

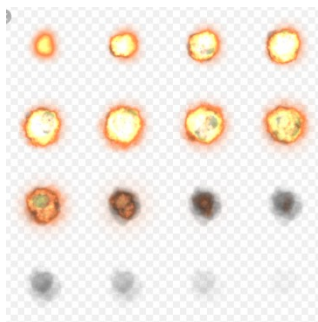


אחד היתרונות הבולטים שיש ל-unity על פני מנועים גרפיים אחרים הוא הקלות שבה ניתן ליצור אנימציות. אנימציות לרוב באות באחת משתי דרכים- או אנימציה קבועה שיש לאובייקט, לרוב ל Prefabs של חפצים או יצורים ש"מקשטים" את המשחק, אבל לא משפיעים עליו ממש. או ע"י מכונת מצבים שמתזמנת את האנימציה המתאימה לאובייקט בהתאם לסיטואציה בה הוא נמצא- הולך, בטל (idle) קופץ וכו'. מה שיפה באנימציות ב-unity הוא האפשרות ליצור את האנימציה ע"י אוסף של sprites שמתקבצים לכדי סרטון קצר שחוזר על עצמו בלופ, וניתן גם "להקליט" את האנימציה- כלומר בצורה אינטראקטיבית ליצור את האנימציה ע"י הזזת האובייקט בכל פריים (של סרטון, לא יחידת זמן של המשחק). עוד מנגנון שיש ל-unity הוא היכולת לחבר כמה sprites שמייצגים חלקים מהאובייקט (למשל ראש, רגליים ידיים וכדו') לכדי אובייקט אחד וכך יהיה ניתן להזיז כל חלק בנפרד וליצור את האפקט של תנועה רב מערכתית מבלי לייצר תמונה במיוחד לכל פריים של סרטון.

אפקט פיצוץ-

האפקט פיצוץ ישמש אותנו לשני מצבים- א. כאשר האויב יושמד, בין ע"י הלייזר ובין בעקבות התנגשות עם השחקן. ב. לשחקן- כאשר נגיע למצב שנגמרו לשחקן החיים נרצה שהוא יבצע אפקט פיצוץ. את האפקט כמעט בטוח שניתן למצוא ב-unity store (בחינם). אם לא מצאנו אל דאגה נשתמש בשיטת ה-"חפש בגוגל".

בגוגל יש לחפש משהו בסגנון של "explosion sprite png". אנחנו צריכים תמונה שמכילה בתוכה מלא תתי-תמונות של מצבים בפיצוץ, אם מצאנו תמונה כזאת בלי רקע מעולה, אחרת קריטה כמו שראינו כבר. דוגמא לתמונה:



לאחר שיש לנו כבר את התמונה נגרור אותה ל-unity ונגדיר אותה כ- sprite(2D and UI) או sprite mode צריך להגדיר אותה כ- multiple (כי עכשיו אנחנו מתעסקים עם כמה תמונות ולא עם תמונה יחידה) ואז נלחץ על ה- sprite editor כדי לערוך את רצף התמונות לכדי אוסף תמונות.

הערה חשובה: יכול להיות שלא מותקן לכם ה- sprite editor בפרויקט, כל מה שצריך כדי להתקין אותו זה ללכת ל- Window -> Package Manager ובשורת החיפוש לחפש 2D Sprite ואז להתקין אותו בכפתור install שמופיע בתחתית החלון.

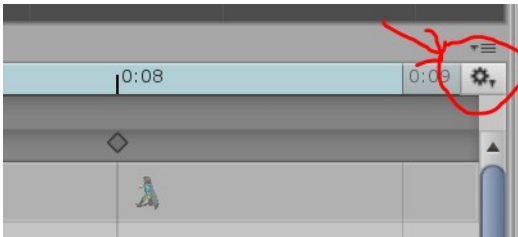
אחרי שיש לנו כבר את הספרייט נפעיל את sprite editor. שם נראה שיש לנו כפתור slice, הוא אחראי לחתוך את התמונה לספרייטים קטנים בהתאם למה שנגדיר לו: ניתן לחתוך בצורה אוטומטית או ע"י איזשהו grid. אחרי שבחרנו דרך לחתוך ואישרנו את ה- slice נסגור את החלון. כעת אפשר לראות את כל הספרייטים שחתכנו מהתמונה המקורית ע"י החץ שמופיע בצד התמונה.

