```
# import tkinter as tk
from tkinter import *
from tkinter import messagebox
import pygame,sys,random

#Main Window
root=Tk() #To initialize tkinter
root.title('ARCADE') #Title of window
root.geometry("900x620") #Size of window
root.resizable(height = False, width = False)

p1 = PhotoImage(file = 'assets/arcade.png') #Display Image
root.iconphoto(False, p1) #Title Bar Icon

bg = PhotoImage(file = "assets/bg1.png")
canvas1 = Canvas( root, width = 900,height = 620)  #rectangular area
for drawing pictures or other complex layouts.
canvas1.pack(fill = "both", expand = True) # geometry manager
organizes widgets in blocks before placing them in the parent widget.
canvas1.create_image( 0, 0, image = bg,anchor = "nw") #Creates an
image item.
                                                        #Anchors are
used to define where text is positioned relative to a reference point.


#Font

f1=("MS UI Gothic", 18, "bold")
f2=(("Franklin Gothic Demi", 8, "bold"))
f3=(("Franklin Gothic Demi", 36, "bold"))
f4=("MS UI Gothic", 12, "bold")


l1=Label(root,text="FLAPPY
BIRD",bg="#360062",fg="white",relief=GROOVE,borderwidth=5,font=f3)
l1.place(x=300,y=20)

#How To Play
def show1():
    messagebox.showinfo("HOW TO PLAY",
                        "1.Tap the SPACEBAR to PLAY. Tap the SPACEBAR
again and \t   again to allow your bird to fly and to start the
game\n"
                        "2.The faster you tap, the higher you go. Each
tap represents a    wing flap and higher flight.\n"
                        "3.Once you stop, you drop towards the ground.
GAME OVER\n"
                        "4.If the bird disappers from the game screen.
GAME OVER\n"
                        "5.If you hit a pipe or the ground. GAME
OVER\n"
                        )
b=Button(root,text="HOW TO
PLAY",command=lambda:show1(),bg="red",fg='white',relief=RAISED,borderw
idth=3,font=f2)
b.place(x=100,y=580)
```

```
#Exit

def close():
    result=messagebox.askquestion('Quit','You are quitting the game,\n
Are you sure ?')
    if result=='yes':
        sys.exit()
B=Button(root,text="EXIT",command=lambda:close(),bg="red",fg='white',r
elief=RAISED,borderwidth=3,font=f2)
B.place(x=700,y=580)

#Day Mode

b1=Label(root,text="DAY
MODE",bg="#cc0099",fg='white',relief=GROOVE,borderwidth=3,font=f1)
b1.place(x=225,y=140)
bN=Button(root,text="PLAY",command=lambda:day(),bg="green",fg='white',
relief=RAISED,borderwidth=3,font=f4)
bN.place(x=540,y=140)

def day():
    try:
        pygame.display.set_caption('Day Mode') #Set the current window
caption

        a = pygame.image.load('assets/bluebird-midflap.png')
        pygame.display.set_icon(a)


pygame.mixer.pre_init(frequency=60000,size=-16,channels=1,buffer=512)
#preset the mixer init arguments
        pygame.init() #initialize all imported pygame modules


        # Pipe Logic

        def create_pipe():
            '''creates pipe at bottom and top'''
            random_pipe_pos=random.choice(pipe_height)
            bottom_pipe=pipe_surface.get_rect(midtop=
(600,random_pipe_pos)) #get the rectangular area of the Surface
            top_pipe=pipe_surface.get_rect(midbottom=
(600,random_pipe_pos-150))
            return bottom_pipe,top_pipe

        def move_pipes(pipes):
            '''moves the pipes'''
            for pipe in pipes:
                pipe.centerx-=9
            return pipes

        def draw_pipes(pipes):
            '''Bottom pipes remain bottom pipes and top pipes are
flipped to become top pipes'''
```

```
            for pipe in pipes:
                if pipe.bottom>=500:
                    screen.blit(pipe_surface,pipe) #Draw the image to
the screen at the given position
                else:

flip_pipe=pygame.transform.flip(pipe_surface,False,True) #flip
vertically and horizontally
                    screen.blit(flip_pipe,pipe)
                    #False for x direction and true for y direction


        #Collision Logic

        def check_collision(pipes):
            '''checks collision of pipes with bird'''
            for pipe in pipes:
                if bird_rect.colliderect(pipe):

                    return False

            if bird_rect.top<=-100 or bird_rect.bottom>=500: #y co-
ordinates
                return False

            return True


        #Bird Logic

        def rotate_bird(bird):
            '''Rotates the bird '''
            new_bird=pygame.transform.rotozoom(bird,-
bird_movement*3,1) #filtered scale and rotation
            return new_bird
            # rotozoom takes 3 arguments: The surface to be rotated,
angle,scale

        def bird_flying():
            '''returns the new bird'''
            new_bird=bird_frames[bird_index]
            new_bird_rect=new_bird.get_rect(center=
(50,bird_rect.centery))
            return new_bird,new_bird_rect


        #Score Logic

        def display_score(game_state):
            '''displays score in different game states'''
            if game_state=="play_game":
                score_surface=game_font.render(f'Score:
{int(score)}',True,(0,0,153)) # used to create a Surface object from
the text, which then can be blit to the screen. It can only render
single lines.
