Проектна задача по предметот
Програмирање на видео игри

Вовед
Играта е симулација на билијард. Технологии кои се користени се Python со библиотеките Pygame и Pymunk. Главната цел на играта е да се стигне до последниот/5 левел.
Во секој левел има одредени топки кои треба да се погодат. Доколку погодите друга топка различна од понудените за тој левел се намалуваат животите што ја прави играта поинтересна и потешка. За секој левел играчот има 3 животи. Секој левел се намалуваат дадените топки.
Линк до кодот

Карактеристки за имплементација на играта:

- Еден од поголемите предизвици со кој се соочивме беше Pymunk библиотеката на Python со која е пореалистична играта во прашање на физиката. Тоа се познава при судири на топките и одбивање од рамките на масата.
- Различни нивоа : секое ниво има различен број достапни топки кои треба да се погодат.
- Приказ на информации: тековно ниво, број на животи, достапни топки.
- Анимација на стап: стапот ротира според движењето на глувчето и го покажува насоката на удар.
- Динамично полнење на ударот: силата на ударот расте и опаѓа со текот на времето додека играчот држи притиснато.

Техничка имплементација:

-Генерирање на топките:
Функцијата create_ball() создава објекти за топките со соодветни физички својства.

```
def create_ball(radius, pos, ball_number):
    body = pymunk.Body()
    body.position = pos
    shape = pymunk.Circle(body, radius)
    shape.mass = 5
    shape.elasticity = 0.8
    shape.ball_number = ball_number  # Assign a number to the ball
    pivot = pymunk.PivotJoint(static_body, body, *args: (0, 0), (0, 0))
    pivot.max_bias = 0
    pivot.max_force = 1000
    space.add( *objs: body, shape, pivot)
    return shape
```

-Поставување на почетна конфигурација

Функцијата setup_level(level) создава топки и ги позиционира во стандардна билијардска форма.

```
def setup_level(level):
    global balls, available_balls, lives, potted_balls

    if 'balls' in globals():
        for ball in balls[:-1]:
            for constraint in ball.body.constraints:
                space.remove(constraint)
            space.remove( *objs: ball.body, ball)
        balls = [balls[-1]]
    else:
        balls = []
        pos = (888, SCREEN_HEIGHT / 2)
        cue_ball = create_ball(dia / 2, pos, -1)
        balls.append(cue_ball)
```

- Проверка за погодени топки

```
for ball in balls[:-1]:
    for pocket in pockets:
        ball_x_dist = abs(ball.body.position[0] - pocket[0])
        ball_y_dist = abs(ball.body.position[1] - pocket[1])
        ball_dist = math.sqrt((ball_x_dist ** 2) + (ball_y_dist ** 2))
        if ball_dist <= pocket_dia / 2:
            ball_number = ball.ball_number
            if ball_number in available_balls:
                available_balls.remove(ball_number)
                potted_balls.append(ball_number)
            else:
                lives -= 1
            space.remove(ball.body)
            balls.remove(ball)
            break
```

- *Контрола на стапот*

Класата Cue ја обработува ротацијата и приказот на стапот.