כדי להפוך את רצף הספרייטים לכדי אנימציה אחת, נצטרך איזשהו אובייקט שיכיל בתוכו את האנימציה, כמין מכולל

אנימציות.

ניצור אובייקט ריק חדש ונתן לו שם, בעודנו על האובייקט נקרא לחלון עורך האנימציות, נבחר - window->animation >animation.

אם ניתן ננסה אפילו להצמיד את החלון שיהיה באותה שורה של חלון הסצנה, כך יהיה נגיש יותר בהמשך. ניצור אנימציה חדשה ונשמור אותה. כדי לייצר את האנימציה של הפיצוץ נצטרך לגרור את כל הספרייטים הרלוונטיים לנו לאנימציה לכן נבחר בכל הספרייטים של תמונת הפיצוץ ונגרור אותם לחלון האנימציה. אם נלחץ על כפתור ה-play נראה שהאנימציה רצה לנו על המסך. יכול להיות שהאנימציה רצה מהר מידי לטעמנו או לאט מידי, אל דאגה ניתן לשנות את מהירות האנימציה ע"י ה-samples. הוא אחראי על המהירות של ריצת הספרייטים. אם לא מופיע לכם ה-samples כשאתם נכנסים לחלון האנימציה בחרו בגלגל השיניים הקטן שנמצא בפינה הימנית העליונה בחלון



- show sample rate < . אפשרות שניה, פחות נוחה, היא להזיז ממש את הפריימים של האנימציה. אם הגודל של האנימציה לא מוצא חן בעיניכם כמובן שניתן לשנות אותו, ואם הצבע של הפיצוץ חזק מידי או חלש מידי ניתן לשחק בצבעים של האנימציה ה-renders השונים שניתן להוסיף ב-add properties בחלון האנימציה. אם אנחנו מתכוונים להשתמש באותה אנימציה לכמה מצבים כדי להפוך אותה ל-prefab. אז יש לנו אנימציה, השאלה איך נחבר אותה לדמויות השונות כך שהיא תפעל ברגע המתאים? בתור התחלה נחבר את האנימציה שתהיה אובייקט 'בן' לאובייקט שעליו הוא פועל כדי שיזוז איתו. לצורך הדוגמא (בה"כ) ניקח את השחקן. השחקן אמור להתפוצץ כאשר ה"חיים" שלו נגמרים, במילים אחרות ברגע שהמשתנה חיים שווה לאפס אנחנו צריכים להפעיל את האובייקט שאחראי על הפיצוץ ואז לעשות destroy () על השחקן.

אם ניזכר רגע באיך יצרנו את המגן, השתמשנו במתודה המיוחדת שמפעילה את האובייקט הבן: `GameObject.SetActive(bool status)`.

גם כאן אנחנו רוצים להפעיל אותה על האנימציה במידה והגענו למצב שלא נשאר לשחקן חיים. חשוב לא לשכוח 'ליידע' את השחקן על האובייקט בן שנוסף אליו, כדי שיוכל להפעיל אותו.

כדי שהשחקן לא יזוז תוך כדי שהוא מפעיל את האנימציה, כי אחרת זה נראה מוזר, פשוט נגדיר את `speed_` להיות 0 כשמפעילים את האנימציה, כך השחקן לא יוכל לנוע בזמן הפיצוץ. לאחר שהפעלנו את האנימציה נוכל לקרוא לפונק' `destroy()`.

הערה: יכול להיות שהפונק' `destroy` תקרא לפני שהאנימציה פעלה, או באמצע הפעולה, במקרה כזה כדי שנוסיף מתודת `coroutine` או שנכנס את כל ה"פרוטוקול הרס" לתוך מתודה כזו, ונקרא ל `yield return` כאורך האנימציה.

```
public void Damage()
{
    life--;
    if(life<1)
    {
        StartCoroutine(Explosion());
    }
}
```

```
IEnumerator Explosion()
{
    _explosion.SetActive(true);
    _speed = 0;
```

```
yield return new WaitForSeconds(0.7f);
Destroy(this.gameObject);
}
```

דבר דומה ניתן לעשות גם עם ה-prefab של האויב, רק לשים לב שהאויב נפגע גם מפגיעה ע"י לייזר וגם ממגע בשחקן.

