מדריך Unity

תוכן עניינים

מבוא ליוניטי ולעולם הגיימינג

יוניטי זהו מנוע גרפי אשר משמש לפיתוח משחקי וידאו, סמארטפונים וקונסולות שונות מיזה שנת 2005.

בגרסא כיום, נוכל להוסיף אלמנטים ליוניטי, לגרום להם לזוז עם החצים, ללחוץ עליהם, להעניק להם 'חיים' ולהפוך אובייקטים לגיבורים. בנוסף, ניתן להוסיף אפשרויות פיזיקה, תזוזה של הרקע ועוד...

במדריך זה, תוכלו לגעת בקצה המזלג ביוניטי. מדובר במדריך בסיסי ביותר אשר מתווספים אליו סרטונים שונים שצולמו בשנת 2016...

בתום הצפייה במדריך תוכלו להשתמש בתוכנה, להוסיף אלמנטים, לגרום להם לזוז ולנוע...ועוד...

אז למה אתם מחכים? הורידו את היוניטי מכאן בלחיצה למחשב שלכם. ההתקנה תקח זמן מה. לחצו על ההתקנה, לכו לשחק קצת בקונסולה שלכם וכשתחזרו כבר היוניטי יהיה מותקן במחשבכם! בהצלחה!

הוספת אלמנטים

בתוכנה עצמה - נראה כי יש לנו סצנה ראשונה והתחלתית.

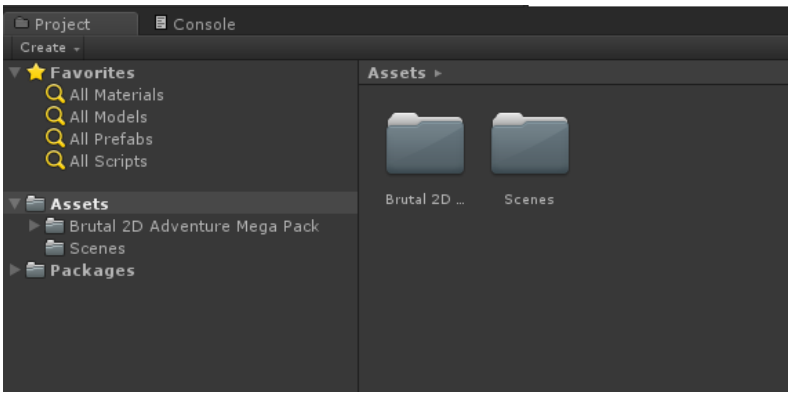
את הסצנה הזו נוכל להפוך למסך הפתיחה שלנו עם כפתורים, מוזיקה ורקע מגניב ממש עוד מעט...

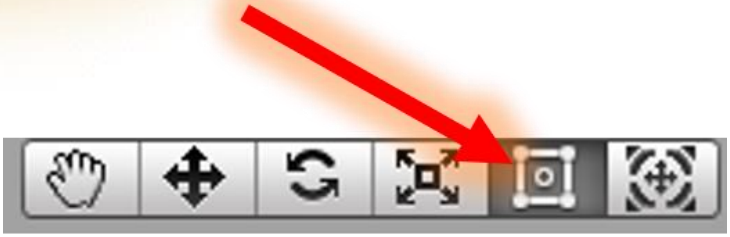
לבינתיים, נוכל לגרור אליה אלמנטים שונים. בין אם מוכנים ובין אם אלמנטים של יוניטי.

כעת, נדבר על שניהם:

- בהנחה שמדובר בתמונות שמוכנות מראש. כמו למשל, תמונה של כדור או רקע גדול. פשוט נצטרך לגרור אותם פנימה אל תוך היוניטי.

- ניתן גם לבצע שימוש באלמנטים מוכנים מראש שתוכלו לבצע באמצעות כפתור COMPONENT בתפריט הראשי.





נוכל לגרור אלמנט מסוים מתוך אובייקטי הסצנה בחלק התחתון אל תוך המסך שלנו.
נבחין כי יש שם מצלמה. מדובר במצלמת המשחק ובריבוע אשר המשתמש רואה.
כאשר נלחץ על אלמנט מסוים שגררנו פנימה, נוכל להזיז אותו/להגדיל/להקטין אותו
בלחיצה על הסימן הבא בסרגל שמופיע בצד שמאל למעלה.

פיזיקה של דברים נופלים

בפרק זה נדבר על שני אלמנטים חשובים מנשוא בכל הנוגע לעולם האובייקטים שלנו. עלינו לעבוד בצורה מאוד מסודרת ולחשוב לאט לאט, שלב אחרי שלב. אסור לנו לפספס אף חלק, כי אם לא נרד לסוף דעת המשחק, כנראה שמשהו לא יעבוד ואולי אפילו יהיו באגים. צפו בסרטון הבא וקבלו אינדיקציה בנושא של הבנת אלגוריתם בעולמינו אנו:

https://www.youtube.com/watch?v=cDA3_5982h8&t=101s

נדמה את עולם הפיזיקה החשוב מנשוא,

באמצעות כדור משחק אשר עלינו לכדרר אותו...

