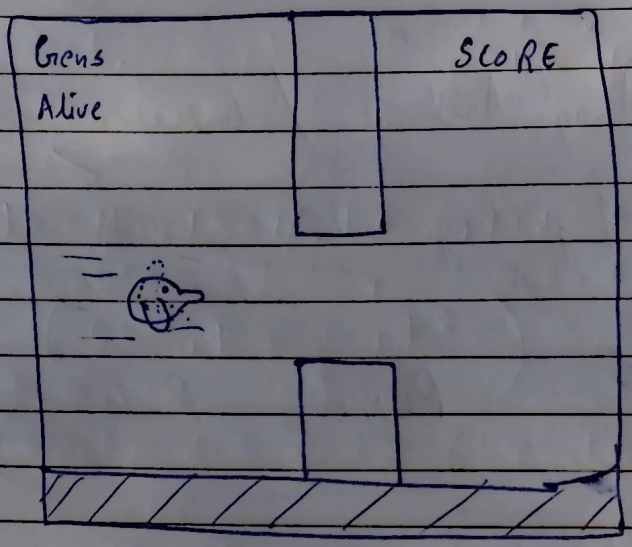
Date
Page No.

FLAPPY BIRD with NEAT

Pygame is a free, open-source, cross platform lib. for writing video games and multimedia application using Python.



WIN_WIDTH & WIN_HEIGHT
↳ constants for size adjustment

Bird_IMGS = [pygame.transform.scale2x(pygame.image.load
↳ make size 2x ↳ Loads images
(os.path.join("imgs","bird1.png")))]

Defining a class

class Bird:
    IMGS = BIRD_IMGS
    MAX_ROTATION = 25 → How much the bird will tilt
    ROT_VEL = 20 → How much we will rotate the bird
    ANIMATION_TIME = 5 → (Flapping wings animation)

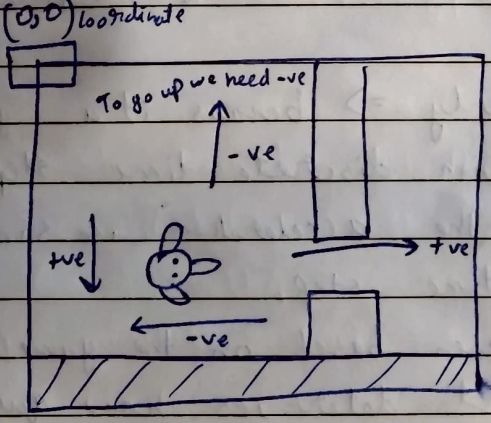
Date
Page No.

obj is created. → init() method is special method used to initialise obj attribute.

The self keyword in Py is used to represent an instance of a class. Used to access attributes & methods of the class.

```
def __init__(self, x, y):
    self.x = x                    ] -> starting position of bird
    self.y = y
    self.tilt = 0                 -> How much the bird tilts
    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
```



Why -ve velocity

In Py, -ve velocity is used to indicate that an object is moving in the opposite direction of positive x-axis. This is because the y-axis is +ve upwards, so if an object is moving upwards it must have a -ve velocity

Going against the usual direction (gravity)

```
self.tick_count = 0    -> when we last jump
```

Date
Page No.

self.height = self.y → where the bird jump from

def move(self): → moving obj
self.tick_count += 1 → A tick happened, u became gone keeping track
~~d = self.vel * self.tick_count + 1.5 * self.tick~~

while True:
bird.move() → I dont have to calculat the move of bird in game i will just call the obj.

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

*

$\left(s = ut + \frac{1}{2}at^2\right)$

(-10.5 + 1.5 = -9 (9 pixel upwards))

why 1.5 only => becaus it a game and we are working with discrete time steps (each tick of the loop), so the acceleration is simplified to a fixed value per time step.
1.5 is chosen based on the game mechanics and desired gameplay. It determines how quickly the bird acc. chooses through trial & error.