מכונת מצבים-

לפעמים יש צורך ביותר מאנימציה אחד לאובייקט, או בתזמון בין האנימציות של אובייקט שיקרה בזמן קבוע או ע"י טריגר.

לכן נעדיף להשתמש במכונת מצבים (animator). מכונת מצבים כשמה כן היא אוטומט שמתזמן את האנימציות של האובייקט בהתאם לטריגר(או פרמטר אחר) שמעביר אותו ממצב למצב. למשל אוטומט שמעביר מצב אם הוא קיבל אחד או אפס ניתן לדמות אותו למעבר מצב של דמות מזמן בטלה (idle) לזמן ריצה ולהפך- אם קיבלנו כקלט 1 אז הדמות תציג את האנימציה של הריצה, אם קיבלנו אפס (או יותר נכון לא קיבלנו קלט) אז היא תציג את האנימציה של הבטלה.

המכונה בנויה ממצבים (או אנימציות), transitions – מעברים בין מצב למצב, parameters- הטריגרים שדרכם המכונה יודעת לאיזה מצב לפנות. מצב שמוגדר כברירת מחדל יהיה בצבע כתום.

בשביל להפעיל את המכונת מצבים דבר ראשון נצטרך להוסיף את חלון ה-animator למסך הראשי, ע"י window-> animator

נבחר את האובייקט עליו נרצה לעבוד. בעדיפות על אובייקט עם כמה אנימציות, אבל גם אובייקט עם אנימציה אחת מספיק לנו ובלבד שיהיה לנו מעבר בין מצב אנימציה למצב בלי אנימציה. דוגמא מדמות שעשינו למשחק אחר: דמות של עורך דין שיש לו שני מצבים-1)מצב עומד או בטל(idle), הדמות נשארת במקום ויש לה אנימציה מהספרייטים הבאים:

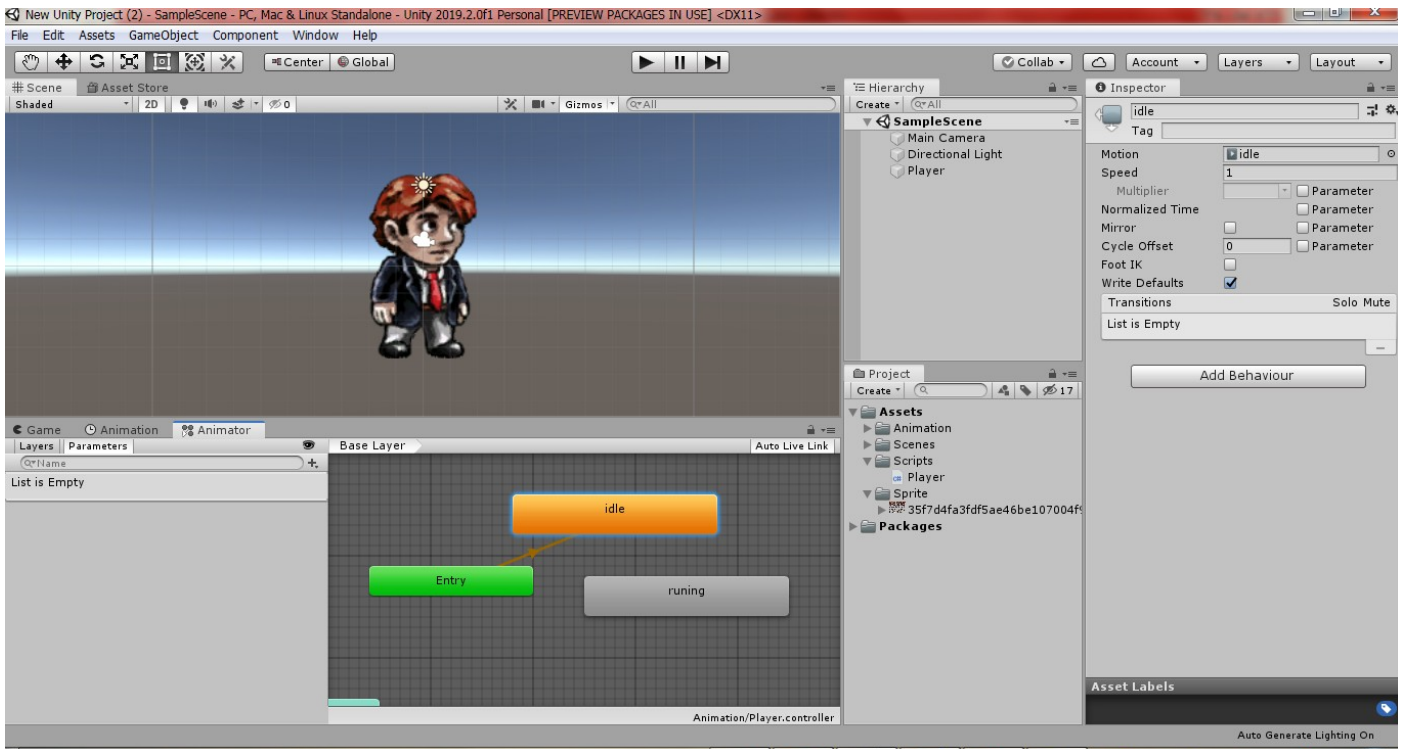


(2) מצב רץ)

(running), הדמות נעה:



לכל רצף ספרייטים יצרנו אנימציה נפרדת. ניתן ליצור כמה אנימציות לאובייקט בחלון ה-animation וליד ה-samples (מצד שמאל) יש לשונית עם שם האנימציה, אם פותחים אותה יש למטה את האפשרות Create new clip. כשפתחנו את ה-animator קיבלנו את החלון הבא:



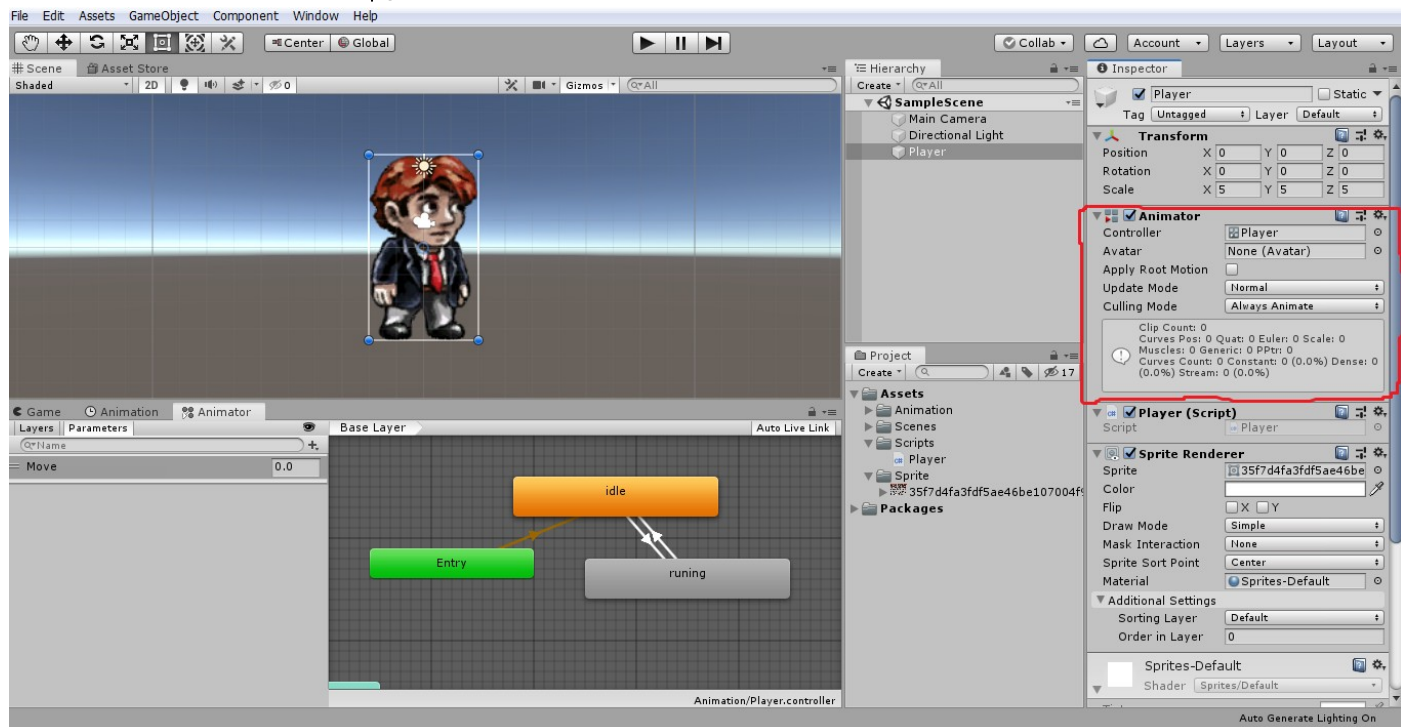
ניתן לראות מהתמונה שהמצב המוגדר כברירת מחדל הוא המצב idle, ובינתיים אין לנו שום קשרים בין מצבים למעט הקשר היחיד שיש לנו שהוא מעבר מ-entry-ה-0 " שלנו, ומשתמשים בו פעם אחת באתחול הדמות ויותר לא חוזרים אליו, ל-idle שהוא האנימציה הראשית.

