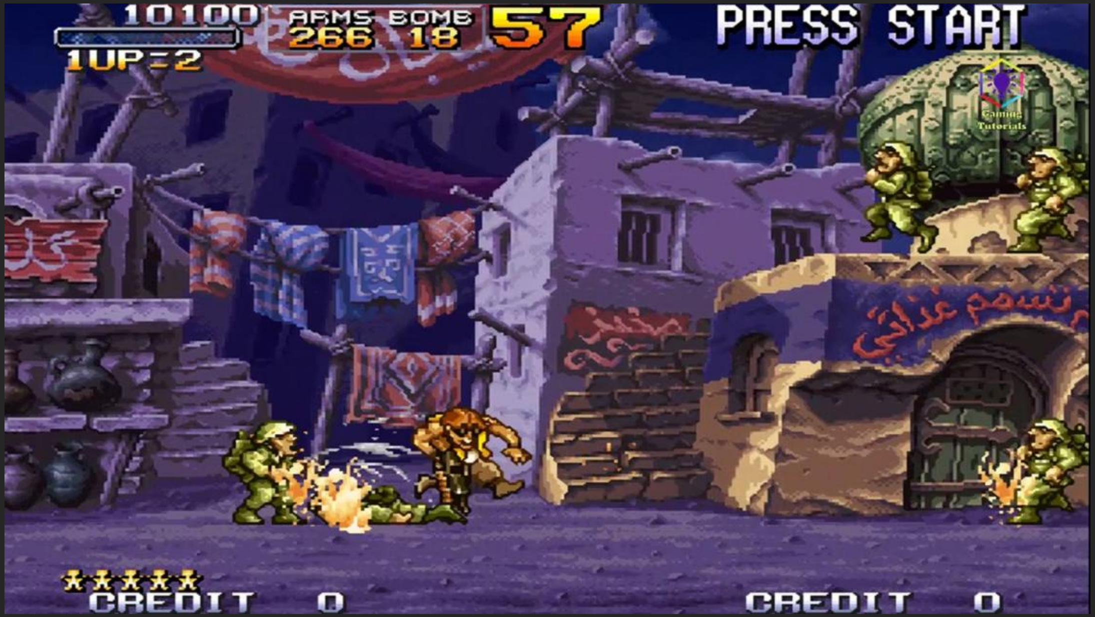
METAL SLUG

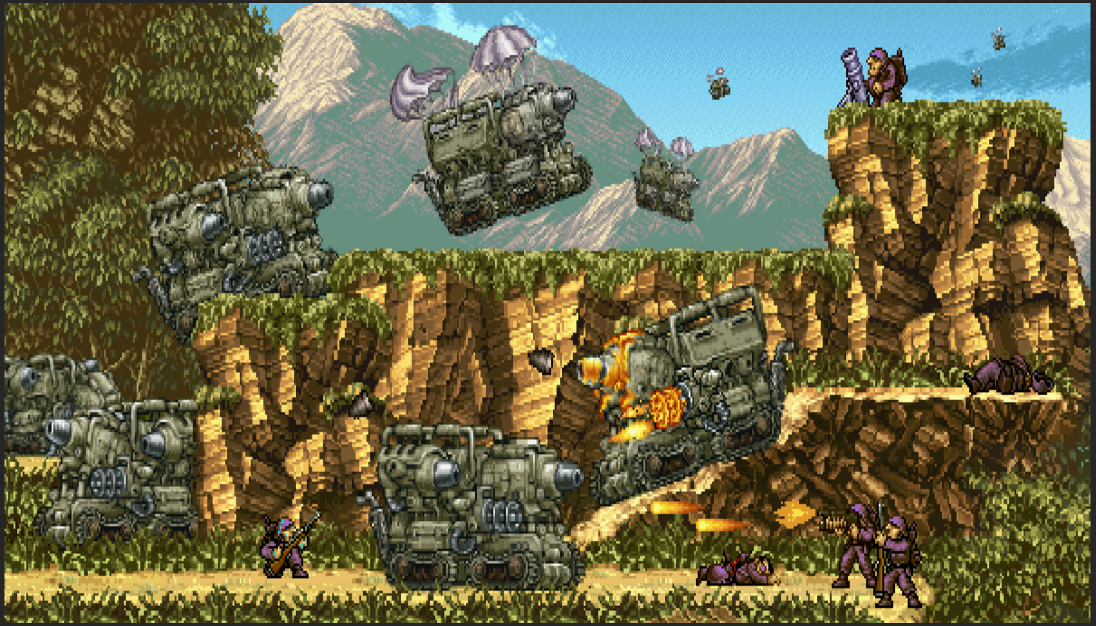
TEAM ONE

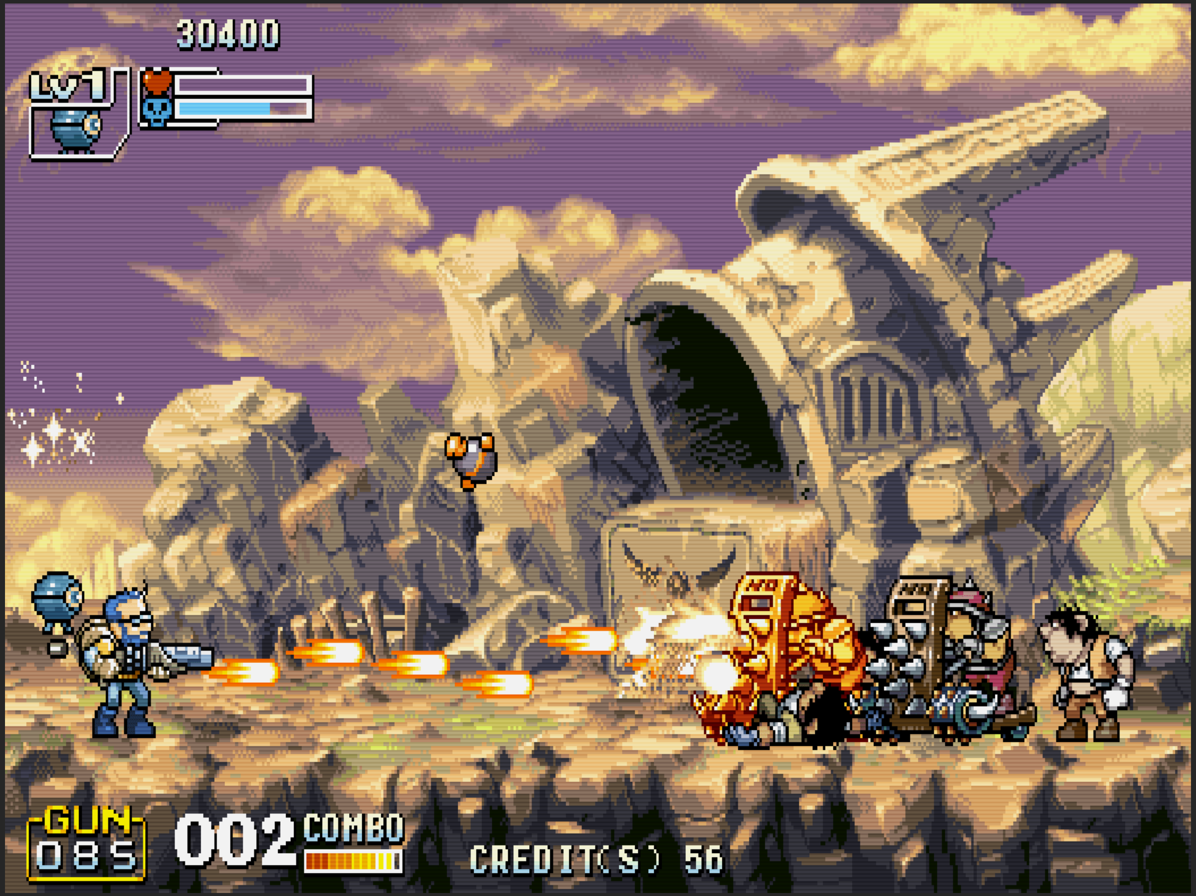
OUTLINE

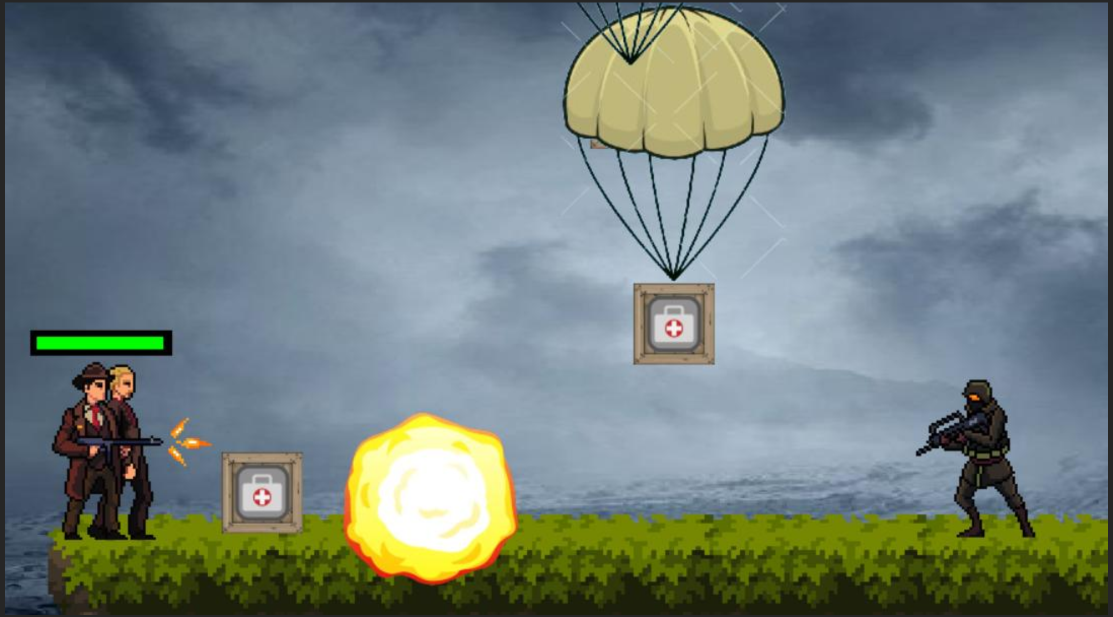
- Introduction
- Idea
- Gameplay
- Design

