Bird Game project is written in Python. The project file contains asset files, python scripts (flappybird.py). The gameplay Graphics is good enough and the controls are simple for the users. Talking about the gameplay, it's one of the most addictive and played games for all. All the playing methods are too simple just like the real one. All you have to do is just try to stay in the middle of the screen until long green pipes appear in front of you. Here, the user has to control the bird flapping up, down using Spacebar, without touching pipes in order to score game points. This means the more you pass through green pipes, more will be the game points. A simple GUI is provided for the easy gameplay. The gameplay design is so simple that user won't find it difficult to use and navigate.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. Python Programming language

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989.Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.

1.2. Applications of Python programming language

1. Web and Internet Development

Python lets you develop a web application without too much trouble. It has libraries for internet protocols like HTML and XML, JSON, e-mail processing, FTP, IMAP, and easy-to-use socket interface. Yet, the package index has more libraries:

- Requests – An HTTP client library
- BeautifulSoup – An HTML parser
- Feedparser – For parsing RSS/Atom feeds
- Paramiko – For implementing the SSH2 protocol
- Twisted Python – For asynchronous network programming

We also have a gamut of frameworks available. Some of these are- Django, Pyramid. We also get microframeworks like flask and bottle.

2. Desktop GUI Applications

Most binary distributions of Python ship with Tk, a standard GUI library. It lets you draft a user interface for an application. Apart from that, some toolkits are available:

- wxWidgets
- Kivy – for writing multitouch applications
- Qt via pyqt or pyside

And then we have some platform-specific toolkits:

- GTK+
- Microsoft Foundation Classes through the win32 extensions
- Delphi

3. Science and Numeric Applications

This is one of the very common applications of Python programming. With its power, it comes as no surprise that Python finds its place in the scientific community. For this, we have:

- SciPy – A collection of packages for mathematics, science, and engineering.
- Pandas – A data-analysis and -modeling library
- IPython – A powerful shell for easy editing and recording of work sessions. It also supports visualizations and parallel computing.
- Software Carpentry Course – It teaches basic skills for scientific computing and running bootcamps. It also provides open-access teaching materials.

- Also, NumPy lets us deal with complex numerical calculations.

4. Games and 3D Graphics

Safe to say, this one is the most interesting. When people hear someone say they're learning Python, the first thing they get asked is – 'So, did you make a game yet?'

PyGame, PyKyra are two frameworks for game development with Python. Apart from this, we also get a variety of 3D-rendering libraries.

4. Software Development

Software developers make use of Python as a support language. They use it for build-control and management, testing, and for a lot of other things:

- SCons – for build-control
- Buildbot**,** Apache Gump – for automated and continuous compilation and testing
- Roundup**,** Trac – for project management and bug-tracking.
- Roster of Integrated Development Environments

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

Language used: Python

IDE/Compiler used: PyCharm

OS used: Ubuntu 18.04 / Windows 10

2.2 Hardware Requirements

Processor: 533 MHz

System: Intel Core 2 Duo U7600

Graphics Card: NVIDIA GeForce 7200 GS

Hard Disk: 0.1 GB

Monitor: 15 VGA color

Mouse: Logitech

Keyboard: 101 keys enhanced

CHAPTER 3

IMPLEMENTATION AND RESULTS

3.1 About Project

Flappy Bird is a 2013 mobile game, developed by Vietnam- based developer Dong Nguyen and published by GEARS Studios, a small independent game developer also based in Vietnam. The game is a s ide-scroller where the player controls a bird, attempting to fly between rows of green pipes without hitting them.

Flappy Bird Game project is written in Python. The project file contains asset files, python scripts (main.py). The gameplay Graphics is good enough and the controls are simple for the users. Talking about the gameplay, it's one of the most addictive and played games for all. All the playing methods are too simple just like the real one. All you have to do is just try to stay in the middle of the screen until long green pipes appear in front of you. Here, the user has to control the bird flapping up, down using Spacebar, without touching pipes in order to score game points. This means the more you pass through green pipes, more will be the game points. A simple GUI is provided for the easy gameplay. The gameplay design is so simple that user won't find it difficult to use and navigate.

