

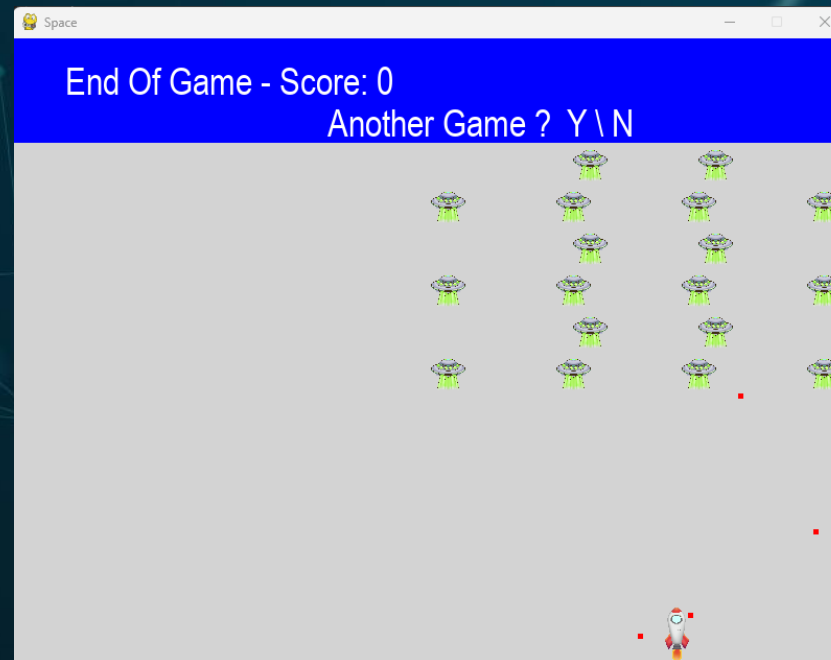


קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

בניית משחק

Space Invaders

גלעד מרקמן



[קישור ל GitHub](#)

מבנה התוכנית

• Action: מספר בין 0-3:

- 0 – החללית לא עושה דבר - נשארת במקום.
- 1 – החללית נעה שמאלה – במהירות $SPACESHIP_SPEED = 5$.
- 2 – החללית נעה ימינה;
- 3 – החללית מבצע ירי צרור מירבי של $SPACE_SHIP_BURST=3$.

• State: מערך של 88 פרמטרים, הכוללים:

- 0-53: מיקום 18 חלליות האויב. כל חללית 3 ערכים: $x, y, speed_x$. חללית הרוגה $0,0,0$.
- 54: מהירות אנכית של חללית האויב $spped_y$.
- 55-74: מיקום 10 טילי האויב. כל טיל 2 ערכים: x, y .
- 75: מהירות טיל אויב על ציר y . $ENEMY_BULLET_SPEED=5$.
- 76-78: מיקום חללית השחקן, ומהירות החללית. $SPACESHIP_SPEED = 5$.
- 79-84: מיקום טילי החללית. כל טיל 2 ערכים x, y .
- 85: מהירות טילי החללית $SPACESHIP_BULLET_SPEED = 15$.
- 86: תחמושת שנותרה לחללית (מקסימום בתחילת שלב $MAX_AMMUNITION = 80$).
- 87: ה level של המשחק.

מבנה התוכנית - המשך

- הסביבה – Environment כוללת את המאפיינים הבאים:
 - Bullets_Group – קבוצת טילי החללית.
 - Spaceship_group – קבוצה של חללית השחקן (קבוצה יחידה).
 - Enemy_bullets_Group – קבוצה של טילי האוייבים.
 - Enemy_Group – קבוצת חלליות האוייב.
 - Ground_Group – קבוצה של הקרקע. הספרייט לא נראה ונעשה בו שימוש כדי לגלות נחיתה של חלליות האוייב.
 - Score – ניקוד של השחקן.
 - Level – רמת המשחק.

