



קריית החינוך
פארק המדע
בית לערכים
למציאות ולחדשנות

PyGame Sprite

גלעד מרקמן



Sprite

- Sprite הם אובייקטים של פייתון המייצגים עצמים גרפיים במשחק, באמצעות תכנות מונחה עצמים.
- הפתרון של PyGame כולל שתי מחלקות המשתלבות ביניהן:
 - המחלקה Sprite לבניית עצם במשחק.
 - המחלקה Sprite.Group באמצעותה ננהל בקלות ספרייטים.
- Sprite.Group יכול גם לכלול sprite אחד.



[קישור ל GitHub](#)

[רשימת הפעולות של Sprite.Group](#)

המחלקה Sprite

- בניית Sprite נעשית על ידי יצירת מחלקה חדשה היורשת מ `pygame.sprite.Sprite`.
- המאפיינים של sprite כוללים:
 - `Self.image` – התמונה או המשטח המוצג על המסך.
 - `Self.rect` – המקבל את הפרמטרים של התמונה.
 - `Self.mask` – לאיתור מדוייק של התנגשויות.
 - מאפיינים נוספים.
- המחלקה תממש את הפעולות הבאות:
 - `Init` – בנאי המחלקה עם קריאה למחלקת האב `super().__init__()`.
 - `Update` – במסגרתה מעדכנים את הפרמטרים של התמונה (כגון מיקום על המסך).
 - `Draw` – הדפסת הספרייט על המסך.

דוגמה: class SpaceShip

```
import pygame
from Graphics import *

class SpaceShip (pygame.sprite.Sprite):
    def __init__(self, image) -> None:
        super().__init__()
        self.image = image
        self.rect = self.image.get_rect()
        self.radius = 30
        self.mask = pygame.mask.from_surface(self.image)

    def move (self, dx, dy):
        x, y = self.rect.midbottom
        x = (x + dx) % WIDTH
        y = (y + dy) % HEIGHT
        self.rect.midbottom = x, y

    def draw (self, surface):
        surface.blit(self.image, self.rect)

    def update(self, dx, dy): # for sprite Group
        self.move(dx,dy)
```

• מאפיינים image, rect
חיוניים.

• מאפיינים mask, radius
אפשריים
התנגשויות.
עבור

• הפעולה update – לצורך
.sprite.Group

שימוש ב sprite

```
import pygame
from Graphics import *
from SpaceShip import SpaceShip

pygame.init()

screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Reversi')
clock = pygame.time.Clock()
screen.fill(LIGHTGRAY)

space_ship_img = pygame.image.load("img/spacecraft.png")
space_ship_img = pygame.transform.scale(space_ship_img, (60, 60))
space_ship = SpaceShip(space_ship_img)
space_ship.rect.midbottom = (400,599)
```

```
run = True
while (run):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False

    screen.fill(LIGHTGRAY)

    space_ship.move(3, 0)
    space_ship.draw(screen)
    pygame.display.update()
    clock.tick(FPS)
```

הדגמה – שליטה ב sprite באמצעות המקלדת

```
import pygame
from Graphics import *
from SpaceShip import SpaceShip

pygame.init()

screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Reversi')
clock = pygame.time.Clock()
screen.fill(LIGHTGRAY)

speed_x, speed_y = 0, 0

space_ship_img = pygame.image.load("img/spacecraft.png")
space_ship_img = pygame.transform.scale(space_ship_img, (60, 60))
space_ship = SpaceShip(space_ship_img)
space_ship.rect.midbottom = (400,599)
```

```
run = True
while (run):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                speed_x += -1
            if event.key == pygame.K_RIGHT:
                speed_x += 1
            if event.key == pygame.K_UP:
                speed_y += -1
            if event.key == pygame.K_DOWN:
                speed_y += 1
            if event.key == pygame.K_SPACE:
                speed_x, speed_y = 0, 0

    screen.fill(LIGHTGRAY)

    space_ship.move(speed_x, speed_y)
    space_ship.draw(screen)

    pygame.display.update()
    clock.tick(FPS)
```

שימוש במספר אובייקטים המחלקה Sun

```
import pygame
from Graphics import *

class Sun (pygame.sprite.Sprite):
    def __init__(self, img) -> None:
        super().__init__()
        self.image = img
        self.image = pygame.transform.scale(self.image, (60, 60))
        self.rect = self.image.get_rect(center = (100, 100))
        self.radius = 20
        self.mask = pygame.mask.from_surface(self.image)

    def draw (self, surface):
        surface.blit(self.image, self.rect)

    def move (self):
        x, y = self.rect.center
        x = (x + 1) % WIDTH
        y = (y + 1) % HEIGHT
        self.rect.center = x, y

    def update(self):
        self.move()
```