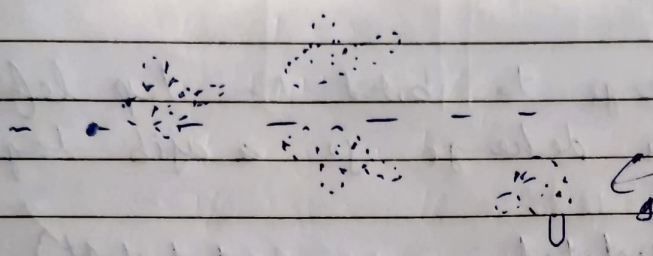
if d >= 16: → [if d is >= 16 terminal vel. moving down/up more than 16 pixel]
d = 16
if d < 0:
d -= 2 [through trial & error makes jumps smooth]

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
```

agar bird jump kiya hai toh agar vo is point d ke upper hai to uska nose upper ke taraf tilt rahega jab tak point d tk nahi aajata. Point d se nich hua toh nose niche tilt hoga.




```
def draw(self, win):          window the bird is drawn
    self.img_count += 1
```

wings change kar ge flapping wings.

```
    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]
```

*3 [2]

*4 [1]

```
    elif self.img_count == self.ANIMATION_TIME*4+1:
        self.img = self.IMGS[0]
        self.img_count = 0
```

~~nose~~ niche gira hai tho flap ni kre ga

```
    if self.tilt <= -80:
        self.img = self.IMGS[1]
        self.img_count = self.ANIMATION_TIME*2
```

Date
Page No.

Rotating img around the center (because all the bird image is level with the screen)

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

ye nahi karenge to bird ka top left corner se tilt hota dekhenge jo ajib lagega

```
win.blit(rotated_image, new_rect.topleft)
```

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

```
def draw_window(win, bird):
    win.blit(BG_IMG, (0,0))
    bird.draw(win)
    pygame.display.update()
```

```
def main():
    bird = Bird(200, 200)
    run = True
    while run:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
        draw_window(win, bird)
    pygame.quit()
    quit()

main()
```

Red ⊠ on top left

Date
Page No.

def main ():

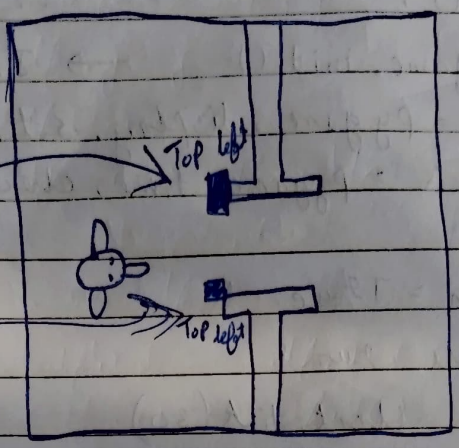
pygame.init() → Initialize Pygame
win = pygame.display.set_mode((WIN_---))
clock = pygame.time.Clock() → speed of the game
(Jitna bhi speed processor ka speed of game fix hoga)

run = True
while run:
clock.tick(30)
↳ 30 tick every seconds

main()

class Pipe:
GAP = 200 → distance of pipe from each other
VEL = 5

def __init__(self, x):
self.x = x
self.height = 0
self.
self.top = 0
self.bottom = 0
self.PIPE_TOP = pygame.transform.flip(PIPE_IMG, False, True)
self.PIPE_BOTTOM = PIP_IMG
self.passed = False → collision ke liye baad mai use hoga
self.set_height()
def set_height(self):
self.height = random.(50, 450)



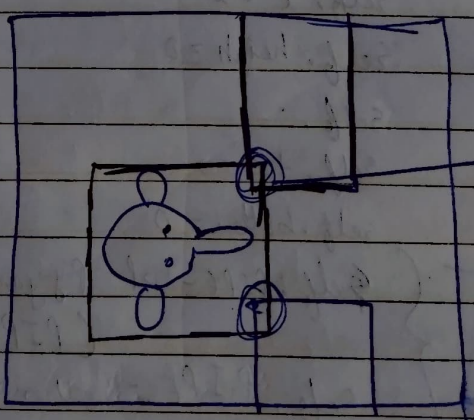
randomly place.

```
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
```
→ Moving pipe to left

drawing pipe

```
def draw(self, win):
    win.blit(self.PIPE_TOP, (self.x, self.top))
    win.blit(self.PIPE_BOTTOM, (self.x, self.bottom))
```



because we generally assign box to the bird. This will say that the collision happend

kyuki bird ko cover kiya box Pipe se touch hua par actual mai bird touch nahi hua hai isliye → mask use karenge.

We create 2 arrays which will have as many pixels in rows & columns.

To have pixel perfect collision

```
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)
```

for bird

Top pipe

bottom Pipe

offset tells how far is our mask from other

If not clashed. They both will return None as no overlap of mask happens.