[קישור ל GitHub](#)

מחלקות התוכנית

- State: מערך הנוצר על ידי הסביבה (`environment.state()`).
- Game – לולאת המשחק הראשית.
- Human_Agent – סוכן אדם הקורא את המקלדת ומחזיר את הפעולה שבחר האדם.
- Environment – הסביבה. המחלקה שמנהלת את המשחק.
- SpaceShip – מחלקת ספרייט של החללית
- Enemy – מחלקת ספרייט של חללית אויב.
- Bullet – מחלקת ספרייט של טיל. מתאים לטיל חללית וטיל אויב.
- Ground – מחלקת ספרייט של אדמה. הספרייט לא נראה ונוצר כדי לאתר נחיתה של חלליות האויב בקצה התחתון של המסך.

לולאת המשחק הראשית

```
def main ():  
  
    pygame.init()  
  
    screen = pygame.display.set_mode((WIDTH, HEIGHT))  
    pygame.display.set_caption('Space')  
    clock = pygame.time.Clock()  
  
    header_surf = pygame.Surface((WIDTH, 100))  
    main_surf = pygame.Surface((WIDTH, HEIGHT - 100))  
    header_surf.fill(BLUE)  
    main_surf.fill(LIGHTGRAY)  
  
    env = Environment(surface=main_surf)  
  
    screen.blit(header_surf, (0,0))  
    screen.blit(main_surf, (0,100))  
  
    player = Human_Agent()  
    # player = DQN_Agent(parametes_path=None, train=False)
```

```
# Main Loop  
run = True  
while (run):  
    main_surf.fill(LIGHTGRAY)  
    header_surf.fill(BLUE)  
  
    events = pygame.event.get()  
    for event in events:  
        if event.type == pygame.QUIT:  
            run = False  
  
    action = player.get_Action(events=events, state=env.state())  
    reward, done = env.move(action=action)  
    if done:  
        write (header_surf, "End Of Game - Score: " + str (env.score))  
        write (header_surf, "Another Game ? Y / N", pos=(300, 60))  
        screen.blit(header_surf, (0,0))  
        pygame.display.update()  
        if another_game():  
            env.restart()  
        else:  
            break  
  
    write(header_surf, "Score: " + str(env.score) ,(200, 60))  
    write(header_surf, "Ammunition: " + str(env.spaceship.ammunition), (400, 60))  
    write(header_surf, "Level: " + str(env.level), (200, 20))  
    screen.blit(header_surf, (0,0))  
    screen.blit(main_surf, (0,100))  
    pygame.display.update()  
    clock.tick(FPS)
```

סוכן אדם – Human Agent

```
import pygame

class Human_Agent:
    def __init__(self):
        self.action = 0

    def get_Action (self, events=None, state= None):
        for event in events:
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    self.action = 1
                if event.key == pygame.K_RIGHT:
                    self.action = 2
                if event.key == pygame.K_SPACE:
                    self.action = 3
            if event.type == pygame.KEYUP:
                self.action = 0
        if self.action == 3: # ירי בבודדת
            self.action = 0
        return 3
    return self.action
```

• סוכן אדם מקבל את רשימת האירועים, בודק אם האירוע מתאים ללחיצת מקש של המשתמש, ומחזיר את הפעולה שבחר המשתמש.

• הפעולה Action היא:

- 0 – נשאר במקום.
- 1 – תנועה של החללית שמאלה.
- 2 – תנועה של החללית ימינה.
- 3 – רווח ירי של החללית.

המחלקה SpaceShip

```
class SpaceShip (pygame.sprite.Sprite):  
    def __init__(self, img_Url, pos, bullets_Group) -> None:  
        super().__init__()  
        self.image = pygame.image.load(img_Url)  
        self.image = pygame.transform.scale(self.image, (60, 60))  
        self.rect = self.image.get_rect(midbottom = pos)  
        self.mask = pygame.mask.from_surface(self.image)  
        self.bullets_Group = bullets_Group  
        self.speed_x = SPACESHIP_SPEED  
        self.ammunition = MAX_AMMUNITION  
        self.burst = SPACE_SHIP_BURST
```

```
> def update(self) -> None: ...
```

```
> def move_right (self): ...
```

```
> def move_left (self): ...
```

```
> def shoot (self): ...
```

```
> def action (self, action): ...
```