RigidBody

- ראשית, נרצה שהכדור ייפול מלמעלה כלפי מטה. כלומר, שיתחיל מלמעלה תמיד ויבצע נחיתה רכה כלפי מטה. לכן, עלינו לייצר פיזיקה. בלחיצה על האלמנט, נבחין כי בצד ימין יש לנו חלון בשם inspector ובתוכו אפשרות להזין component חדש... ליחצו והוסיפו rigitbody חדש. התכונה הזו תאפשר לאלמנט שלכם ליפול כלפי מטה... נסו בעצמיכם! כעת הכדור יפול!

בנוסף, נרצה לבצע מצב בו יש קיר תחתון והכדור מגיע עד אליו ונעצר בו. נוסיף גם לקיר rigitbody והפעם בתכונותיו נחליף את gravity scale במקום 1 ל0, כך שהוא לא ייפול ויישאר במקומו. בנוסף, נשנה לקיר שלנו את body type לstatic.

Collider

- אנו רוצים כעת ליצור התנגשויות בין הכדור שלנו לבין אובייקט אחר (במקרה זה: הקיר). לכן, נצטרך להוסיף גם לכדור וגם לקיר collider רלוונטי אליו דרך הcomponent. למעשה, לכדור נוסיף circle collider ולקיר נוסיף box collider. למעשה, כל אובייקט שיש לו את הcollider שלו, יוכל להתנגש באובייקט אחר וכך כעת - כאשר הכדור ייפול (באמצעות הגרביטציה של הrigitbody) הוא ייפול ויתנגש בקיר ויעצר שם (בזכות הcollider).

סקריפטים

ביוניטי נכתוב בשפת c#.

שפת תכנות זו הומצאה על ידי מייקרוסופט ואנו משתמשים באפשרויות שונות שהיא מציעה. על מנת להיסגר על השפה מומלץ לצפות בסרטון המצורף וכן גם ראו ערך 'לוגיקה' כאן בעולם ההייטק. הרשאות כמו private וכן גם public בעלי ערך כאן מאוד. משתנים ניתן להגדיר פה לפי סוג המחלקה שלהם. בנוסף, יש מחלקה אחת עיקרית וחשובה מאוד ביוניטי וזוהי `update()`. כל מה שאנו רוצים שיקרה בכל מסך שלנו, נכתוב בתוך הפונקציה הזו.

ביוניטי נוכל להוסיף סקריפט חדש באמצעות לחיצה על component בתפריט העליון. לאחר מכן, הוא יתווסף לחלונית inspector התחתונה מטה. עלינו יהיה לכתוב בתוכו את כל האפשרויות של התזוזה, הטקסט שנרצה לשנות, הלחיצות וכדומה - ולאחר מכן בסיום העבודה, נגרור אותו ימינה לתוך האיזור של תכונות האובייקט שלנו.

במידה ומונחים כמו משתנים, הרשאות, פונקציות ועוד לא מובנים לכם עדיין - נא לעבור על מדריך הלוגיקה בשקידה.

לחיצה על כפתורים

ברגע שחיברנו את הסקריפט שלנו לאובייקט מסוים, נוכל כעת להגדיר בתוך הסקריפט שבלחיצה על מקש מסוים או בלחיצה על האובייקט עצמו שיבצע פעולה כלשהי...

כיצד עושים זאת בשני שלבים ברורים. ראשית, נתחיל מלחיצות עכבר/טאצ' בסמארטפון ובשיעור הבא במדריך נעבור גם ללחיצה על חצי המקלדת וכפתורי המקלדת במחשב PC.

- ראשית, כדי לזהות לחיצות של עכבר, צריך להוסיף קוליידרים לאוביקטים. הביטו בנושא שיעור מספר 3 מעלה 'פיזיקה של דברים נופלים.'

- לאחר מכן, נשתמש ב ()OnMouseUp – זוהי מתודה אשר מתעסקת בלחיצת עכבר על האובייקט שלנו...

```
public void OnMouseUp()
{
}
```

זהו! כעת, בתוך המתודה של ()OnMouseUp נוכל להזין את הפקודות שנרצה לבצע.

למשל, לגרום לאובייקט שלנו להיעלם. אזי נכתוב:

```
transform.Translate(0, 0, 0);
```

או למשל נרצה שבלחיצה האובייקט יקפוץ כלפי מעלה, אזי נשתמש בשורות הבאות:

```
GetComponent<Rigidbody>().AddForce(0.3 * 0.3 * Time.deltaTime);
```

פרוייקט 1 – כדור אוסף אובייקטים

צרו פרוייקט חדש 3D ביוניטי ובתוכו הוסיפו כדור אחד ואובייקט אחד.

