REPORT FOR FLAPPY BIRD GAME

As a project work for the course

PYTHON PROGRAMMING(INT213)

Name : Bharat Kumar Mareedu

Registration Number : 12008463

Name : Deepanshu Singh Raghav

Registration Number : 12008812

Program : BTech CSE

Semester : 3rd

College Name : Lovely Professional University

Date of submission : 20th November 2021

Flappy Bird Game

Acknowledgement

I would like to thank my mentor – Prof. Sagar Dhanraj Pande for teaching the course INT213 and giving the choice for us to take the project. Many thanks to my friends who spent their precious time to provide feedbacks to this project.

Abstract

Students like us who are interested in gaming are very curious to learn more about technology and grasp how it works. So, for this project we thought to make a basic game which everyone know and to head start our career in game development we made this project with different implementation of code and few added functionalities in it.

TABLE OF CONTENTS

INTRODUCTION

Context

Under the supervision of Prof-Sagar Pande, I have done this project as part of my Computer Science Engineering (CSE) course at Lovely Professional University. We took 1 month of coding and 15 days of research to complete the requirements to pass the module.

Motivation

As we are interested in gaming, we chose a project that include games. So, we thought that a popular and basic game like Flappy Bird will be perfect as our project and started working on it.

Idea

The idea was to create three classes named as Bird, Pipe, and Base where we defined all the properties and functions for the objects in the game. Images and audio files which are required throughout the game are created as constants, the main game function which captures key strokes from the keyboard and controls the game.

TEAM LEADER

Bharat Kumar

Contributions:

1. Game Constants and Required Files
2. Base Class
3. BIMG class
4. Main game function
5. Life of user function
6. Welcome Screen Function
7. Report

TEAM MEMBER

Deepanshu Singh Raghav

Contributions:

1. Bird Class
2. Pipe Class
3. Draw Window Function
4. Start Game function
5. Background Music
6. Function which rotates the bird
7. Report

LIBRARIES USED (references mentioned at last)

1. 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.

2. OS

The OS module in python provides functions for interacting with the operating system. OS comes under Python's standard utility modules (It will be installed along with python).

3. SYS

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

4. RANDOM

Python random module is an in-built module of python which is used to generate random numbers. These are pseudo-random numbers means these are not truly random. This module can be used to perform random actions such as generating random numbers.

5. TIME

This module provides various time-related functions.

Reason for choosing these modules

- PYGAME

The pygame library is an open-source module for python specifically made to make games and other multimedia applications. This is built on top of the highly portable Simple DirectMedia Layer Development library, This can run across many platforms.

By using this module we can control logic and graphics of our game without worrying about the backend complexity required for working with video and audio.

- OS

To obtain the audio files and images we have used “os.path.join()” which we gave two parameters one is the location of the file and another is the file name.

For example, if the images are in “/imgs*” folder “os.path.join()” returns the full path irrespective of the operating system.

- SYS

We have used “sys.exit()” function to stop the execution and come out from the program when the user press escape key or closed the window.