- מחלקת ספרייט של החללית.
- הפעולה action – מקבלת מספר בין 0-3, ומבצעת את הפעולות המתאימות.

```
def action (self, action):  
    if action == 1:  
        self.move_left()  
    elif action == 2:  
        self.move_right()  
    elif action == 3:  
        self.shoot()
```

המחלקה SpaceShip

- הפעולה shot מבצעת יריה של טיל היוצא מן החללית כלפי מעלה.
- החללית רשאית לירות אם נותרה לה תחמושת ולא יותר מ- 3 טילים בבת אחת.
- יריית הטיל נעשית באמצעות הוספת Bullet ל bullets_Group שמיקומו ההתחלתי בחלק העליון האמצעי של החללית, ומהירותו כלפי מעלה.

```
def shoot (self):  
    if self.ammunition > 0 and len(self.bullets_Group) < self.burst:  
        self.bullets_Group.add(Bullet(self.rect.midtop,speed_y= -SPACESHIP_BULLET_SPEED))  
        self.ammunition -= 1
```


המחלקה Enemy

```
class Enemy (pygame.sprite.Sprite):  
    shoots_factor = ENEMY_SHOOTS_FACTOR  
    speed_add = 0  
    speed_y = 40  
    def __init__(self, img, pos, Enemy_bullets_Group) -> None:  
        super().__init__()  
        self.image = img  
        self.rect = self.image.get_rect(topleft = pos)  
        self.mask = pygame.mask.from_surface(self.image)  
        self.speed_x = ENEMY_START_SPEED  
        self.speed_dir = 1  
        self.Enemy_bullets_Group = Enemy_bullets_Group  
  
    def update(self) -> None:  
        self.move()  
        self.shoot()  
  
    def move (self): ...  
  
    def shoot (self): ...
```

• חללית אויב מוגדרת במחלקה.

• לאויב יש משתנים סטטיים
המשותפים לכל חלליות
האויב, על מנת שינועו ביחד.

• בכל עדכון של קבוצת חלליות
האויב מתבצעות הפעולות:
.move, shoot

המחלקה Enemy - המשך

- הפעולה shoot היא רנדומלית ומתבצעת רק ב 0.2% מהמקרים, ואם מספר טילי האוייב קטן צ MAX_ENEMY_BULLETS.
- התנועה בציר Y רק אם מגיעים לקצה המסך, ואז משנים כיוון.

```
def move (self):  
    if self.rect.right > WIDTH:  
        self.rect.y += Enemy.speed_y  
        self.speed_dir = -self.speed_dir  
    if self.rect.left < 0:  
        self.rect.y += Enemy.speed_y  
        self.speed_dir = -self.speed_dir  
    self.rect.x += (self.speed_x + Enemy.speed_add) * self.speed_dir  
  
def shoot (self):  
    if random.random() < Enemy.shoots_factor/1000 and len(self.Enemy_bullets_Group) < MAX_ENEMY_BULLETS:  
        self.Enemy_bullets_Group.add(Bullet(self.rect.midbottom, speed_y=ENEMY_BULLET_SPEED))
```

המחלקה Bullet

```
class Bullet (pygame.sprite.Sprite):  
    def __init__(self, pos, speed_x=0, speed_y= -10) -> None:  
        super().__init__()  
        self.image = pygame.Surface((5,5))  
        self.image.fill(RED)  
        self.rect = self.image.get_rect(midbottom = pos)  
        self.mask = pygame.mask.from_surface(self.image)  
        self.speed_x = speed_x  
        self.speed_y = speed_y  
  
    def update(self) -> None:  
        self.move()  
  
    def move (self):  
        self.rect.y += self.speed_y  
        if self.rect.y < 0:  
            self.kill()  
        if self.rect.y > MAIN_SURF_HEIGHT:  
            self.kill()
```

- המחלקה מייצגת טיל הן של חללית והן של אויב. השיוך של הטיל נקבע לפי הקבוצה אליו מוספים אותו.
- הטיל מתקדם בכל update במהירות שלו על ציר y.
- אם הטיל מגיע לקצה מסך הוא מתבטל `.self.kill()`.