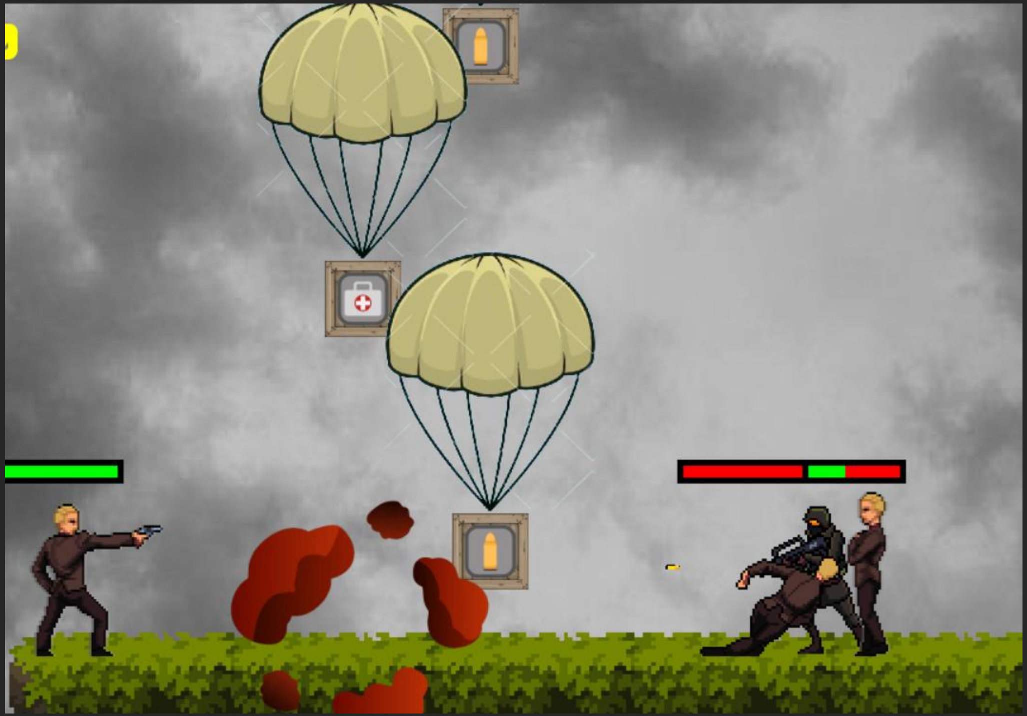
ORIGINAL

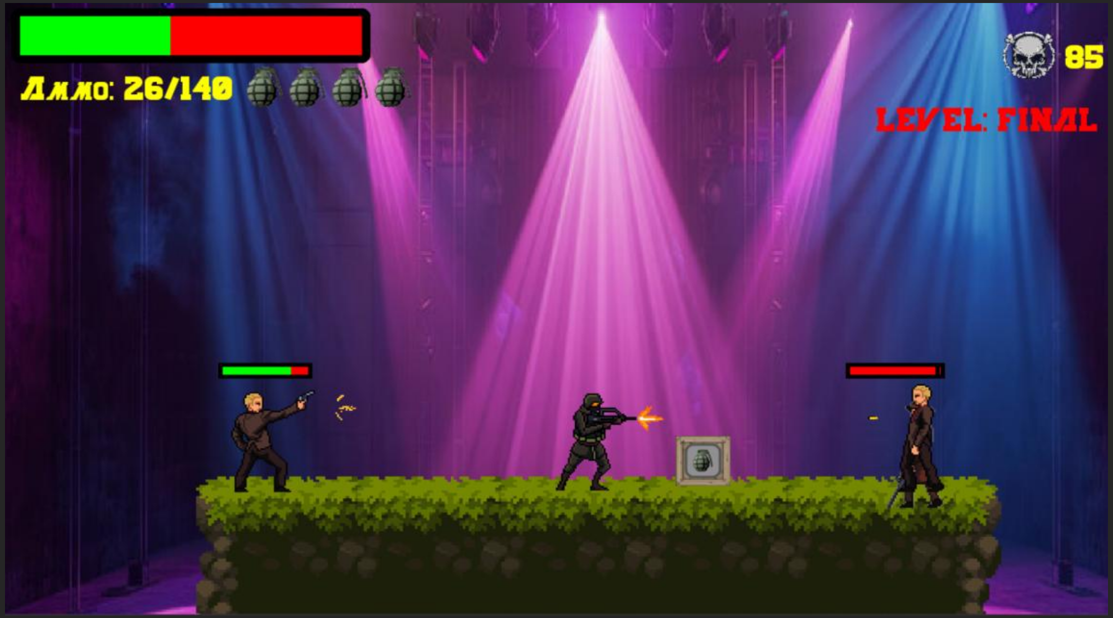

10100
ARMS BOMB
266 18
57
1UP=2
PRESS START
CREDIT 0
CREDIT 0

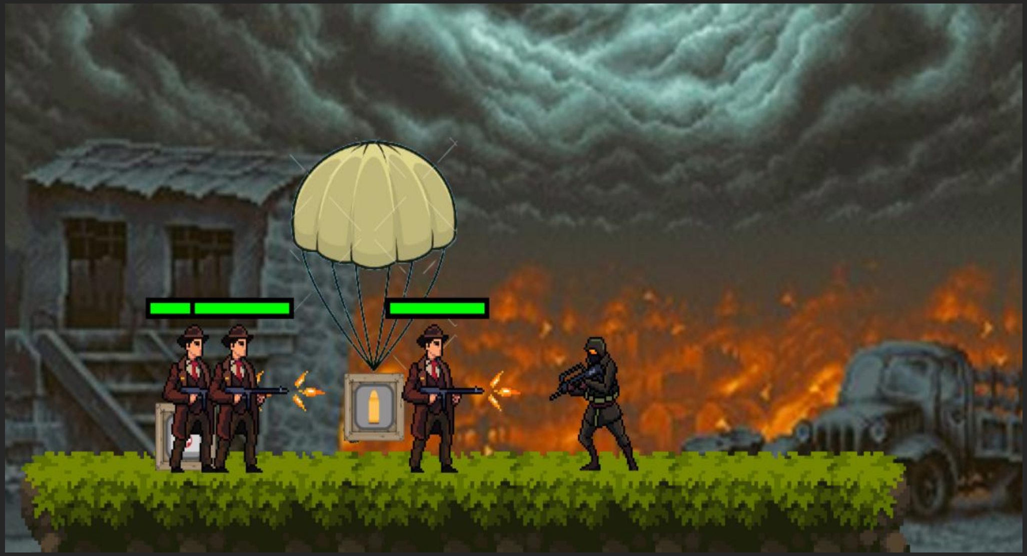



30400
Lv1
GUN
085
002 COMBO
CREDIT(S) 56






85
Ammo: 26/140
LEVEL: FINAL

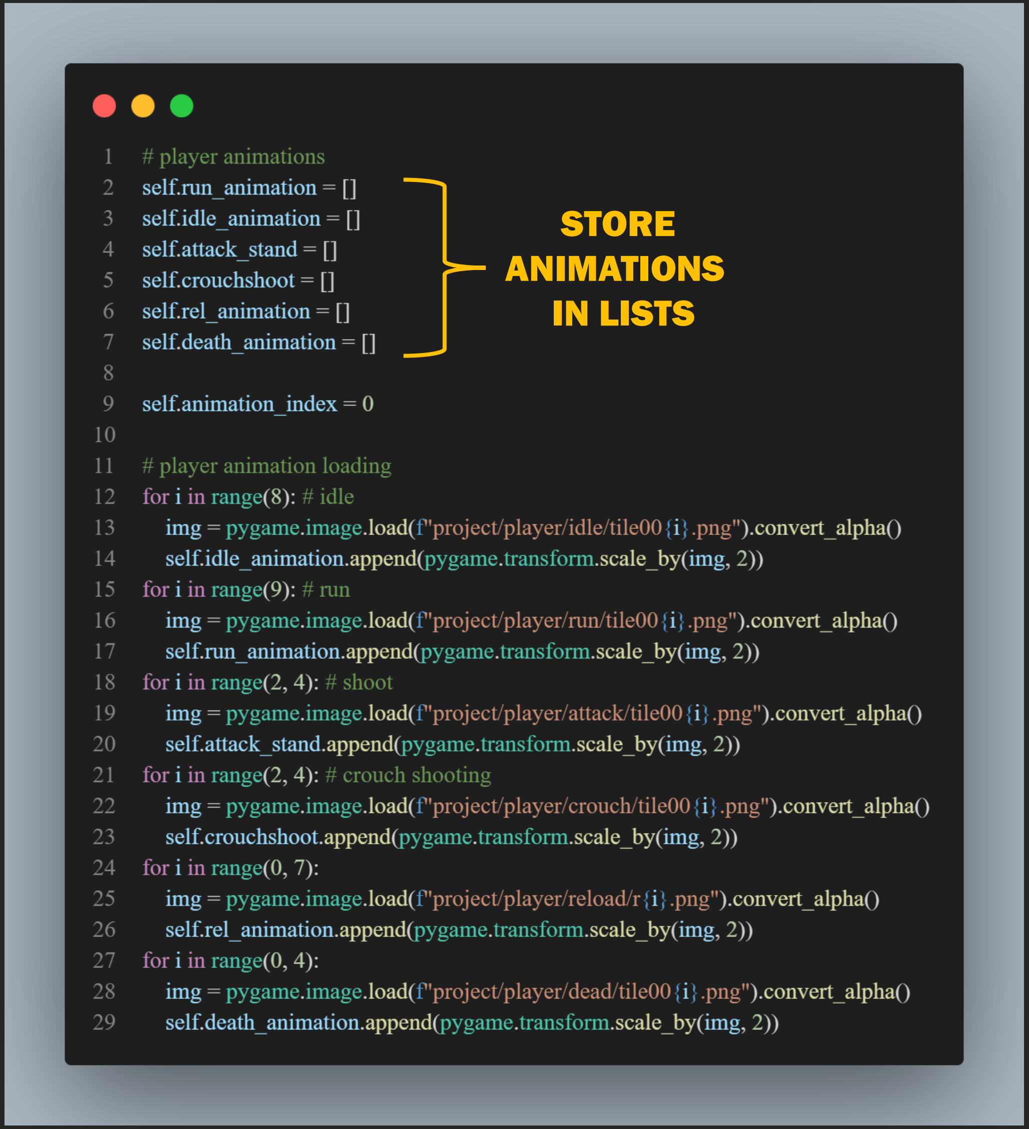


PLAYER

- Animations
- Physics
- Interactions

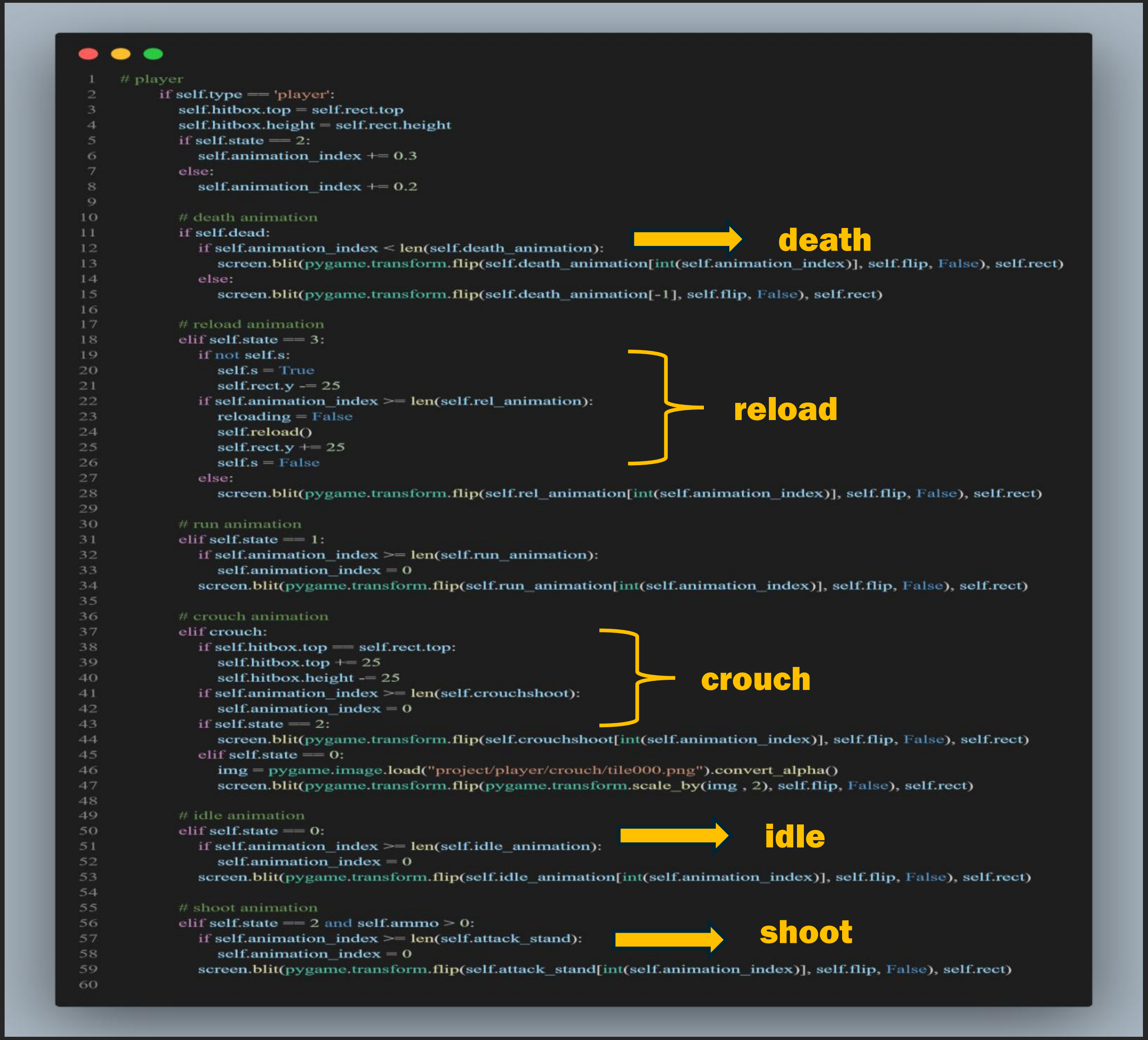
LOADING

PLAYING



```python
# player animations
self.run_animation = []
self.idle_animation = []
self.attack_stand = []
self.crouchshoot = []
self.rel_animation = []
self.death_animation = []

self.animation_index = 0

# player animation loading
for i in range(8): # idle
    img = pygame.image.load(f"project/player/idle/tile00{i}.png").convert_alpha()
    self.idle_animation.append(pygame.transform.scale_by(img, 2))
for i in range(9): # run
    img = pygame.image.load(f"project/player/run/tile00{i}.png").convert_alpha()
    self.run_animation.append(pygame.transform.scale_by(img, 2))
for i in range(2, 4): # shoot
    img = pygame.image.load(f"project/player/attack/tile00{i}.png").convert_alpha()
    self.attack_stand.append(pygame.transform.scale_by(img, 2))
for i in range(2, 4): # crouch shooting
    img = pygame.image.load(f"project/player/crouch/tile00{i}.png").convert_alpha()
    self.crouchshoot.append(pygame.transform.scale_by(img, 2))
for i in range(0, 7):
    img = pygame.image.load(f"project/player/reload/r{i}.png").convert_alpha()
    self.rel_animation.append(pygame.transform.scale_by(img, 2))
for i in range(0, 4):
    img = pygame.image.load(f"project/player/dead/tile00{i}.png").convert_alpha()
    self.death_animation.append(pygame.transform.scale_by(img, 2))
```

STORE ANIMATIONS IN LISTS



```python
# player
        if self.type == 'player':
            self.hitbox.top = self.rect.top
            self.hitbox.height = self.rect.height
            if self.state == 2:
                self.animation_index += 0.3
            else:
                self.animation_index += 0.2

            # death animation
            if self.dead:
                if self.animation_index < len(self.death_animation):
                    screen.blit(pygame.transform.flip(self.death_animation[int(self.animation_index)], self.flip, False), self.rect)
                else:
                    screen.blit(pygame.transform.flip(self.death_animation[-1], self.flip, False), self.rect)

            # reload animation
            elif self.state == 3:
                if not self.s:
                    self.s = True
                    self.rect.y -= 25
                if self.animation_index >= len(self.rel_animation):
                    reloading = False
                    self.reload()
                    self.rect.y += 25
                    self.s = False
                else:
                    screen.blit(pygame.transform.flip(self.rel_animation[int(self.animation_index)], self.flip, False), self.rect)

            # run animation
            elif self.state == 1:
                if self.animation_index >= len(self.run_animation):
                    self.animation_index = 0
                screen.blit(pygame.transform.flip(self.run_animation[int(self.animation_index)], self.flip, False), self.rect)

            # crouch animation
            elif crouch:
                if self.hitbox.top == self.rect.top:
                    self.hitbox.top += 25
                    self.hitbox.height -= 25
                if self.animation_index >= len(self.crouchshoot):
                    self.animation_index = 0
                if self.state == 2:
                    screen.blit(pygame.transform.flip(self.crouchshoot[int(self.animation_index)], self.flip, False), self.rect)
                elif self.state == 0:
                    img = pygame.image.load("project/player/crouch/tile000.png").convert_alpha()
                    screen.blit(pygame.transform.flip(pygame.transform.scale_by(img , 2), self.flip, False), self.rect)

            # idle animation
            elif self.state == 0:
                if self.animation_index >= len(self.idle_animation):
                    self.animation_index = 0
                screen.blit(pygame.transform.flip(self.idle_animation[int(self.animation_index)], self.flip, False), self.rect)

            # shoot animation
            elif self.state == 2 and self.ammo > 0:
                if self.animation_index >= len(self.attack_stand):
                    self.animation_index = 0
                screen.blit(pygame.transform.flip(self.attack_stand[int(self.animation_index)], self.flip, False), self.rect)
```