- RANDOM

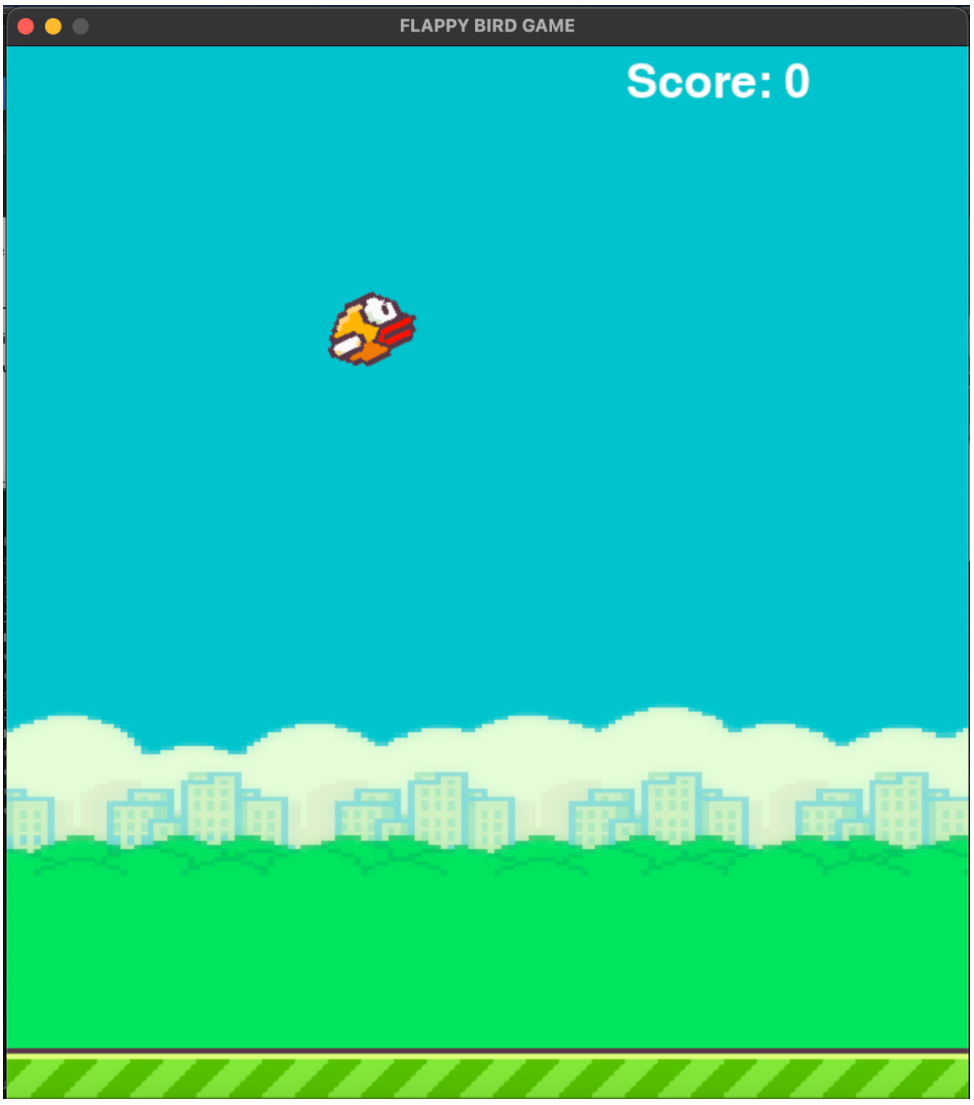
To set the height of the pipes in the game we used random.randrange() function to set random heights for the pipes throughout the game.