הדגמה – שימוש במספר אובייקטים - ב

- החללית נשלטת על ידי המשתמש. השמש נעה אוטומטית בכל פריים.

```
pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Reversi')
clock = pygame.time.Clock()
screen.fill(LIGHTGRAY)

space_ship_img = pygame.image.load("img/spacecraft.png")
space_ship_img = pygame.transform.scale(space_ship_img, (60, 60))
space_ship = SpaceShip(space_ship_img)
space_ship.rect.midbottom = (200,599)

sun_img = pygame.image.load("img/sun.png")
sun = Sun(sun_img)

speed_x, speed_y = 0, 0
```

```
run = True
while (run):
    for event in pygame.event.get():
        if event.type == pygame.QUIT: ...
        if event.type == pygame.KEYDOWN: ...

    screen.fill(LIGHTGRAY)

    space_ship.move(speed_x,speed_y)
    space_ship.draw(screen)

    sun.move()
    sun.draw(screen)

    pygame.display.update()
    clock.tick(FPS)
```


גילוי התנגשויות באמצעות rect

- במשחקים אנחנו נדרשים לגלות ולטפל בהתנגשות בין אובייקטים. pyGame נותן לנו כלים נוחים לאתר התנגשויות.
- הפעולה `pygame.sprite.collide_rect` מחזירה אמת אם קיימת חפיפה בין ה `rect` של שני ספרייטים.
- על מנת שהפעולה תעבוד יש להגדיר מאפיין `self.rect` לספרייט.
- נדגים באמצעות תוכנית בה החללית נעה על יד המשתמש והשמש נעה בצורה אוטומטית. אם קיימת התנגשות ביניהם צבע הרקע מתשנה לכחול בהיר.
- גילוי התנגשויות זה אינו מדוייק שכן ה `rect` כולל גם שוליים ריקים של האובייקט. יחד עם זאת, הפעולה מאוד מהירה.

גילוי התנגשויות באמצעות rect

```
pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Reversi')
clock = pygame.time.Clock()
screen.fill(LIGHTGRAY)

space_ship_img = pygame.image.load("img/spacecraft.png")
space_ship_img = pygame.transform.scale(space_ship_img, (60, 60))
space_ship = SpaceShip(space_ship_img)
space_ship.rect.midbottom = (200, 599)

sun_img = pygame.image.load("img/sun.png")
sun = Sun(sun_img)

speed_x, speed_y = 0, 0
color = LIGHTGRAY
```

```
run = True
while (run):
    for event in pygame.event.get():
        if event.type == pygame.QUIT: ...
        if event.type == pygame.KEYDOWN: ...

    screen.fill(color)

    space_ship.move(speed_x, speed_y)
    space_ship.draw(screen)

    sun.move()
    sun.draw(screen)

    if pygame.sprite.collide_rect(space_ship, sun):
        color = pygame.Color('LightBlue')
    else:
        color = LIGHTGRAY

    pygame.display.update()
    clock.tick(FPS)
```

גילוי התנגשויות באמצעות מעגל

- ניתן לגלות התנגשות באמצעות מעגל דמיוני המצוייר מסביב לאובייקט.
- לצורך כך עלינו להגדיר מאפיין `self.radius` בשני הספרייטים, כאשר מרכז ה `rect` הוא מרכז המעגל.

```
if pygame.sprite.collide_circle(space_ship, sun):  
    color = pygame.Color('LightBlue')  
else:  
    color = LIGHTGRAY
```

