```
import pygame

import neat

import time

import os

import random

import sys

pygame.mixer.pre_init(frequency = 44100, size = 16, channels = 1, buffer = 512)

pygame.init()


WIN_WIDTH = 500

WIN_HEIGHT = 650

GEN = 0




BIRD_IMGS = [pygame.transform.scale2x(pygame.image.load(os.path.join("redbird-
downflap.png"))),pygame.transform.scale2x(pygame.image.load(os.path.join("redbird-
midflap.png"))),pygame.transform.scale2x(pygame.image.load(os.path.join("redbird-upflap.png")))]

PIPE_IMG = pygame.transform.scale2x(pygame.image.load(os.path.join("pipe-red.png")))

BASE_IMG = pygame.transform.scale2x(pygame.image.load(os.path.join("base.png")))

BG_IMG = pygame.transform.scale2x(pygame.image.load(os.path.join("background-day.png")))


STAT_FONT = pygame.font.SysFont("comicsans",50)

GAME_SOUND = pygame.mixer.Sound("KGF.wav")

TIWARI_SOUND = pygame.mixer.Sound("Joker-Lai-Lai-Lai.wav")

SHRISH_SOUND = pygame.mixer.Sound("Tera-Baap-Aaya.wav")

class Bird:

    IMGS = BIRD_IMGS

    MAX_ROTATION = 25

    ROT_VEL = 20

    ANIMATION_TIME = 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 = -10.5
    self.tick_count = 0
    self.height = self.y

def move(self):
    self.tick_count += 1

    d = self.vel*self.tick_count + 1.5*self.tick_count**2

    if d >= 16:
        d = 16

    if d < 0:
        d -= 2

    self.y = self.y + d
    if d < 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

        rotated_image = pygame.transform.rotate(self.img, self.tilt)
        new_rect = rotated_image.get_rect(center=self.img.get_rect(topleft =
(int(self.x),int(self.y))).center)
        win.blit(rotated_image, new_rect.topleft)



    def get_mask(self):
        return pygame.mask.from_surface(self.img)


class Pipe:
    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, 400)
        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 - 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 t_point or b_point:
            return True
        return False

class Base:
    TIWARI_SOUND.play()
    #GAME_SOUND.play()
    #SHRISH_SOUND.play()
    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 draw_window(win,bird, pipes, base, score, gen):
    win.blit(BG_IMG,(0,0))

    for pipe in pipes:
        pipe.draw(win)

    text = STAT_FONT.render("Score: " + str(score), 1,(255,255,255))
    win.blit(text, (WIN_WIDTH - 10 - text.get_width(), 10))
    text = STAT_FONT.render("Gen: " + str(gen), 1,(255,255,255))
    win.blit(text, (10, 10))
    base.draw(win)
    for birds in bird:
        birds.draw(win)
    pygame.display.update()



def main(genomes, config):
    global GEN
    GEN += 1
    nets =[]
    ge = []
    birds = []
    for _, g in genomes:
        net = neat.nn.FeedForwardNetwork.create(g, config)
        nets.append(net)
        birds.append(Bird(230, 350))
        g.fitness = 0
        ge.append(g)




    base = Base(900)
    pipes = [Pipe(600)]
    win = pygame.display.set_mode((WIN_WIDTH,WIN_HEIGHT))
    clock = pygame.time.Clock()

    score = 0


    run = True
    while run:
        clock.tick(40)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.quit()
                quit()
                game_sound.play()

        pipe_ind = 0
        if len(birds) > 0:
            if len(pipes) > 1 and birds[0].x > pipes[0].x + pipes[0].PIPE_TOP.get_width():
                pipe_ind = 1
        else:
            run = False
            break

        for x, bird in enumerate(birds):
            bird.move()
            ge[x].fitness += 0.1

            output = nets[x].activate((bird.y, abs(bird.y - pipes[pipe_ind].height),abs(bird.y -
pipes[pipe_ind].bottom)))



            if output[0] > 0.5:
                bird.jump()

        add_pipe = False
        rem =[]
        for pipe in pipes:
            for x, bird in enumerate(birds):
                if pipe.collide(bird):
```

```
            ge[x].fitness -= 1
            birds.pop(x)
            nets.pop(x)
            ge.pop(x)

        if not pipe.passed and pipe.x < bird.x:
            pipe.passed = True
            add_pipe = True
    if pipe.x + pipe.PIPE_TOP.get_width() < 0:
        rem.append(pipe)

    pipe.move()
if add_pipe:
    score += 1
    for g in ge:
        g.fitness +=5
    pipes.append(Pipe(600))

for r in rem:
    pipes.remove(r)
for x,bird in enumerate(birds):
    if bird.y + bird.img.get_height() >= 730 or bird.y < 0:
        birds.pop(x)
        nets.pop(x)
        ge.pop(x)
# if score > 50:
#    break

base.move()
draw_window(win, birds, pipes, base, score, GEN)
```

```
def run(config_path):
    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction,
                                neat.DefaultSpeciesSet, neat.DefaultStagnation,
                                config_path)

    p = neat.Population(config)

    p.add_reporter(neat.StdOutReporter(True))
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)

    winner = p.run(main,50)

if __name__=="__main__":
    local_dir = os.path.dirname(__file__)
    config_path = os.path.join(local_dir, "config-feedforward.txt")
    run(config_path)
```