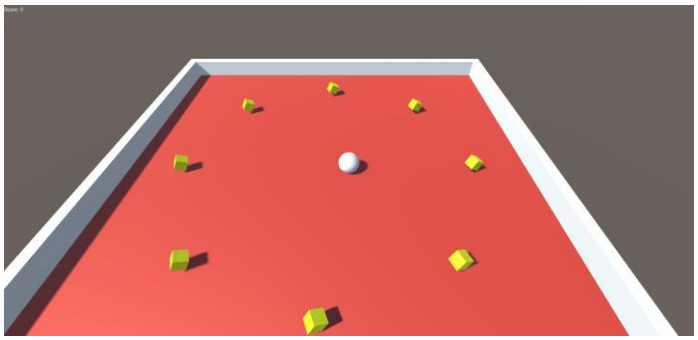
לאחר מכן, העניקו להם את הקוליידרים והריג'ידבאדי הרלוונטי...

לסיום, הצמידו לכדור שלנו גם סקריפט ובתוך מתודת הupdate שלו כתבו הקוד הבא:

```
void Update () {
        float xInput = Input.GetAxis('Horizontal');
        float yInput = Input.GetAxis('Vertical');
        transform.Translate(xInput * Time.deltaTime, yInput * Time.deltaTime, 0);
}
```

אנחנו רוצים להעלים את האובייקטים כאשר הכדור נוגע בהם, כלומר אנחנו צריכים להבין מתי יש לנו התנגשות ביניהם... לכן - נשתמש במתודה מיוחדת. כמו שיוניטי קוראת לפונקציה ()Update בסקריפט שלנו כאשר יש עדכון של המסך, כך גם היא קוראתל-()OnCollisionEnter2D כאשר יש התנגשות!

```
private void OnCollisionEnter2D(Collision2D collision) {
Destroy(collision.gameObject);
}
```



שימו לב כי ה-gameObject.collision , מכיל את האוביקט שבו התנגשנו... לסיום, בגרסת Unity המתקדמת ביותר, עלינו להפעיל את האפשרות להתנגשות אובייקט מסוג קינמטי להתנגשות באובייקט סטטי. נוכל לטפל בכך בrigitbody של הכדור ושל כל אחד מהאובייקטים.

תזוזות עם החצים

על מנת לבצע תזוזה של הכדור שלנו (ממיני פרוייקט) שלא על ידי העכבר, נוכל לבצע באמצעות החצים. כל שעלינו לעשות הוא בתוך הupdate לוודא על איזה מקש לחצנו. הביטו בקוד הבא...

```
public KeyCode upKey;
public KeyCode downKey;

if (Input.GetKey(upKey)) {

        transform.Translate(0, 3 * Time.deltaTime, 0);
}
```

שימו לב - כעת, בחלון הימני בתכונות האובייקט שלנו, נוכל לראות כי תחת האובייקט יש מקום להוסיף את הלחצנים הרלוונטים. הכוונה היא שUpKey יוכל להיות כל מקש שתבחרו לנכון. אבל עליכם להגדיר מבין הרשימה שנפתחת, באיזה מקש מדובר.

הוספת טקסט

נוכל להוסיף טקסט לדף שלנו!

הטקסט יוכל להציג כמה פעמים לחצנו או כמה אלמנטים אכלנו (במיני פרוייקט)...

ראשית, עלינו לייצר component חדש שהוא מסוג של canvas.

לאחר מכן, נצטרך בתוך הסקריפט לכתוב משתנה חדש מסוג `public Text countText;`

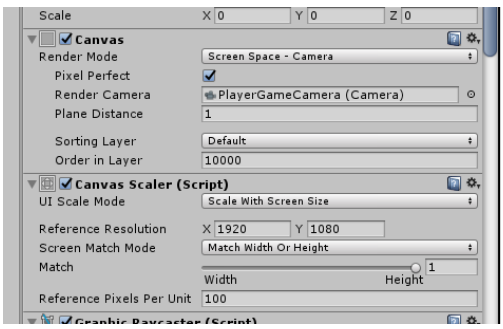
בתוך המתודה של הלחיצה או התזוזה, נשלב גם שורה נוספת אשר תציג את המשתנה שמונה את כמות הפעמים למשל, לבחירתכם.

```
countText.text = 'Count: ' + count;
```

שימו לב- בגרסאות יוניטי שונות עלינו להוסיף ייבוא של ספריית הUI המכילה גם את הפקד של הטקסט – `import UnityEngine.UI`

הערה חשובה נוספת היא לגבי הcanvas עצמו. לעיתים הוא יהיה מרוח על פני כל הspace שלנו. נרצה שהוא יהיה סביב המצלמה שלנו או סביב השחקן שלנו. לכן, בסרגל הימני של הinspector נוכל לשנות את הrender mode שיהיה שייך למצלמת השחקן.

הביטו בתמונה הבאה –



לסיכום, במספר שלבים ברורים:

1. הקליקו על המקש הימני בעכבר בתוך חלון ההירכייה. ייפתח לפניכם תפריט. ביחרו בו את האפשרות של Text תחת פקדי הUI.

