

משימת פיתוח - ספריית Arcade - תנועה במשחק



במהלך היחידה האחרונה עברנו מסע מרתוך מפיתוח קוד בסיסי לייצור פרויקט משחק מורכב. התחלנו בכתיבת קוד בקובץ יחיד, ושם התקדמנו לשיטות עבודה של תכנות מודולרי (פיזול הקוד למספר קבצים), המאפשרות ניהול פרויקטים רחבי היקף. בעזרת ספריית Arcade, הענקנו למשחקים שלנו ממשק ייחודי והזכנו את הלוגיקה מהמסוף אל המסר. לצד הפיתוח הטכני, שמננו דגש על 'הצד האדמיניסטרטיבי': ניהול פרויקט נכון ותיעוד קוד (Documentation). המאפשרים לבדוק צוות פוריה והבנה מהירה של הקוד על ידי גורמים חיצוניים.

במפגש האחרון השלכנו את הפהzel עם הוספת מנגנון שליטה בשחקן ומערכת זיהוי התנגשויות (Collision Detection).

עתה, אנחנו מוכנים לשלב הסופי: הפיכת הפרויקט שבו השקענו בארכיטקטורת המפגשים האחורוניים למשחק מובסס לוגיקה בלבד לחוויה משחק מלאה, אינטראקטיבית ומהנה.



מזהה update_on

מקבלת את self ואת delta_time (זמן שעבר מאז העדכון האחרון). עליו לבצע את השלבים הבאים לפי הסדר:

1. בדיקת מצב משחק: בצעו בדיקה – אם דגל ה-game_over פועל (True), הפסיקו את פעולת המתודה מיד (כדי שהשחקן "יקפא").
2. ניהול תנועת השחקן (Pacman):
 - שמרו את המיקום הנוכחי של הפקמן (center_x, center_y) כ"גיבו" בתוך משתנים זמינים.
 - הפעילו את פקודת התנועה של השחקן.
 - בדיקת קירות: בדקו האם נוצרה התנגשות בין השחקן לבין רשימת הקירות (wall_list). במקרה וכך – החזירו את השחקן למקום הקודם ששמרתם (כדי שלא יוכל לעبور דרך הקיר).
3. ניהול תנועת הרוחות:
 - עברו בלולאה על כל רוח שנמצאת בתוך ghost_list. עבור כל רוח:
 - שמרו את המיקום הנוכחי שלה כגיבוי.

- עדכנו את תנועת הרוח (באמצעות `delta_time`).
■ בדיקת קירות לרוח: בדקנו האם הרוח התנגשה בקיר. במידה וכן – החזרו אותה למקום הקודם ובקשו מהרוח לבחור כיוון חדש.
4. מנגן איסוף מטבעות:
- בדקנו האם השחקן התנגש באובייקטים מתוך רשימת המטבעות.
 - עברו כל מטבע שנאנסף: הוסיפו את הערך שלו לניקוד של השחקן והסירו את המטבע מהמסך.
5. מנגן פגעה (התנגשות עם רוח):
- בדקנו האם השחקן התנגש באחת הרוחות.
 - במיידה ונוצרה פגעה:
 - הורידו חיים לשחקן.
 - החזרו את השחקן לנקודת ההתחלה שלו (`start_x, start_y`).
 - אפסו את מהירות התנועה שלו.
 - בדיקת הפוד: אם מספר החיים של השחקן הגיע ל-0 או פחות, עדכנו את דגל `game_over` ל-True.

מתודת `on_key_press`

מקבלת כארוגומנטים את `self`, `key` ו-`modifiers` (כך חותמת המתודה מגיעה בירושה מ-`View`). המתודה מטפלת בלחיצה על מקשי המקלדת. מבצעת את הפעולות הבאות:

1. במצב הפקד - SPACE – מתחילה משחק מחדש (`setup`).
2. בזמן המשחק:
 - a. חץ למעלה ישנה את `y` ל-1.
 - b. חץ למטה ישנה את `y` ל-1.
 - c. חץ שמאלה ישנה את `x` ל-1.
 - d. חץ ימינה ישנה את `x` ל-1.

מתודת `on_key_release`

מקבלת כארוגומנטים את `self`, `key` ו-`modifiers` (כך חותמת המתודה מגיעה בירושה מ-`View`). המתודה מאפסת את התנועה כאשר המשתמש משחרר את המKeySpec. המתודה בודקת את התנאים הבאים:

1. אם המKeySpec שהשתחרר הוא Up/Down - המתודה תאפס את `y_change`.
2. אם המKeySpec שהשתחרר הוא Left/Right - המתודה תאפס את `x_change`.



- המתודה `check_for_collision_with_list()` מחזירה רשימה של כל הספריות מתחום ה-`list`.
- שנמצאים כרגע בתנגשות עם `sprite`.
- מתודת `key_press_on` לא מזינה את השחקן מיד אלא רק קובעת את הכיוון. בפועל התנועה נעשית ב-`update_on`.
- מתודת `key_release_on` היא הסיבה שהדמות לא ממשיכה לנוע ללא הפסקה, אלא רק כאשר אנחנו לוחצים על מקש מסוים.
- הקפידו להמשיך ולתעד את הקוד החדש שאתם כתובים, תיעוד טוב היום הוא הדרך הטובה ביותר ליותר לוודא שתבינו בדיק מה הקוד שלכם עושה גם בעוד שבועיים.



על מנת להריץ את המשחק, יש ליצור תחילה חלון משחק באמצעות המחלקה `window` מספריית `Arcade`. לאחר מכן יש ליצור מופע של מחלקת המשחק (`PacmanGame`) ולבצע את הולך מלא של רכיבי המשחק באמצעות המתודה `setup()`.
בסיום, יש להציג את המספר הראשי של המשחק באמצעות(`show_view()`, וזה להתחליל את לולאת המשחק של `Arcade` באמצעות הקריאה `arcade.run()`. לולאה זו מפעילה ברכז את כל מחזורי המשחק.

הפעולות חייבות להופיע בסדר זה, כיון שככל שלב תלוי בקדמיםלו:

1. יצירת חלון – ממשק גרפי שבו המשחק יוצג.
2. יצירת אובייקט המשחק – שמנוהל את הלוגיקה, המפה והספריות.
3. את הולך המשחק (`setup()`) – בניית חדש של קירות, שחקן, מטבעות ורוחות.
4. הצגת ה-View – מחייב הצבתת המשחק בחולון.
5. הפעלת לולאת המשחק (`arcade.run()`) – מפעילה את המבוקע שמריץ את המשחק בפועל.