FLAPPY BIRD USING PYGAME

MINIPROJECT REPORT

Submitted by

ARUN V. (710018104007)

GOWTHAM S. (710018104302)

RANJITHKUMAR K. (710018104035)

SHYAM SUNDAR S. (710018104046)

in partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING IN
COMPUTER SCIENCE AND ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING**

**ANNA UNIVERSITY
REGIONAL CAMPUS, COIMBATORE
COIMBATORE – 641046**

APRIL 2021

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Report titled "**FLAPPY BIRD USING PYGAME**" is the bonafide work of **ARUN V. (710019405001), GOWTHAM S. (710018104302), RANJITHKUMAR K. (710018104035), SHYAM SUNDAR S. (710018104046)** have carried out the work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. P. MARIKKANNU M.TECH.,Ph.D.,	**Dr. M. NEWLIN RAJKUMAR M.S., M.B.A., Ph.D.,**
Head of the Department,	Supervisor,
Assistant Professor,	Assistant Professor,
Department of CSE,	Department of CSE,
Anna University	Anna University
Regional Campus,	Regional Campus,
Coimbatore – 641046.	Coimbatore – 641046.

Submitted for the Mini-Project Viva - Voce Examination held on ___________

------------------------	--------------------------
Internal Examiner	**External Examiner**

INDEX

ABSTRACT

Simple 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 that user has to do is just try to stay in the middle of the screen until long green pipes appear in front of user. 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.

ACKNOWLEDGEMENT

First and foremost, I would like to express my heartfelt thanks to my beloved parents for their blessing and wishes for the successful completion of this mini-project.

I wish to express my deep sense of gratitude and hearted thanks to **Dr. K. RAVICHANDRAN M.Tech., Ph.D.,** Dean, Anna University Regional Campus, Coimbatore, for providing necessary facilities in the university to complete my mini-project work successfully.

I express my heartfelt thanks to **Dr. P. MARIKKANNU M.TECH., Ph.D.,** Head of the Department, Department of Computer Science and Engineering, for his encouragement and inspiration.

I take this golden opportunity to extend my sincere thanks to my Mini-Project Guide **Dr. M. NEWLIN RAJKUMAR M.S., M.B.A., Ph.D.,** Assistant Professor, Department of Computer Science and Engineering, for his valuable guidance and suggestions in all aspects that aided us to ameliorate our skills.

I am thankful to all the **Teaching and Non-Teaching Staff Members** of Department of Computer Science and Engineering, and to all those who have directly and indirectly extended their help to us in completing this mini-project work successfully.

ARUN V. (710018104007),
GOWTHAM S. (710018104302),
RANJITHKUMAR K. (710018104035),
SHYAM SUNDAR S. (710018104046),
B.E. Computer Science and Engineering

INTRODUCTION

Flappy Bird is a mobile game developed by Vietnamese video game artist and programmer Dong Nguyen, under his game development company dotGears. The game is a side-scroller where the player controls a bird, attempting to fly between columns of green pipes without hitting them. Nguyen created the game over the period of several days, using a bird protagonist that he had designed for a cancelled game in 2012.

The game was released in May 2013 but received a sudden rise in popularity in early 2014 and became a sleeper hit. Flappy Bird received poor reviews from some critics, who criticized its high level of difficulty and alleged plagiarism in graphics and game mechanics, while other reviewers found it addictive. At the end of January 2014, it was the most-downloaded free game in the App Store for iOS. During this period, its developer said that Flappy Bird was earning $50,000 a day from in-app advertisements as well as sales.

OBJECTIVE:

The objective of the game is to direct a flying bird, named Faby, the conditions are

1. It should move continuously to the right, between sets of Mario-like pipes.
2. If the player touches the pipes the game ends as the user fails to help the Faby from preventing it from hitting the pipe.
3. Faby briefly flaps upward each time that the player presses the space bar.
4. If the space bar is not pressed, Faby falls because of the gravity.
5. Each pair of pipes that he navigates between earns the player a single point.