כדי להוסיף קשרים (transition) נלחץ מקש ימני על המצב שממנו יוצא הקשר- <make transition ונגרור אותו למצב אליו הוא אמור להגיע, וכנ"ל בכיוון ההפוך.

בהרצה של המשחק הדמות מחליפה בין אנימציות לאחר כמה שניות, זאת משום שה-transition עדיין מוגדרים ב-inspector כ-has exit time. אנחנו רוצים להוסיף לדמות פרמטרים כדי שהמעבר יהיה לפי דרישה. אם נשים לב מהתמונה למעלה בצד שמאל של המכונת מצבים יש לנו Parameters-layers, כדי להוסיף פרמטר חדש נצטרך לעמוד על חלון הפרמטרים וללחוץ על הפלוס הקטן מצד ימין לכפתור החיפוש. יש לנו אפשרות לבחור את הסוג של הפרמטר (int, float, bool, trigger). משום שאנחנו מתעסקים בתזוזה כרגע נבחר float. ניתן שם לפרמטר (אצלנו קוראים לו Move), נלחץ על אחד הקשרים ונסתכל על האינספקטור. נשים לב שבאינספקטור מתחת ל-setting של ה-exit time יש לנו רכיב שקוראים לו Conditions הרכיב אחראי להוסיף לקשר פרמטר שלפיו הוא עובד. נוסיף את הפרמטר שיצרנו לקשר ע"י לחיצה על הפלוס הקטן מצד ימין (add to list). התנאי שהוספנו כרגע בנוי משלושה חלקים- שם התנאי, במקרה שלנו Move, לשונית שמגדירה האם גדול (Greater) או קטן (less) וחלון להזין את המספר שממנו הוא אמור להיות גדולקטן.

היות וההתנהלות שלנו היא בוקטורים, אז אם אנחנו במצב של vector2d.right, כלומר אם קיבלנו בפלט לפנות ימינה ואנחנו מתקדמים אז אנחנו כבר לא עומדים במקום אלא נעים, כלומר אם קיבלנו קלט מהמשתמש נעבור ממצב של אנימציה עומדת לאנימציה רצה, לכן עדיף כאן שהתנאי יהיה אם Move גדול מ-0 (או 0.1 אם אנחנו עוברים בין מצבים שיש להם כמה אפשרויות), וכנ"ל לכיוון ההפוך רק עם less.

