פעולות מתוזמנות - CoRoutines

– Spawner

אז לאחר שיש לנו כבר שחקן שמסוגל לירות ואויב שניתן להרוג השלב הבא שמתבקש הוא להכניס עוד כמה אויבים למשחק. אז איך נעשה את זה? אם נכניס אותם פשוט אחד אחד מחלון הפרויקט אומנם יהיו לנו כמה אויבים במסך, אך כולם יופיעו ישירות על המסך, והשאיפה היא להכניס אותם בסדר מסוים כך שכל פעם יעלו מספר האויבים הפוטנציאלים בקצב אחיד ולא בפעם אחת, מה גם שזאת סתם עבודה קשה להכניס עכשיו עשרים ומעלה אויבים.

למעשה אנחנו כבר מכירים שיטה ליצור אינסטנס של prefab במהלך המשחק, אם ניזכר בשלב יצירת הלייזר השתמשנו בפונקציה instantiate בכדי ליצור את היריות, למה שלא נשתמש בה גם כאן? למה שלא ניצור אינסטנסים של אויבים בתדירות של כל חמש שניות למשל? ברמת העיקרון המתזמן הוא לא אובייקט, לא נראה אותו על המסך עם סטופר מזניק תור של אויבים כל אחד בתורו. למזלנו unity דאגו גם לזה ונתנו לנו את האפשרות ליצור אובייקט ריק (Empty object).
כדי ליצור אובייקט ריק , נלחץ על מקש ימני-> create empty. ניתן לאובייקט שם, בהגה המקצועית נהוג לקרוא למתזמן spawn manager, או בקיצור Spawner. נחבר לאובייקט שיצרנו סקריפט עם אותו שם.

יצירת thread היא בעייתית במקצת במיוחד במשחקים מורכבים, היות והיא דורשת מהאובייקטים להיות מסונכרנים לתרד הראשי, פעולה שצורכת יותר מידי משאבים יחסית למה שהיא מועילה לנו, למרבה המזל ישנן דרכים נוספות שנוכל להשתמש בהן בכדי לקבל את האפקט של התזמון, דרך נוספת היא באמצעות מתודות coroutines.
כשאנו קוראים לפונקציה היא רצה עד שהיא משלימה את עצמה, ורק אז מחזירה ערך אם בכלל. למעשה זה אומר שכל פעולה שמתרחשת בפונקציה קורה בתוך פריים אחד של המתודה ()update. קריאה לפונקציה לא יכולה לשמש בכדי להכיל תהליך של אנימציה או רצף אירועים התלויים בזמן.
מתודות coroutines הן מתודות שמאפרות להחזיר ערך 'זמני' מתוך הפונקציה עד הקריאה הבא למתודה שמשם היא ממשיכה מאותו מקום שבו עצרה בפעם הקודמת. למשל ספירה עד עשר: המתודה תחזיר כל פעם שנקרא לה מספר מאחד עד עשר.
בכדי שהמתודה תוכל להחזיר משתנה זמני עליה לקיים שני דברים: 1) המשתנה בחתימת הפונקציה אותו היא מחזירה צריך להיות מסוג IEnumerator שהוא כמו משתנה מסוג iterable ב-c++, כלומר משתנה שמציג את הערך האחרון בו אנחנו נמצאים במתודה שלנו שלנו. 2) במקום שאנחנו מחזירים ערך עם return אנחנו מחזירים עם yield return.

כברירת מחדל מתודות coroutine נקראות מחדש כל פריים, אך ניתן להשהות את זמן הקריאה שלהן, כך שנקרא להן אחת לפרק זמן מסוים ע"י שנחזיר משתנה waitforseconds עם כמה זמן להשהות עד הקריאה הבאה:

```
yield return new WaitForSeconds(float time);
```

חזרה לענייננו: בדיוק כמו שעשינו עם השחקן נצטרך לעשות גם כאן- כדי לייצר אינסטנסים של אובייקט 'אויב', ניצור שדה חדש למחלקה spawnManager מסוג GameObject (לצורך הדוגמא נקרא לו enemyPrefab), נגדיר אותו כ- [serializefield] ונגרור את האובייקט 'אויב' מחלון הפרויקט להיכן שמופיע האובייקט שיצרנו ב-inspector של ה spawn manager. אח"כ נוסיף את פונקציה coroutine שתיצור אויב כל פרק זמן שאותו נבחר ותגריל לו מיקום חדש להתחיל ממנו(על ציר ה-x כי אנחנו מתחילים בראש המסגרת בציר ה-y). בכדי שהמתודה תמשיך עד לסוף המשחק נכניס את האיטרציות של המתודה ללולאה, כך בפעם הבאה שנקרא לה היא תתחיל מתחילת הלולאה, ולא תמשיך מyield return האחרון עד לסוף המתודה, ואז לא יהיה ניתן להשתמש בה יותר בתזמון רציף:

```
IEnumerator SpawnRoutine()
    {
        while (true)
        {
```

```
            Vector3 postospawn = new Vector3(Random.Range(-8f, 8f), 7, 0);
            GameObject new_enemy =Instantiate(_enemyPrefabs, postospawn, Quaternion.identity);
            yield return new WaitForSeconds(time);
        }
    }
```

עוד לא סיימנו. כדי לקרוא למתודה שעשינו צריך להשתמש במתודה המיוחדת StartCoroutine שמפעילה מתודות מסוג coroutine, המתודה מקבלת כפרמטר את הפונקציה עצמה אליה היא קוראת (ניתן גם לשלוח לה מחרוזת עם שם המתודה). למתודה StartCoroutine נקרא דווקא מהפונקציה ()start היות ואין צורך לקרוא לה כל פריים, אלא היא קוראת לעצמה בכל פרק זמן החל מהפריים הראשון של המשחק:

```
void Start()
    {
        StartCoroutine("SpawnRoutine");// StartCoroutine(SpawnRoutine())is also valid.

    }
```