death

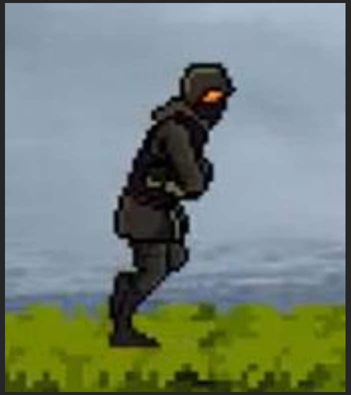
reload

crouch

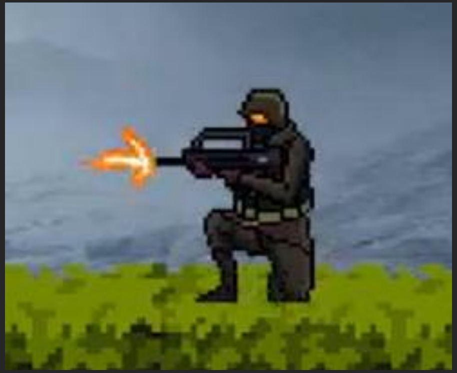
idle

shoot

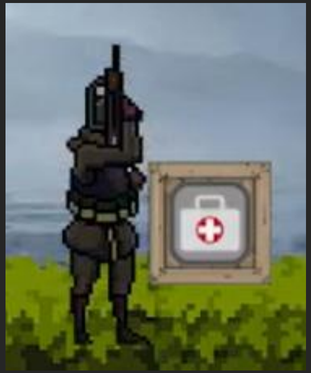
RUNNING



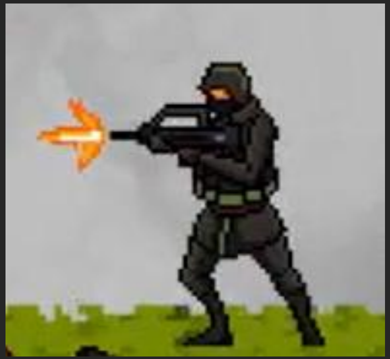
CROUCHING



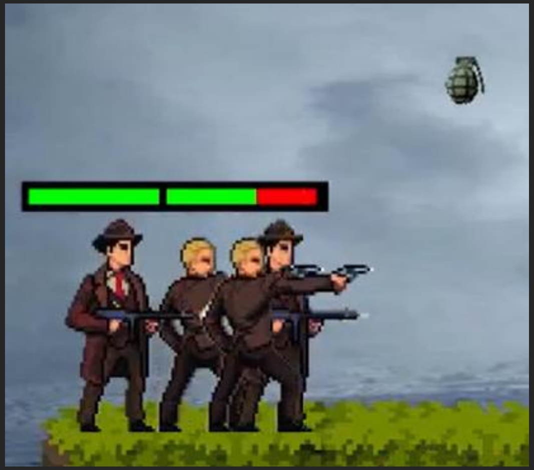
RELOADING



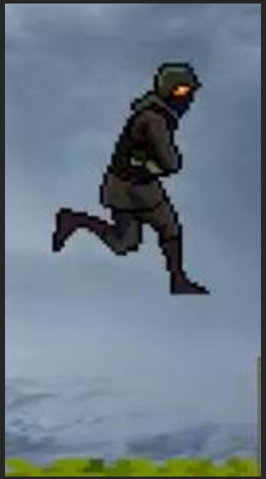
SHOOTING



GRENADE

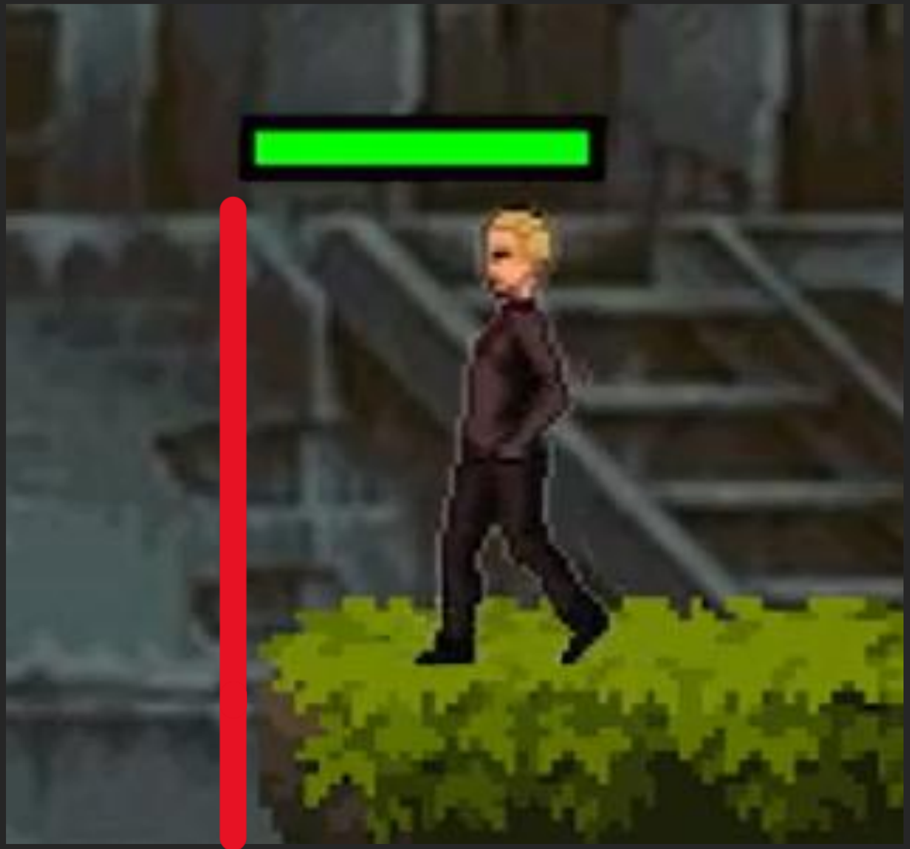


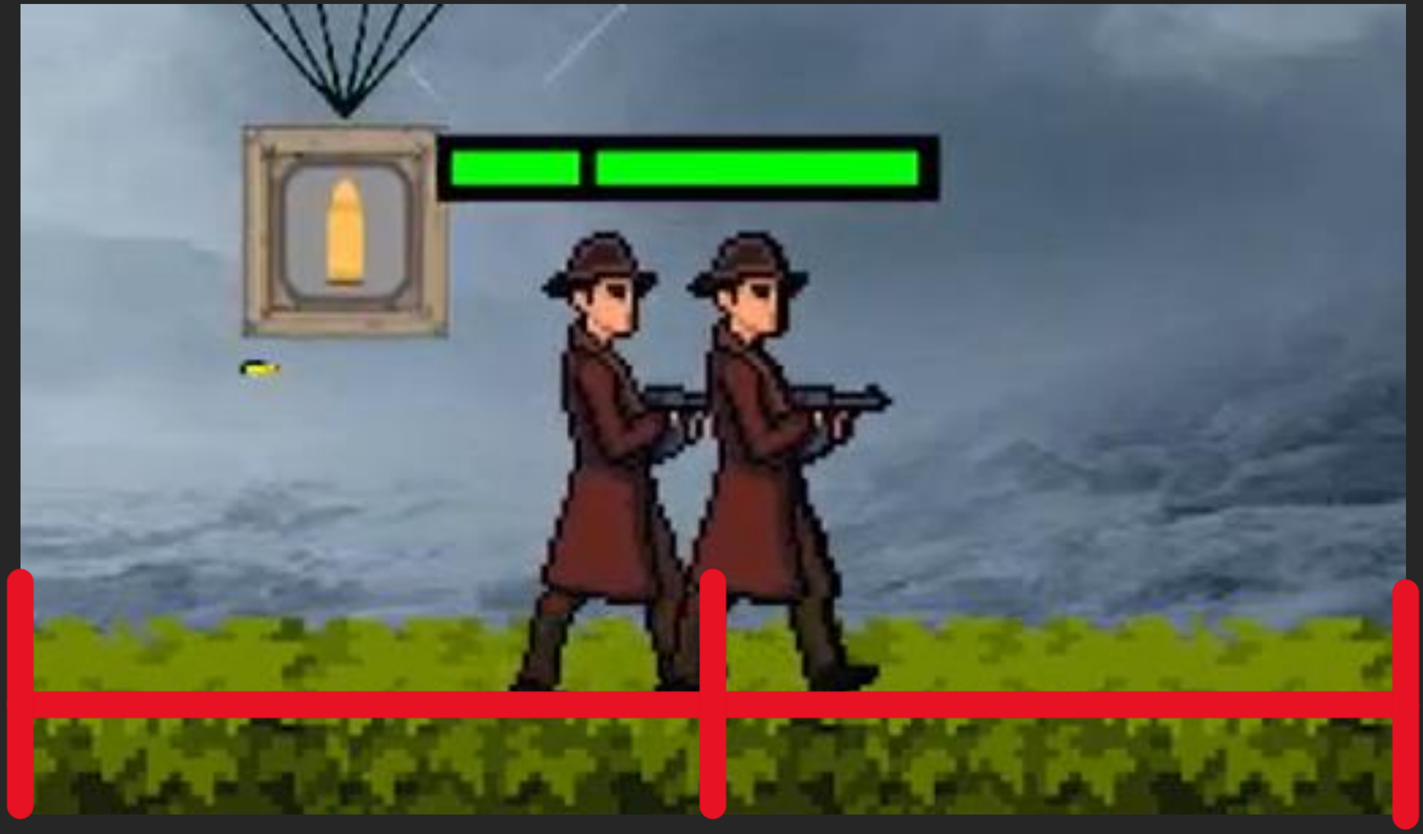
JUMPING



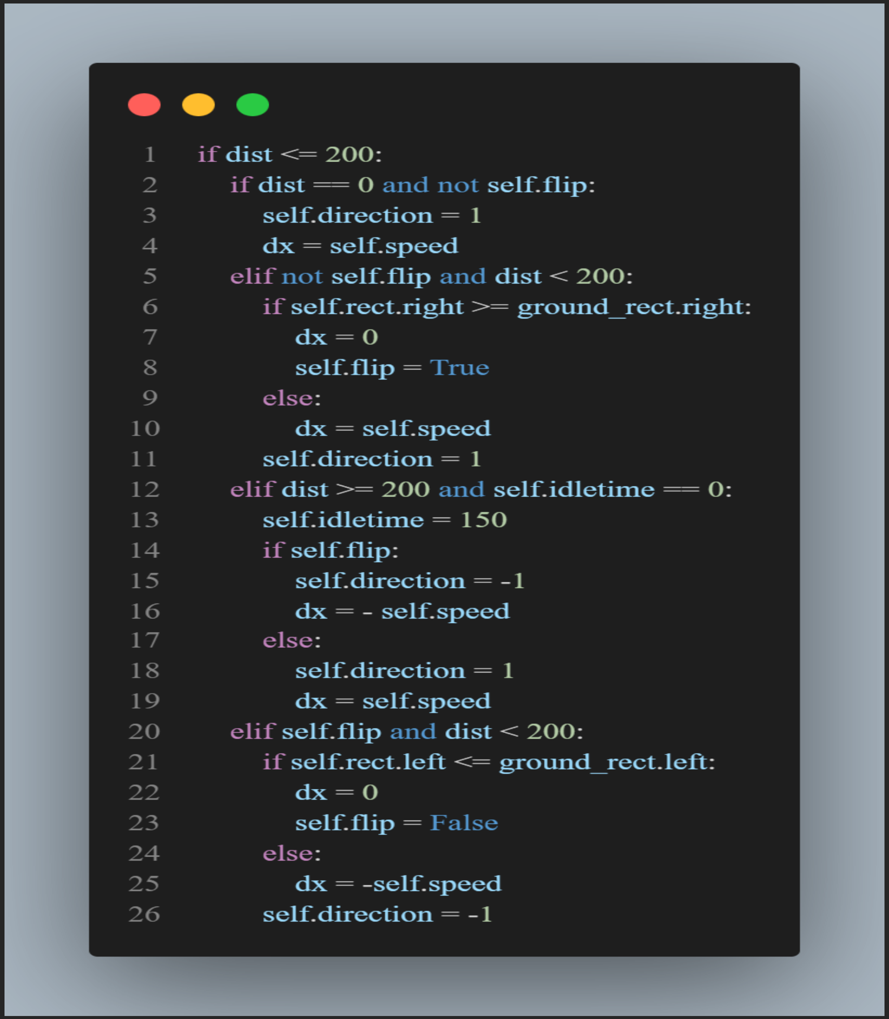
ENEMIES

AI BEHAVIOUR

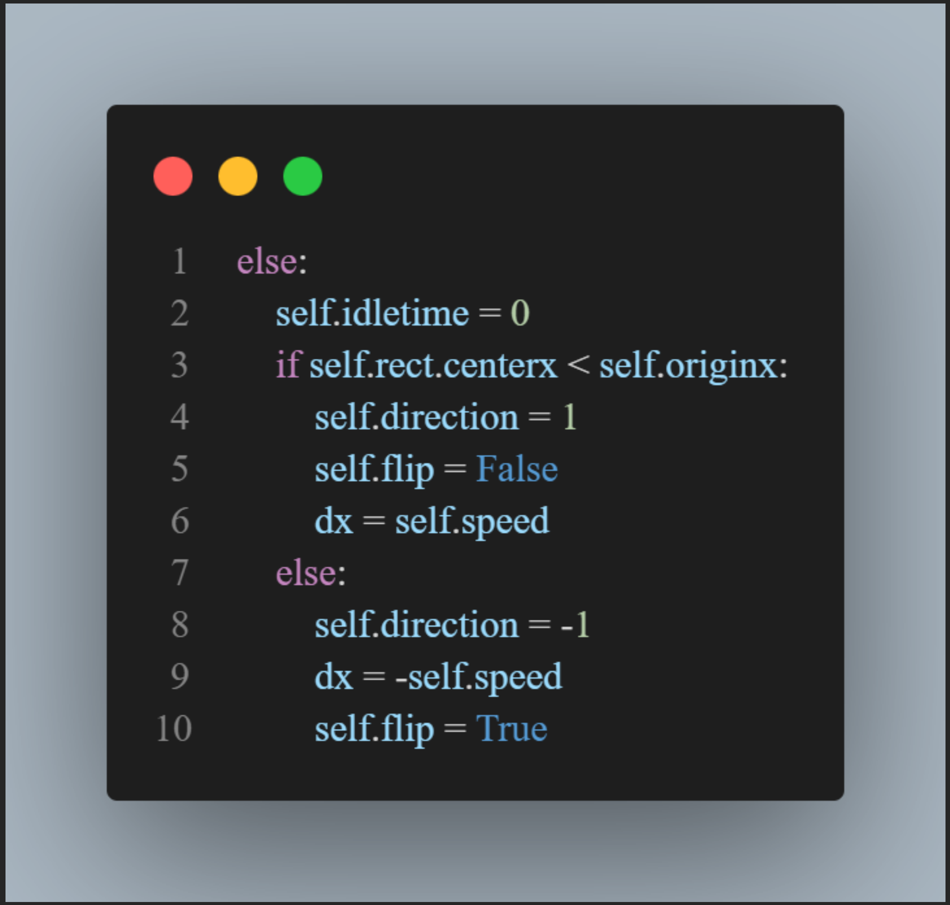




AI BEHAVIOUR

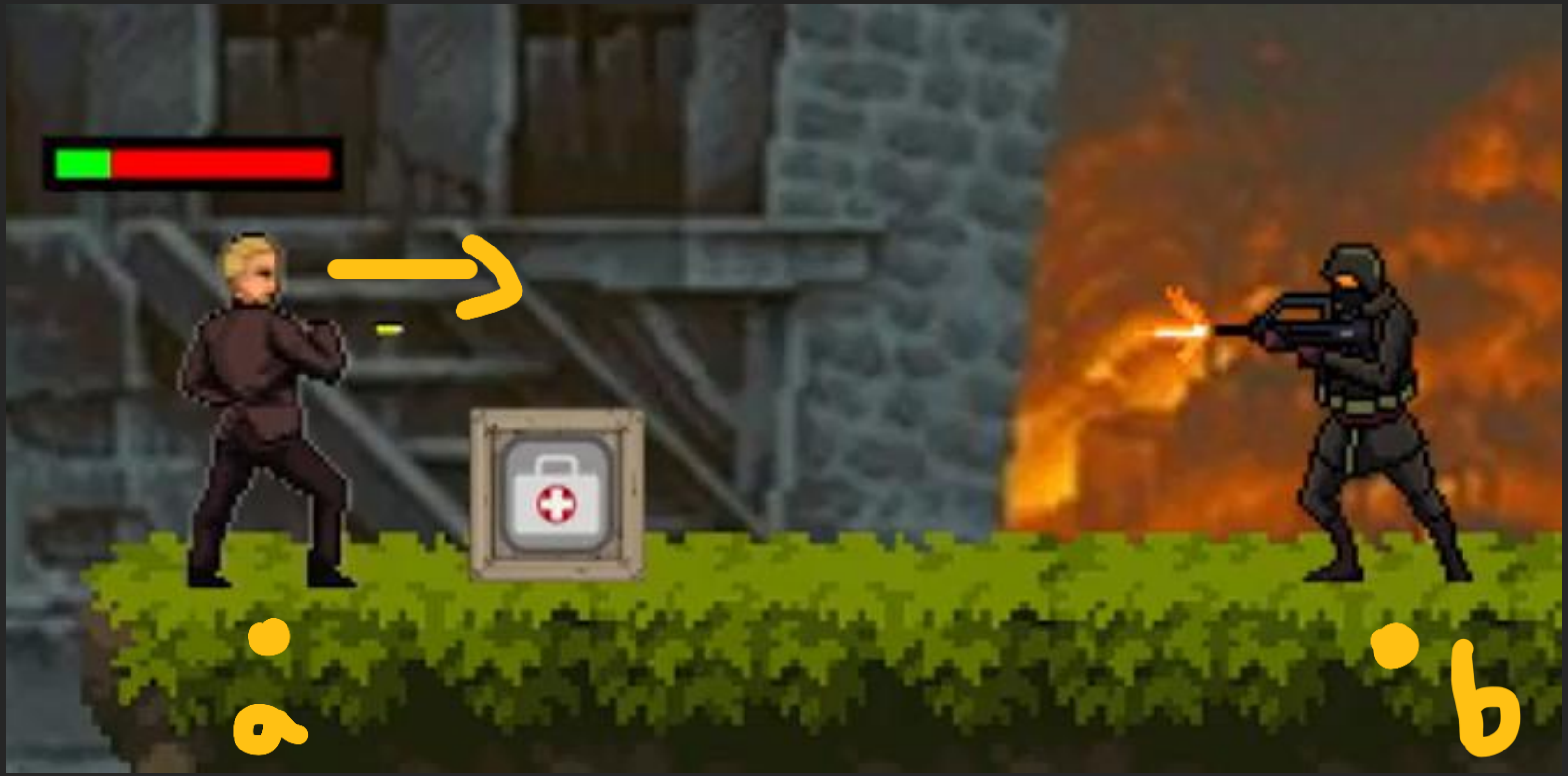


```
if dist <= 200:
    if dist == 0 and not self.flip:
        self.direction = 1
        dx = self.speed
    elif not self.flip and dist < 200:
        if self.rect.right >= ground_rect.right:
            dx = 0
            self.flip = True
        else:
            dx = self.speed
        self.direction = 1
    elif dist >= 200 and self.idletime == 0:
        self.idletime = 150
        if self.flip:
            self.direction = -1
            dx = - self.speed
        else:
            self.direction = 1
            dx = self.speed
    elif self.flip and dist < 200:
        if self.rect.left <= ground_rect.left:
            dx = 0
            self.flip = False
        else:
            dx = -self.speed
        self.direction = -1
```