Environment

```
class Environment:  
    def __init__(self, surface) -> None: ...  
  
    def make_enemy_group (self, row=ENEMY_ROWS, col=ENEMY_COLS, space_row = 80, space_col = 120): ...  
  
    def update (self): ...  
  
    def draw (self): ...  
  
    def restart (self, add_speed = 0, add_shoot_factor = 0, new_game = True): ...  
  
    def move (self, action): ...  
  
    def is_end_of_stage (self): ...  
  
    def is_end_of_Game (self): ...  
  
    def hits (self): ...  
  
    def state (self): ...
```

Environment - המשך

- בנאי המחלקה יוצר את כל האובייקטים שמשתתפים במשחק.
- כל האובייקטים מצטרפים לקבוצות לצורך הצגה נוחה במסך.
- בנוסף מוגדרים פרמטרים נוספים: ניקוד, רמה וכד'.

```
def __init__(self, surface) -> None:  
    self.bullets_Group = pygame.sprite.Group()  
    self.spaceship = Spaceship(SPACESHIP_URL, (WIDTH //2, HEIGHT - 100), self.bullets_Group)  
    self.spaceship_Group = pygame.sprite.GroupSingle(self.spaceship)  
    self.enemy_bullets_Group = pygame.sprite.Group()  
    self.enemy_img = pygame.image.load(ENEMY_URL)  
    self.enemy_img = pygame.transform.scale(self.enemy_img, (40, 40))  
    self.enemy_Group = self.make_enemy_group()  
    self.score = 0  
    self.surface = surface  
    self.level = 1  
    self.ground = Ground()  
    self.ground_Group = pygame.sprite.GroupSingle(self.ground)
```

Environment - המשך

- בתחילת משחק ובכל מעבר רמה נוצרות 18 חלליות של האוייב המוכנות לתוך קבוצת חלליות האוייב.
- עדכון של הקבוצה יבצע update לכל אחת מהחלליות בקבוצה, הכולל תנועה וירייה (רנדומלית).

```
def make_enemy_group (self, row=ENEMY_ROWS, col=ENEMY_COLS, space_row = 80, space_col = 120):  
    enemy_Group = pygame.sprite.Group()  
    row , col = 3 , 6  
    for r in range (row):  
        for c in range (col):  
            enemy_Group.add(Enemy(self.enemy_img, (c * space_col, r * space_row, ), self.enemy_bullets_Group))  
    return enemy_Group
```

Environment - המשך

```
def update (self):  
    self.spaceship_Group.update()  
    self.enemy_Group.update()  
    self.bullets_Group.update()  
    self.enemy_bullets_Group.update()  
  
def draw (self):  
    surface = self.surface  
    self.ground_Group.draw(surface)  
    self.spaceship_Group.draw(surface)  
    self.enemy_Group.draw(surface)  
    self.bullets_Group.draw(surface)  
    self.enemy_bullets_Group.draw(surface)
```

- הפעולה update מניעה את המשחק, ומבצעת עדכון של כל הקבוצות. כל עדכון כזה מניע את החלליות והטילים בהתאם למחלקות.

- עדכון של spaceship מיותר ולמעשה לא עושה דבר. התנועה של חללית השחקן נעשית באמצעות הסוכן ולא באופן אוטומטי על ידי המשחק.

- הפעולה draw מציירת את כל הספרייטים בקבוצות על המסך, לאחר שעשינו להם עדכון.