                score_rect=score_surface.get_rect(center=(250,25))
                screen.blit(score_surface,score_rect)
```

```
            if game_state=="Game_over":
                score_surface=game_font.render(f'Score:
{int(score)}',True,(255,255,0))
                score_rect=score_surface.get_rect(center=(250,200))
                screen.blit(score_surface,score_rect)

                high_score_surface=game_font.render(f'High Score:
{int(high_score)}',True,(255,204,0))
                high_score_rect=high_score_surface.get_rect(center=
(250,300))
                screen.blit(high_score_surface,high_score_rect)

                Game_restart_surface=game_font.render('Click to
Restart Game',True,(255,153,51))

Game_restart_rect=Game_restart_surface.get_rect(center=(250,400))
                screen.blit(Game_restart_surface,Game_restart_rect)


        def score_update(score,high_score):
            '''updates highscore'''
            if score>high_score:
                high_score=score
            return high_score

        def cross_prev_highscore(score,highscore):
            ''' plays ringing sound if previous highscore is
crossed'''
            if score>highscore:
                return highscore_sound.play()

        screen = pygame.display.set_mode((500,600)) #Initialize a
window or screen for display
        clock=pygame.time.Clock()
        game_font=pygame.font.Font('assets/04B_19.ttf',30)


        # Game variables
        Gravity= 0.25
        bird_movement=0
        score=0
        high_score=0


        #Game Dispaly

        game_active=True

        #Background
        bg_surface=pygame.image.load('assets/background-
day.png').convert() #pixel format not as same as the requested source,
but its optimized for fast alpha blitting to the destination.
        bg_surface=pygame.transform.scale(bg_surface,(500,600))
#resize to new resolution

        #Base/Floor
        floor_surface=pygame.image.load('assets/base.png').convert()
```

```
#load new image from a file
        floor_surface=pygame.transform.scale(floor_surface,(500,150))
        floor_x_position=0

        #Bird
        bird_dsurface=pygame.image.load('assets/bluebird-
downflap.png').convert_alpha() #change the pixel format of an image
including per pixel alphas
        bird_dsurface=pygame.transform.scale(bird_dsurface,(40,30))
        bird_msurface=pygame.image.load('assets/bluebird-
midflap.png').convert_alpha()
        bird_msurface=pygame.transform.scale(bird_msurface,(40,30))
        bird_usurface=pygame.image.load('assets/bluebird-
downflap.png').convert_alpha()
        bird_usurface=pygame.transform.scale(bird_usurface,(40,30))
        bird_downflap=pygame.transform.scale(bird_dsurface,(40,30))
        bird_midflap=pygame.transform.scale(bird_msurface,(40,30))
        bird_upflap=pygame.transform.scale(bird_usurface,(40,30))
        bird_frames=[bird_downflap,bird_midflap,bird_upflap]
        bird_index=0
        bird_surface=bird_frames[bird_index]
        bird_rect=bird_surface.get_rect(center=(50,150))


        ''' User Events :
            1.Flapping of Bird
            2.Creation of Pipes '''

        BIRDFLAP=pygame.USEREVENT+1 #Pygame Events created by the user
        pygame.time.set_timer(BIRDFLAP,300) #repeatedly create an
event on the event queue

        pipe_surface=pygame.image.load('assets/pipe-
green.png').convert()
        pipe_surface=pygame.transform.scale(pipe_surface,(75,400))
        pipe_surface_y_position=300
        pipe_list=[]
        SPAWNPIPE=pygame.USEREVENT
        pygame.time.set_timer(SPAWNPIPE,900)
        pipe_height=[400,300,200]


Game_over_surface=pygame.image.load('assets/oover.png').convert()
        Game_over_surface=pygame.transform.scale(Game_over_surface,
(500,500))
        Game_over_rect=Game_over_surface.get_rect(center=(250,250))