```
    if t_point or b_point:        ⟶ (if not none)
        return True
    return False
```

```
class Base:
    VEL = 5        ⟶ same as pipe
    WIDTH = BASE_IMG.get_width()
    IMG = BASE_IMG
```

```
def __init__(self, y):
    self.y = y
    self.x1 = 0
    self.x2 = self.width
```

→ one img with position 0
→ another directly behind.

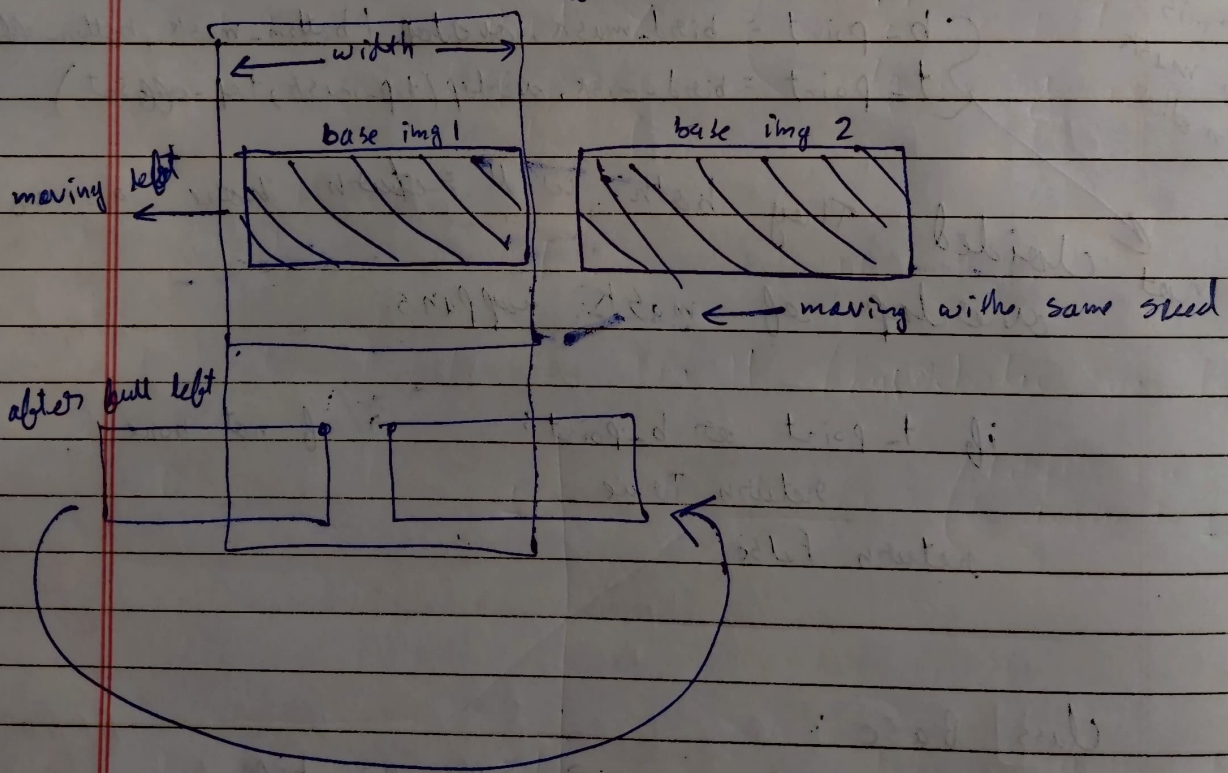
```
def move(self):
    self.x1 -= self.VEL
    self.x2 -= self.VEL
```

→ same speed

if one of them goes out of screen it will cycle back to the end of Bannerson.