LITERATURE REVIEW

1. T. H. Laine and R. S. N. Lindberg, "Designing Engaging Games for Education: A Systematic Literature Review on Game Motivators and Design Principles," in *IEEE Transactions on Learning Technologies*, vol. 13, no. 4, pp. 804-821, 1 Oct.-Dec. 2020, doi: 10.1109/TLT.2020.3018503.
2. McGugan, Will. "Beginning Game Development with Python and Pygame: From Novice to Professional." (2007).
3. Heo, Junyoung and Seukwon Kang. "Simple shooting game engine in Python." *Int. J. Comput. Vis. Robotics* 5 (2015): 130-137.
4. Riley, S.. "Game Programming with Python." (2003).
5. Wang, Hong. "Teaching CS1 with Python GUI Game Programming." *WCE 2010* (2010).
6. Sahni, N. et al. "A Review on Developing an Arcade Game Machine and an Arcade Game using Raspberry Pi and Pygame." *International Journal of Computer Applications* 120 (2015): 37-39.
7. Yang, Cheer-Sun D. and J. W. Yu. "Using Incremental Worked Examples for Teaching Python and Game Programming Teaching Python with Tkinter and Pygame Modules." (2011).
8. Sweigart, A.. "Invent Your Own Computer Games with Python, 2nd Edition." (2010).
9. Sweigart, A.. "Automate the Boring Stuff with Python: Practical Programming for Total Beginners." (2015).

SOFTWARE PROJECT DESCRIPTION

Story

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

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 space bar 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. Pygame are expected to run on multiple operating systems with different hardware specifications. Threads give the most accurate timing solutions.

Objective Selection

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

Moving Wall

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

Moving Background

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

Score Counting

Score counting is the interesting for user. By the score the player knows his/her performance. If the bird crosses a pipe without collision or not fall in ground his/her score increment one.

Proposed Process Model

A **s**oftware process model is a simplified representation of a software process. Each model represents a process from a specific perspective.

Types of model:

1. The Incremental Development Model.
2. The spiral model.
3. Evolutionary model.
4. The phases of iterative development.
5. The principles of agile methods.
6. The Waterfall Model.

We used evolutionary model in our project. In this way, we accomplish work in a iterative way.

Evolutionary Model

Evolutionary model is a combination of iterative and incremental approach to software development. The Evolutionary development model divides the development cycle into smaller, incremental **waterfall** models in which users are able to get access to the product at the end of each cycle. Waterfall project management is a sequential, linear process of project management. It consists of several discrete phases. No phase begins until the prior phase is complete, and each phase's completion is terminal waterfall management does not allow you to return to a previous phase. We use this process model.