2 . התאימו את הrender mode של הcanvas שלכם, אשר מכיל את הטקסט, שיהיה סביב המצלמה. עליכם לגרור את המצלמה פנימה (גרירה מחלון ההירכייה לתוך הinput הממתין בrender mode שבחלון הinspector של הcanvas).

3. לאחר מכן, נמשיך לC# ובכתיבת הסקריפט ניצור ייבוא של ספריית הUI

4. ניצור שני משתנים בתוך הclass שלנו. האחד משתנה מסוג int הנוגע למספר המונה והשני משתנה מסוג Text הנוגע לטקסט שבו נרצה להציג את המספר המונה.

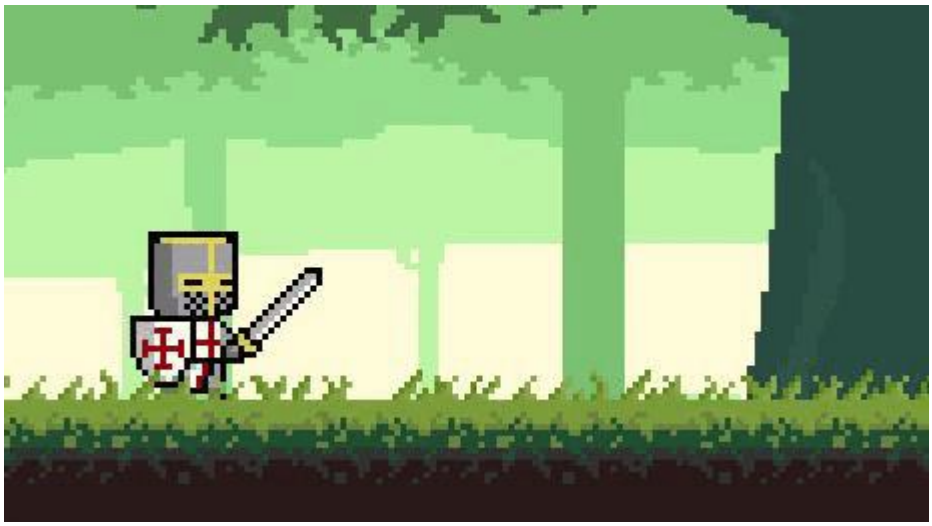
5.לסיום, בתוך אזור ההתנגשות נבצע את הבדיקה הרלוונטית לגבי האובייקט בו נרצה להתנגש ובתוכה ניצור את שורות הקוד של `count++` דהיינו העלאה ב1 במונה וכן גם `points.text = " 'Points: ' + count "` דהיינו הצגה של הטקסט עם הכיתוב Points ולאחריו המונה. 6. לסיום, נשייך את הסקריפט שיצרנו לגיבור. נשים לב שיש לנו input הממתין לקבל טקסט כלשהו. נזריק את הטקסט מחלון ההירכייה ישירות לinput שבתוך הסקריפט שבחלון הinspector.

יצירת תפריט ומעבר בין סצנות

צפו בסרטון ההדרכה הבא, הלקוח מן הרשת, המציג כיצד נוכל לבנות תפריט ובו קישורים למעברי הסצנות בקבצי הסקריפט. בהצלחה

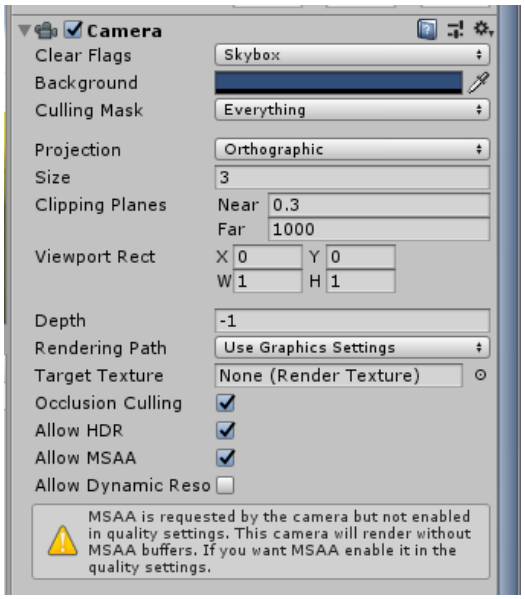
מעקב המצלמה אחר השחקן

אחד הסקריפטים היותר חשובים במשחק שלנו הוא תזוזת המצלמה. בעולם הגיימינג, אף פעם לא נחשוף את המשתמש שלנו לכל מסך המשחק בבת אחת... נגרום למצלמת המשחק לנוע ולזוז בהתאם למיקום של הPlayer שלנו!



לכן, נצטרך לבצע מספר שלבים...

1. להגדיר לsize במצלמה שלנו 3 לערך.

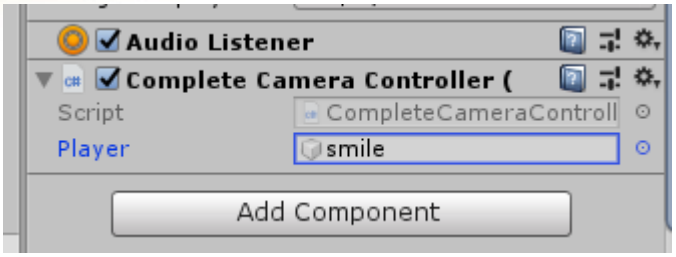


2. לכוון את המצלמה בסצנה שלנו למרכז הדמות שאחריה היא תעקוב...

