```
import pygame
import math

pygame.init()

WIDTH, HEIGHT = 800, 600
win = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Brick Breaker")

FPS = 60
PADDLE_WIDTH = 100
PADDLE_HEIGHT = 15
BALL_RADIUS = 10

LIVES_FONT = pygame.font.SysFont("comicsans", 40)

class Paddle:
    VEL = 5

    def __init__(self, x, y, width, height, color):
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.color = color

    def draw(self, win):
        pygame.draw.rect(
            win, self.color, (self.x, self.y, self.width, self.height))

    def move(self, direction=1):
        self.x = self.x + self.VEL * direction


class Ball:
    VEL = 5

    def __init__(self, x, y, radius, color):
        self.x = x
        self.y = y
        self.radius = radius
        self.color = color
        self.x_vel = 0
        self.y_vel = -self.VEL

    def move(self):
        self.x += self.x_vel
        self.y += self.y_vel

    def set_vel(self, x_vel, y_vel):
        self.x_vel = x_vel
        self.y_vel = y_vel

    def draw(self, win):
        pygame.draw.circle(win, self.color, (self.x, self.y),
self.radius)

class Brick:
    def __init__(self, x, y, width, height, health, colors):
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.health = health
        self.max_health = health
        self.colors = colors
        self.color = colors[0]

    def draw(self, win):
        pygame.draw.rect(
            win, self.color, (self.x, self.y, self.width, self.height))

    def collide(self, ball):
        if not (ball.x <= self.x + self.width and ball.x >= self.x):
            return False
        if not (ball.y - ball.radius <= self.y + self.height):
            return False

        self.hit()
        ball.set_vel(ball.x_vel, ball.y_vel * -1)
        return True

    def hit(self):
        self.health -= 1
        self.color = self.interpolate(
            *self.colors, self.health/self.max_health)

    @staticmethod
    def interpolate(color_a, color_b, t):

        return tuple(int(a + (b - a) * t) for a, b in zip(color_a,
color_b))


def draw(win, paddle, ball, bricks, lives):
    win.fill("white")
    paddle.draw(win)
    ball.draw(win)

    for brick in bricks:
        brick.draw(win)

    lives_text = LIVES_FONT.render(f"Lives: {lives}", 1, "black")
    win.blit(lives_text, (10, HEIGHT - lives_text.get_height() - 10))

    pygame.display.update()


def ball_collision(ball):
    if ball.x - BALL_RADIUS <= 0 or ball.x + BALL_RADIUS >= WIDTH:
        ball.set_vel(ball.x_vel * -1, ball.y_vel)
    if ball.y + BALL_RADIUS >= HEIGHT or ball.y - BALL_RADIUS <= 0:
        ball.set_vel(ball.x_vel, ball.y_vel * -1)


def ball_paddle_collision(ball, paddle):
    if not (ball.x <= paddle.x + paddle.width and ball.x >= paddle.x):
        return
    if not (ball.y + ball.radius >= paddle.y):
        return

    paddle_center = paddle.x + paddle.width/2
    distance_to_center = ball.x - paddle_center

    percent_width = distance_to_center / paddle.width
    angle = percent_width * 90
    angle_radians = math.radians(angle)

    x_vel = math.sin(angle_radians) * ball.VEL
    y_vel = math.cos(angle_radians) * ball.VEL * -1

    ball.set_vel(x_vel, y_vel)


def generate_bricks(rows, cols):
    gap = 2
    brick_width = WIDTH // cols - gap
    brick_height = 20

    bricks = []
    for row in range(rows):
        for col in range(cols):
            brick = Brick(col * brick_width + gap * col, row *
brick_height +
                          gap * row, brick_width, brick_height, 2, [(0,
255, 0), (255, 0, 0)])
            bricks.append(brick)

    return bricks


def main():
    clock = pygame.time.Clock()

    paddle_x = WIDTH/2 - PADDLE_WIDTH/2
    paddle_y = HEIGHT - PADDLE_HEIGHT - 5
    paddle = Paddle(paddle_x, paddle_y, PADDLE_WIDTH, PADDLE_HEIGHT,
"black")
    ball = Ball(WIDTH/2, paddle_y - BALL_RADIUS, BALL_RADIUS, "black")

    bricks = generate_bricks(3, 10)
    lives = 3

    def reset():
        paddle.x = paddle_x
        paddle.y = paddle_y
        ball.x = WIDTH/2
        ball.y = paddle_y - BALL_RADIUS


    def display_text(text):
        text_render = LIVES_FONT.render(text, 1, "red")
        win.blit(text_render, (WIDTH/2 - text_render.get_width() /
                               2, HEIGHT/2 - text_render.get_height()/2))
        pygame.display.update()
        pygame.time.delay(3000)

    run = True
    while run:
        clock.tick(FPS)

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

        keys = pygame.key.get_pressed()

        if keys[pygame.K_LEFT] and paddle.x - paddle.VEL >= 0:
            paddle.move(-1)
        if keys[pygame.K_RIGHT] and paddle.x + paddle.width + paddle.VEL
<= WIDTH:
            paddle.move(1)

        ball.move()
        ball_collision(ball)
        ball_paddle_collision(ball, paddle)

        for brick in bricks[:]:
            brick.collide(ball)

            if brick.health <= 0:
                bricks.remove(brick)

        # lives check
        if ball.y + ball.radius >= HEIGHT:
            lives -= 1
            ball.x = paddle.x + paddle.width/2
            ball.y = paddle.y - BALL_RADIUS
            ball.set_vel(0, ball.VEL * -1)

        if lives <= 0:
            bricks = generate_bricks(3, 10)
            lives = 3
            reset()
            display_text("You Lost!")

        if len(bricks) == 0:
            bricks = generate_bricks(3, 10)
            lives = 3
            reset()
            display_text("You Won!")

        draw(win, paddle, ball, bricks, lives)

    pygame.quit()
    quit()


if __name__ == "__main__":
    main()
```