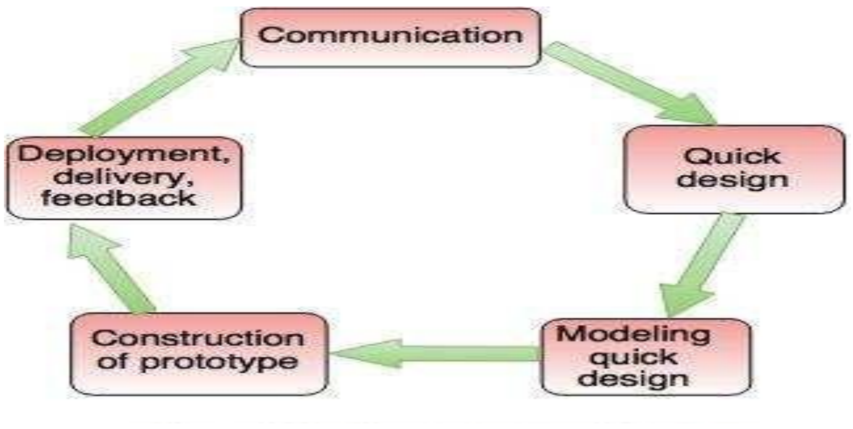


Figure 1: Evolutionary Process Model

Tools

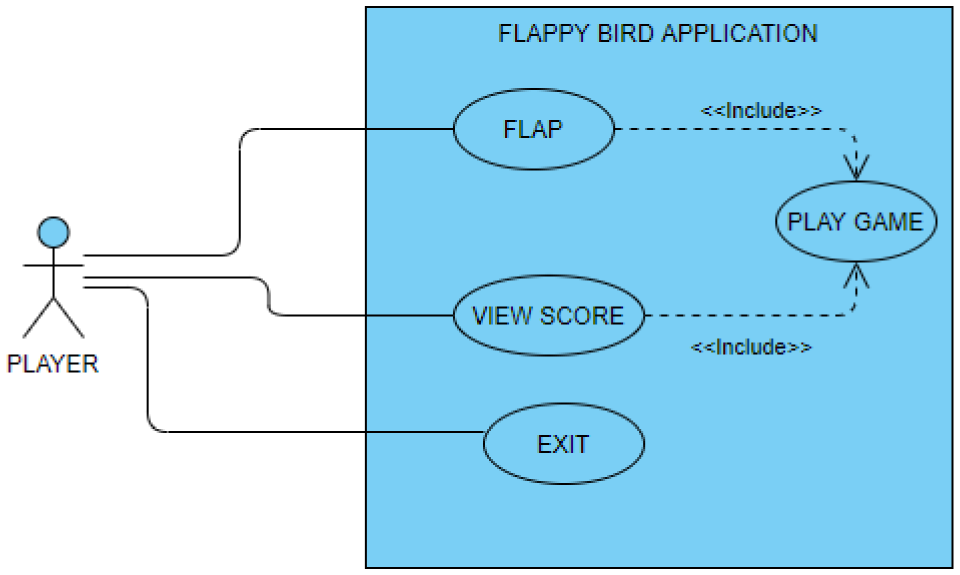
Language: Python

IDE: An IDE (Integrated Development Environment) contains a code editor, a compiler or interpreter, and a debugger, accessed through a single graphical user interchange (GUI). Our IDE is Python IDLE.

SYSTEM DESIGN:

USE CASE DIAGRAM

- Use case diagrams are **behavioural diagram** used to describe **a set of actions (use cases)** that systems perform in collaboration with **one or more external users** of the system(**actors**).
- Use case diagram depicts the human interaction with the system to give the context of who uses, which part of the system and for what purposes.
- Use case diagrams are typically developed in the early stage of development and people often apply use case modelling for the following purposes:
  - Specify the context of a system
  - Capture the requirements of a system
  - Validate a systems architecture
  - Drive implementation and generate test cases
  - Developed by analysts together with domain experts



STATE- MACHINE DIAGRAM

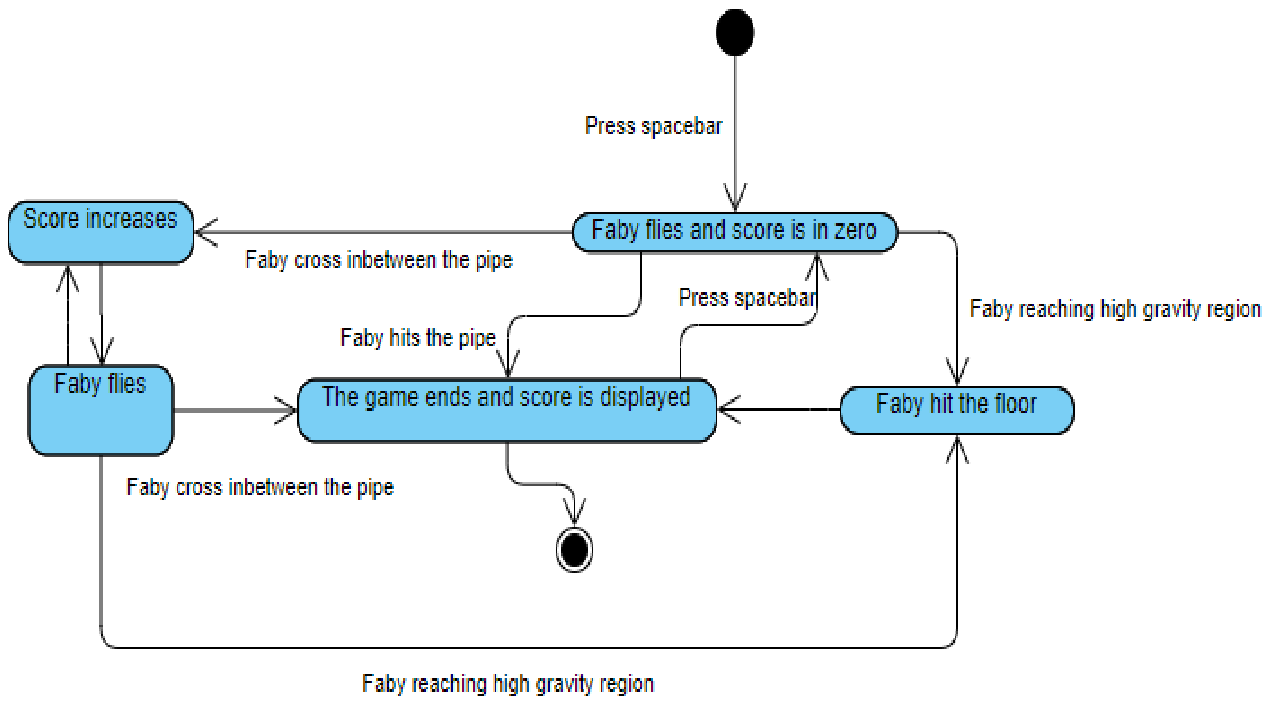
State machine diagram is a **behaviour diagram** which shows discrete behaviour of a part of designed system through finite state transitions. State machine diagrams can also be used to express the usage protocol of part of a system. Two kinds of state machines defined in **UML 2.4** are