        birdflap_sound=pygame.mixer.Sound('assets/sound_sfx_wing.wav')
#Create a new Sound object from a file or buffer object
        dead_sound=pygame.mixer.Sound('assets/sound_sfx_hit.wav')

highscore_sound=pygame.mixer.Sound('assets/sound_sfx_point.wav')




        while True:
```

```
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit() #opp of pygame.init(); it runs code
that deactivates the pygame lib
                if event.type == pygame.MOUSEBUTTONDOWN:
                    if event.button == 1 and game_active==True :
                        bird_movement=0
                        bird_movement-=5
                        birdflap_sound.play()
                    if event.button == 1 and game_active==False:
                        game_active==True
                        pipe_list.clear()
                        bird_rect.center= (50,150)
                        bird_movement=0
                        score=0
                if event.type == pygame.KEYDOWN:

                    if event.key == pygame.K_SPACE and
game_active==True:
                        bird_movement=0
                        bird_movement-=5
                        birdflap_sound.play()
                    if event.key == pygame.K_SPACE and
game_active==False:
                        game_active==True
                        pipe_list.clear()
                        bird_rect.center= (50,150)
                        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_flying()


            #background

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



            if game_active:

                # Loading and playing background music:
                pygame.mixer.music.load('assets/mixkit-sad-game-over-
trombone-471.wav')
                pygame.mixer.music.play()
                #bird
                bird_movement+=Gravity
                rotated_bird=rotate_bird(bird_surface)
```

```
                bird_rect.centery+=bird_movement
                screen.blit(rotated_bird,bird_rect)
                #pipe
                pipe_list=move_pipes(pipe_list)
                draw_pipes(pipe_list)

                score+=0.02
                cross_prev_highscore(score,high_score)
                display_score('play_game')
            else:
                screen.blit(Game_over_surface,Game_over_rect)
                high_score=score_update(score,high_score)
                display_score('Game_over')


            #check collision
            game_active=check_collision(pipe_list)

            #floor
            screen.blit(floor_surface,(floor_x_position,500))
            screen.blit(floor_surface,(floor_x_position+500,500))



        #floor
            floor_x_position-=3

            if floor_x_position<=-500:
                floor_x_position=0



            pygame.display.update() # Updating the display surface
            clock.tick(50)

    except pygame.error:
        pass

#Night Mode

b2=Label(root,text="NIGHT
MODE",bg="#cc0099",fg='white',relief=GROOVE,borderwidth=3,font=f1)
b2.place(x=165,y=270)
bN=Button(root,text="PLAY",command=lambda:night(),bg="green",fg='white
',relief=RAISED,borderwidth=3,font=f4)
bN.place(x=580,y=270)

def night():
    try:
        pygame.display.set_caption('Night Mode') #Set the current
window caption

        a = pygame.image.load('assets/redbird-midflap.png')
        pygame.display.set_icon(a)


pygame.mixer.pre_init(frequency=60000,size=-16,channels=1,buffer=512)
```

```
#preset the mixer init arguments
        pygame.init() #initialize all imported pygame modules


        # Pipe Logic

        def create_pipe():
            '''creates pipe at bottom and top'''
            random_pipe_pos=random.choice(pipe_height)
            bottom_pipe=pipe_surface.get_rect(midtop=
(600,random_pipe_pos)) #get the rectangular area of the Surface
            top_pipe=pipe_surface.get_rect(midbottom=