```
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))
```

all the classes made up this now.

Moving Pipes and generating more pipes

main ()

rem = [ ] → we will add the pipes that crossed the window

For pipe in pipes:

if pipe.collide (bird): // abhi K liye bird ko kuch nahi kar rahe.

pass

if pipe is completely out of screen ← if pipe.x + pipe.PIPE_TOP.get_width() < 0:

rem.append(pipe) → adding pipe to rem []

if we have passed the pipe ← if not pipe.passed and pipe.x < bird.x:

pipe.passed = True

add_pipe = True

pipe.move()

if add_pipe: → checking score ++

score += 1

pipes.append(Pipe(600)) → x position [distance between to pipes □ ⟷ □ 600]

For r in rem: → removing pipes in rem

pipes.remove(r)

If the bird hit the ground.

if bird.y + bird.img.get_height() >= 730:

pass ⟹ deal with it later

Date
Page No.

Adding Score

Pygame.font.init()

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

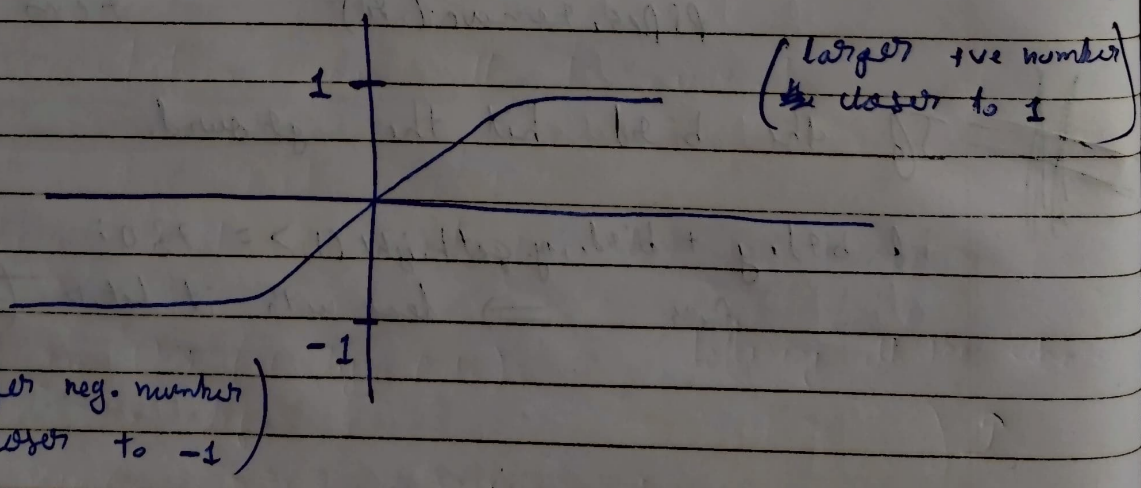
in ⇒ def draw_window(win, bird, pipes, base, score):
    test = stAT_FONT.render("Score:" + str(score), 1, white)
    win.blit(text, (WIN.WIDTh - 10 - text.get_wid(255,255,255))

No matter how big the score get it will display on screen.

Inputs ← Bird Y (: The bird moves only in y axis)
Top Pipe
Bottom Pipe

Output ← Jump?

Activation Function ← Tan H (hyperbolic Tan functn)



It will squash value from this output neuron to be between 1 & -1 & This way we can evaluate value. and tan tell to Jump or not.

We can also use segmoid func but its simpler to use tan H cuz

if value is > 0.5 bird will jump

~~if~~ ~~bird will not jump~~

(why 0.5 only gives a smooth effect better visual)

Population Size ← How many birds in each generation

IMP ★ Fitness Function ← The further the bird goes the more fitter it is from others (distance)

Max generation ← 30 (if 30 gen k baad bhi perfect bird ni mila → "end")

↳ random

---

feed forward (config file)

Fitness_criterion = max → birds with highest fitness

(min, max options to choose)

↳ birds with lowest fitness

Fitness_threshold = 100 → the value that needs to be reached by a bird to say we dont need any more generations

[Default Genome]

activation_default = tanh → (hyperbolic Tan function)

activation_mutate_rate = 0.0

↳ (If we want to change the activation function in next gen we can set (0.1) ie (10%) chance to change activation)

activation_options = tanh
(yaha se activ_mutate_rate select karega)

node bias options

bias_init_mean = 0.0 (How likely to change)
max = 30 } sampling values between
min = -30 } -30, 30

mutate_power = 0.5
mutate_rate = 0.7
replace_rate = 0.1
} How we want to change the gen. These values are tried & tested and works best

Connection add/~~remove~~

conn_add = 0.5
... delete = 0.5
} 50%

Network Parameters

num_hidden = 0
input = 3
o/p = 1
} default setting.

Documentation has all it covered.

---

adding feet_forward neural net to our prog.

def run (config_path): defining all the subheading
config = ———) ↑
if __name__ == "__main__":
local_dir = os.path.dirname(__file__)
config_path = os.path.join(local_dir, "config-feedforward.txt")

Date
Page No.

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

required by neat.