- TIME

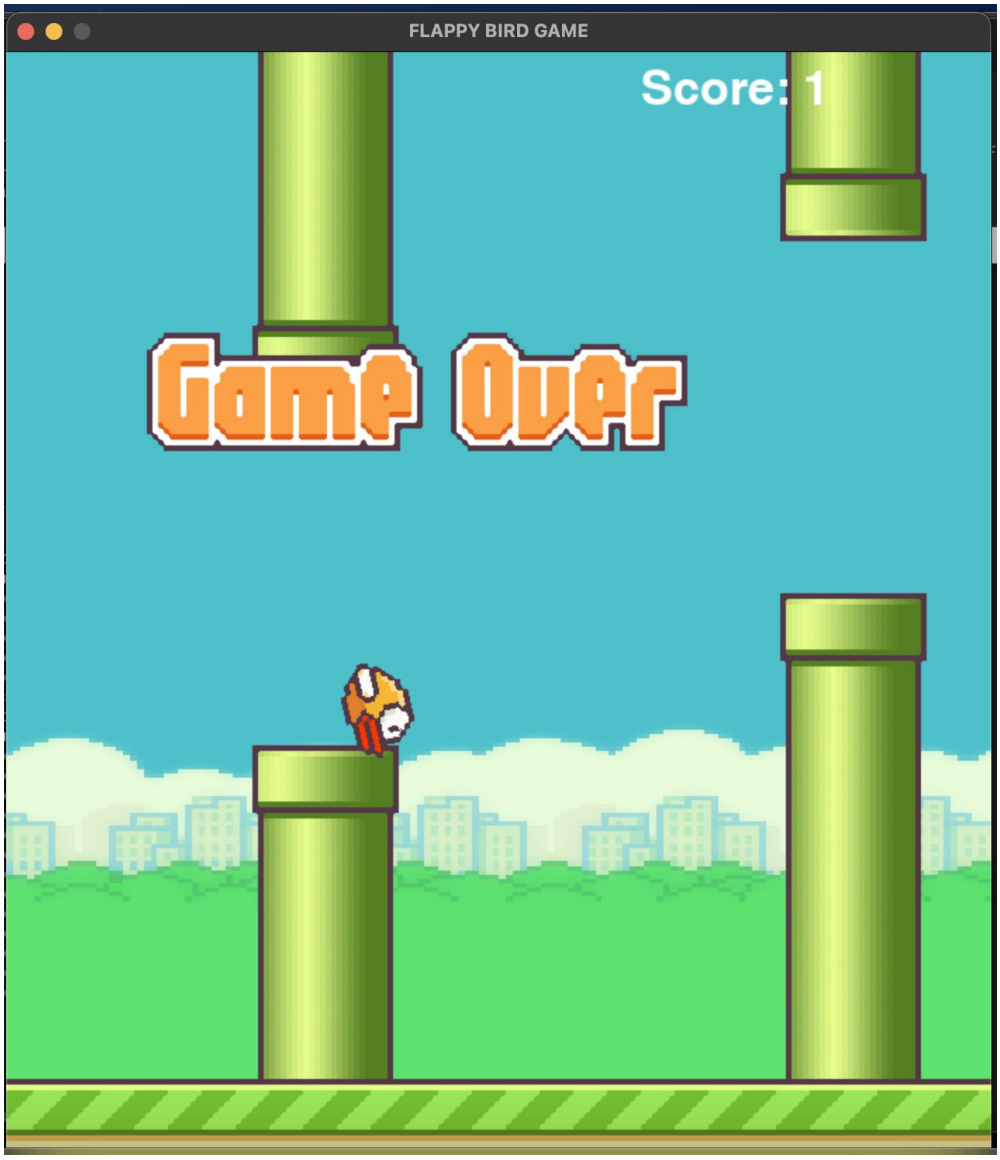
“time.sleep()” function is used to pause the program after the death of the bird.

