

LAB 2

SPACESHIP GAME

Програмирање на видео игри



Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Решение : [github link](#)

Објаснување на функционалности:

Се иницијализира `pygame`, се поставува прозорец со димензии 800x600 и се вчитуваат потребните ресурси (слики, звуци, музика).

```
# Screen dimensions
WIDTH, HEIGHT = 800, 600

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

#Materials
spaceship_img = pygame.image.load('spaceship.png')
asteroid_img = pygame.image.load('asteroid.png')
energy_crystal_img = pygame.image.load('energy_crystal.png')
background_music = 'background_music.wav'
clash_sound = pygame.mixer.Sound('clash_sound.wav')
```

Класа Spaceship

Креира објект за контролирање на вселенското летало. Се управува со стрелките на тастатурата, движејќи се низ екран.

```
class Spaceship(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = spaceship_img
        self.rect = self.image.get_rect(center=(WIDTH // 2, HEIGHT // 2))
        self.speed = 5
    def update(self):
        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT] and self.rect.left > 0:
            self.rect.x -= self.speed
        if keys[pygame.K_RIGHT] and self.rect.right < WIDTH:
            self.rect.x += self.speed
        if keys[pygame.K_UP] and self.rect.top > 0:
            self.rect.y -= self.speed
        if keys[pygame.K_DOWN] and self.rect.bottom < HEIGHT:
            self.rect.y += self.speed
```

Класа Asteroid

Креира астероиди кои влегуваат од различни страни на екранот со случајна брзина. Ако излезат надвор од екранот, се отстрануваат.

```
# Asteroid class
class Asteroid(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = asteroid_img
        side = random.choice(['left', 'right', 'top', 'bottom'])
        if side == 'left':
            self.rect = self.image.get_rect(center=(random.randint(-50, 0),
random.randint(0, HEIGHT)))
        elif side == 'right':
            self.rect = self.image.get_rect(center=(random.randint(WIDTH,
WIDTH + 50), random.randint(0, HEIGHT)))
        elif side == 'top':
            self.rect = self.image.get_rect(center=(random.randint(0, WIDTH),
random.randint(-50, 0)))
        else:
            self.rect = self.image.get_rect(center=(random.randint(0, WIDTH),
random.randint(HEIGHT, HEIGHT + 50)))
        self.speed_x = random.choice([-1, 1]) * random.uniform(0.5, 2)
        self.speed_y = random.choice([-1, 1]) * random.uniform(0.5, 2)
```

Класа EnergyCrystal

исто така влегуваат од различни страни и се движат случајно. Играчот собира кристали за да го зголеми својот резултат.

```
# EnergyCrystal class
class EnergyCrystal(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = energy_crystal_img
        side = random.choice(['left', 'right', 'top', 'bottom'])
        if side == 'left':
            self.rect = self.image.get_rect(center=(random.randint(-50, 0),
random.randint(0, HEIGHT)))
        elif side == 'right':
            self.rect = self.image.get_rect(center=(random.randint(WIDTH, WIDTH
+ 50), random.randint(0, HEIGHT)))
        elif side == 'top':
            self.rect = self.image.get_rect(center=(random.randint(0, WIDTH),
random.randint(-50, 0)))
        else:
            self.rect = self.image.get_rect(center=(random.randint(0, WIDTH),
```

```
random.randint(HEIGHT, HEIGHT + 50)))
    self.speed_x = random.choice([-1, 1]) * random.uniform(0.5, 2)
    self.speed_y = random.choice([-1, 1]) * random.uniform(0.5, 2)
```

Класа Grid

Поставува позадина со мрежа што симулира движење. Мрежата се ажурира врз основа на движењето на играчот.

```
# Background Grid
class Grid:
    def __init__(self):
        self.offset_x = 0
        self.offset_y = 0
        self.cell_size = 40

    def update(self, dx, dy):
        self.offset_x = (self.offset_x + dx) % self.cell_size
        self.offset_y = (self.offset_y + dy) % self.cell_size

    def draw(self, surface):
        for x in range(-self.cell_size, WIDTH, self.cell_size):
            for y in range(-self.cell_size, HEIGHT, self.cell_size):
                rect = pygame.Rect(x + self.offset_x, y + self.offset_y,
self.cell_size, self.cell_size)
                pygame.draw.rect(surface, WHITE, rect, 1)
```

Главна логика

- Во секоја итерација на играта се проверуваат:
 - Движење на играчот, астероидите и кристалите.
 - Судири меѓу играчот и објектите.
- При судир со астероид, се активира звук и играта завршува.
- Играчот може да рестартира играта со копчето **R**.

```
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
```

```

if not game_over:
    # Spawn asteroids
    if random.randint(1, 30) == 1:
        asteroid = Asteroid()
        all_sprites.add(asteroid)
        asteroids.add(asteroid)

    # Spawn energy crystals
    if random.randint(1, 60) == 1:
        energy_crystal = EnergyCrystal()
        all_sprites.add(energy_crystal)
        energy_crystals.add(energy_crystal)

    # Update
    dx = dy = 0
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        dx = player.speed
    elif keys[pygame.K_RIGHT]:
        dx = -player.speed
    if keys[pygame.K_UP]:
        dy = player.speed
    elif keys[pygame.K_DOWN]:
        dy = -player.speed

    grid.update(dx, dy)
    all_sprites.update()

    # Move asteroids and crystals with the grid
    for sprite in asteroids.sprites() + energy_crystals.sprites():
        sprite.rect.x += dx
        sprite.rect.y += dy

    # Check collisions
    if pygame.sprite.spritecollideany(player, asteroids):
        clash_sound.play()
        game_over = True

    collected_crystals = pygame.sprite.spritecollide(player,
energy_crystals, True)
    score += len(collected_crystals)

```