- **behavioural state machine**
- **protocol state machine**

State machine diagram typically are used to describe state-dependent behaviour for an object.

An object responds differently to the same event depending on what state it is in.

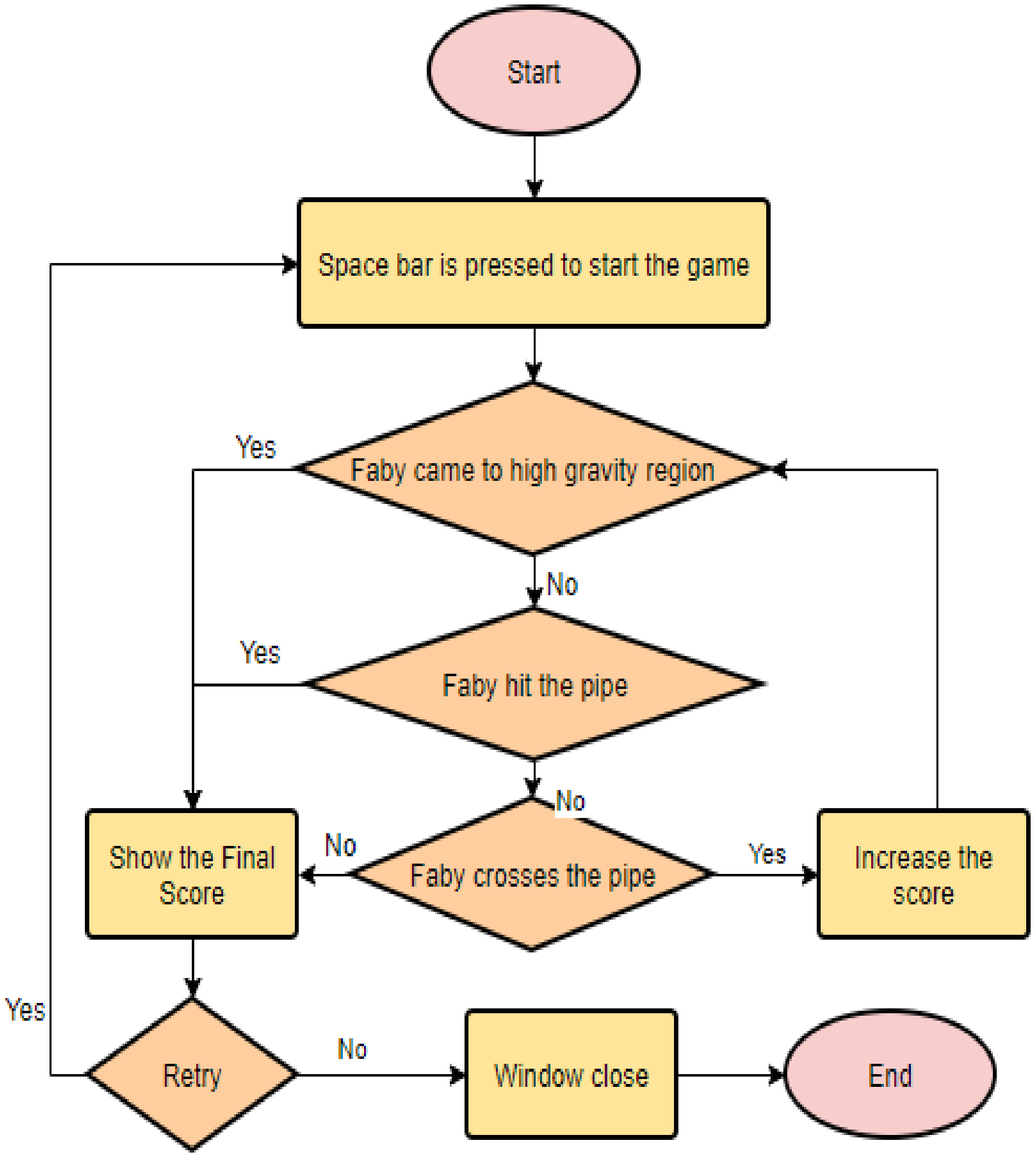
State machine diagrams are usually applied to objects but can be applied to any element that has behavior to other entities such as: actors, use cases, methods, subsystems systems and etc. and they are typically used in conjunction with interaction diagrams (usually sequence diagrams).