This simple flappy bird game provides the simplest gameplay of flappy bird. The gameplay works well. In order to run the project, you must have installed Python, Pygame on your PC.

3.2 Project code

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

# Global Variables for the game
FPS = 32
SCREENWIDTH = 285
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
```

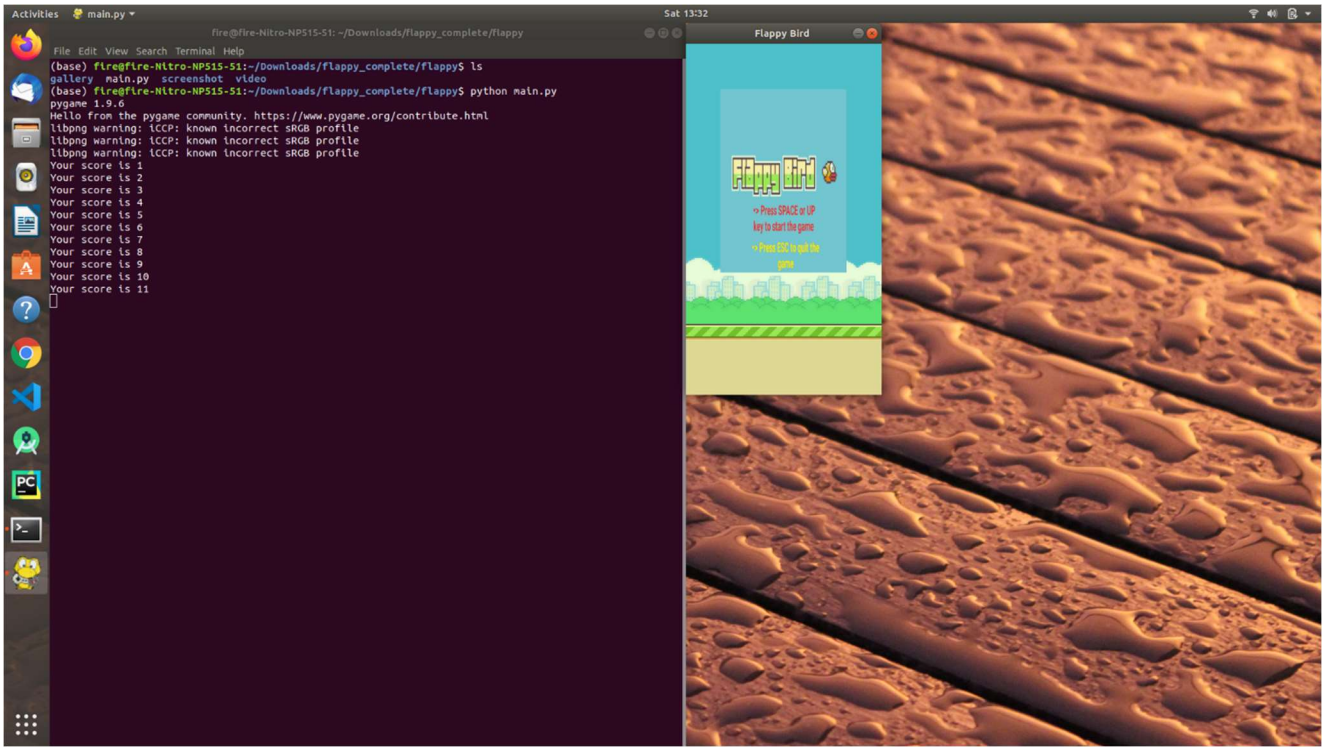
```
        #check for score
        playerMidPos = playerx + GAME_SPRITES['player'].get_width()/2
        for pipe in upperPipes:
            pipeMidPos = pipe['x'] + GAME_SPRITES['pipe'][0].get_width()/2
            if pipeMidPos<= playerMidPos < pipeMidPos +4:
                score +=1
                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):
    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
    """
```

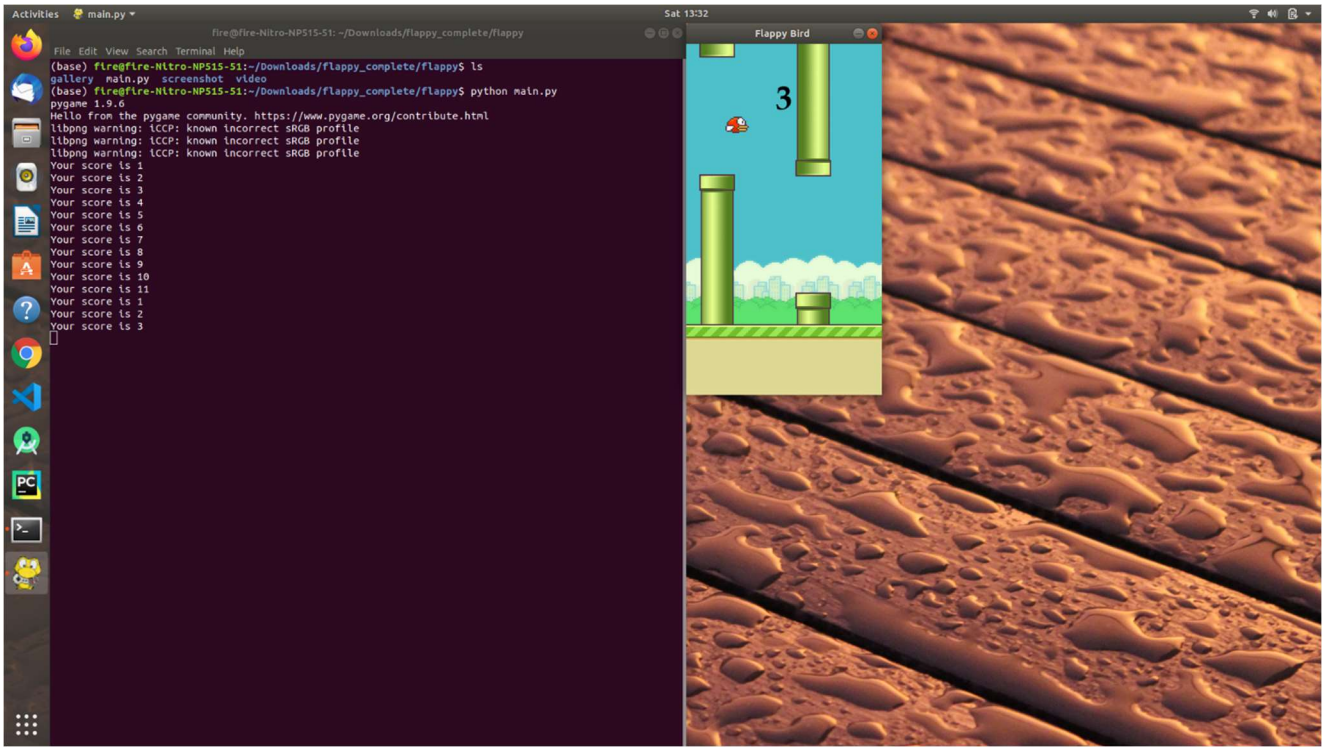
```
    pipeHeight = GAME_SPRITES['pipe'][0].get_height()