ועכשיו לחלק התכנותי. בהסתכלות על ה-inspector של הדמות ניתן לראות שיש לנו כבר משתנה מסוג animator:



לכן מספיק לנו לקרוא לרכיב במתודת start() של הדמות בכדי לאתחל אותה. ניצור משתנה עצם מסוג אנימציה שאותו נאתחל להיות הרכיב אנימציה:

```
private Animator _anim;
void Start()
{
    _anim = GetComponent<Animator>();
}
```

המטרה היא שהדמות תעבור למצב אנימציה "רץ" במידה וקיבלנו ערך שהוא גדול מאפס, לשם כך נצטרך לעדכן את הפרמטר שיצרנו. ב-unity יש מתודה במיוחד לכך: `<Parameter type>(<Parameter> name>, <var>)`

הצגנו את זה בצורה שנראית מבלבלת אך למעשה זה בכלל לא מסובך: למשל אצלנו אנחנו רוצים "לערוך" את הפרמטר של `_anim` (כך קראנו לאנימציה) והוא מסוג `Float` לכן נשתמש במתודה `SetFloat` (ואם זה היה `int` אז היינו משתמשים ב-`SetInt`, וכו'), שמקבלת את שם הפרמטר (התנאי) כמחרוזת, ולפי מי התנאי מתקיים. נניח אנחנו רוצים משתנה שהערך שלו הוא פלט מהפונקציה `input` אם לחצנו על איזשהו חץ אופקי ("horizontal"), נקרא לו `direction` לצורך הדוגמא, אזי המתודה תראה כך:

```
void Update()
{
    float direction = Input.GetAxis("Horizontal");
    _anim.SetFloat("Move", direction);
    ...
}
```



כשמריצים את המשחק רואים שהשחקן באמת עובר בין המצבים אבל עדיין יכול להיות שהוא לא עובר ישר בין אנימציות אלא מחכה לסוף האנימציה האחת בכדי לעבור לאחרת. כנראה שהסיבה שזה קורה היא כי ה- `Has Exit` Time עדיין מסומן בשני ה- `transitions`, אם נבטל אותם המעבר יהיה חד יותר בהתאם לקלט אותו הוא מקבל, ואם נרצה מעבר חד אפילו יותר, בתוך ה- `setting` (מתחת ל- `has exit time`) יש `transition duration` הוא אחראי לדיילי בין המעברים, ואם נשווה אותו ל-0 המעבר יהיה חד הרבה יותר, אך לפעמים נעדיף דווקא שהמעבר לא יהיה חד מידי, כי אחרת זה לא נראה "ראליסטי" מספיק.

אומנם הצגנו דוגמא פשוטה יחסית לשימוש במכונת המצבים, ואם ננסה ליישם את זה במשחק שלנו יהיה לנו קצת יותר מסובך כי הקוד קצת יותר עמוס, אך הבסיס הוא אותו בסיס בשניהם. במשחק שלנו אפשר ליישם את מכונת המצבים בהרבה מקרים: בחללית או בדמות הראשית כשהיא יורה או כשהיא זזה;

אם נרצה להוסיף "בוסים" למשחק, כלומר אויבים יותר גדולים מהאויבים הקטנים שמתים אחרי ירייה אחת, אפשר להוסיף להם אנימציות ביניים כל פעם שפוגעים בהם עד שהם מתפוצצים לחלוטין, ויש עוד אינספור דוגמאות. הערה: בדוגמא לעיל הראנו רק שימוש בפרמטר מסוג `float` הדבר דומה מאוד במקריים של `int` ו- `bool`, אבל במקרה של פרמטר מסוג `trigger` הוא קצת שונה. בניגוד לשלושת הקודמים, במצב טריגר אנחנו מחכים שיקרה איזשהו אירוע, ולא דווקא קלט, למשל אם היינו עושים אויב "בוס" היינו משתמשים בטריגר כאשר הדמות הראשית פגעה באויב כמה פעמים ועכשיו הוא להתפוצץ ("להיות מושמד").

פרמטר מסוג טריגר אנחנו מפעילים כאשר אנחנו רוצים שיהיה מעבר בין האנימציות במקרה קיצוני ולא משהו נשלט לגמרי ע"י השחקן, נגיד לאויב "בוס" לפני שנעשה עליו `destroy` () נפעיל את הטריגר (ולרוב נשהה את האובייקט כמה שניות בשביל שתרוץ האנימציה) ואז נשמיד את הדמות סופית.