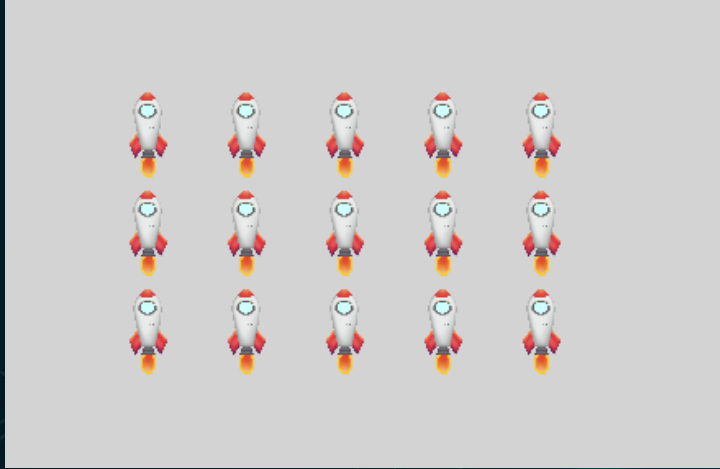
גילוי התנגשויות באמצעות mask

- הדרך המדוייקת ביותר לאתר התנגשות בין שני ספרייטים היא באמצעות mask. הפעולה משווה בין כל פיקסל בספרייטים ומחזירה אמת אם יש שני פיקסלים שאינם שקופים המתנגשים.
- לצורך שימוש בפעולה עלינו להגדיר מאפיין mask בשני הספרייטים:
- הפעולה מופעלת כך:

```
self.mask = pygame.mask.from_surface(self.image)
```

```
if pygame.sprite.collide_mask(space_ship, sun):  
    color = pygame.Color('LightBlue')  
else:  
    color = LIGHTGRAY
```

Sprite Group



- המחלקה Sprite.Group עוזרת לנו להציג ולנהל מספר תמונות.

- נדגים כיצד אנחנו בונים בקלות צי של חלליות הנשלטות על ידי המשתמש, כאשר המשתמש מזיז את כל הקבוצה יחדיו.

- השימוש בקבוצה מחייב:

```
space_ship_Group = pygame.sprite.Group()
```

- הגדרת קבוצה

```
space_ship_Group.add(space_ship)
```

- הוספת ספרייט לקבוצה:

Sprite Group

- אנחנו בונים 15 ספרייטים של חללית באמצעות לולאה. לכל אחד מהם קובעים מיקום התחלתי שונה ומוסיפים אותו לקבוצה.
- הפעולה `group.update` – מפעילה את ה `update` בכל ספרייט בקבוצה.
- הפעולה `group.draw` – מדפיסה את כל הספרייטים (ללא ה `draw` של הספרייט).

```
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Sprites')
clock = pygame.time.Clock()
screen.fill(LIGHTGRAY)

speed_x, speed_y = 0, 0

space_ship_img = pygame.image.load("img/spacecraft.png")
space_ship_img = pygame.transform.scale(space_ship_img, (60, 60))
space_ship_Group = pygame.sprite.Group()

for i in range(5):
    for j in range(3):
        space_ship = SpaceShip(space_ship_img)
        space_ship.rect.midbottom = (60*i, 599-j*60)
        space_ship_Group.add(space_ship)
```

```
run = True
while (run):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.KEYDOWN: ...

    screen.fill(LIGHTGRAY)

    space_ship_Group.update(speed_x, speed_y)
    space_ship_Group.draw(screen)
    pygame.display.update()
    clock.tick(FPS)
```

זיהוי התנגשויות בין קבוצות

- ה `sprite.Group` מאפשר לנו לזהות בקלות התנגשויות בין ספרייטים מקבוצות שונות.
- נגדים התנגשות בין החלליות (קבוצה של ספרייטים) לבין השמש שהינה `single.Group` של ספרייט אחד.
- איתור ההתנגשות נעשה באמצעות ה `rect` של הספרייטים.
- ניתן להעביר לפעולה גם הוראה ל"בטל" ספרייט בקבוצה מסויימת שהשתתף בהתנגשות. במקרה זה הספרייט מבוטל ונמחק מהקבוצה.

```
pygame.sprite.groupcollide(sun_Group, space_ship_Group, False, True)
```

זיהוי התנגשות בין קבוצות באמצעות mask

- ניתן לבצע זיהוי של התנגשויות לכל פיקסל של התמונה באמצעות `.mask`.
- לצורך כך יש להגדיר לכל `sprite` שבקבוצה את המאפשיין `.self.mask`.
- גם במסגרת פעולה זו ניתן לקבוע האם הספרייט המתנגש "יחוסל".
- כמו כן, הפעולה מחזירה את הספרייטים שהשתתפו בהתנגשות:

```
colisions = pygame.sprite.groupcollide(sun_Group, space_ship_Group, False, True, pygame.sprite.collide_mask )
if colisions:
    for sprite in colisions:
        print(sprite.rect.center)
```


המשחק space invaders

- בשיעורים הבאים נדגים כיצד לבנות משחק space invaders באמצעות pyGame.

- המשחק יבנה בהתאם לעקרונות סביבה-סוכן, כבסיס לבניית סוכנים חכמים שילמדו כיצד לשחק במשחק.