Screenshots

Gameplay



Crash with pipe



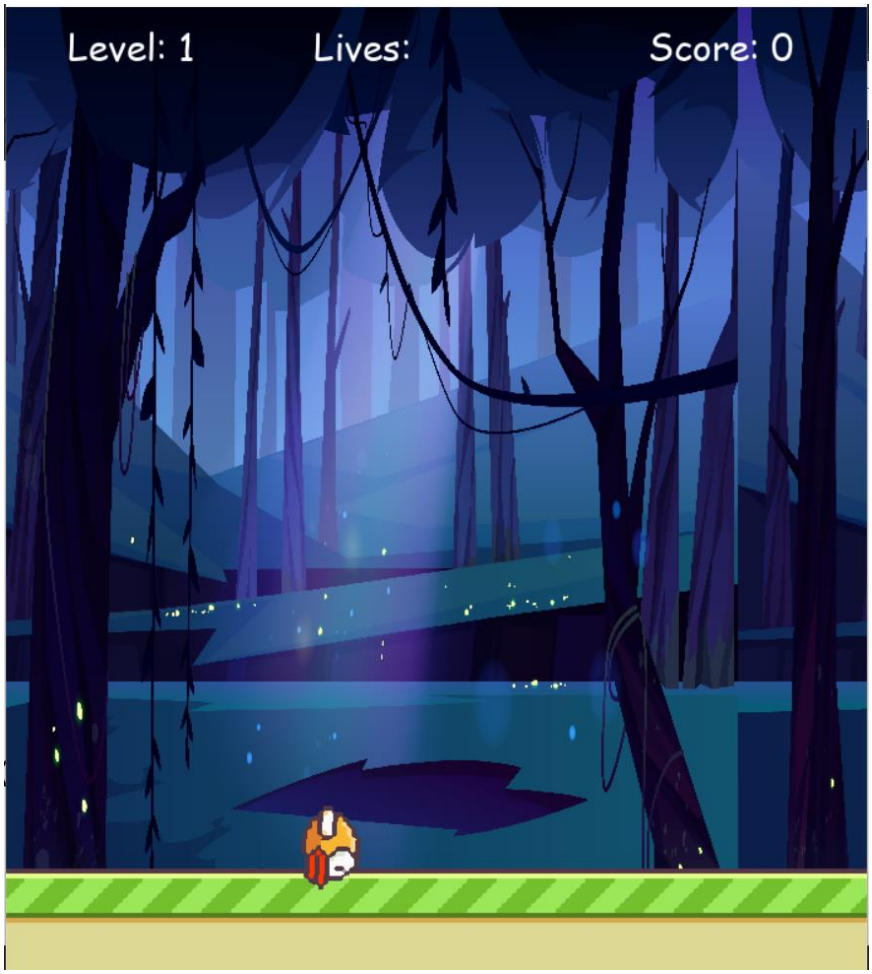
SCORE CARD

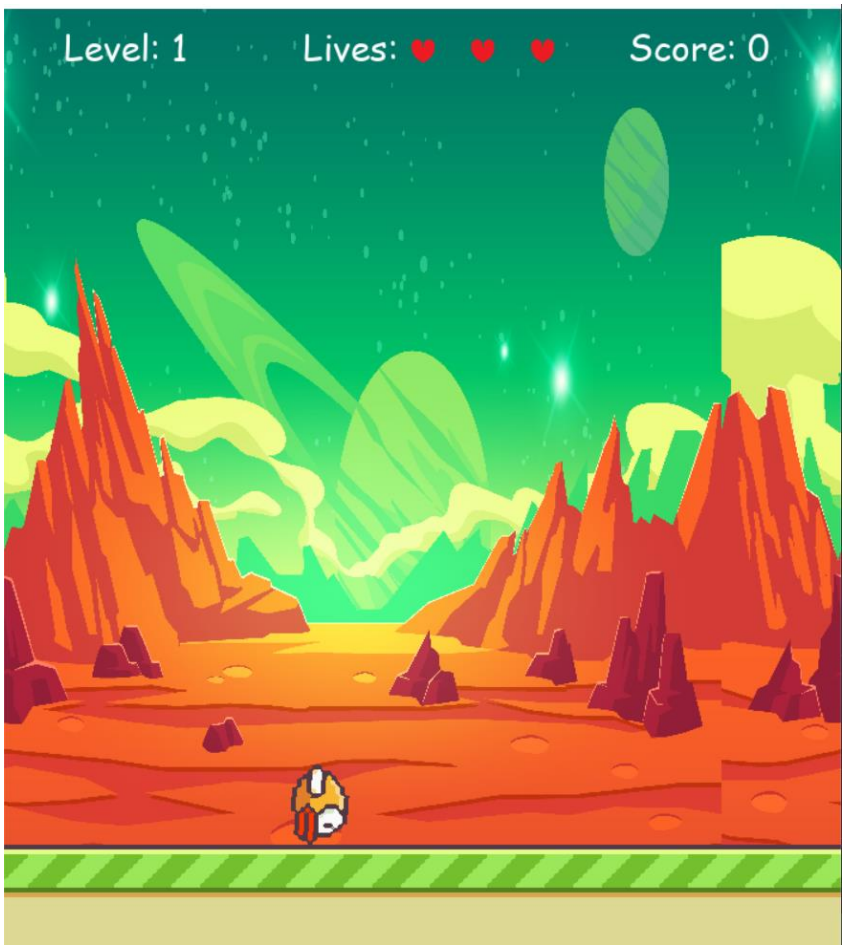
```
deepanshusingh@Deepanshus-MacBook-Air flappy-bird % python3 game.py
pygame 2.1.0 (SDL 2.0.16, Python 3.10.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  0
your score is :  1
```

There is a level indicator which will be increased for each 5 successful points scored by the player and for each level the pipes and move faster towards the bird.