    offset = SCREENHEIGHT/3
    y2 = offset + random.randrange(0, int(SCREENHEIGHT -
GAME_SPRITES['base'].get_height()  - 1.2 *offset))
    pipeX = SCREENWIDTH + 10
    y1 = pipeHeight - y2 + offset
    pipe = [
        {'x': pipeX, 'y': -y1}, #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 ')
    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
```

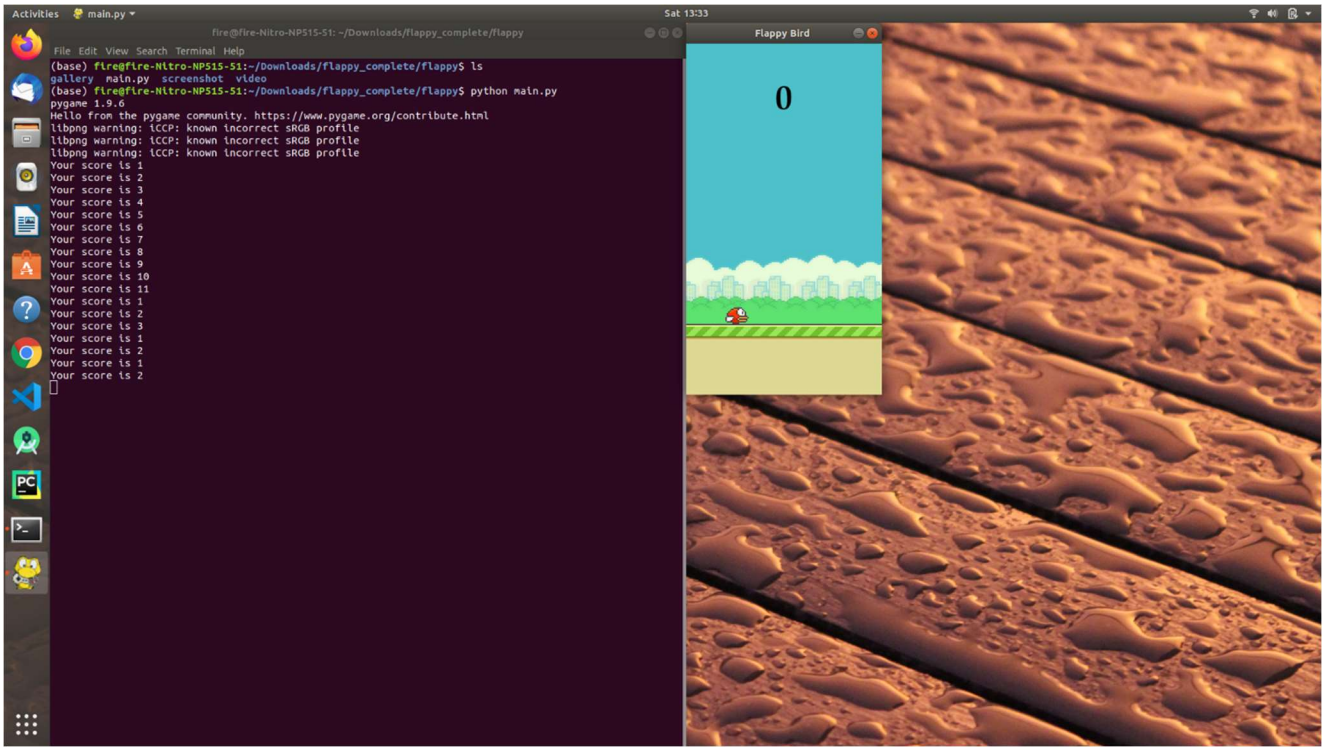
3.3 Results



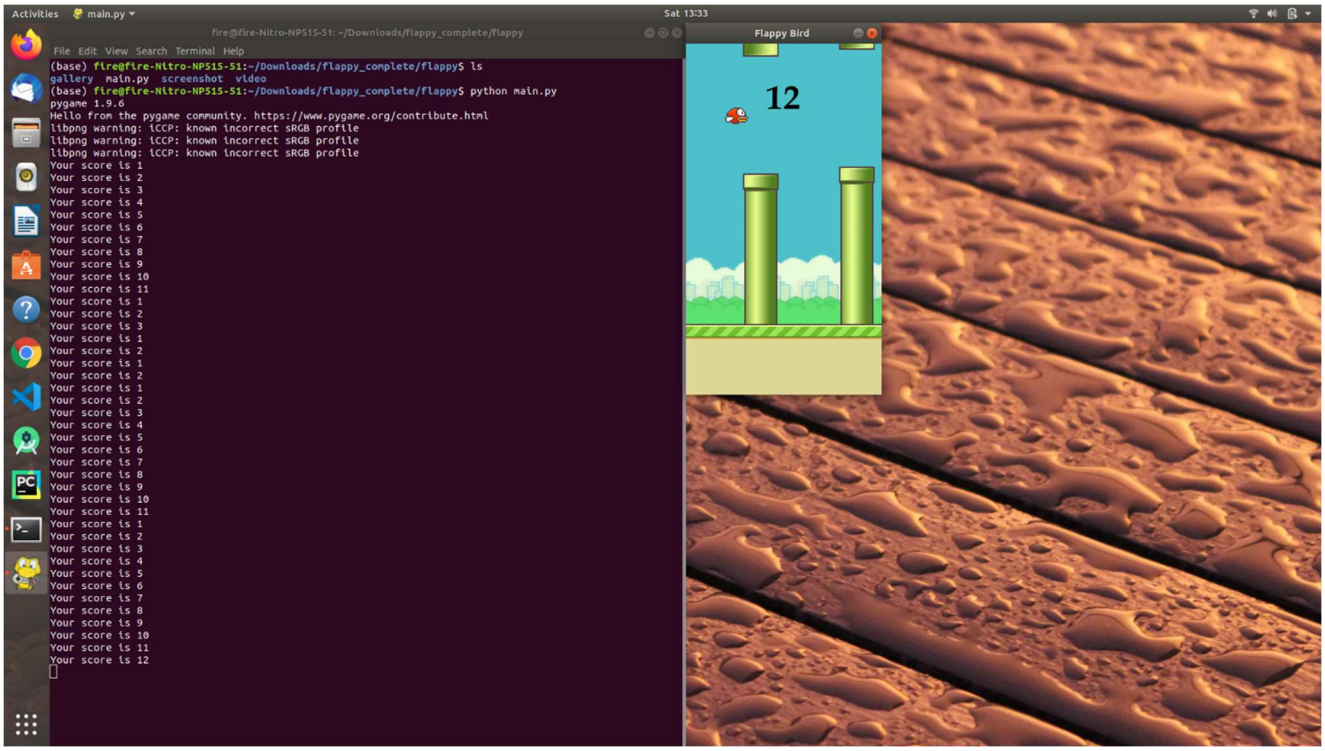
Snapshot 3.1:



Snapshot 3.2:



Snapshot 3.3:


Activities
main.py
Sat 13:33
fire@fire-Nitro-NP515-51: ~/Downloads/flappy_complete/flappy
File Edit View Search Terminal Help
(base) fire@fire-Nitro-NP515-51:~/Downloads/flappy_complete/flappy$ ls
gallery main.py screenshot video
(base) fire@fire-Nitro-NP515-51:~/Downloads/flappy_complete/flappy$ python main.py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
Your score is 1
Your score is 2
Your score is 3
Your score is 4
