## Flappy Bird - Full Code Explanation

```
# Flappy Game

## Import Libraries
import pygame  # Main game library for graphics and input handling
import os      # For path operations like locating image assets
import random  # For generating random pipe heights

## Initialize fonts in pygame
pygame.font.init()

## Set window dimensions
width = 500
height = 800

## Load and scale game images
bird_img = [pygame.transform.scale2x(pygame.image.load(os.path.join('assets', 'bird1.png'))), ...]
# Loads three bird images and doubles their size for animation

pipe_img, base_img, bg_img, game_over_img = ...  # Load and scale pipe, base, background, and game over image

## Define font for score display
start_font = pygame.font.SysFont('arial', 25)

## Base Class - represents the scrolling ground
class Base:
    def __init__(self, y):
        self.y = y  # y-position of the ground
        self.x1 = 0  # Starting position of first base image
        self.x2 = base_img.get_width()  # Second image follows first for seamless loop
        self.VEL = 5  # Velocity of movement

    def move(self):
        # Moves the ground images leftward to simulate motion
        self.x1 -= self.VEL
        self.x2 -= self.VEL
        if self.x1 + base_img.get_width() < 0:
            self.x1 = self.x2 + base_img.get_width()
        if self.x2 + base_img.get_width() < 0:
            self.x2 = self.x1 + base_img.get_width()

    def draw(self, window):
        # Draws both images to screen
        window.blit(base_img, (self.x1, self.y))
        window.blit(base_img, (self.x2, self.y))

## Pipe Class - handles pipe generation, movement, and collision
```

# Flappy Bird Game - Code Explanation

```
class Pipe:
    gap = 200  # Gap between top and bottom pipes
    velocity = 5

    def __init__(self, x):
        self.x = x
        self.height = random.randint(50, 400)  # Random height
        self.top = self.height - pipe_img.get_height()
        self.bottom = self.height + self.gap
        self.pipe_top = pygame.transform.flip(pipe_img, False, True)
        self.pipe_bottom = pipe_img
        self.passed = False

    def move(self):
        self.x -= self.velocity  # Moves pipes left

    def draw(self, window):
        window.blit(self.pipe_top, (self.x, self.top))
        window.blit(self.pipe_bottom, (self.x, self.bottom))

    def off_screen(self):
        return self.x + pipe_img.get_width() < 0  # Check if pipe is off screen

    def collide(self, bird):
        # Collision detection using masks
        bird_mask = bird.get_mask()
        top_mask = pygame.mask.from_surface(self.pipe_top)
        bottom_mask = pygame.mask.from_surface(self.pipe_bottom)
        offset_top = (self.x - bird.x, self.top - round(bird.y))
        offset_bottom = (self.x - bird.x, self.bottom - round(bird.y))
        return bird_mask.overlap(top_mask, offset_top) or bird_mask.overlap(bottom_mask, offset_bottom)

## Bird Class - handles bird physics, animation, and drawing
class Bird:
    imgs = bird_img
    max_rotation = 25
    rot_velocity = 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):
```

# Flappy Bird Game - Code Explanation

```
        self.vel = -8
        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_velocity

    def draw(self, window):
        # Controls animation frames and rotation
        self.img_count += 1
        self.img = self.imgs[self.img_count // self.animation_time % len(self.imgs)]
        if self.tilt <= -80:
            self.img = self.imgs[1]
            self.img_count = self.animation_time
        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)
        window.blit(rotated_image, new_rect.topleft)

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

## Draw everything in window
def draw_window(window, bird, pipes, base, score):
    window.blit(bg_img, (0, 0))
    for pipe in pipes:
        pipe.draw(window)
    text = start_font.render(f"Score: {score}", 1, (255, 255, 255))
    window.blit(text, (width - 10 - text.get_width(), 10))
    base.draw(window)
    bird.draw(window)
    pygame.display.update()

## Show Game Over screen
def game_over(window):
    game_over_x = (width - game_over_img.get_width()) // 2
    game_over_y = (height - game_over_img.get_height()) // 2
    window.blit(game_over_img, (game_over_x, game_over_y))
    pygame.display.update()
```

```
    pygame.time.delay(3000)
    main()  # Restart game

## Main loop
def main():
    pygame.display.set_caption("Flappy Bird with Code4FewBucks")
    bird = Bird(230, 350)
    base = Base(730)
    pipes = [Pipe(600)]
    score = 0
    window = pygame.display.set_mode((width, height))
    clock = pygame.time.Clock()
    game_started = False

    run = True
    while run:
        clock.tick(30)
        base.move()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return
            if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                game_started = True
                bird.jump()

        if game_started:
            bird.move()
            for pipe in pipes:
                pipe.move()
                if pipe.collide(bird) or bird.y + bird_img[0].get_height() >= base.y:
                    game_over(window)
                if not pipe.passed and pipe.x + pipe_img.get_width() < bird.x:
                    pipe.passed = True
                    score += 1
                if pipe.off_screen():
                    pipes.remove(pipe)
            if pipes[-1].x < 400:
                pipes.append(Pipe(730))

        draw_window(window, bird, pipes, base, score)

## Start the game
if __name__ == "__main__":
    main()
```