(600,random_pipe_pos-150))
            return bottom_pipe,top_pipe

        def move_pipes(pipes):
            '''moves the pipes'''
            for pipe in pipes:
                pipe.centerx-=9
            return pipes

        def draw_pipes(pipes):
            '''Bottom pipes remain bottom pipes and top pipes are
flipped to become top pipes'''
            for pipe in pipes:
                if pipe.bottom>=500:
                    screen.blit(pipe_surface,pipe) #Draw the image to
the screen at the given position
                else:

flip_pipe=pygame.transform.flip(pipe_surface,False,True) #flip
vertically and horizontally
                    screen.blit(flip_pipe,pipe)
                    #False for x direction and true for y direction


        #Collision Logic

        def check_collision(pipes):
            '''checks collision of pipes with bird'''
            for pipe in pipes:
                if bird_rect.colliderect(pipe):

                    return False

            if bird_rect.top<=-100 or bird_rect.bottom>=500: #y co-
ordinates
                return False

            return True


        #Bird Logic

        def rotate_bird(bird):
            '''Rotates the bird '''
            new_bird=pygame.transform.rotozoom(bird,-
```

```
bird_movement*3,1) #filtered scale and rotation
            return new_bird
            # rotozoom takes 3 arguments: The surface to be rotated,
angle,scale

        def bird_flying():
            '''returns new bird'''
            new_bird=bird_frames[bird_index]
            new_bird_rect=new_bird.get_rect(center=
(50,bird_rect.centery))
            return new_bird,new_bird_rect


        #Score Logic

        def display_score(game_state):
            '''displays the score in different game states'''
            if game_state=="play_game":
                score_surface=game_font.render(f'Score:
{int(score)}',True,(0,0,153)) # used to create a Surface object from
the text, which then can be blit to the screen. It can only render
single lines.
                score_rect=score_surface.get_rect(center=(250,25))
                screen.blit(score_surface,score_rect)
            if game_state=="Game_over":
                score_surface=game_font.render(f'Score:
{int(score)}',True,(255,255,0))
                score_rect=score_surface.get_rect(center=(250,200))
                screen.blit(score_surface,score_rect)

                high_score_surface=game_font.render(f'High Score:
{int(high_score)}',True,(255,204,0))
                high_score_rect=high_score_surface.get_rect(center=
(250,300))
                screen.blit(high_score_surface,high_score_rect)

                Game_restart_surface=game_font.render('Click to
Restart Game',True,(255,153,51))

Game_restart_rect=Game_restart_surface.get_rect(center=(250,400))
                screen.blit(Game_restart_surface,Game_restart_rect)


        def score_update(score,high_score):
            '''updates highscore'''
            if score>high_score:
                high_score=score
            return high_score

        def cross_prev_highscore(score,highscore):
            ''' plays ringing sound if previous highscore is
crossed'''
            if score>highscore:
                return highscore_sound.play()

        screen = pygame.display.set_mode((500,600)) #Initialize a
window or screen for display
```

```
        clock=pygame.time.Clock()
        game_font=pygame.font.Font('assets/04B_19.ttf',30)


        # Game variables
        Gravity= 0.25
        bird_movement=0
        score=0
        high_score=0


        #Game Dispaly

        game_active=True

        #Background
        bg_surface=pygame.image.load('assets/background-
night.png').convert() #pixel format not as same as the requested
source, but its optimized for fast alpha blitting to the destination.