3. לצרף את הסקריפט הבא למצלמה.

```
using UnityEngine;
using System.Collections;

public class CompleteCameraController : MonoBehaviour {

    public GameObject player;

    private Vector3 offset;

    void Start ()
    {
        offset = transform.position - player.transform.position;
    }

    void LateUpdate ()
    {
        transform.position = player.transform.position + offset;
    }
}
```

4. שימו לב, הסקריפט שהוספנו למצלמה, מכיל בתוכו משתנה של Player. עליכם לגרור את השחקן מחלונית ההירכייה (הכי שמאלית) אל תוך הריבוע הקטן הריק...



וזהו זה!

כעת יש מעקב של המצלמה אחרי הדמות שלכם!

מסך Game Over

אחד מהקודים הבסיסיים הוא סיום של המשחק... עלינו לוודא בupdate בכל רגע, האם נגמרו החיים לשחקן/האם השחקן נפסל/האם השחקן נפל לתהום ונגמר המשחק? נעשה זאת בקוד יחסית פשוט...

```
void Update ()
{
    if (???){
        Application.LoadLevel(Application.loadedLevel);
    }
}
```

במקום סימני השאלה בקוד נוכל לשלב בדיקות שנרצה לעשות כאמור.

למשל: `transform.position.y <= -5`

או אולי `lifes == 0`

אנימציות - Sprites

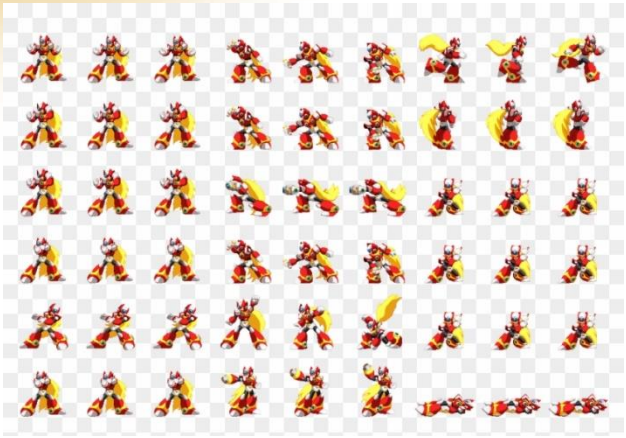
עוד כלי חשוב ביותר בעולם הגיימינג הוא הנפשות (אנימציות).

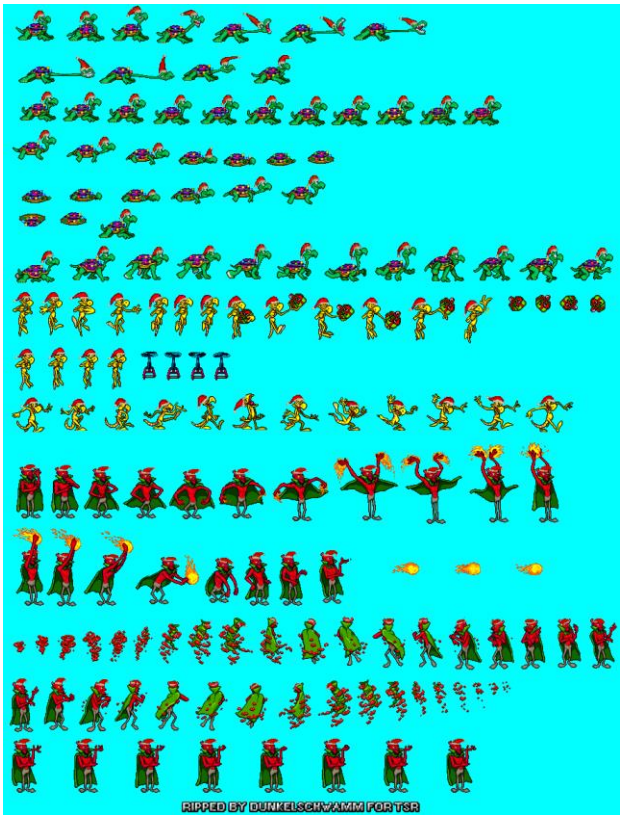
הכוונה היא שהדמויות שלנו יהיו דינאמיות ולעורר בהן חיים. הביטו על הדמות הבאה, שהרי היא פשוט עומדת בלי תנועה:

המצב הסטטי ניכר וברור, הוא משבש את המשחקיות ויכול לגרום גם למשתמשים להפסיק לשחק בעקבות כך. וזה כל הרעיון שעומד מאחורי שיעור זה.