There are 3 lives for each game and after all the lives are completed the game will start from initial point. The score will be added until the last life.

.





The above screen shots which has 2 different background images these will be dynamically selected each time the player starts the game

CODE

Importing all the modules

```
import pygame
import os
import sys
import random
import time
from pygame.locals import *
```

Initializing pygame window

```
pygame.font.init()
pygame.mixer.init()

W_WID=600
W_HEI=800
BASE=700
FONT=pygame.font.SysFont("comicsans",30)
WIN=pygame.display.set_mode(size=(700,800))
pygame.display.set_caption("FLAPPY BIRD GAME")
```

Loading the required images

```
welcome_img=pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","message.png"))
.convert_alpha())
pipe_img=pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","pipe.png")).conve
rt_alpha())
bg_img=pygame.transform.scale(pygame.image.load(os.path.join("imgs","bg{0}.jpg".format(ran
dom.randrange(1,3)))).convert_alpha(),(700,800))
bird_images =
[pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","bird"+str(x) +
".png")).convert_alpha()) for x in range(1,4)]
base_img=pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","base.png")).conve
rt_alpha())
game_over=pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","gameover.png")).
convert_alpha())
coins_img=pygame.transform.scale(pygame.image.load(os.path.join("imgs","coin.png")).conver
t_alpha(),(30,30))
lives=pygame.transform.scale(pygame.image.load(os.path.join("imgs","heart.png")).convert_a
lpha(),(20,20))
```

Loading the required sounds

```
SOUNDS={}
SOUNDS['music']=pygame.mixer.Sound('audio/music.mp3')
SOUNDS['die']=pygame.mixer.Sound('audio/die.wav')
SOUNDS['death']=pygame.mixer.Sound('audio/die1.wav')
SOUNDS['hit']=pygame.mixer.Sound('audio/hit.wav')
SOUNDS['point']=pygame.mixer.Sound('audio/point.wav')
SOUNDS['wing']=pygame.mixer.Sound('audio/wing.wav')
SOUNDS['swoosh']=pygame.mixer.Sound('audio/swoosh.wav')
SOUNDS['death'].set_volume(0.1)
SOUNDS['hit'].set_volume(0.1)
SOUNDS['swoosh'].set_volume(0.4)
SOUNDS['point'].set_volume(0.1)
SOUNDS['die'].set_volume(0.1)
SOUNDS['music'].set_volume(0.3)
SOUNDS['wing'].set_volume(0.1)
```

Bird Class

```
lifeCounter=3
score=0
class Bird:
    ROTATION=25
    IMGS=bird_images
    VELOCITY=17
    ANIMATION_DELAY=5

    def __init__(self,x,y):
        self.x= x
        self.y= y
        self.tilt=0
        self.tick_count=0
        self.vel=0
        self.height= self.y
        self.img_count=0
        self.img=self.IMGS[0]

    def jump(self):
        self.vel = -8
        self.tick_count=0
        self.height= self.y
    def move(self):
        self.tick_count +=1
        displacement=self.vel*(self.tick_count)+0.5*3*(self.tick_count)**2
        # displacement=5
        # terminal velocity
        if displacement>=16:
            displacement=16
```

```
        if displacement <0:
            displacement -=2
        self.y=self.y+displacement
        if displacement < 0 or self.y <self.height+50:
            if self.tilt<self.ROTATION:
                self.tilt=self.ROTATION
        else:
            if self.tilt>-90:
                self.tilt -= self.VELOCITY
    def draw(self,win):
        self.img_count+=1
        if self.img_count <= self.ANIMATION_DELAY:
            self.img = self.IMGS[0]
        elif self.img_count <= self.ANIMATION_DELAY*2:
            self.img = self.IMGS[1]
        elif self.img_count <= self.ANIMATION_DELAY*3:
            self.img = self.IMGS[2]
        elif self.img_count <= self.ANIMATION_DELAY*4:
            self.img = self.IMGS[1]
        elif self.img_count == self.ANIMATION_DELAY*4 + 1:
            self.img = self.IMGS[0]
            self.img_count = 0
        if self.tilt <= -80:
            self.img = self.IMGS[1]
            self.img_count = self.ANIMATION_DELAY*2
        blitRotateCenter(win,self.img,(self.x,self.y),self.tilt)
    def get_mask(self):
        return pygame.mask.from_surface(self.img)
```