Press spacebar
Score increases
Faby flies and score is in zero
Faby cross inbetween the pipe
Press spacebar
Faby reaching high gravity region
Faby hits the pipe
Faby flies
The game ends and score is displayed
Faby hit the floor
Faby cross inbetween the pipe
Faby reaching high gravity region

FLOW CHART DIAGRAM

A flowchart is a graphical representation of steps. It was originated from computer science as a tool for representing algorithms and programming logic but had extended to use in all other kinds of processes. Nowadays, flowcharts play an extremely important role in displaying information and assisting reasoning.

They help us visualize complex processes, or make explicit the structure of problems and tasks. A flowchart can also be used to define a process or project to be implemented.



IMPLEMENTATION

PROGRAM CODE:

```
import pygame, sys, random

def draw_floor():
screen.blit(floor_surface,(floor_x_pos,900))
screen.blit(floor_surface,(floor_x_pos + 576,900))

def create_pipe():
random_pipe_pos = random.choice(pipe_height)
bottom_pipe = pipe_surface.get_rect(midtop = (700,random_pipe_pos))
top_pipe = pipe_surface.get_rect(midbottom = (700,random_pipe_pos - 300))
return bottom_pipe,top_pipe

def move_pipes(pipes):
for pipe in pipes:
        pipe.centerx -= 5
return pipes

def draw_pipes(pipes):
for pipe in pipes:
        if pipe.bottom >= 1024:
                screen.blit(pipe_surface,pipe)
        else:
                flip_pipe = pygame.transform.flip(pipe_surface,False,True)
                screen.blit(flip_pipe,pipe)
def remove_pipes(pipes):
for pipe in pipes:
        if pipe.centerx == -600:
                pipes.remove(pipe)
return pipes
def check_collision(pipes):
```

```
for pipe in pipes:
        if bird_rect.colliderect(pipe):
                death_sound.play()
                return False

if bird_rect.top <= -100 or bird_rect.bottom >= 900:
        return False

return True

def rotate_bird(bird):
new_bird = pygame.transform.rotozoom(bird,-bird_movement * 3,1)
return new_bird

def bird_animation():
new_bird = bird_frames[bird_index]
new_bird_rect = new_bird.get_rect(center = (100,bird_rect.centery))
return new_bird,new_bird_rect

def score_display(game_state):
if game_state == 'main_game':
        score_surface = game_font.render(str(int(score)),True,(255,255,255))
        score_rect = score_surface.get_rect(center = (288,100))
        screen.blit(score_surface,score_rect)
if game_state == 'game_over':
        score_surface = game_font.render(f'Score: {int(score)}' ,True,(255,255,255))
        score_rect = score_surface.get_rect(center = (288,100))
        screen.blit(score_surface,score_rect)

        high_score_surface = game_font.render(f'High score:
{int(high_score)}',True,(255,255,255))
        high_score_rect = high_score_surface.get_rect(center = (288,850))
        screen.blit(high_score_surface,high_score_rect)
```

```
def update_score(score, high_score):
if score > high_score:
        high_score = score
return high_score

pygame.mixer.pre_init(frequency = 44100, size = 16, channels = 1, buffer = 512)
pygame.init()
screen = pygame.display.set_mode((576,1024))
clock = pygame.time.Clock()
game_font = pygame.font.Font('04B_19.ttf',40)

# Game Variables
gravity = 0.25
bird_movement = 0
game_active = True
score = 0
high_score = 0

bg_surface = pygame.image.load('assets/background-day.png').convert()
bg_surface = pygame.transform.scale2x(bg_surface)

floor_surface = pygame.image.load('assets/base.png').convert()
floor_surface = pygame.transform.scale2x(floor_surface)
floor_x_pos = 0

bird_downflap = pygame.transform.scale2x(pygame.image.load('assets/bluebird-downflap.png').convert_alpha())
bird_midflap = pygame.transform.scale2x(pygame.image.load('assets/bluebird-midflap.png').convert_alpha())
bird_upflap = pygame.transform.scale2x(pygame.image.load('assets/bluebird-upflap.png').convert_alpha())
bird_frames = [bird_downflap,bird_midflap,bird_upflap]
```

