



קריית החינוך  
פארק המדע  
בית לערכים  
למציאות ולחדשנות

# PyGame

גלעד מרקמן



# ספריית PyGame

- ספריית PyGame היא מנוע משחק חינוכי בשני מימדים.
- הספרייה מיועדת לפיתוח משחקים פשוטים. משחקים מודרניים נבנים עם מנועי משחק מתוחכמים יותר כמו Unity.
- הספרייה מתאימה במיוחד ללימוד תכנות ולמשחקי לוח פשוטים.



# התקנת PyGame

• התקנת הספרייה היא פשוטה מאוד באמצעות פקודת pip בטרמינל:

```
pip install pygame
```

בתחילת התוכנית יש ליבא את הספרייה באמצעות הפקודה:

```
Import pygame
```

[PyGame tutorial](#)



# התוכנית הראשית

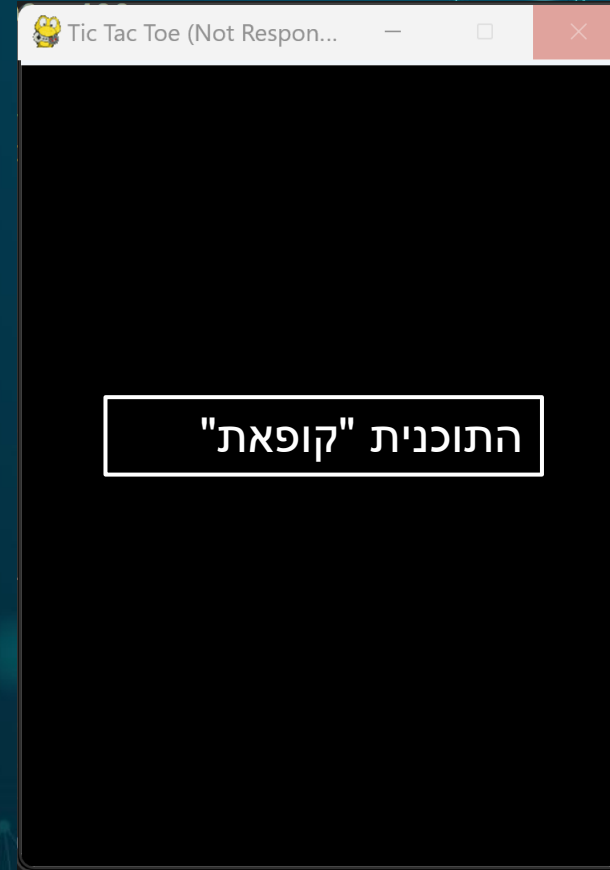
```
1 import pygame
2
3 pygame.init()
4
5 WIDTH, HEIGHT = 300, 400
6
7 screen = pygame.display.set_mode((WIDTH, HEIGHT))
8 pygame.display.set_caption('Tic Tac Toe')
9
10 def main ():
11     run = True
12
13     while (run):
14         pygame.display.update()
15
16
17 if __name__ == '__main__':
18     main()
19
```

- כל תוכנית יש להתחיל ב `pygame.init()` אחרת הספרייה לא תעבוד.
- התוכנית הראשית מורכבת משני חלקים עיקריים:
  - משטח ראשי `screen`.
  - לולאה המעדכנת את המסך.



# הלולאה הראשית

```
1 import pygame
2
3 pygame.init()
4
5 WIDTH, HEIGHT = 300, 400
6
7 screen = pygame.display.set_mode((WIDTH, HEIGHT))
8 pygame.display.set_caption('Tic Tac Toe')
9
10 def main ():
11     run = True
12
13     while (run):
14         pygame.display.update()
15
16
17 if __name__ == '__main__':
18     main()
19
```



# אירועים - events

- PyGame מחזיקה מחסנית עם אירועים המשמשים כממשק משתמש.
- כל אירוע כגון: לחיצה על עכבר, לחיצה על המקלדת, הזזת עכבר וכד' מתועד במחסנית האירועים.
- הפעולה `pygame.event.get()` מרוקנת את המחסנית ומחזירה רשימה של האירועים במחסנית.
- נעבור בלולאה על הרשימה ונבדוק אילו אירועים נוצרו על ידי המשתמש.
- נוסיף טיפול באירוע של סגירת החלון של המשחק.

```
8 def main ():
9     run = True
10
11     while (run):
12         events = pygame.event.get()
13         for event in events:
14             if event.type == pygame.QUIT:
15                 run = False
16
17         pygame.display.update()
18
```