Pipe Class

```
class Pipe():
    GAP = 250
    def __init__(self,x,vel):
        self.x=x
        self.VEL=vel
        self.height=0
        self.top=0
        self.bottom=0
        self.passed=False
        self.PIPE_TOP=pygame.transform.flip(pipe_img, False, True)
        self.PIPE_BOTTOM=pipe_img
        self.set_height()
    def set_height(self):
        self.height=random.randrange(80,450)
        self.top=self.height-self.PIPE_TOP.get_height()
        self.bottom=self.height+self.GAP
    def move(self):
        self.x-=self.VEL
```

```
    def draw(self,win):
        win.blit(self.PIPE_TOP,(self.x,self.top))
        win.blit(self.PIPE_BOTTOM,(self.x,self.bottom))
    def collide(self,bird):
        bird_mask=bird.get_mask()
        top_mask=pygame.mask.from_surface(self.PIPE_TOP)
        bottom_mask=pygame.mask.from_surface(self.PIPE_BOTTOM)
        top_offset=(self.x-bird.x,self.top-bird.y)
        bottom_offset=(self.x-bird.x,self.bottom-bird.y)
        b_point=bird_mask.overlap(bottom_mask,bottom_offset)
        t_point=bird_mask.overlap(top_mask,top_offset)
        if b_point or t_point:
            return True
        return False
```

Base Class

```
class Base:
    VEL=5
    WIDTH=base_img.get_width()
    IMG=base_img
    def __init__(self,y):
        self.y=y
        self.x1=0
        self.x2=self.WIDTH
    def move(self):
        self.x1-=self.VEL
        self.x2-=self.VEL
        if self.x1+self.WIDTH<0:
            self.x1=self.x2+self.WIDTH
        if self.x2+self.WIDTH<0:
            self.x2=self.x1+self.WIDTH
    def draw(self,win):
        win.blit(self.IMG,(self.x1,self.y))
        win.blit(self.IMG,(self.x2,self.y))
    def collide(self,bird):
        bird_mask=bird.get_mask()
        baseMask=pygame.mask.from_surface(self.IMG)
        base_offset=(0-bird.x,self.y-bird.y)
        b_point=bird_mask.overlap(baseMask,base_offset)
        if b_point:
            return True
        return False
```

Background image class

```
class BIMG:
    VEL=5
    WIDTH=bg_img.get_width()
    IMG=bg_img
    def __init__(self,x,y):
        self.posX=x
        self.posY=y
        self.posX2=self.WIDTH
    def move(self):
        self.posX-=self.VEL
        self.posX2-=self.VEL
        if self.posX+self.WIDTH<0:
            self.posX=self.posX2+self.WIDTH
        if self.posX2+self.WIDTH<0:
            self.posX2=self.posX+self.WIDTH
    def draw(self,win):
        win.blit(self.IMG,(self.posX,self.posY))
        win.blit(self.IMG,(self.posX2,self.posY))
```

Functions for rotating bird and drawing the window

```
def blitRotateCenter(win,image,topleft,angle):
    rotated_image=pygame.transform.rotate(image, angle)
    new_rect=rotated_image.get_rect(center=image.get_rect(topleft=topleft).center)
    win.blit(rotated_image,new_rect.topleft)

def draw_window(win,bimg,bird,pipes,base,score,level):
    # win.blit(bg_img,(0,0))
    bimg.draw(win)
    for pipe in pipes:
        pipe.draw(win)
    base.draw(win)
    bird.draw(win)
    life_label=FONT.render("Lives:",1,(255,255,255))
    score_label=FONT.render("Score: "+str(score),1,(255,255,255))
    level_label=FONT.render("Level: "+str(level),1,(255,255,255))
    win.blit(score_label,(W_WID-score_label.get_width()+40,10))
    win.blit(level_label,(50,10))
    win.blit(life_label,(250,10))
```

```
    return livesOfUser(win)
def livesOfUser(win):
    global lifeCounter
    xPos=340
    for i in range(lifeCounter):
        win.blit(lives,(xPos,25))
        xPos+=50
    if(lifeCounter==0):
        pygame.display.update()
        return True
    pygame.display.update()
    return False
```