        bg_surface=pygame.transform.scale(bg_surface,(500,600))
#resize to new resolution

        #Base/Floor
        floor_surface=pygame.image.load('assets/base.png').convert()
#load new image from a file
        floor_surface=pygame.transform.scale(floor_surface,(500,150))
        floor_x_position=0

        #Bird
        bird_dsurface=pygame.image.load('assets/redbird-
downflap.png').convert_alpha() #change the pixel format of an image
including per pixel alphas
        bird_dsurface=pygame.transform.scale(bird_dsurface,(40,30))
        bird_msurface=pygame.image.load('assets/redbird-
midflap.png').convert_alpha()
        bird_msurface=pygame.transform.scale(bird_msurface,(40,30))
        bird_usurface=pygame.image.load('assets/redbird-
downflap.png').convert_alpha()
        bird_usurface=pygame.transform.scale(bird_usurface,(40,30))
        bird_downflap=pygame.transform.scale(bird_dsurface,(40,30))
        bird_midflap=pygame.transform.scale(bird_msurface,(40,30))
        bird_upflap=pygame.transform.scale(bird_usurface,(40,30))
        bird_frames=[bird_downflap,bird_midflap,bird_upflap]
        bird_index=0
        bird_surface=bird_frames[bird_index]
        bird_rect=bird_surface.get_rect(center=(50,150))


        ''' User Events :
            1.Flapping of Bird
            2.Creation of Pipes '''

        BIRDFLAP=pygame.USEREVENT+1 #Pygame Events created by the user
        pygame.time.set_timer(BIRDFLAP,300) #repeatedly create an
event on the event queue

        pipe_surface=pygame.image.load('assets/pipe-
```

```
brown.png').convert()
        pipe_surface=pygame.transform.scale(pipe_surface,(75,400))
        pipe_surface_y_position=300
        pipe_list=[]
        SPAWNPIPE=pygame.USEREVENT
        pygame.time.set_timer(SPAWNPIPE,900)
        pipe_height=[400,300,200]


Game_over_surface=pygame.image.load('assets/oover.png').convert()
        Game_over_surface=pygame.transform.scale(Game_over_surface,
(500,500))
        Game_over_rect=Game_over_surface.get_rect(center=(250,250))

        birdflap_sound=pygame.mixer.Sound('assets/sound_sfx_wing.wav')
#Create a new Sound object from a file or buffer object
        dead_sound=pygame.mixer.Sound('assets/sound_sfx_hit.wav')

highscore_sound=pygame.mixer.Sound('assets/sound_sfx_point.wav')




        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit() #opp of pygame.init(); it runs code
that deactivatesthe pygame lib
                if event.type == pygame.MOUSEBUTTONDOWN:
                    if event.button == 1 and game_active==True :
                        bird_movement=0
                        bird_movement-=5
                        birdflap_sound.play()
                    if event.button == 1 and game_active==False:
                        game_active==True
                        pipe_list.clear()
                        bird_rect.center= (50,150)
                        bird_movement=0
                        score=0
                if event.type == pygame.KEYDOWN:

                    if event.key == pygame.K_SPACE and
game_active==True:
                        bird_movement=0
                        bird_movement-=5
                        birdflap_sound.play()
                    if event.key == pygame.K_SPACE and
game_active==False:
                        game_active==True
                        pipe_list.clear()
                        bird_rect.center= (50,150)
                        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_flying()


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


        if game_active:

            # Loading and playing background music:
            pygame.mixer.music.load('assets/mixkit-sad-game-over-
trombone-471.wav')
            pygame.mixer.music.play()
            #bird
            bird_movement+=Gravity
            rotated_bird=rotate_bird(bird_surface)
            bird_rect.centery+=bird_movement
            screen.blit(rotated_bird,bird_rect)
            #pipe
            pipe_list=move_pipes(pipe_list)
            draw_pipes(pipe_list)

            score+=0.02
            cross_prev_highscore(score,high_score)
            display_score('play_game')
        else:
            screen.blit(Game_over_surface,Game_over_rect)
            high_score=score_update(score,high_score)
            display_score('Game_over')

        #check collision
        game_active=check_collision(pipe_list)

        #floor
        screen.blit(floor_surface,(floor_x_position,500))
        screen.blit(floor_surface,(floor_x_position+500,500))



        #floor
        floor_x_position-=3

        if floor_x_position<=-500:
            floor_x_position=0


        pygame.display.update() # Updating the display surface
        clock.tick(50)
    except pygame.error:
        pass
```