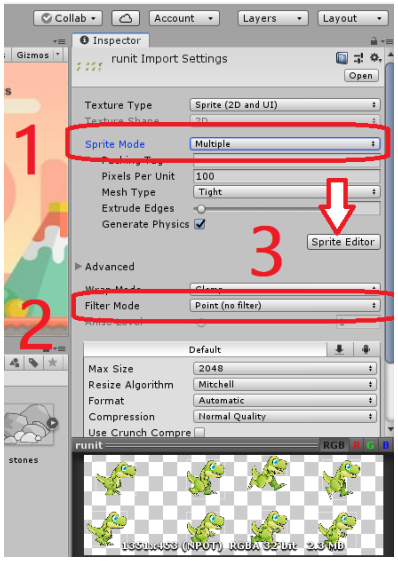
עלינו לקחת את התמונה שלנו ולגרום לה לזוז ולחיות אותה...

בעולם האנימציות, הכל עובד לפי frames, כלומר כל 0.24 שניות למשל תבצע תמונה אחרת. ככל שנדייק בתמונות שנזין, כך נקבל דמות שזזה בצורה 'חלקה יותר'. למשל, את העין שנסגרת ונפתחת לחייזר למעלה, על המעצב הגרפי היה להביא כ100 תמונות של עצימת עין בהילוך איטי... 100 תמונות שמרכיבות תמונה אחת נקראת ביוניטי Sprite, ואם תכתבו בגוגל 'Sprite' ולידו שם של דמות משחק שאתם מכירים תקבלו המון אפשרויות... למשל: Sprite Jazz Jackrabbit , Sprite Sonic , Sprite Rayman ועוד...

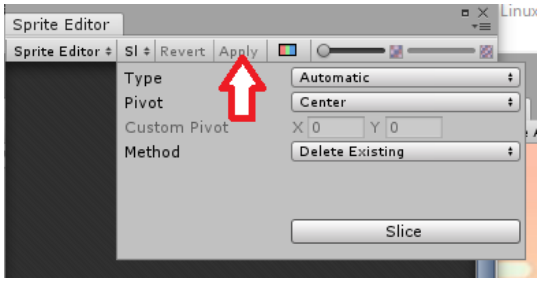




לאחר שהשגתם/יצרתם תמונה אחת גדולה שמורכבת מלפחות 7 תמונות (שכל אחת כאמור היא חלק קטן מהאנימציה), אנא גררו אותה לתוך חלונית הassets התחתונה שלנו... בלחיצה על התמונה שהוספתם, ראו כי בצד ימין בחלונית הinspector יש לנו אפשרויות שטרם היו - אנא שנו את הinspector שלכם כמו בתמונה - לפי שלבים 1,2 ולסיום לחצו על הכפתור Sprite Editor לפי שלב 3



בחלונית זו, עליכם ללחוץ על Sl בתפריט למעלה ואז לוודא שהאפשרות לחיתוך היא Automatic. לאחר מכן, ליחצו על כפתור Apply. למעשה, פעולה זו מצליחה לקרוא ולזהות איפה יש דמויות ואיפה יש רקע שקוף. וכך היא חותכת את כל הדמויות שלכם בצורה אוטומטית...



גררו את האנימציה מהassets שלמטה אל תוך המשחק שלכם.
היוניטי יבקש מכם לשמור את האנימציה בתקייה וכך תעשו. לאחר מכן, תוכלו למחוק את האנימציה מתוך חלון הGame אבל תשימו לב שיש לכם כעת בתוך הassets גם דברים נוספים שקשורים לאנימציה ולתמונה עצמה...

הוסיפו לדמות שלכם בחלון הinspector רכיב של Animator , כמו בתמונה הבאה. אל תשכחו לגרור לשדה הראשון שלה את האנימציה שנוצרה כעת מהassets... וכעת הדמות תזוז בלחיצה על הPLAY! אם משהו לא תקין או לא עובד, אנא השאירו הודעה כאן למטה ונענה לשאלותיכם.
בונוס עלינו לגרום בלחיצה לדמות שלנו לשנות את האנימציה שלה. למשל, לחיצה על מקש הרווח, אז ישתנה האנימציה...
ראשית, אנא ודאו שאתם סגורים על שיעור מספר 6 - תזוזות עם החצים.

בתוך הupdate נוכל לעשות בדיקה - אם המקש שנלחץ הוא רווח...
אז תבצע את שתי השורות הבאות:

```
Animator animator = this.gameObject.GetComponent();
animator.runtimeAnimatorController = anim1;
```

הוא יכעס כעת מפני שאינו מכיר את anim1. anim1 זהו משתנה מסוג RuntimeAnimatorController שהגדרנו כמשתנה גלובלי למעלה... עליכם להוסיף אותו גם. כעת אין errors בקוד... אבל - בחלונית הinspector אנא גררו פנימה את האנימציה של הקפיצה שלכם! וזהו! כעת, בלחיצה על הרווח - הדמות שלכם תקפץ!!! בהצלחה!

