```
import pygame
import random
import math
import sys

# Initialize Pygame
pygame.init()

# CGA Color Palette (4 colors)
CGA_COLORS = {
    'BLACK': (0, 0, 0),
    'WHITE': (255, 255, 255),
    'MAGENTA': (255, 85, 255),
    'CYAN': (85, 255, 255)
}

# Game constants
SCREEN_WIDTH = 640
SCREEN_HEIGHT = 480
FPS = 60

class Particle:
    """Visual effect particle for explosions"""
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.vx = random.uniform(-3, 3)
        self.vy = random.uniform(-3, 3)
        self.life = 30
        self.color = color
        self.size = random.randint(2, 4)

    def update(self):
        self.x += self.vx
        self.y += self.vy
        self.life -= 1
        self.vy += 0.1  # Gravity effect

    def draw(self, screen):
        if self.life > 0:
            alpha_size = max(1, int(self.size * (self.life / 30)))
            pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), alpha_size)

    def is_alive(self):
```

```
        return self.life > 0

class PowerUp:
    """Power-up collectibles"""
    def __init__(self, x, y, power_type):
        self.x = x
        self.y = y
        self.width = 12
        self.height = 12
        self.speed = 2
        self.type = power_type  # 'health', 'rapid_fire', 'shield'
        self.colors = {
            'health': CGA_COLORS['MAGENTA'],
            'rapid_fire': CGA_COLORS['CYAN'],
            'shield': CGA_COLORS['WHITE']
        }

    def update(self):
        self.y += self.speed

    def draw(self, screen):
        color = self.colors.get(self.type, CGA_COLORS['WHITE'])
        # Draw rotating diamond shape
        points = [
            (self.x + self.width // 2, self.y),
            (self.x + self.width, self.y + self.height // 2),
            (self.x + self.width // 2, self.y + self.height),
            (self.x, self.y + self.height // 2)
        ]
        pygame.draw.polygon(screen, color, points, 2)
        # Inner cross
        pygame.draw.line(screen, color,
                         (self.x + self.width // 2, self.y + 3),
                         (self.x + self.width // 2, self.y + self.height - 3), 2)
        pygame.draw.line(screen, color,
                         (self.x + 3, self.y + self.height // 2),
                         (self.x + self.width - 3, self.y + self.height // 2), 2)

class Fighter:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.width = 20
        self.height = 16
```

```
        self.speed = 5
        self.bullets = []
        self.health = 100
        self.max_health = 100
        self.shoot_cooldown = 0
        self.rapid_fire_timer = 0
        self.shield_timer = 0
        self.invincible_timer = 0

    def move(self, dx, dy):
        self.x += dx * self.speed
        self.y += dy * self.speed

        # Keep fighter on screen
        self.x = max(0, min(self.x, SCREEN_WIDTH - self.width))
        self.y = max(0, min(self.y, SCREEN_HEIGHT - self.height))

    def shoot(self):
        if self.shoot_cooldown <= 0:
            # Normal shot
            bullet = Bullet(self.x + self.width // 2 - 1, self.y, 0, -8, True)
            self.bullets.append(bullet)

            # Rapid fire adds side shots
            if self.rapid_fire_timer > 0:
                left_bullet = Bullet(self.x + 2, self.y + 4, -2, -8, True)
                right_bullet = Bullet(self.x + self.width - 4, self.y + 4, 2, -8, True)
                self.bullets.append(left_bullet)
                self.bullets.append(right_bullet)
                self.shoot_cooldown = 5
            else:
                self.shoot_cooldown = 10

    def activate_power_up(self, power_type):
        if power_type == 'health':
            self.health = min(self.max_health, self.health + 30)
        elif power_type == 'rapid_fire':
            self.rapid_fire_timer = 300  # 5 seconds at 60 FPS
        elif power_type == 'shield':
            self.shield_timer = 180  # 3 seconds

    def take_damage(self, damage):
        if self.shield_timer > 0 or self.invincible_timer > 0:
            return False
```

```
        self.health -= damage
        self.invincible_timer = 30  # Brief invincibility after hit
        return True

    def update(self):
        # Update timers
        if self.shoot_cooldown > 0:
            self.shoot_cooldown -= 1
        if self.rapid_fire_timer > 0:
            self.rapid_fire_timer -= 1
        if self.shield_timer > 0:
            self.shield_timer -= 1
        if self.invincible_timer > 0:
            self.invincible_timer -= 1

        # Update bullets
        for bullet in self.bullets[:]:
            bullet.update()
            if bullet.y < -10:
                self.bullets.remove(bullet)

    def draw(self, screen):
        # Draw shield if active
        if self.shield_timer > 0:
            shield_color = CGA_COLORS['CYAN'] if (self.shield_timer // 5) % 2 == 0 else
CGA_COLORS['WHITE']
            pygame.draw.circle(screen, shield_color,
                               (int(self.x + self.width // 2), int(self.y + self.height // 2)),
                               15, 2)

        # Blink when invincible
        if self.invincible_timer > 0 and self.invincible_timer % 4 < 2:
            return

        # Draw fighter jet (simple CGA style)
        # Main body
        pygame.draw.rect(screen, CGA_COLORS['WHITE'],
                         (self.x + 6, self.y + 4, 8, 12))
        # Wings
        wing_color = CGA_COLORS['CYAN'] if self.rapid_fire_timer > 0 else
CGA_COLORS['WHITE']
        pygame.draw.rect(screen, wing_color,
                         (self.x, self.y + 8, 20, 4))
        # Cockpit
```

```
        pygame.draw.rect(screen, CGA_COLORS['MAGENTA'],
                         (self.x + 8, self.y, 4, 8))

        # Draw bullets
        for bullet in self.bullets:
            bullet.draw(screen)

class Enemy:
    def __init__(self, x, y, enemy_type='basic'):
        self.x = x
        self.y = y
        self.type = enemy_type

        # Type-specific properties
        if enemy_type == 'fast':
            self.width = 12
            self.height = 10
            self.speed = 3.5
            self.health = 1
            self.shoot_delay = 80
        elif enemy_type == 'tank':
            self.width = 20
            self.height = 16
            self.speed = 1
            self.health = 3
            self.shoot_delay = 40
        else:  # basic
            self.width = 16
            self.height = 12
            self.speed = 2
            self.health = 1
            self.shoot_delay = 60

        self.bullets = []
        self.shoot_timer = 0
        self.move_timer = 0
        self.direction = random.choice([-1, 1])

    def update(self):
        self.y += self.speed

        # Add some horizontal movement for variety
        self.move_timer += 1
        if self.move_timer > 30:
```

```
            self.x += self.direction * 1
            if self.x <= 0 or self.x >= SCREEN_WIDTH - self.width:
                self.direction *= -1

        # Shoot occasionally
        self.shoot_timer += 1
        if self.shoot_timer > self.shoot_delay:
            if random.random() < 0.3:
                self.shoot()
            self.shoot_timer = 0

        # Update bullets
        for bullet in self.bullets[:]:
            bullet.update()
            if bullet.y > SCREEN_HEIGHT + 10:
                self.bullets.remove(bullet)

    def shoot(self):
        if self.type == 'tank':
            # Tank shoots spread pattern
            for angle in [-0.3, 0, 0.3]:
                dx = math.sin(angle) * 5
                dy = 5
                bullet = Bullet(self.x + self.width // 2, self.y + self.height, dx, dy, False)
                self.bullets.append(bullet)
        else:
            bullet = Bullet(self.x + self.width // 2, self.y + self.height, 0, 6, False)
            self.bullets.append(bullet)

    def take_damage(self):
        self.health -= 1
        return self.health <= 0

    def draw(self, screen):
        # Different appearance based on type
        if self.type == 'fast':
            pygame.draw.polygon(screen, CGA_COLORS['CYAN'], [
                (self.x + self.width // 2, self.y),
                (self.x, self.y + self.height),
                (self.x + self.width, self.y + self.height)
            ])
        elif self.type == 'tank':
            pygame.draw.rect(screen, CGA_COLORS['WHITE'],
                             (self.x, self.y, self.width, self.height), 2)
```

```
            pygame.draw.rect(screen, CGA_COLORS['MAGENTA'],
                             (self.x + 4, self.y + 4, self.width - 8, self.height - 8))
        else:  # basic
            pygame.draw.rect(screen, CGA_COLORS['MAGENTA'],
                             (self.x + 4, self.y, 8, 12))
            pygame.draw.rect(screen, CGA_COLORS['WHITE'],