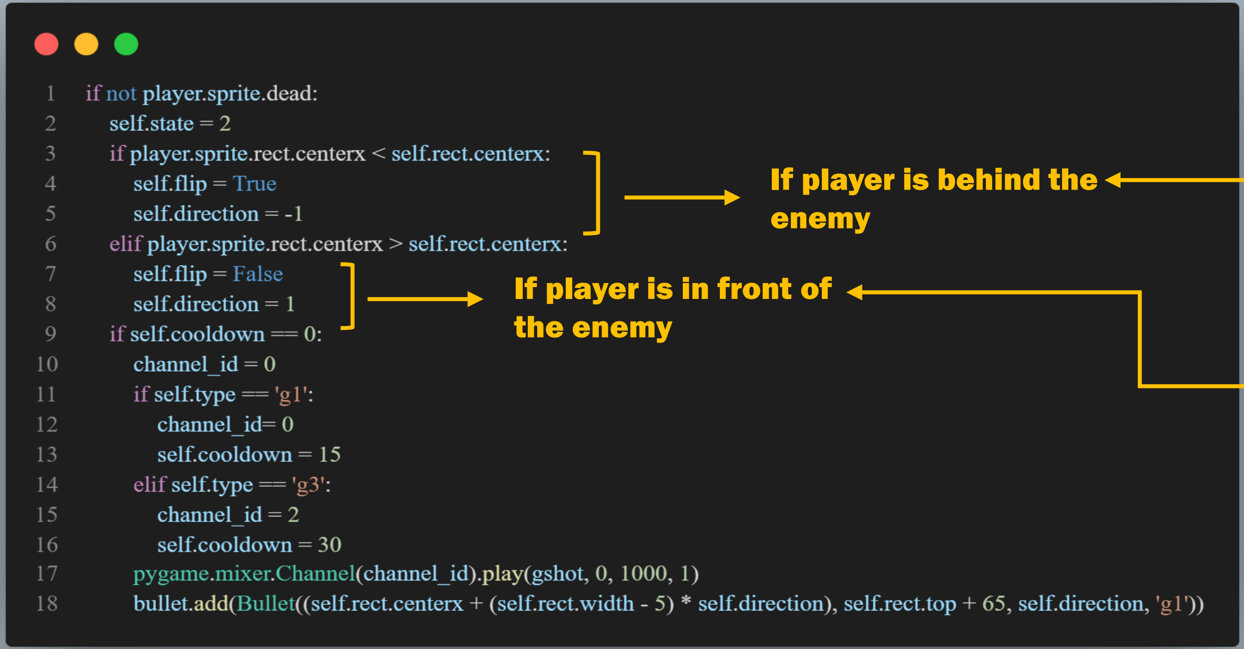


```
else:
    self.idletime = 0
    if self.rect.centerx < self.originx:
        self.direction = 1
        self.flip = False
        dx = self.speed
    else:
        self.direction = -1
        dx = -self.speed
        self.flip = True
```

AI BEHAVIOUR



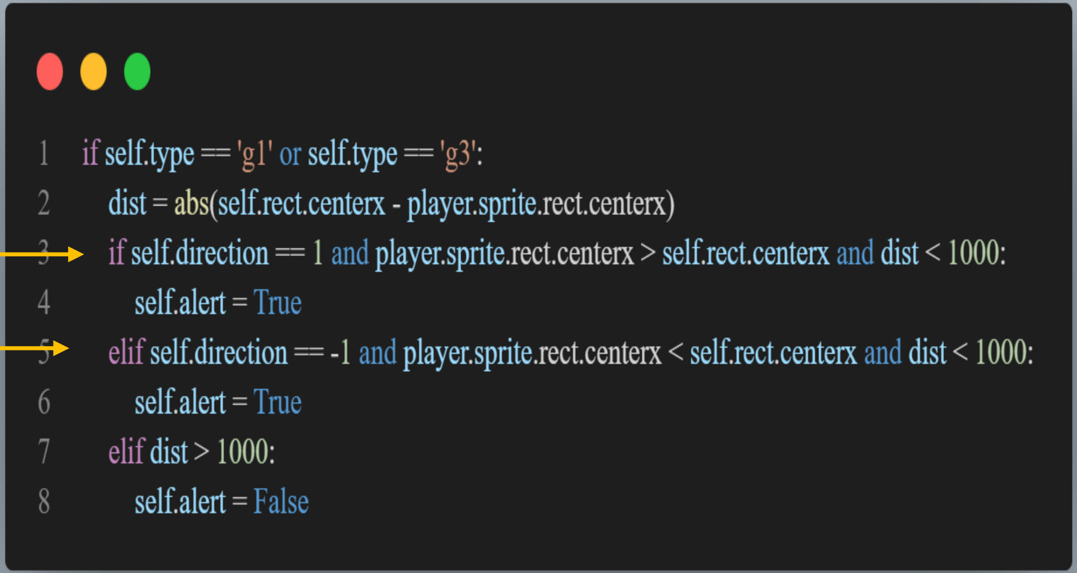
AI BEHAVIOUR



```
if not player.sprite.dead:
    self.state = 2
    if player.sprite.rect.centerx < self.rect.centerx:
        self.flip = True
        self.direction = -1
    elif player.sprite.rect.centerx > self.rect.centerx:
        self.flip = False
        self.direction = 1
    if self.cooldown == 0:
        channel_id = 0
        if self.type == 'g1':
            channel_id= 0
            self.cooldown = 15
        elif self.type == 'g3':
            channel_id = 2
            self.cooldown = 30
        pygame.mixer.Channel(channel_id).play(gshot, 0, 1000, 1)
        bullet.add(Bullet((self.rect.centerx + (self.rect.width - 5) * self.direction), self.rect.top + 65, self.direction, 'g1'))
```