יריות ולבבות

מהאלמנטים החשובים ביותר בעולם הגיימינג זה יריות, פגיעות, לבבות ופסילות... בשיעור זה נלמד כיצד ניתן ביוניטי לייצר ירייה. יחד עם זאת, לגבי הלוגיקה של לבבות, אנא חיזרו על פרקי הלוגיקה כאן בעולם ההייטק.

כאשר אנו רוצים לייצר יריה מהדמות שלנו בלחיצה על מקש הרווח למשל, קודם כל עלינו להביא את המיקום של הדמות שלנו! לדעת מהיכן להוציא את הירייה.

כדי לקבל את המיקום של הדמות שלנו בציר הX ובציר הY נשתמש בקוד הבא:

```
this.gameObject.transform.position = new
Vector3(transform.position.x + 2, transform.position.y + 2 ,
transform.position.z);
```

לאחר שיהיה לנו את המיקום במקום הX – `transform.position.x`

ואת המיקום במקום הY – `transform.position.y`

נוכל להוציא מאותו מיקום בדיוק את הassets של הירייה שלנו.

הasset של הירייה יכול להיות ממש עיגול קטן צהבהב שתייצרו בעצמכם בPowerPoint, אבל תהיו חייבים ליצור לו גם בupdate שלכם את ההתקדמות על גבי ציר הX (למשל, יותר ימינה או יותר שמאלה בהתאמה):

```
var tmp = transform.position;
tmp.x -= 0.09f;
transform.position = tmp;
```

בואו נתקדם למשל למצב שבו יש יריה של רשע.

ברגע שהירייה תפגע בכם - נרצה למשל להוריד לב או להציג מסך של GameOver המסמן על סוף המשחק.

1. נגדיר משתנה בוליאני באובייקט של הגיבור שלנו בשם dead.
2. בתוך ה update נעשה בדיקה - אם dead הוא true, הפסק משחק והגע למסך חדש למשל... או הצג assets עם תמונה של 'GameOver!' אחרת - אל תעשה דבר. כמו במצב ברירת המחדל (תמיד bool הוא false)
3. כמובן לא לשכוח, כמו שלמדנו במיני פרויקט אודות הפונקציה

```
private void OnCollisionEnter2D(Collision2D collision)
```

עלינו לכתוב בדיקה... עלינו לבדוק מיהו האובייקט שהתנגש בגיבור שלנו.

אם זו הירייה עלינו לבצע פעולה מסוימת בעת ההתנגשות...

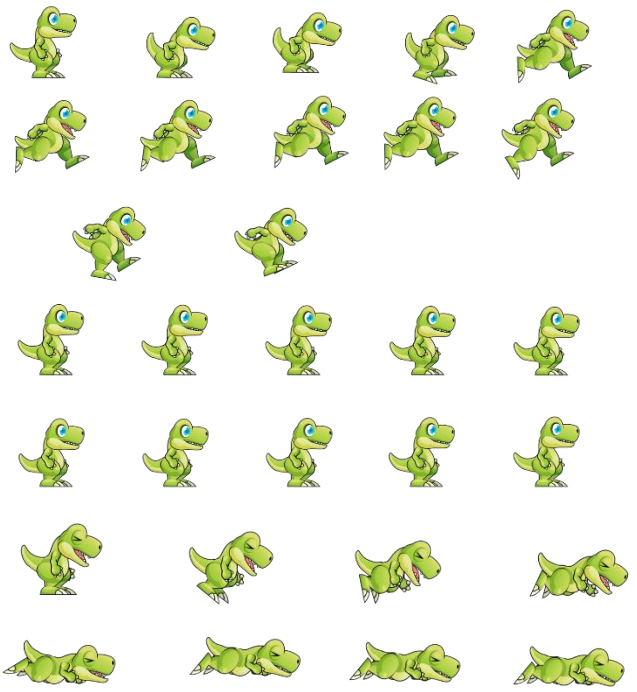
TAG

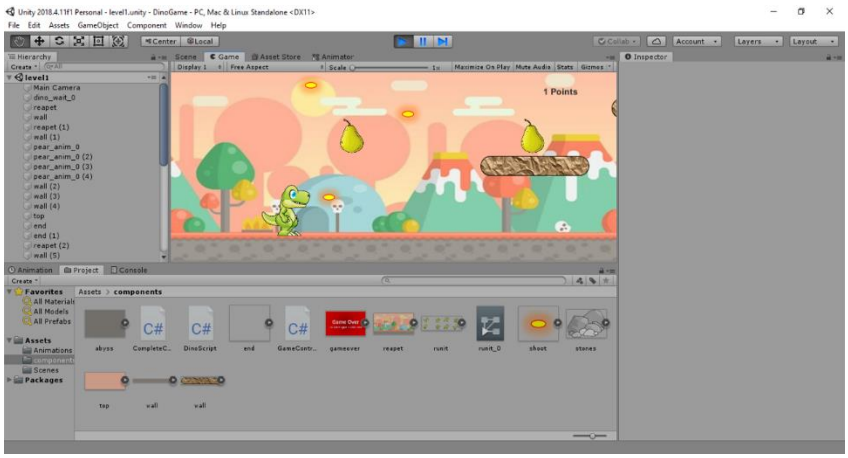
למעשה, נוכל להגדיר לכל אובייקט איזשהו tag. זה קורה בצד ימין בחלונית ה inspect שלנו, למעלה. הגדירו tag חדש בשם shoot ובידקו בתוך הפונקציה מסעיף 3 למעלה, אם התגית הוא shoot אנא תעלים את האובייקט למשל...

```
private void OnCollisionEnter2D(Collision2D collision) {
if (collision.gameObject.tag == 'shoot')

        Destroy(collision.gameObject);
}
```