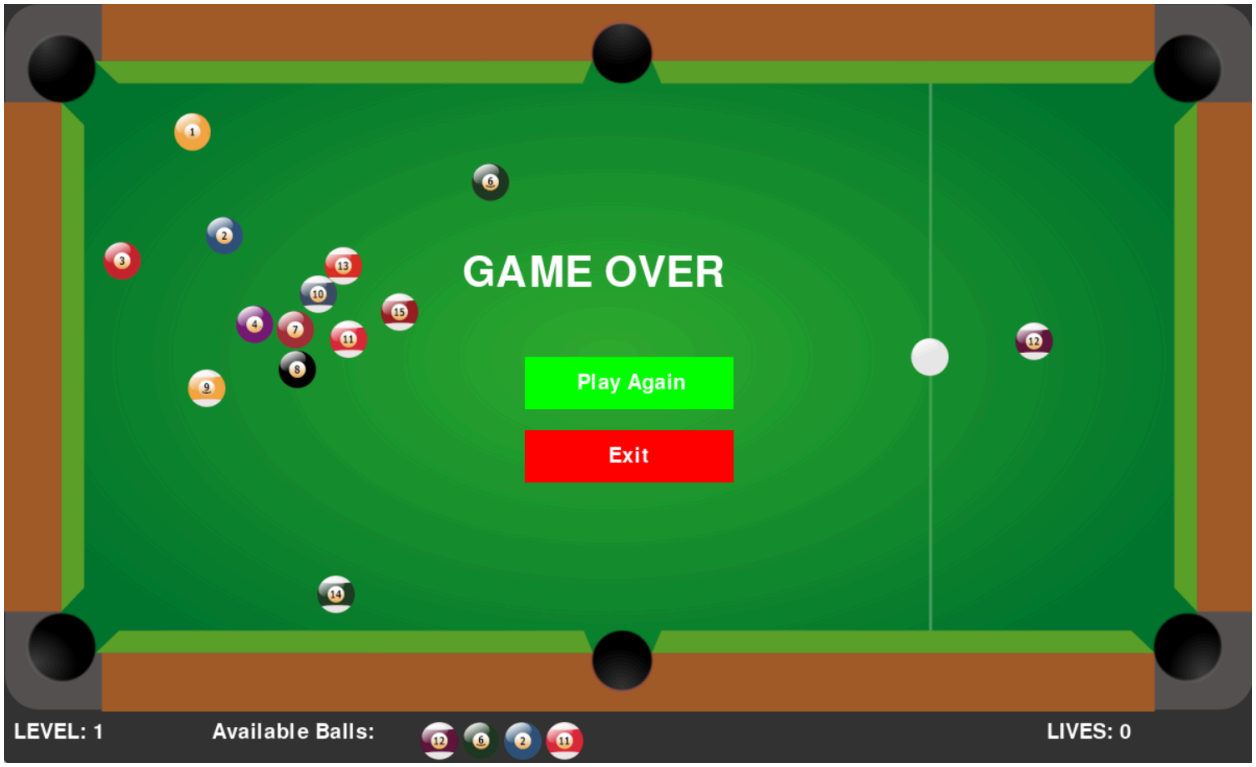
```
class Cue():
    def __init__(self, pos):
        self.original_image = cue_image
        self.angle = 0
        self.image = pygame.transform.rotate(self.original_image, self.angle)
        self.rect = self.image.get_rect()
        self.rect.center = pos

    1 usage
    def update(self, angle):
        self.angle = angle

    1 usage
    def draw(self, surface):
        self.image = pygame.transform.rotate(self.original_image, self.angle)
        surface.blit(self.image,
                     (self.rect.centerx - self.image.get_width() / 2,
                      self.rect.centery - self.image.get_height() / 2)
                     )
```

-Удар и импулс

Кога играчот го отпушта кликот на глувчето, топката добива импулс.

```
if powering_up and game_running:
    force += 100 * force_direction
    if force >= max_force or force <= 0:
        force_direction *= -1
    for b in range(math.ceil(force / 2000)):
        screen.blit(power_bar,
                    dest: (balls[-1].body.position[0] - 30 + (b * 15),
                    balls[-1].body.position[1] + 30))
elif not powering_up and taking_shot:
    x_impulse = math.cos(math.radians(cue_angle))
    y_impulse = math.sin(math.radians(cue_angle))
    balls[-1].body.apply_impulse_at_local_point( impulse: (force * -x_impulse, force * y_impulse),  point: (0, 0))
    force = 0
    force_direction = 1
```

Неколку слики од играта:


GAME OVER
Play Again
Exit
LEVEL: 1
Available Balls:
LIVES: 0


LEVEL: 2
Available Balls:
LIVES: 3

Референци

- Pygame: https://www.pygame.org/docs/
- Pymunk: http://www.pymunk.org/en/latest/
- Python: https://www.python.org/

Изработиле:
Стефан Вељаноски 213098
Маја Манева 223167