דגמים - Prefabs

נח לבנות אובייקט משחק חדש (GameObject) בסצנה ע"י הוספת רכיב ועריכת המאפיינים שלו לערכים המתאימים. אולם זה יכול ליצור בעיות כאשר אנחנו מתעסקים עם אובייקטים כמו NPC-non-player character כלומר דמות שנשלטת באמצעות המכונה ולא ע"י השחקן, חלק מנוף - עץ למשל או סלעים, או סתם עזרים שיש לשחקן, שהמשותף לכולם שכולם אובייקטים שיכולים להופיע יותר מפעם אחת במהלך המשחק וחולקים מאפיינים דומים. לשכפל את האובייקטים אומנם יצור העתק שלהם, אך השכפול יגרום לכל עותק לעמוד בפני עצמו, כך שאם נרצה לשנות את המבנה של האובייקטים נצטרך לעבור כל העתק בנפרד ולשנות אותו, במקום שיהיה לנו איזשהו אובייקט אב שכל שינוי שיתבצע בו יתבטא בכל העתקים שלו ישירות.

למרבה המזל ב-unity יש אפשרות ליצור **prefab** (קיצור של pre-fabricated – מיוצר מראש). זה מעין דגם המאפשר לאחסן אובייקט-משחק שלם, עם רכיבים ומאפיינים, כ'תבנית' לכל העותקים שלו. בדומה למחלקות וממשקים בשפות תכנות - כל שינוי שיתחולל במחלקת (או ממשק) האב יתבטא גם באינסטנסים שלו. בנוסף ניתן *לדרוס* (override) רכיבים ולשנות מאפיינים של דגם-האב, בדומה מאוד לירושה.

אז כיצד יוצרים דגם? נהוג ליצור תיקיה ייעודית לכל הדגמים במשחק. בכדי ליצור דגם חדש, נגרור אובייקט שכבר יצרנו בתוך הסצנה. נגרור אותו מה-hierarchy view או מחלון הסצנה (scene view), לתיקיה הייעודית בחלון הפרויקט (project view). אם האובייקט מופיע בחלון הפרויקט והוא צבוע כחול ב- hierarchy view, סימן שהפעולה הצליחה! עכשיו, כל פעם שנרצה להוסיף עוד העתק של אותו הprefab-, נוכל פשוט לגרור את האובייקט מחלון הפרויקט לחלון הסצנה או ל- hierarchy view.

כמו שהוזכר קודם, עריכה של הדגם עצמו, כלומר אותו אובייקט prefab שניתן למצוא בחלון הפרויקט, ישפיע על כל העותקים שלו, אך ניתן גם לשנות כל העתק אינדיבידואלית. זה שימושי כאשר אנחנו רוצים ליצור כמה NPC דומים אבל בווריאציות שונות כדי שהמשחק יראה יותר ראליסטי. כדי להבהיר מתי אינסטנס של prefab דורס את אובייקט האב שלו הוא מוצג ב-inspector עם תווית שם **מודגשת** (וכאשר רכיב חדש נוסף לאובייקט שהוא אינסטנס של prefab, כל המאפיינים שלו מודגשים). בהמשך נראה דוגמא לשני prefabs מהמשחק שלנו- לייזר ואויבים.

לייזר-

יריות או לייזר אלה דוגמאות טובות לסוג של prefab היות וכל היריות צריכות להיות זהות, אנחנו נקרא להן די הרבה במהלך המשחק ואין לשחקן שליטה על מנגנון הפעולה שלהן למעט לכוון אותן.

ראשית נעמוד על כמה תכונות של היריות: 1)כל היריות מתנהגות באופן זהה עם אותה מטרה. 2) כל ירייה מגיעה רק אם התרחש איזשהו מאורע (השחקן הראשי ירה ברובה למשל). 3)לכל ירייה יש טווח, היריות נעלמות מהמשחק אם הן פגעו במטרה או אם הן יצאו מהמסגרת של המשחק. 4) סוגי לייזרים שונים הם פשוט העתק של אובייקט לייזר ראשוני עם דריסה של כמה רכיבים.

בתור התחלה ניצור אובייקט פרימיטיבי שיהווה דוגמת ירי, אישית אני הייתי בוחר בצילינדר כי יש לו דמיון לקליע. נשנה את הגודל של האובייקט כך שיראה בגודל ראליסטי יחסית לשחקן שלנו, נגיד השחקן ב-scale של (1,1,1) נעשה את הלייזר ב-scale של 20% ממנו כלומר (0.2, 0.2, 0.2). ניתן ואף מומלץ להוסיף ללייזר חומר (material) בכדי להבליט את הלייזר ברקע.

כמו שאמרנו כבר קודם, כדי להפוך את הלייזר שלנו ל-prefab נצטרך לגרור אותו מהחלון בסצנה (או ה hierarchy view) לתיקייה הייעודית ל-prefabs בחלון הפרויקט.