```
#Retro Mode

b3=Label(root,text="RETRO
MODE",bg="#cc0099",fg='white',relief=GROOVE,borderwidth=3,font=f1)
b3.place(x=215,y=410)
bN=Button(root,text="PLAY",command=lambda:retro(),bg="green",fg='white
',relief=RAISED,borderwidth=3,font=f4)
bN.place(x=520,y=410)

def retro():
    try:
        pygame.display.set_caption('Retro Mode') #Set the current
window caption

        a = pygame.image.load('assets/yellowbird-midflap.png')
        pygame.display.set_icon(a)


pygame.mixer.pre_init(frequency=60000,size=-16,channels=1,buffer=512)
#preset the mixer init arguments
        pygame.init() #initialize all imported pygame modules


        # Pipe Logic

        def create_pipe():
            '''creates pipe at bottom and top'''
            random_pipe_pos=random.choice(pipe_height)
            bottom_pipe=pipe_surface.get_rect(midtop=
(600,random_pipe_pos)) #get the rectangular area of the Surface
            top_pipe=pipe_surface.get_rect(midbottom=
(600,random_pipe_pos-150))
            return bottom_pipe,top_pipe

        def move_pipes(pipes):
            '''moves the pipes'''
            for pipe in pipes:
                pipe.centerx-=9
            return pipes

        def draw_pipes(pipes):
            '''Bottom pipes remain bottom pipes and top pipes are
flipped to become top pipes'''
            for pipe in pipes:
                if pipe.bottom>=500:
                    screen.blit(pipe_surface,pipe) #Draw the image to
the screen at the given position
                else:

flip_pipe=pygame.transform.flip(pipe_surface,False,True) #flip
vertically and horizontally
                    screen.blit(flip_pipe,pipe)
                    #False for x direction and true for y direction


        #Collision Logic
```

```
        def check_collision(pipes):
            '''checks collision of pipes with bird'''
            for pipe in pipes:
                if bird_rect.colliderect(pipe):

                    return False

            if bird_rect.top<=-100 or bird_rect.bottom>=500: #y co-
ordinates
                return False

            return True


        #Bird Logic

        def rotate_bird(bird):
            '''Rotates the bird '''
            new_bird=pygame.transform.rotozoom(bird,-
bird_movement*3,1) #filtered scale and rotation
            return new_bird
            # rotozoom takes 3 arguments: The surface to be rotated,
angle,scale

        def bird_flying():
            '''returns the new bird'''
            new_bird=bird_frames[bird_index]
            new_bird_rect=new_bird.get_rect(center=
(50,bird_rect.centery))
            return new_bird,new_bird_rect


        #Score Logic

        def display_score(game_state):
            '''displays score in different game states'''
            if game_state=="play_game":
                score_surface=game_font.render(f'Score:
{int(score)}',True,(255,255,0)) # used to create a Surface object from
the text, which then can be blit to the screen. It can only render
single lines.
                score_rect=score_surface.get_rect(center=(250,25))
                screen.blit(score_surface,score_rect)
            if game_state=="Game_over":
                score_surface=game_font.render(f'Score:
{int(score)}',True,(255,255,0))
                score_rect=score_surface.get_rect(center=(250,200))
                screen.blit(score_surface,score_rect)

                high_score_surface=game_font.render(f'High Score:
{int(high_score)}',True,(255,204,0))
                high_score_rect=high_score_surface.get_rect(center=
(250,300))
                screen.blit(high_score_surface,high_score_rect)

                Game_restart_surface=game_font.render('Click to
Restart Game',True,(255,153,51))
```

```
Game_restart_rect=Game_restart_surface.get_rect(center=(250,400))
                screen.blit(Game_restart_surface,Game_restart_rect)


        def score_update(score,high_score):
            '''updates highscore'''
            if score>high_score:
                high_score=score
            return high_score

        def cross_prev_highscore(score,highscore):
            ''' plays ringing sound if previous highscore is
crossed'''
            if score>highscore:
                return highscore_sound.play()

        screen = pygame.display.set_mode((500,600)) #Initialize a
window or screen for display
        clock=pygame.time.Clock()
        game_font=pygame.font.Font('assets/04B_19.ttf',30)


        # Game variables
        Gravity= 0.25
        bird_movement=0
        score=0
        high_score=0


        #Game Dispaly

        game_active=True

        #Background
        bg_surface=pygame.image.load('assets/background-
retro.png').convert() #pixel format not as same as the requested
source, but its optimized for fast alpha blitting to the destination.