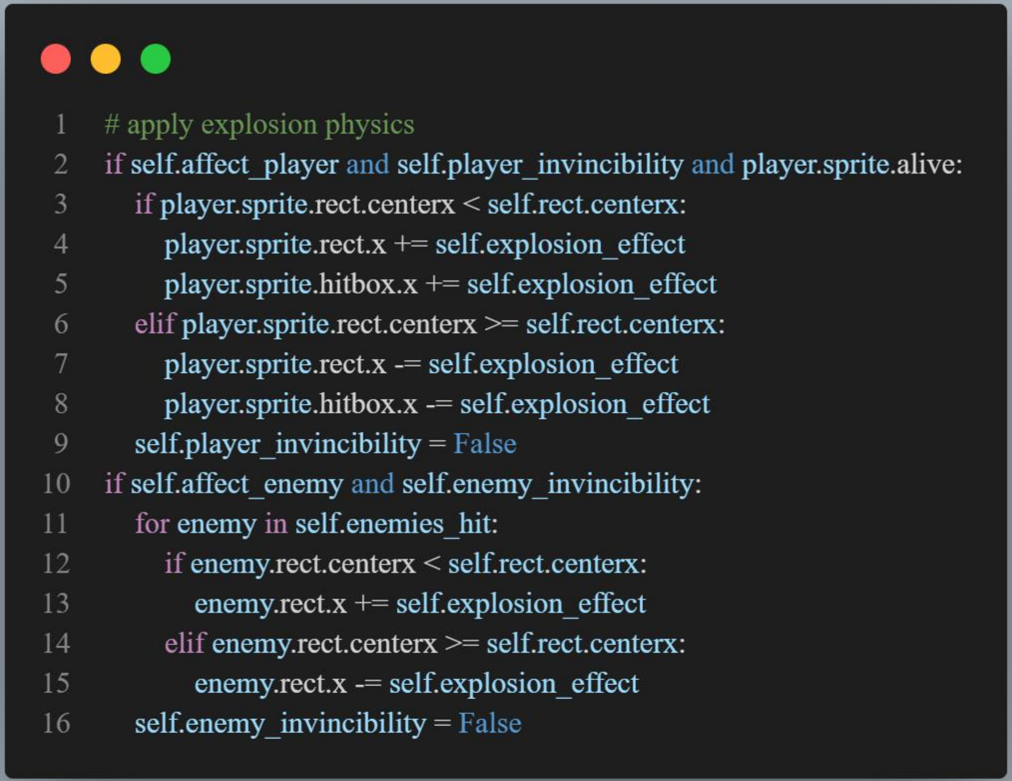
If player is behind the enemy

If player is in front of the enemy

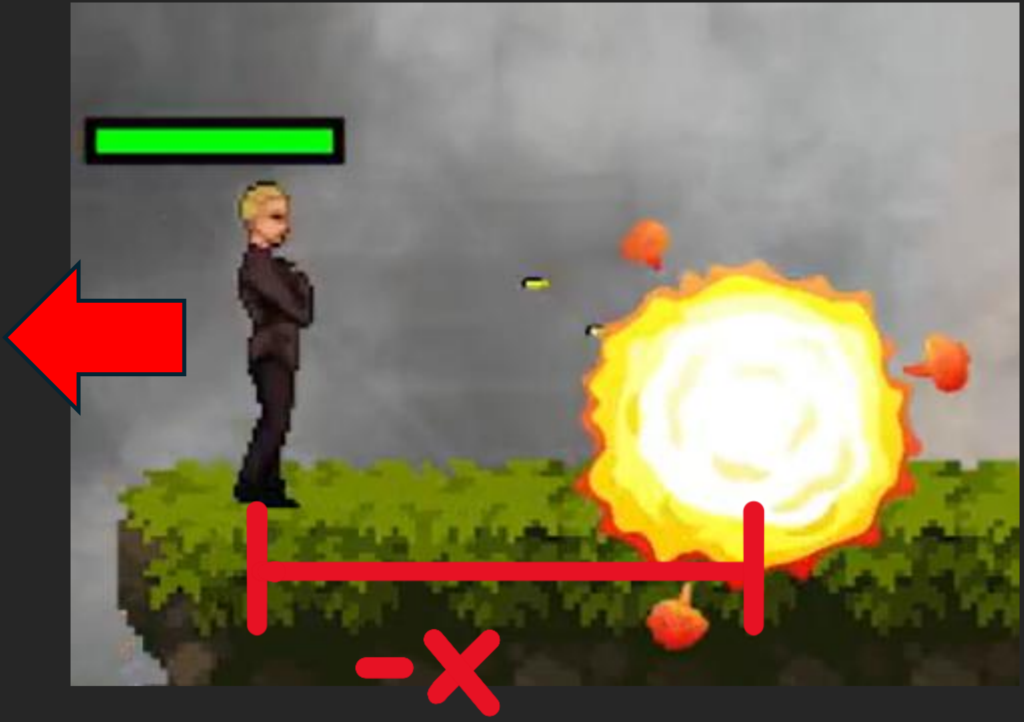


```
if self.type == 'g1' or self.type == 'g3':
    dist = abs(self.rect.centerx - player.sprite.rect.centerx)
    if self.direction == 1 and player.sprite.rect.centerx > self.rect.centerx and dist < 1000:
        self.alert = True
    elif self.direction == -1 and player.sprite.rect.centerx < self.rect.centerx and dist < 1000:
        self.alert = True
    elif dist > 1000:
        self.alert = False
```

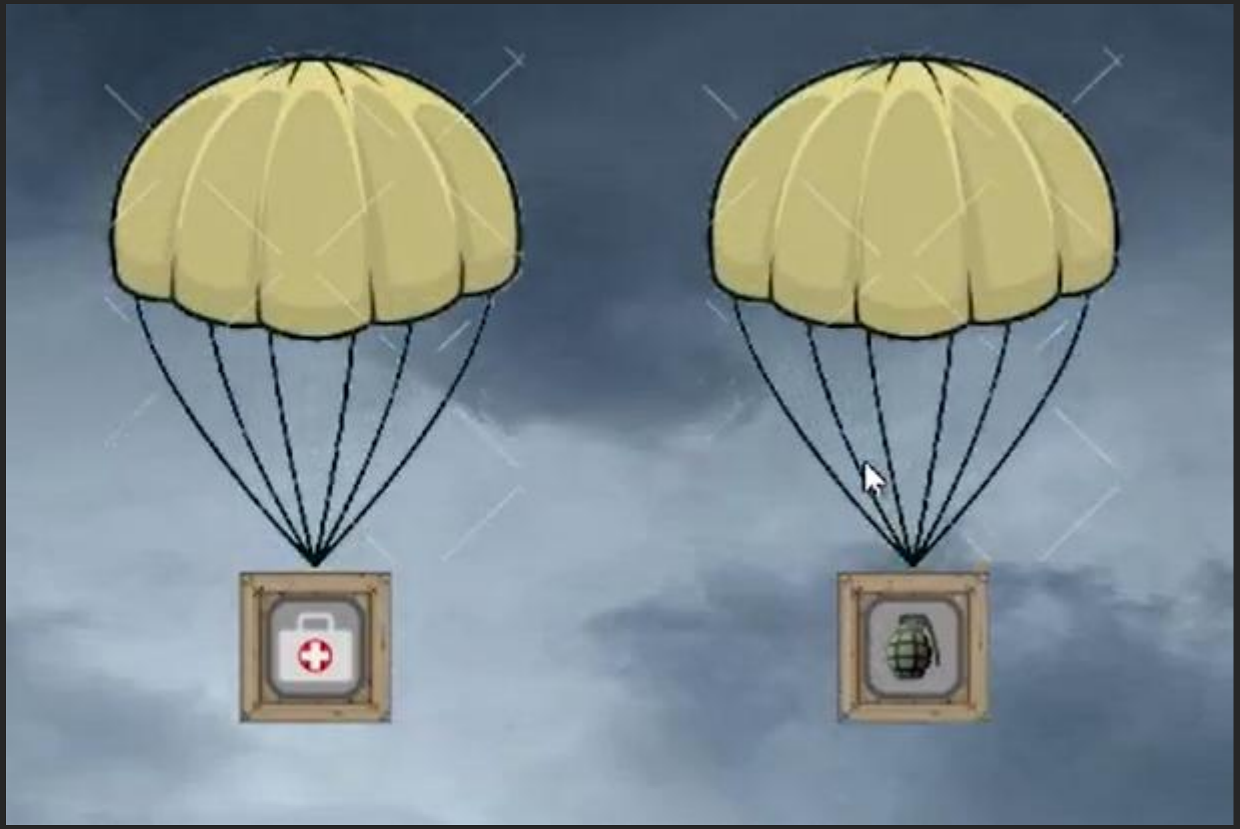
GRENADE



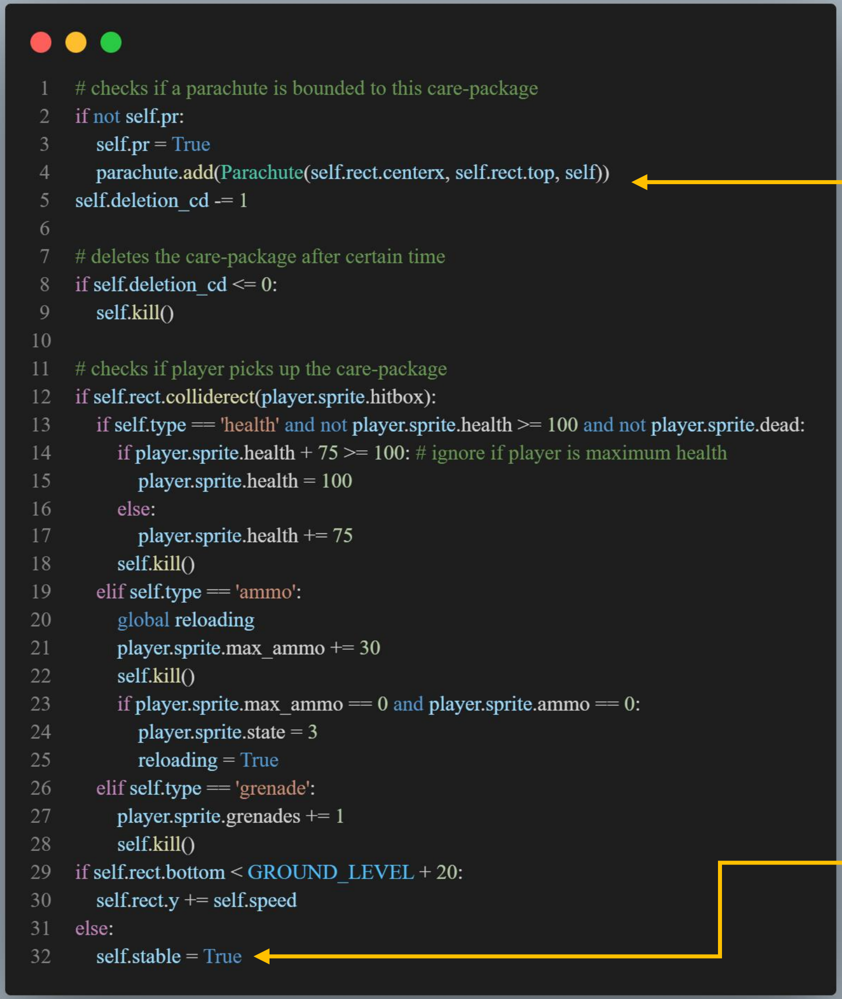
```python
# apply explosion physics
if self.affect_player and self.player_invincibility and player.sprite.alive:
    if player.sprite.rect.centerx < self.rect.centerx:
        player.sprite.rect.x += self.explosion_effect
        player.sprite.hitbox.x += self.explosion_effect
    elif player.sprite.rect.centerx >= self.rect.centerx:
        player.sprite.rect.x -= self.explosion_effect
        player.sprite.hitbox.x -= self.explosion_effect
    self.player_invincibility = False
if self.affect_enemy and self.enemy_invincibility:
    for enemy in self.enemies_hit:
        if enemy.rect.centerx < self.rect.centerx:
            enemy.rect.x += self.explosion_effect
        elif enemy.rect.centerx >= self.rect.centerx:
            enemy.rect.x -= self.explosion_effect
    self.enemy_invincibility = False
```