```
def run(config_path)
    config = - - - - - - (all the sub heading)

    p = neat.Population(config) → Setting Population
    p.add_reporter(neat.StdOutReporter(True))
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)

giving output on console(stats)

winner = p.run(main, 50)
```

`main` → Calling main with 50 times, genomes & config

`50` → no. of gen will run

In theory we "can" run one bird at a time but its not efficient, to run all the birds at same time we need to make some changes.

```
bird = Bird(230, 350)
birds = []
```

Date
Page No.

def main():
    def main(genomes, config):

        birds = []

adding code → for pipe in pipes:
                for bird in birds:
                    if pipe.collide(bird):
                        pass

checking every bird collides with every pipe

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

Not running the loop because inside it for only one pipe → if pipe.x + pipe.PIPE_TOP.get_width() < 0:
                                                                rem.append(pipe)

                pipe.move()

if bird's the ground. → for bird in birds:
                            if bird.y + bird.img.get_height() >= 730:
                                pass # later dead karna.

Date
Page No.

Adding more array in main.

```
nets = []        ← neural network
ge = []          ← genome
birds = []
```

ge = [0] will be for one bird i.e [0]

```
for g in genomes:
    nets = neat.nn.FeedForwardNetwork(g, config)
    nets.append(net)
    birds.append(Bird(230, 350))
    ge.append(g)
    g.fitness = 0
```

all the birds starts at same position.

In collide part -

~~pipe.collide(bird)~~ ⇒ a built in function that allows to keep track of the no. of iterations (loops) in a loop.

```
for pipe in pipes:
    for x, bird in enumerate(birds):
        if pipe.collide(bird):
            ge[x].fitness -= 1
```

Every time a bird hits a pipe its going to remove 1 from its fitness

~~birds.remove(bird)~~

```
birds.pop(x)
nets.pop(x)
ge.pop(x)
```

} removing neural network & genome of the bird with x who performs that.

in add_pipe :

```
if add_pipe:
    score += 1
    for g in ge:
        g.fitness += 5
```

encouraging birds to perform better by giving them +5 fitness when passing pipe

```
for r in rem:
    pipes.remove(r)
```

adding code

```
for x, bird in enumerate(birds):
    if bird.y + bird.img.get_height() >= 730:
        birds.pop(x)
        nets.pop(x)
        ge.pop(x)
```

if bird hits the ground removing fitness of the bird, remove the bird, rather than

Moving birds acc to there neural net.

what input we are giving to our neural network

→ Distance between top & bottom pipe

[why both top & bottom why not one
→ To make it easy for neural net to perform as it wont have to calculate distance]

→ y position of the bird

Date
Page No.

What if there is more than one pipe in the screen.

→ Setting pipe index to be 0

pipe_ind = 0
if len(birds) > 0: → if length of bird is 0 (not to continue) but if > 0 then ↓
    if len(pipes) > 1 and birds[0].x > pipes[0].x + pipes[0].PIPE_TOP.get_width():
        pipe_ind = 1

if the length of pipe is > 1 we will get the bird (in x position) if that is greater than [since x position is going to be same for all birds we are just using [0]] if we have passed the pipe & change the index & we will look for second pipe in the window.

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

giving bird a little fitness for surviving this long, to encourage it to keep moving forward.

why only 0.1 → because this loop is going to run 30 times [clock.tick(30)] so in theory for 1 sec being alive it gain 1 fitness point.

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

Output is a list

Our example has only one list thats why [0]

pTO =>

Date
Page No.

holding value of output from neural network

absolute value

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

((first value of bird in y axis, distance between top & bottom pipe))

```
if output > 0.5:
    bird.jump()
```

What if bird try to go to the sky to pass the pipe from out of the window.

```
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 gone far up in the sky

means bird is removed.

---

for _, g in genomes

why _ ? → because genomes is a tuple which has the genome ID & object

genome ID → _

object → g.

Date
Page No.

Drawing lines -

bird ka width by 2

pygame.draw.line(win, (57, 255, 20), (bird.x + bird.img.width()/2, bird.y + bird.img.get_height()/2),

(green color)

pipe ka width/2

, pipe[pipe_ind].height), 5)

pipe ka height, 5 units distance

Plotting graph

matplotlib.pyplot & numpy

AI se uthaya code.

Trouble shoot -

coz jupyter ek notebook hai -

current_dir = os.getcwd() → current dir

config_path = os.path.join(current_dir, "feed_forward.txt")

run(config_path)