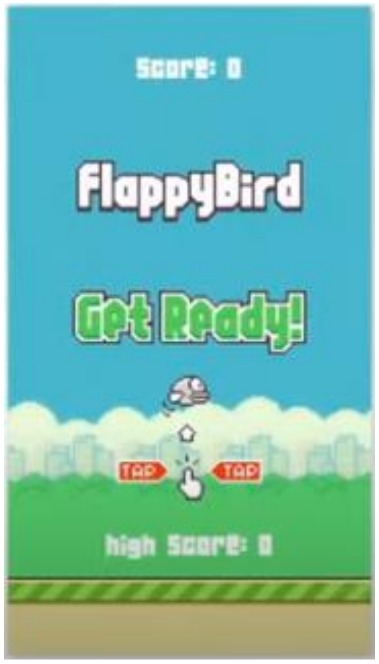
```
bird_index = 0
bird_surface = bird_frames[bird_index]
bird_rect = bird_surface.get_rect(center = (100,512))

BIRDFLAP = pygame.USEREVENT + 1
pygame.time.set_timer(BIRDFLAP,200)

# bird_surface = pygame.image.load('assets/bluebird-midflap.png').convert_alpha()
# bird_surface = pygame.transform.scale2x(bird_surface)
# bird_rect = bird_surface.get_rect(center = (100,512))

pipe_surface = pygame.image.load('assets/pipe-green.png')
pipe_surface = pygame.transform.scale2x(pipe_surface)
pipe_list = []
SPAWNPIPE = pygame.USEREVENT
pygame.time.set_timer(SPAWNPIPE,1200)
pipe_height = [400,600,800]

game_over_surface                                                          =
pygame.transform.scale2x(pygame.image.load('assets/message.png').convert_alpha())
game_over_rect = game_over_surface.get_rect(center = (288,512))

flap_sound = pygame.mixer.Sound('sound/sfx_wing.wav')
death_sound = pygame.mixer.Sound('sound/sfx_hit.wav')
score_sound = pygame.mixer.Sound('sound/sfx_point.wav')
score_sound_countdown = 100

while True:
for event in pygame.event.get():
        if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
        if event.type == pygame.KEYDOWN:
```

```
        if event.key == pygame.K_SPACE and game_active:
            bird_movement = 0
            bird_movement -= 12
            flap_sound.play()
        if event.key == pygame.K_SPACE and game_active == False:
            game_active = True
            pipe_list.clear()
            bird_rect.center = (100,512)
            bird_movement = 0
            score = 0

    if event.type == SPAWNPIPE:
        pipe_list.extend(create_pipe())

    if event.type == BIRDFLAP:
        if bird_index < 2:
            bird_index += 1
        else:
            bird_index = 0

        bird_surface,bird_rect = bird_animation()

screen.blit(bg_surface,(0,0))

if game_active:
    # Bird
    bird_movement += gravity
    rotated_bird = rotate_bird(bird_surface)
    bird_rect.centery += bird_movement
    screen.blit(rotated_bird,bird_rect)
    game_active = check_collision(pipe_list)

    # Pipes
```