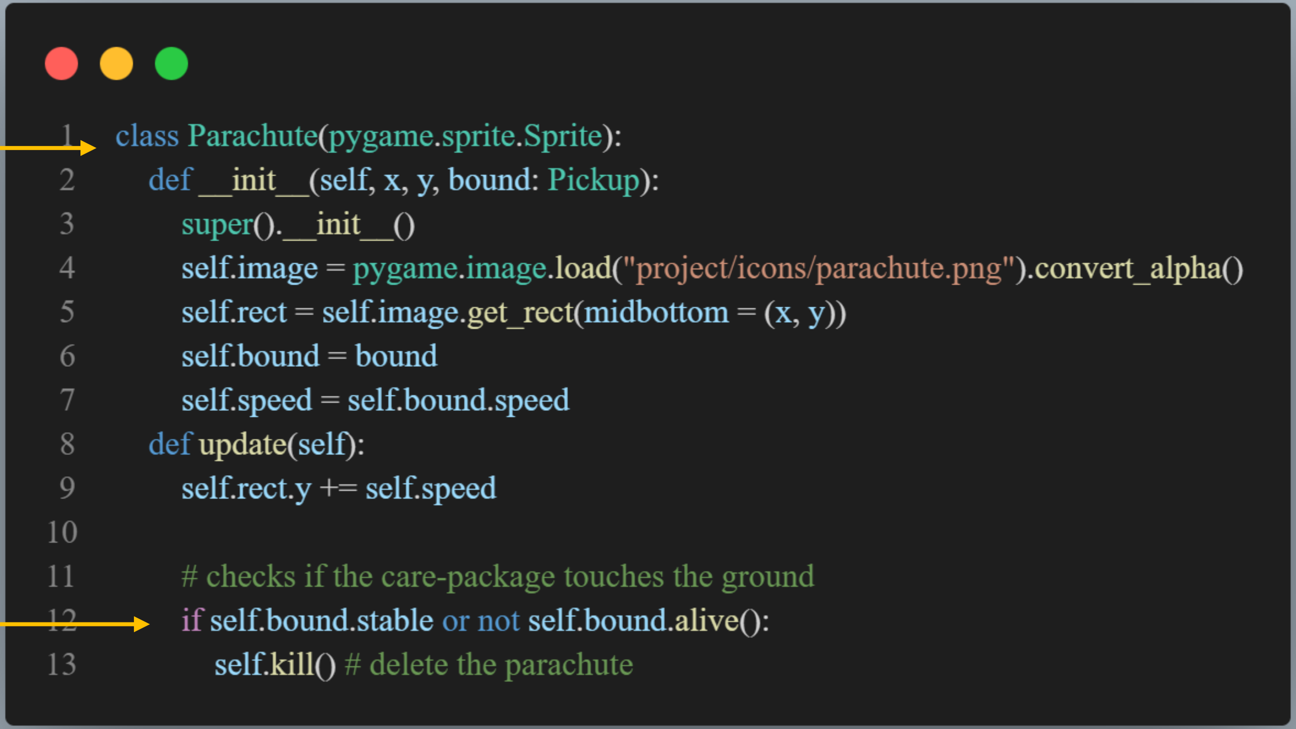
CARE-PACKAGES



CARE-PACKAGES

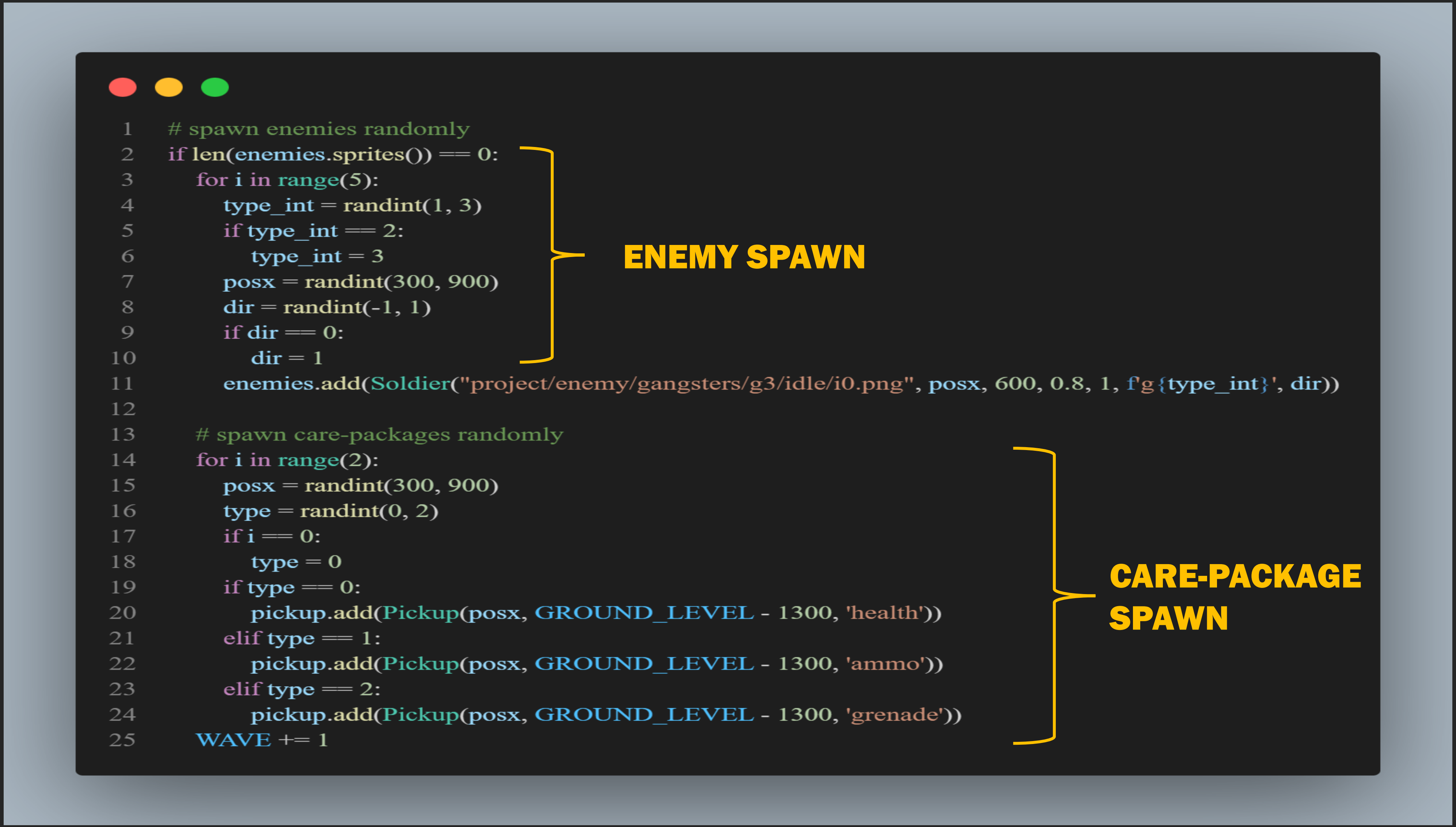


```python
# checks if a parachute is bounded to this care-package
if not self.pr:
    self.pr = True
    parachute.add(Parachute(self.rect.centerx, self.rect.top, self))
self.deletion_cd -= 1

# deletes the care-package after certain time
if self.deletion_cd <= 0:
    self.kill()

# checks if player picks up the care-package
if self.rect.colliderect(player.sprite.hitbox):
    if self.type == 'health' and not player.sprite.health >= 100 and not player.sprite.dead:
        if player.sprite.health + 75 >= 100: # ignore if player is maximum health
            player.sprite.health = 100
        else:
            player.sprite.health += 75
        self.kill()
    elif self.type == 'ammo':
        global reloading
        player.sprite.max_ammo += 30
        self.kill()
        if player.sprite.max_ammo == 0 and player.sprite.ammo == 0:
            player.sprite.state = 3
            reloading = True
    elif self.type == 'grenade':
        player.sprite.grenades += 1
        self.kill()
if self.rect.bottom < GROUND_LEVEL + 20:
    self.rect.y += self.speed
else:
    self.stable = True
```



```python
class Parachute(pygame.sprite.Sprite):
    def __init__(self, x, y, bound: Pickup):
        super().__init__()
        self.image = pygame.image.load("project/icons/parachute.png").convert_alpha()
        self.rect = self.image.get_rect(midbottom = (x, y))
        self.bound = bound
        self.speed = self.bound.speed
    def update(self):
        self.rect.y += self.speed

        # checks if the care-package touches the ground
        if self.bound.stable or not self.bound.alive():
            self.kill() # delete the parachute
```

WAVE DESIGN



```
# spawn enemies randomly
if len(enemies.sprites()) == 0:
    for i in range(5):
        type_int = randint(1, 3)
        if type_int == 2:
            type_int = 3
        posx = randint(300, 900)
        dir = randint(-1, 1)
        if dir == 0:
            dir = 1
        enemies.add(Soldier("project/enemy/gangsters/g3/idle/i0.png", posx, 600, 0.8, 1, f'g{type_int}', dir))

    # spawn care-packages randomly
    for i in range(2):
        posx = randint(300, 900)
        type = randint(0, 2)
        if i == 0:
            type = 0
        if type == 0:
            pickup.add(Pickup(posx, GROUND_LEVEL - 1300, 'health'))
        elif type == 1:
            pickup.add(Pickup(posx, GROUND_LEVEL - 1300, 'ammo'))
        elif type == 2:
            pickup.add(Pickup(posx, GROUND_LEVEL - 1300, 'grenade'))
    WAVE += 1
```

ENEMY SPAWN

CARE-PACKAGE SPAWN

THANK YOU !