

משימת פיתוח - ספריית Arcade - תנועה במשחק

הקדמה

במהלך היחידה האחרונה עברנו מסע מרתק מפיתוח קוד בסיסי ליצירת פרויקט משחקי מורכב. התחלנו בכתיבת קוד בקובץ יחיד, ומשם התקדמנו לשיטות עבודה של תכנות מודולרי (פיצול הקוד למספר קבצים), המאפשרות ניהול פרויקטים רחבי היקף. בעזרת ספריית Arcade, הענקנו למשחקים שלנו ממשק ויזואלי והוצאנו את הלוגיקה מהמסוף אל המסך. לצד הפיתוח הטכני, שמנו דגש על 'הצד האדמיניסטרטיבי': ניהול פרויקט נכון ותיעוד קוד (Documentation), המאפשרים עבודת צוות פורייה והבנה מהירה של הקוד על ידי גורמים חיצוניים. במפגש האחרון השלמנו את הפאזל עם הוספת מנגנוני שליטה בשחקן ומערכת זיהוי התנגשויות (Collision Detection). כעת, אנחנו מוכנים לשלב הסופי: הפיכת הפרויקט שבו השקענו בארבעת המפגשים האחרונים ממשחק מבוסס לוגיקה בלבד לחווית משחק מלאה, אינטראקטיבית ומהנה.

משימה

מתודת on_update

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

1. בדיקת מצב משחק: בצעו בדיקה – אם דגל ה-game_over פעיל (True), הפסיקו את פעולת המתודה מיד (כדי שהמשחק "יקפא").
2. ניהול תנועת השחקן (Pacman):
 - שמרו את המיקום הנוכחי של הפק-מן (center_x ו-center_y) כ"גיבוי" בתוך משתנים זמניים.
 - הפעילו את פקודת התנועה של השחקן.
 - בדיקת קירות: בדקו האם נוצרה התנגשות בין השחקן לבין רשימת הקירות (wall_list). במידה וכן – החזירו את השחקן למיקום הקודם ששמרתם (כדי שלא יוכל לעבור דרך הקיר).
3. ניהול תנועת הרוחות:
 - עברו בלולאה על כל רוח שנמצאת בתוך ה-ghost_list. עבור כל רוח:
 - שמרו את המיקום הנוכחי שלה כגיבוי.

- עדכנו את תנועת הרוח (באמצעות `delta_time`).
- בדיקת קירות לרוח: בדקו האם הרוח התנגשה בקיר. במידה וכן - החזירו אותה למיקום הקודם ובקשו מהרוח לבחור כיוון חדש.
- 4. מנגנון איסוף מטבעות:
 - בדקו האם השחקן התנגש באובייקטים מתוך רשימת המטבעות.
 - עבור כל מטבע שנאסף: הוסיפו את הערך שלו לניקוד של השחקן והסירו את המטבע מהמסך.
- 5. מנגנון פגיעה (התנגשות עם רוח):
 - בדקו האם השחקן התנגש באחת הרוחות.
 - במידה ונוצרה פגיעה:
 - הורידו חיים לשחקן.
 - החזירו את השחקן לנקודת ההתחלה שלו (`start_x, start_y`).
 - אפסו את מהירות התנועה שלו.
 - בדיקת הפסד: אם מספר החיים של השחקן הגיע ל-0 או פחות, עדכנו את דגל ה-`game_over` ל-`True`.

מתודת `on_key_press`

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

1. במצב הפסד - `SPACE` מאתחל משחק מחדש (`setup`).
2. בזמן המשחק:
 - a. חץ למעלה ישנה את `change_y` ל-1.
 - b. חץ למטה ישנה את `change_y` ל-1-.
 - c. חץ שמאלה ישנה את `change_x` ל-1-.
 - d. חץ ימינה ישנה את `change_x` ל-1.

מתודת `on_key_release`

מקבלת כארגומנט את `self` ו-`modifiers` (כך חותמת המתודה מגיעה בירושה מ-`View`). המתודה מאפסת את

התנועה כאשר המשתמש משחרר את המקש. המתודה בודקת את התנאים הבאים:

1. אם המקש שהשתחרר הוא `Up/Down` - המתודה תאפס את `change_y`.
2. אם המקש שהשתחרר הוא `Left/Right` - המתודה תאפס את `change_x`.

דגשים



- המתודה `check_for_collision_with_list()` מחזירה רשימה של כל הספרייטים מתוך `sprites_list` שנמצאים כרגע בהתנגשות עם `sprite`.
- מתודת `on_key_press` לא מזיזה את השחקן מיד אלא רק קובעת את הכיוון. בפועל התנועה נעשית ב-`on_update`.
- מתודת `on_key_release` היא הסיבה שהדמות לא ממשיכה לנוע ללא הפסקה, אלא רק כאשר אנחנו לוחצים על מקש מסויים.
- הקפידו להמשיך ולתעד את הקוד החדש שאתם כותבים, תיעוד טוב היום הוא הדרך הטובה ביותר לוודא שתבינו בדיוק מה הקוד שלכם עושה גם בעוד שבועיים.

בדיקה



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 אתחול מלא של רכיבי המשחק באמצעות המתודה `setup()`. בסיום, יש להציג את המסך הראשי של המשחק באמצעות `show_view()`, ואז להתחיל את לולאת המשחק של `Arcade` באמצעות הקריאה `arcade.run()`. לולאה זו מפעילה ברצף את כל מחזורי המשחק.

הפעולות חייבות להופיע בסדר זה, כיוון שכל שלב תלוי בקודמים לו:

1. יצירת חלון – ממשק גרפי שבו המשחק יוצג.
2. יצירת אובייקט המשחק – שמנהל את הלוגיקה, המפה והספרייטים.
3. אתחול עולם המשחק (`setup`) – בנייה מחדש של קירות, שחקן, מטבעות ורוחות.
4. הצגת ה-`View` – מחייב הצבת המשחק בחלון.
5. הפעלת לולאת המשחק (`arcade.run`) – מפעילה את המנוע שמריץ את המשחק בפועל.