        bg_surface=pygame.transform.scale(bg_surface,(500,600))
#resize to new resolution

        #Base/Floor
        floor_surface=pygame.image.load('assets/base-
retro.png').convert() #load new image from a file
        floor_surface=pygame.transform.scale(floor_surface,(500,150))
        floor_x_position=0

        #Bird
        bird_dsurface=pygame.image.load('assets/yellowbird-
downflap.png').convert_alpha() #change the pixel format of an image
including per pixel alphas
        bird_dsurface=pygame.transform.scale(bird_dsurface,(40,30))
        bird_msurface=pygame.image.load('assets/yellowbird-
midflap.png').convert_alpha()
        bird_msurface=pygame.transform.scale(bird_msurface,(40,30))
        bird_usurface=pygame.image.load('assets/yellowbird-
downflap.png').convert_alpha()
```

```
        bird_usurface=pygame.transform.scale(bird_usurface,(40,30))
        bird_downflap=pygame.transform.scale(bird_dsurface,(40,30))
        bird_midflap=pygame.transform.scale(bird_msurface,(40,30))
        bird_upflap=pygame.transform.scale(bird_usurface,(40,30))
        bird_frames=[bird_downflap,bird_midflap,bird_upflap]
        bird_index=0
        bird_surface=bird_frames[bird_index]
        bird_rect=bird_surface.get_rect(center=(50,150))


        ''' User Events :
            1.Flapping of Bird
            2.Creation of Pipes '''

        BIRDFLAP=pygame.USEREVENT+1 #Pygame Events created by the user
        pygame.time.set_timer(BIRDFLAP,300) #repeatedly create an
event on the event queue

        pipe_surface=pygame.image.load('assets/pipe-
brown.png').convert()
        pipe_surface=pygame.transform.scale(pipe_surface,(65,400))
        pipe_surface_y_position=300
        pipe_list=[]
        SPAWNPIPE=pygame.USEREVENT
        pygame.time.set_timer(SPAWNPIPE,900)
        pipe_height=[400,300,200]


Game_over_surface=pygame.image.load('assets/oover.png').convert()
        Game_over_surface=pygame.transform.scale(Game_over_surface,
(500,500))
        Game_over_rect=Game_over_surface.get_rect(center=(250,250))

        birdflap_sound=pygame.mixer.Sound('assets/sound_sfx_wing.wav')
#Create a new Sound object from a file or buffer object
        dead_sound=pygame.mixer.Sound('assets/sound_sfx_hit.wav')

highscore_sound=pygame.mixer.Sound('assets/sound_sfx_point.wav')




        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit() #opp of pygame.init(); it runs code
that deactivatesthe pygame lib
                if event.type == pygame.MOUSEBUTTONDOWN:
                    if event.button == 1 and game_active==True :
                        bird_movement=0
                        bird_movement-=5
                        birdflap_sound.play()
                    if event.button == 1 and game_active==False:
                        game_active==True
                        pipe_list.clear()
                        bird_rect.center= (50,150)
                        bird_movement=0
```

```
                    score=0
                if event.type == pygame.KEYDOWN:

                    if event.key == pygame.K_SPACE and
game_active==True:
                        bird_movement=0
                        bird_movement-=5
                        birdflap_sound.play()
                    if event.key == pygame.K_SPACE and
game_active==False:
                        game_active==True
                        pipe_list.clear()
                        bird_rect.center= (50,150)
                        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_flying()


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


            if game_active:

                # Loading and playing background music:
                pygame.mixer.music.load('assets/mixkit-sad-game-over-
trombone-471.wav')
                pygame.mixer.music.play()
                #bird
                bird_movement+=Gravity
                rotated_bird=rotate_bird(bird_surface)
                bird_rect.centery+=bird_movement
                screen.blit(rotated_bird,bird_rect)
                #pipe
                pipe_list=move_pipes(pipe_list)
                draw_pipes(pipe_list)

                score+=0.02
                cross_prev_highscore(score,high_score)
                display_score('play_game')
            else:
                screen.blit(Game_over_surface,Game_over_rect)
                high_score=score_update(score,high_score)
                display_score('Game_over')

            #check collision
            game_active=check_collision(pipe_list)
```

```
            #floor
            screen.blit(floor_surface,(floor_x_position,500))
            screen.blit(floor_surface,(floor_x_position+500,500))



        #floor
            floor_x_position-=3

            if floor_x_position<=-500:
                floor_x_position=0



            pygame.display.update() # Updating the display surface
            clock.tick(50)
    except pygame.error:
        pass

root.mainloop() # Tkinter to start running the application
                #loop forever until the user exits the window or waits
for any events from the user.
```