                             (self.x, self.y + 4, 16, 4))

        # Draw bullets
        for bullet in self.bullets:
            bullet.draw(screen)

class Bullet:
    def __init__(self, x, y, dx, dy, is_player):
        self.x = x
        self.y = y
        self.dx = dx
        self.dy = dy
        self.width = 2
        self.height = 4 if is_player else 6
        self.is_player = is_player

    def update(self):
        self.x += self.dx
        self.y += self.dy

    def draw(self, screen):
        color = CGA_COLORS['CYAN'] if self.is_player else CGA_COLORS['MAGENTA']
        pygame.draw.rect(screen, color,
                         (int(self.x), int(self.y), self.width, self.height))

class StarField:
    """Scrolling star background"""
    def __init__(self):
        self.stars = []
        for _ in range(50):
            x = random.randint(0, SCREEN_WIDTH)
            y = random.randint(0, SCREEN_HEIGHT)
            speed = random.choice([1, 2, 3])
            self.stars.append([x, y, speed])

    def update(self):
        for star in self.stars:
            star[1] += star[2]
```

```
            if star[1] > SCREEN_HEIGHT:
                star[1] = 0
                star[0] = random.randint(0, SCREEN_WIDTH)

    def draw(self, screen):
        for star in self.stars:
            size = 1 if star[2] == 1 else 2
            color = CGA_COLORS['WHITE'] if star[2] > 1 else CGA_COLORS['CYAN']
            pygame.draw.circle(screen, color, (star[0], int(star[1])), size)

class Game:
    def __init__(self):
        self.screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
        pygame.display.set_caption("CGA Fighter Jet - Enhanced Edition")
        self.clock = pygame.time.Clock()

        # Game objects
        self.fighter = Fighter(SCREEN_WIDTH // 2, SCREEN_HEIGHT - 50)
        self.enemies = []
        self.power_ups = []
        self.particles = []
        self.starfield = StarField()

        # Game state
        self.enemy_spawn_timer = 0
        self.powerup_spawn_timer = 0
        self.score = 0
        self.wave = 1
        self.enemies_killed_this_wave = 0
        self.enemies_per_wave = 10

        self.font = pygame.font.Font(None, 36)
        self.small_font = pygame.font.Font(None, 24)

        self.running = True
        self.game_over = False
        self.paused = False

        # High score
        self.high_score = 0

    def spawn_enemy(self):
        x = random.randint(0, SCREEN_WIDTH - 20)
```

```
    # Difficulty increases with waves
    rand = random.random()
    if self.wave >= 3 and rand < 0.2:
        enemy_type = 'tank'
    elif self.wave >= 2 and rand < 0.4:
        enemy_type = 'fast'
    else:
        enemy_type = 'basic'

    enemy = Enemy(x, -20, enemy_type)
    self.enemies.append(enemy)

def spawn_powerup(self):
    x = random.randint(20, SCREEN_WIDTH - 20)
    power_type = random.choice(['health', 'rapid_fire', 'shield'])
    powerup = PowerUp(x, -12, power_type)
    self.power_ups.append(powerup)

def create_explosion(self, x, y, color=CGA_COLORS['WHITE']):
    for _ in range(15):
        particle = Particle(x, y, color)
        self.particles.append(particle)

def check_collisions(self):
    # Check fighter bullets hitting enemies
    for bullet in self.fighter.bullets[:]:
        bullet_rect = pygame.Rect(bullet.x, bullet.y, bullet.width, bullet.height)
        for enemy in self.enemies[:]:
            enemy_rect = pygame.Rect(enemy.x, enemy.y, enemy.width, enemy.height)
            if bullet_rect.colliderect(enemy_rect):
                if bullet in self.fighter.bullets:
                    self.fighter.bullets.remove(bullet)

                if enemy.take_damage():
                    self.enemies.remove(enemy)
                    self.score += 10 if enemy.type == 'basic' else (15 if enemy.type == 'fast' else 25)
                    self.enemies_killed_this_wave += 1
                    self.create_explosion(enemy.x + enemy.width // 2,
                                          enemy.y + enemy.height // 2,
                                          CGA_COLORS['MAGENTA'])
                break

    # Check enemy bullets hitting fighter
    for enemy in self.enemies:
```

```
            for bullet in enemy.bullets[:]:
                bullet_rect = pygame.Rect(bullet.x, bullet.y, bullet.width, bullet.height)
                fighter_rect = pygame.Rect(self.fighter.x, self.fighter.y,