# שעון ומהירות רענון מסך

```
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Reversi')
clock = pygame.time.Clock()
FPS = 60

def main ():
    run = True

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

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

- במשחקים דינאמיים מהירות המשחק תלויה במהירות המחשב.
- אנחנו מעוניינים שמהירות המשחק תהיה קבועה ללא קשר לכוח החישוב של המחשב עליו אנחנו מריצים את התוכנה.
- לצורך כך, אנחנו מגדירים שעון אשר מבצע רענון של מסך המחשב במהירות קבועה. למעשה אנחנו מגבילים את מהירות הלולאה הראשית המעדכנת את המסך.
- FPS – Frame per second

# שכבות וצבעים – surface & colors

- PyGame יוצר את מסך המשחק באמצעות שכבות המודפסות אחת על גבי השניה.
- כל רכיב בתצוגה נקראת שכבה (בין אם צורה, שחקן, משטח וכד').
- המשטח הראשי הוא: `pygame.display.screen`.
- ניתן להדפיס משטח אחד על גבי משטח אחר באמצעות `surface.blit()`.

```
5 screen = pygame.display.set_mode((WIDTH, HEIGHT))
6 pygame.display.set_caption('Reversi')
7 clock = pygame.time.Clock()
8 header_surf = pygame.Surface((H_WIDTH, H_HEIGHT))
9 main_surf = pygame.Surface((M_WIDTH, M_HEIGHT))
10 header_surf.fill(BLUE)
11 main_surf.fill(LIGHTGRAY)
12
13 screen.blit(header_surf, (0,0))
14 screen.blit(main_surf, (0,100))
```

Blit = Block transfer •

צבעים: RGB •

```
BLUE = (0, 0, 255)
```

```
LIGHTGRAY = (211,211,211)
```



# הדפסת צורות גאומטריות

- הדפסת צורות על גבי משטח נעשית באמצעות הפונקציה `pygame.draw`.

```
pygame.draw.line(surface=main_surf, color=BLACK, start_pos=(10,10), end_pos=(100,100), width=5)  
pygame.draw.circle(surface=main_surf, color=GREEN, center= (50,50), radius=20, width=2)  
pygame.draw.circle(surface=header_surf, color=RED, center= (150,50), radius=30, width=0)
```

- ניתן להדפיס צורות מלאות או ריקות, בצבעים שונים, ובגדלים שונים.

- לרשימה מלאה של כל הצורות ראו:

<https://www.pygame.org/docs/ref/draw.html>

# הדפסת תמונות

- ניתן להציג על משטח תמונות, ולקבוע את הגודל שלהם, והמיקום שלהם.
- `pygame.image.load()` – טוען תמונה לזכרון המחשב.
- `pygame.transform.scale()` – משנה את מידות התמונה.
- באמצעות הפעולה `blit` ניתן להציג את התמונה על כל משטח.

```
x_img = pygame.image.load("img/x_img.png")  
x_img = pygame.transform.scale(x_img, (40, 40))  
main_surf.blit(x_img, (50, 200))
```

# אינטראקציה עם המשתמש

- ניתן לבצע אינטראקציה עם המשתמש באמצעות האירועים – `events`.
- דוגמה לקבלת קלט מהמשתמש באמצעות העכבר והמקלדת.
- שימו לב שהדפסת משטח מאוחר עלולה להסתיר את המשטחים הקודמים.

```
while (run):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            x,y = pygame.mouse.get_pos()
            pos = x - 20, y - 100 - 20
            main_surf.blit(x_img,pos)
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                header_surf.fill(GREEN)
            if event.key == pygame.K_RIGHT:
                header_surf.fill(CADETBBLUE1)
```

# אנימציה

- אנימציה במחשב (כמו בסרט) נעשית באמצעות הדפסות חוזרות ונשנות של צורות במקומות שונים ובמהירות.
- בדרך זו נוצר הרושם של תנועת הצורה.
- לפני כל הדפסה של הצורה הנעה יש למחוק את המשטח ואת הדפסת הצורה הקודמת.

```
while (run):  
    for event in pygame.event.get(): ...  
  
    header_surf.fill(BLUE)  
    header_surf.blit(x_img, (x1,y1))  
    x1 = (x1 + 2) % 300  
    screen.blit(header_surf, (0,0))  
    screen.blit(main_surf, (0,100))  
    pygame.display.update()  
    clock.tick(FPS)
```