Your score is 5
Your score is 6
Your score is 7
Your score is 8
Your score is 9
Your score is 10
Your score is 11
Your score is 1
Your score is 2
Your score is 3
Your score is 1
Your score is 2
Your score is 1
Your score is 2
Your score is 1
Your score is 2
Your score is 3
Your score is 4
Your score is 5
Your score is 6
Your score is 7
Your score is 8
Your score is 9
Your score is 10
Your score is 11
Your score is 1
Your score is 2
Your score is 3
Your score is 4
Your score is 5
Your score is 6
Your score is 7
Your score is 8
Your score is 9
Your score is 10
Your score is 11
Your score is 12
Flappy Bird
12

Snapshot 3.4:

CONCLUSION

In this project basic python concepts are used along with some other so it helped to clear our basics of python by implementing them in real world. Pygame module is used that helped us to make game which is highly portable SDL(Simple Direct Media Layer) development library, it can run across many platforms and operating system and also control the logic and graphics of the game without worrying about the backend complexities required with video and audio. Random module is also used which helped in creating the obstacles randomly of different length and made the game more interesting. This project helped us to gain the hands on experience of python programming language and increased our knowledge about it.

The project helped us to gain the skills like time management, team work, ownership and many more as different tasks were assigned to each member which gives the sense of ownership and also have to be completed within a given time period.

REFERENCES

Bibliography:

[1] Charles R. Severance, “Python for Everybody”

[2] Allen B. Downey, “Think Python: How to Think Like a Computer Scientist”

[3] Chaeles Dierbach, “Introduction to Computer Science Using Python”

Websites:

[1] www.pygame.org/docs/

[2] www.realpython.com

[3] www.python.org