                        self.fighter.width, self.fighter.height)
                if bullet_rect.colliderect(fighter_rect):
                    if bullet in enemy.bullets:
                        enemy.bullets.remove(bullet)
                    if self.fighter.take_damage(15):
                        self.create_explosion(self.fighter.x + self.fighter.width // 2,
                                self.fighter.y + self.fighter.height // 2,
                                CGA_COLORS['CYAN'])
                    if self.fighter.health <= 0:
                        self.game_over = True

        # Check enemies colliding with fighter
        fighter_rect = pygame.Rect(self.fighter.x, self.fighter.y,
                self.fighter.width, self.fighter.height)
        for enemy in self.enemies[:]:
            enemy_rect = pygame.Rect(enemy.x, enemy.y, enemy.width, enemy.height)
            if fighter_rect.colliderect(enemy_rect):
                self.enemies.remove(enemy)
                if self.fighter.take_damage(25):
                    self.create_explosion(enemy.x + enemy.width // 2,
                            enemy.y + enemy.height // 2,
                            CGA_COLORS['MAGENTA'])
                if self.fighter.health <= 0:
                    self.game_over = True

        # Check power-up collection
        for powerup in self.power_ups[:]:
            powerup_rect = pygame.Rect(powerup.x, powerup.y, powerup.width, powerup.height)
            if fighter_rect.colliderect(powerup_rect):
                self.power_ups.remove(powerup)
                self.fighter.activate_power_up(powerup.type)
                self.score += 5

    def handle_input(self):
        keys = pygame.key.get_pressed()
        dx, dy = 0, 0

        if keys[pygame.K_LEFT] or keys[pygame.K_a]:
            dx = -1
        if keys[pygame.K_RIGHT] or keys[pygame.K_d]:
            dx = 1
```

```
        if keys[pygame.K_UP] or keys[pygame.K_w]:
            dy = -1
        if keys[pygame.K_DOWN] or keys[pygame.K_s]:
            dy = 1

        self.fighter.move(dx, dy)

        if keys[pygame.K_SPACE]:
            self.fighter.shoot()

    def update(self):
        if self.paused or self.game_over:
            return

        # Update starfield
        self.starfield.update()

        # Check for wave completion
        if self.enemies_killed_this_wave >= self.enemies_per_wave and len(self.enemies) == 0:
            self.wave += 1
            self.enemies_killed_this_wave = 0
            self.enemies_per_wave += 5
            self.score += 50  # Wave completion bonus

        # Spawn enemies (faster spawning as waves progress)
        self.enemy_spawn_timer += 1
        spawn_rate = max(30, 90 - (self.wave * 5))
        if self.enemy_spawn_timer > spawn_rate and self.enemies_killed_this_wave <
self.enemies_per_wave:
            self.spawn_enemy()
            self.enemy_spawn_timer = 0

        # Spawn power-ups occasionally
        self.powerup_spawn_timer += 1
        if self.powerup_spawn_timer > 600:  # Every 10 seconds
            if random.random() < 0.5:
                self.spawn_powerup()
            self.powerup_spawn_timer = 0

        # Update game objects
        self.fighter.update()

        for enemy in self.enemies[:]:
            enemy.update()
```

```
            if enemy.y > SCREEN_HEIGHT:
                self.enemies.remove(enemy)

        for powerup in self.power_ups[:]:
            powerup.update()
            if powerup.y > SCREEN_HEIGHT:
                self.power_ups.remove(powerup)

        for particle in self.particles[:]:
            particle.update()
            if not particle.is_alive():
                self.particles.remove(particle)

        # Check collisions
        self.check_collisions()

        # Update high score
        if self.score > self.high_score:
            self.high_score = self.score

    def draw(self):
        # Fill screen with black
        self.screen.fill(CGA_COLORS['BLACK'])

        # Draw starfield
        self.starfield.draw(self.screen)

        if not self.game_over:
            # Draw particles first (behind everything)
            for particle in self.particles:
                particle.draw(self.screen)

            # Draw game objects
            self.fighter.draw(self.screen)
            for enemy in self.enemies:
                enemy.draw(self.screen)