Environment - המשך

```
def move (self, action):  
    reward = 0  
    if action == 1:  
        self.spaceship.move_left()  
    elif action == 2:  
        self.spaceship.move_right()  
    elif action == 3:  
        self.spaceship.shoot ()  
    self.update()  
    self.draw()  
    reward += self.hits()  
    if self.is_end_of_stage():  
        reward += 0  
        self.restart(add_speed=0.5, add_shoot_factor=0.1, new_game=False)  
    self.score += reward  
    done = self.is_end_of_Game()  
    if done:  
        reward -= 3  
    return reward, done
```

- הפעולה move מספר דברים:

- מקבלת action ומפעילה את החללית.
- מבצעת עדכון וציור של כל האובייקטים על המסך.
- בודקת מספר פגיעות בחלליות אויב ונותן נקודה לכל פגיעה.
- בודקת סוף שלב ומבצעת אתחול של המסך בסוף שלב.

- בודקת אם סוף משחק. אתחול סוף משחק נעשה בתוכנית הראשית.

- מחזירה: ניקוד (reward) והאם סוף משחק.

Environment - המשך

- בדיקת סוף שלב – אם אין יוצר חלליות אויב בקבוצה.
- בדיקת סוף משחק – אם האויב נחת (האויב התנגש עם ground_group) או החללית נפגעה מטיל (התנגשות בין קבוצת טילי האויב לחללית).
- Hits – פגיעות בחללית אויבץ התנגשות בין קבוצת טילי החללית לקבוצת חלליות האויב.

```
def is_end_of_stage (self):  
    return len(self.enemy_Group) == 0  
  
def is_end_of_Game (self):  
    enemy_landed = pygame.sprite.spritecollide(self.ground, self.enemy_Group, dokill=True)  
    spaceship_hit = pygame.sprite.spritecollide(self.spaceship, self.enemy_bullets_Group, dokill=True, collided= pygame.sprite.collide_mask)  
    return len(enemy_landed) > 0 or len(spaceship_hit) > 0  
  
def hits (self):  
    collisions = pygame.sprite.groupcollide(self.enemy_Group, self.bullets_Group, True, True, pygame.sprite.collide_mask)  
    return len(collisions)
```

Environment - המשך

```
state_list = []  
# 0 - 53  
index = 0  
for sprite in self.enemy_Group:  
    state_list.append(sprite.rect.centerx)  
    state_list.append(sprite.rect.centery)  
    state_list.append(sprite.speed_x)  
    index += 3  
for i in range(enemy_ships-index):  
    state_list.append(0)  
state_list.append(Enemy.speed_y) # 54  
index = 0  
for sprite in self.enemy_bullets_Group: # 55 - 74  
    state_list.append(sprite.rect.centerx)  
    state_list.append(sprite.rect.centery)  
    index += 2  
for i in range(enemy_bullets-index):  
    state_list.append(0)  
state_list.append(ENEMY_BULLET_SPEED) # 75  
state_list.append(self.spaceship.rect.centerx) # 76  
state_list.append(self.spaceship.rect.centery) # 77  
state_list.append(SPACESHIP_SPEED) # 78  
index = 0  
for sprite in self.bullets_Group: # 79 - 84  
    state_list.append(sprite.rect.centerx)  
    state_list.append(sprite.rect.centery)  
    index += 2  
for i in range(SpaceShip_Bullet_pos_shape-index):  
    state_list.append(0)  
state_list.append(SPACESHIP_BULLET_SPEED) # 85  
state_list.append(self.spaceship.ammunition) # 86  
state_list.append(self.level) # 87  
# state_list.append(self.score) # 88  
return torch.tensor(state_list, dtype=torch.float32)
```

- הפעולה state מחזירה מערך עם ערכים המתארים את המצב של המסך, כגון:
 - מיקום חלליות האוייב וכיוון הנסיעה שלהן.
 - מיקום חללית האם.
 - מיקום טילי אוייב וטילי החלליית שלב המשחק.
- במאמר של קבוצת deep mind על משחקי אטארי השתמשו החוקרים בתמונות של המסך. אנחנו נשתמש במערך נתונים על מנת להקל על הלמידה.

נפגש בהרצאות על בניית סוכן AI למשחק

