```
import pygame
import random
import os
pygame.font.init()

WIN_WIDTH = 600
WIN_HEIGHT = 800
PIPE_VEL = 3
FLOOR = 730
STAT_FONT = pygame.font.SysFont("comicsans", 50)
END_FONT = pygame.font.SysFont("comicsans", 40)

WIN = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
pygame.display.set_caption("Flappy Bird")
pipe_img =
pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","pipe.png"
)).convert_alpha())
bg_img =
pygame.transform.scale(pygame.image.load(os.path.join("imgs","bg.png")).c
onvert_alpha(), (600, 900))
bird_images =
[pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","bird" +
str(x) + ".png"))) for x in range(1,4)]
base_img =
pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","base.png"
)).convert_alpha())

class Bird:
    WIN_HEIGHT = 0
    WIN_WIDTH = 0
    MAX_ROTATION = 25
    IMGS = bird_images
    ROT_VEL = 20
    ANIMATION_TIME = 5

    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.gravity = 9.8
        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 = -10.5
        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

        if displacement >= 16:
            displacement = (displacement/abs(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.MAX_ROTATION:
                self.tilt = self.MAX_ROTATION
        else:
            if self.tilt > -90:
                self.tilt -= self.ROT_VEL

    def draw(self, win):

        self.img_count += 1

        if self.img_count <= self.ANIMATION_TIME:
            self.img = self.IMGS[0]
        elif self.img_count <= self.ANIMATION_TIME*2:
            self.img = self.IMGS[1]
        elif self.img_count <= self.ANIMATION_TIME*3:
            self.img = self.IMGS[2]
        elif self.img_count <= self.ANIMATION_TIME*4:
            self.img = self.IMGS[1]
        elif self.img_count == self.ANIMATION_TIME*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_TIME*2

        blitRotateCenter(win, self.img, (self.x, self.y), self.tilt)

    def get_mask(self):

        return pygame.mask.from_surface(self.img)

class Pipe():

    WIN_HEIGHT = WIN_HEIGHT
    WIN_WIDTH = WIN_WIDTH
    GAP = 200
    VEL = 5

    def __init__(self, x):

        self.x = x
        self.height = 0
        self.gap = 100


        self.top = 0
        self.bottom = 0

        self.PIPE_TOP = pygame.transform.flip(pipe_img, False, True)
        self.PIPE_BOTTOM = pipe_img

        self.passed = False

        self.set_height()

    def set_height(self):

        self.height = random.randrange(50, 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, win):

        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 - round(bird.y))
        bottom_offset = (self.x - bird.x, self.bottom - round(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

class Base:

    VEL = 5
    WIN_WIDTH = WIN_WIDTH
    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 blitRotateCenter(surf, image, topleft, angle):

    rotated_image = pygame.transform.rotate(image, angle)
    new_rect = rotated_image.get_rect(center = image.get_rect(topleft =
topleft).center)

    surf.blit(rotated_image, new_rect.topleft)

def menu_screen(win):

    pass

def end_screen(win):

    run = True
    text_label = END_FONT.render("Aperte EspaÃ§o para comeÃ§ar", 1,
(255,255,255))
    while run:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False

            if event.type == pygame.KEYDOWN:
                main(win)

        win.blit(text_label, (WIN_WIDTH/2 - text_label.get_width()/2,
500))
        pygame.display.update()

    pygame.quit()
    quit()

def draw_window(win, bird, pipes, base, score):

    win.blit(bg_img, (0,0))

    for pipe in pipes:
        pipe.draw(win)

    base.draw(win)
    bird.draw(win)

    score_label = STAT_FONT.render("PontuaÃ§Ã£o: " +
str(score),1,(255,255,255))
    win.blit(score_label, (WIN_WIDTH - score_label.get_width() - 15, 10))

    pygame.display.update()

def main(win):
```

```python
bird = Bird(230,350)
base = Base(FLOOR)
pipes = [Pipe(700)]
score = 0

clock = pygame.time.Clock()
start = False
lost = False

run = True
while run:
    pygame.time.delay(30)
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            pygame.quit()
            quit()
            break

        if event.type == pygame.KEYDOWN and not lost:
            if event.key == pygame.K_SPACE:
                if not start:
                    start = True
                bird.jump()

    if start:
        bird.move()
    if not lost:
        base.move()

        if start:
            rem = []
            add_pipe = False
            for pipe in pipes:
                pipe.move()

                if pipe.collide(bird, win):
                    lost = True

                if pipe.x + pipe.PIPE_TOP.get_width() < 0:
                    rem.append(pipe)

                if not pipe.passed and pipe.x < bird.x:
                    pipe.passed = True
                    add_pipe = True

            if add_pipe:
                score += 1
                pipes.append(Pipe(WIN_WIDTH))

            for r in rem:
                pipes.remove(r)


    if bird.y + bird_images[0].get_height() - 10 >= FLOOR:
        break
        draw_window(WIN, bird, pipes, base, score)
    end_screen(WIN)
main(WIN)
```