            for powerup in self.power_ups:
                powerup.draw(self.screen)

            # Draw UI
            score_text = self.small_font.render(f"SCORE: {self.score}", True,
CGA_COLORS['WHITE'])
            self.screen.blit(score_text, (10, 10))
```

```
        wave_text = self.small_font.render(f"WAVE: {self.wave}", True, CGA_COLORS['CYAN'])
        self.screen.blit(wave_text, (10, 35))

        # Health bar
        health_width = 100
        health_height = 10
        health_x = 10
        health_y = 60
        pygame.draw.rect(self.screen, CGA_COLORS['WHITE'],
                         (health_x, health_y, health_width, health_height), 1)
        health_fill = int((self.fighter.health / self.fighter.max_health) * health_width)
        if health_fill > 0:
            health_color = CGA_COLORS['MAGENTA'] if self.fighter.health < 30 else
CGA_COLORS['CYAN']
            pygame.draw.rect(self.screen, health_color,
                             (health_x + 1, health_y + 1, health_fill - 2, health_height - 2))

        # Power-up indicators
        if self.fighter.rapid_fire_timer > 0:
            rapid_text = self.small_font.render("RAPID FIRE", True, CGA_COLORS['CYAN'])
            self.screen.blit(rapid_text, (SCREEN_WIDTH - 130, 10))

        if self.fighter.shield_timer > 0:
            shield_text = self.small_font.render("SHIELD", True, CGA_COLORS['WHITE'])
            self.screen.blit(shield_text, (SCREEN_WIDTH - 90, 35))

        # Wave progress
        progress = min(self.enemies_killed_this_wave, self.enemies_per_wave)
        progress_text = self.small_font.render(f"{progress}/{self.enemies_per_wave}", True,
CGA_COLORS['WHITE'])
        self.screen.blit(progress_text, (10, 80))

        if self.paused:
            pause_text = self.font.render("PAUSED", True, CGA_COLORS['WHITE'])
            self.screen.blit(pause_text, (SCREEN_WIDTH//2 - pause_text.get_width()//2,
SCREEN_HEIGHT//2))
    else:
        # Draw particles
        for particle in self.particles:
            particle.draw(self.screen)

        # Game Over screen
        game_over_text = self.font.render("GAME OVER", True, CGA_COLORS['MAGENTA'])
```

```
        final_score_text = self.font.render(f"SCORE: {self.score}", True,
CGA_COLORS['WHITE'])
        wave_text = self.small_font.render(f"Reached Wave: {self.wave}", True,
CGA_COLORS['CYAN'])
        high_score_text = self.small_font.render(f"High Score: {self.high_score}", True,
CGA_COLORS['CYAN'])
        restart_text = self.small_font.render("Press R to Restart or Q to Quit", True,
CGA_COLORS['WHITE'])

        self.screen.blit(game_over_text, (SCREEN_WIDTH//2 - game_over_text.get_width()//2,
SCREEN_HEIGHT//2 - 80))
        self.screen.blit(final_score_text, (SCREEN_WIDTH//2 - final_score_text.get_width()//2,
SCREEN_HEIGHT//2 - 30))
        self.screen.blit(wave_text, (SCREEN_WIDTH//2 - wave_text.get_width()//2,
SCREEN_HEIGHT//2))
        self.screen.blit(high_score_text, (SCREEN_WIDTH//2 - high_score_text.get_width()//2,
SCREEN_HEIGHT//2 + 25))
        self.screen.blit(restart_text, (SCREEN_WIDTH//2 - restart_text.get_width()//2,
SCREEN_HEIGHT//2 + 60))

    pygame.display.flip()

def restart(self):
    self.fighter = Fighter(SCREEN_WIDTH // 2, SCREEN_HEIGHT - 50)
    self.enemies = []
    self.power_ups = []
    self.particles = []
    self.enemy_spawn_timer = 0
    self.powerup_spawn_timer = 0
    self.score = 0
    self.wave = 1
    self.enemies_killed_this_wave = 0
    self.enemies_per_wave = 10
    self.game_over = False
    self.paused = False

def run(self):
    while self.running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False
            elif event.type == pygame.KEYDOWN:
                if self.game_over:
                    if event.key == pygame.K_r:
```

```
                    self.restart()
                elif event.key == pygame.K_q:
                    self.running = False
            else:
                if event.key == pygame.K_p:
                    self.paused = not self.paused
                elif event.key == pygame.K_ESCAPE:
                    self.paused = not self.paused

        if not self.game_over and not self.paused:
            self.handle_input()

        self.update()
        self.draw()
        self.clock.tick(FPS)

    pygame.quit()
    sys.exit()

if __name__ == "__main__":
    game = Game()
    game.run()
```