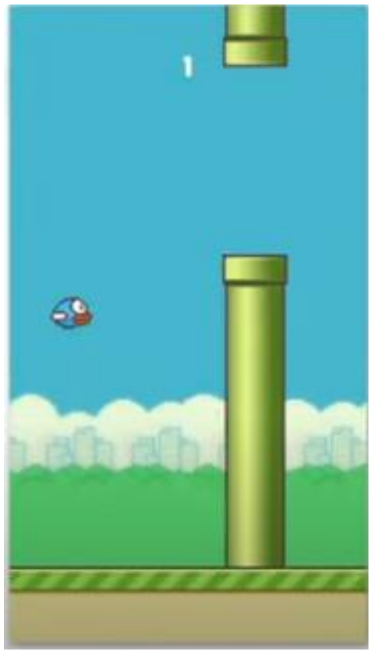
```
        pipe_list = move_pipes(pipe_list)
        pipe_list = remove_pipes(pipe_list)
        draw_pipes(pipe_list)

        score += 0.01
        score_display('main_game')
        score_sound_countdown -= 1
        if score_sound_countdown <= 0:
            score_sound.play()
            score_sound_countdown = 100
    else:
        screen.blit(game_over_surface,game_over_rect)
        high_score = update_score(score,high_score)
        score_display('game_over')


    # Floor
    floor_x_pos -= 1
    draw_floor()
    if floor_x_pos <= -576:
        floor_x_pos = 0


    pygame.display.update()
    clock.tick(120)
```

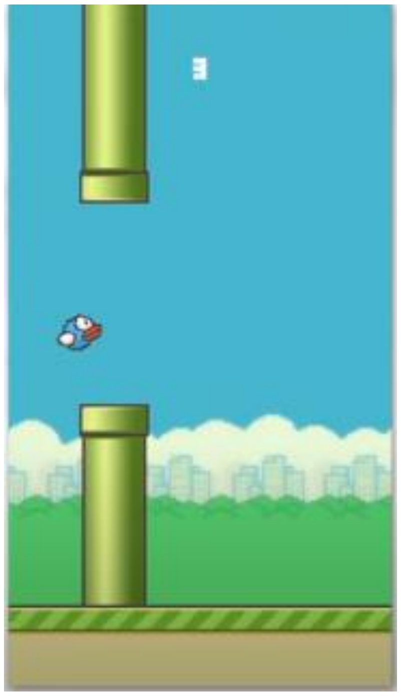
OUTPUT:

BEFORE STARTING THE GAME

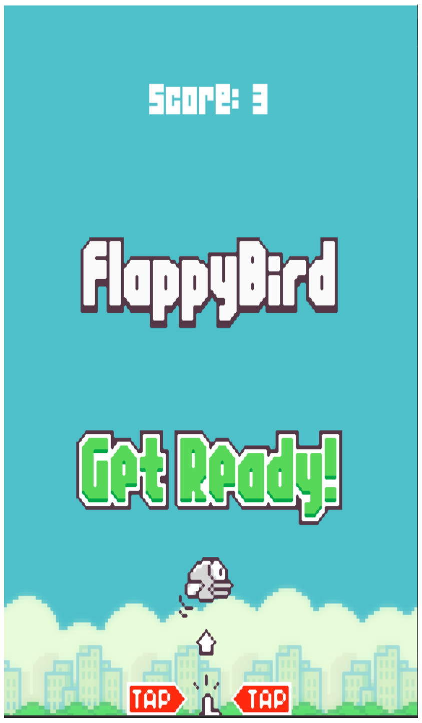


AFTER STARTING THE GAME





ON GAME OVER



CONCLUSION

We choice the miniproject to gain proper knowledge to make desktop application. It increased our knowledge for Object Oriented Language (Python). Getting experience Pygame. Another important issue is, it’s a game application and it will be recreation for all.

Future Direction

We will add more feature to the game and will change the bird and background scenery according to user choice. The status and the history will be saved and we show a graph where user can see his total performance whether it increasing or decreasing.