The main game function for game

```
def main_game():
    global lifeCounter
    global score
    global WIN
    win=WIN
    pipeVelocity=5
    bimg=BIMG(0,0)
    bird=Bird(230,350)
    base=Base(BASE)
    pipes=[Pipe(1000,pipeVelocity)]
    level=1
    clock=pygame.time.Clock()
    while True:
        clock.tick(30)
        bird.move()
        bimg.move()
        for event in pygame.event.get():
            if event.type==QUIT or (event.type==KEYDOWN and event.key==K_ESCAPE):
                print("your score is : ",score)
                pygame.quit()
                sys.exit()
            if event.type==KEYDOWN and (event.key==K_SPACE or event.key==K_UP):
                SOUNDS['wing'].play()
                bird.jump()

        base.move()
        if base.collide(bird) or bird.y<-5:
            if(lifeCounter>0):
                lifeCounter-=1
                # livesOfUser(win)
                SOUNDS['die'].play()
                time.sleep(0.5)
                main_game()
        rem=[]
```

```
        add_pipe=False

        for pipe in pipes:
            pipe.move()
            if pipe.collide(bird):
                if(lifeCounter>0):
                    lifeCounter-=1
                    # livesOfUser(win)
                    SOUNDS['die'].play()
                    time.sleep(0.5)
                    main_game()
            if pipe.x+pipe.PIPE_TOP.get_width()<0:
                rem.append(pipe)
            if not pipe.passed and pipe.x+30<bird.x:
                pipe.passed=True
                add_pipe=True
        if add_pipe:
            score+=1
            if(score%5==0):
                pipes.pop(0)
                pipeVelocity+=1
                base.VEL+=1
                bimg.VEL+=1
                level+=1
            SOUNDS['point'].play()
            pipes.append(Pipe(W_WID+100,pipeVelocity))
        for r in rem:
            pipes.remove(r)
        died=draw_window(WIN,bimg,bird,pipes,base,score,level)
        if died:
            return
```

Welcome Screen

```
def welcomeScreen():
    global WIN
    win=WIN
    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):
                SOUNDS['music'].play(loops=-1,fade_ms=100)
                SOUNDS['swoosh'].play()
                return
            else:
                win.blit(bg_img,(0,0))
                win.blit(base_img,(0,BASE))
```

```
                win.blit(base_img,(672,BASE))
                win.blit(welcome_img,(180,150))
                pygame.display.update()
```

This start game function calls the welcome screen and main game

```
def startGame():
    while True:
        global lifeCounter,score
        lifeCounter,score=3,0
        welcomeScreen()
        main_game()
        SOUNDS['music'].fadeout(100)
        SOUNDS['death'].play()
        time.sleep(1)
startGame()
```

Conclusions

It is our team's hope that this document will be of huge with understanding of our little project as we have used a different approach which has proved beneficial for us and easy for us to understand the vast community Gaming. We have reached the maximum accuracy of 90% after making few of the changes in the game. We will try to achieve the remaining accuracy later on

References

https://www.pygame.org/docs/

https://docs.python.org/3/

https://www.101soundboards.com/boards/10178-flappy-bird-sounds

https://www.pngitem.com/so/flappy-bird/

https://stackoverflow.com/questions/tagged/pygame

https://stackoverflow.com/questions/29640685/how-do-i-detect-collision-in-pygame

https://stackoverflow.com/questions/14087609/smooth-keyboard-movement-in-pygame

https://www.reddit.com/r/pygame/comments/2cu9x2/animations_a_little_choppy_dropped_frames_any_way/