(*** להשלים הסבר על Apply all ועל Revert all ***)

(*** להשלים הסבר על Find references in scene ועל Open prefab asset ***)

סקריפט-

עתה נתעסק בלוגיקה של האובייקט. ניצור סקריפט 'Laser' ונחבר אותו ל**דגם** (האובייקט שנמצא **בחלון הפרויקט** - כי אחרת זו דריסה). נרצה שהלייזר שלנו פשוט יתקדם קדימה מהרגע שהתרחש המאורע שקרא לו. איך להזיז אובייקט בקו ישר כבר ראינו כאשר התעסקנו בתזוזה של הדמות הראשית. כל מה שנצטרך זה להשתמש בפונקציה transform.translate עם ווקטור3 שמתקדם כלפי מעלה כפול הדלתא טיים ומהירות (כי הרי הלייזר אמור להתקדם מהר בהרבה מהשחקן) ומבחינת סינטקס:

```
void Update() {

;transform.Translate(Vector3.up * Time.deltaTime*_speed)

. . .

{
```

כמובן שאנחנו רוצים לתת ללייזר שלנו טווח, הרי לא נרצה שהוא יתקדם עד אין סוף, כי זה סתם תופס משאבים.
לשם כך נצטרך להכיר את המתודה `Destroy` שהיא משמידה את האובייקט משחק אותו היא מקבלת כפרמטר.

אפשר להוסיף לרכיב שלנו שדה בשם maxY, ולשים בו ערך בהתאם לגודל המסך, למשל 8. ואז להשמיד את this.gameObject במקרה שיצאנו מגבולות המשחק:

```
if(transform.position.y>maxY)
        {
            Destroy(this.gameObject);
{
```

אם נריץ את המשחק בינתיים נראה שאותו לייזר שיצרנו מתקדם כלפי מעלה בקו ישר. עתה אנחנו יכולים למחוק אותו מהסצנה (אבל לא מחלון הפרויקט) היות ולא נצטרך אותו אלא אם כן יתרחש מאורע שקרא לו (למשל לחיצה על רווח במקלדת).
בשביל לעשות את זה נצטרך לחזור לסקריפט של השחקן הראשי.
נרצה שהשחקן הראשי יפעיל, או יותר נכון ייצור אינסטנס של הלייזר שלנו, כאשר הוא מקבל קלט מהמקלדת, למשל רווח.
כבר נפגשנו עם מחלקת input שמתעסקת עם קלטים מהמשתמש ובמיוחד במתודה "GetAxis" שמחזירה ערך בהתאם להתקדמות לכיוון החיובי או השלילי של הצירים, כרגע נשתמש במתודה אחרת של המחלקה- 'GetKeyDown' שמקבלת כפרמטר שם של מקש (כמחרוזת) ומחזירה ערך בוליאני אם הקשנו על אותו מקש או לא. במילים אחרות נרצה ליצור תנאי: אם קיבלנו ערך חיובי מהפונקציה (כלומר לחצנו על המקש) אז ייווצר אינסטנס של לייזר.

בשביל ליצור אינסטנס של prefab נוכל להשתמש במתודה Instantiate(gameObject, transform, Quaternion rotation)
הסבר: המתודה מקבלת כפרמטר איזשהו אובייקט משחק כלשהו, vector3 עם המיקום של האובייקט, ואיזשהו וקטור לסיבוב האובייקט אם נרצה ליצור אותו עם סיבוב כלשהו. לרוב לא נרצה שהאובייקט שאותו אנחנו מפעילים יגיע מסיבוב, לכן נוכל להשתמש במשתנה `Quaternion.identity` שיוצר את האובייקט החדש באותה זווית סיבובית של האובייקט שקרא לו, כלומר הלייזר שלנו יהיה באותו כיוון כמו השחקן הראשי.
מבחינת מיקום נרצה שהלייזר יתחיל קצת מעל לשחקן אבל באותו קו שלו, לשם כך נצטרך להשתמש במיקום של השחקן ולשנות רק את הנקודה y של השחקן כך שתהיה גבוהה יותר במעט:

```
;Vector3 laser_position = new Vector3(transform.position.x , transform.position.y+1, 0)
```

בשביל לשלוח אובייקט מסוג לייזר כפרמטר נצטרך לשמור gameObject מסוג לייזר כמשתנה עצם של המחלקה.
זה המקום להזכיר שמשתני עצם של המחלקה שאנחנו יוצרים כפרטיים אנחנו עדיין יכולים לראות ב-inspector אם הגדרנו מעליהם [serializeFiled]. עתה ניצור אובייקט כזה, לצורך הפשטות נקרא לו laser ונחזור למנוע הגרפי.
נשים לב שבחלון ה-inspector של השחקן הראשי, מתחת לסקריפט מופיעים המשתני עצם שלו .
בכדי להגדיר שהאובייקט עצם של player הוא מסוג לייזר נצטרך לגרור מחלון הפרויקט את ה- prefab 'לייזר' לאיפה שמופיע המשתנה laser ב- inspector של השחקן הראשי. עתה שהגדרנו את האובייקט אפשר להשתמש בפונקציה instantiate:

```
if(Input.GetKeyDown(KeyCode.Space))
```

```
        {
            Vector3 laser_position = new Vector3(transform.position.x , transform.position.y+1, 0);
            Instantiate(laser, laser_position, Quaternion.identity);
{
```

תרגיל: נסו לחשוב איך ניתן ליצור דיילי בין יריה ליריה כך שנצטרך לחכות קצת זמן בין היריות והן לא יתחילו אוטומטית כל פעם שנלחץ 'רווח'.