פרויקט 2 – דינו הדינוזאור

צרו פרוייקט המכיל דינוזאור שהולך ימינה, שמאלה וקופץ. הדינוזאור אוכל אגסים וצובר 10 נקודות עבור כל אגס. כאשר הדינוזאור נתקל באסטרואיד שפוגע בו, הוא מת ומופיע כיתוב של GameOver. השתמשו בASSETS הבאים:





הוספת ג'ויסטיק למשחק מובייל

זהו מדריך מיוחד המסביר איך ניתן להוסיף joystick שיישאר טבוע וקבוע על המסך ובו נוכל לגרום לדמות שלנו לזוז ימינה, שמאלה, למטה, למעלה וכן גם ניצור כפתור של קפיצה.

אתם מוזמנים לצפות טרם קריאת מדריך זה, בסרטון המצולם המתאר את שלבי ההטמעה של הג'ויסטיק ושל כפתור הקפיצה. כמו כן, אנא צרו לעצמכם את הכדור שתרצו להזיז על גבי המשטח, הוסיפו לו rigidbody וכן סקריפט בשם myScript. בנוסף, צרו canvas שגודלו יותאם לגודל המצלמה שלכם והיכנסו לasset store וייבאו מאזור זה את החבילה של joystick pack החינמית הראשונה כמו בסרטון.

הטמעת הJoyStick הראשי

ראשית, ניצור את הג'ויסטיק שלנו. נייבא אותו מתוך תקיית הPrefabs החדשה שנוצרה ובתוכה נבחר את Fixed Joystick. נוודא שהוא נמצא בתוך הCanvas שיצרנו קודם לכן. בתוך myScript שמקושר לכדור, נזין את קטעי הקוד הבאים - גם קריאה ישירה לJoyStick וגם איתחול של מיקום תזוזת הכדור בUpdate בהתאם לגרירת הג'ויסטיק –

```
public class myScript : MonoBehaviour {

protected Joystick joystick;

void Start(){
joystick = FindObjectOfType<Joystick>();
}

void Update(){
var rigidbody = GetComponent<Rigidbody>();
rigidbody.velocity = new Vector3(joystick.Horizontal * 6f,
rigidbody.velocity.y, joystick.Vertical * 15f);
}

}
```

קפיצת הכדור

כעת, נרצה לגרום לכדור המתגלגל שלנו גם לקפץ לו. הקפיצה תתבצע בעת לחיצה על תמונה עגולה, וזהו השלב הראשון שלכם - צרו את התמונה בתוך הCavnas שיצרנו. מקש ימני בחלון ההירכייה ובתפריט שנפתח לפנינו בקטגוריית UI נוסיף Image. נייבא תמונה מתוך האוסף החדש שנוצר עם ייבוא ספריית

הג'ויסטיק. כמו כן, לאחר מכן קשרו לתמונה שלכם סקריפט חדש בשם JoyButton ובו יש משתנה בוליאני הבודק האם העיגול לחוץ או לא –

```
public class Joybutton : MonoBehaviour , IPointerUpHandler,
IPointerDownHandler{

public bool Pressed;

public void OnPointerDown(PointerEventData eventData){
Pressed = true;
}

public void OnPointerUp(PointerEventData eventData){
Pressed = false;
}
```

וכשנחזור לסקריפט של הכדור שלנו (myScript) נרצה לבצע את הפקודה שתרים את הכדור כלפי מעלה ואחרי הרפייה מהלחיצה, תוריד אותו חזרה מטה -נוסיף את הרפרנס הבא כמשתנה – `protected Joybutton joybutton;`

וכן בתוך הStart, נקשר את המשתנה שיצרנו אל הלחצן –

```
joybutton = FindObjectOfType<Joybutton>();
```

וכעת, בUpdate נבצע בדיקה פשוטה - האם הלחצן לחיץ או לאו.

אם לחיץ - תקפיץ את הכדור:

```
if (joybutton.Pressed){
if (!jump){
rigidbody.velocity += Vector3.up * 10f;
jump = true;
}
}
```

Instantiate

בשיעור זה נלמד כיצד ניתן לשכפל אובייקטים מסוימים בפרויקט שלנו. לצורך העניין, יש לפנינו משחק עם מגוון רשעים וסוגי יהלומים לאיסוף. הגיבור שלנו נע במרחב שלו ומגיע לרשע מסוים מסוג צב (כמו במשחק: Jazz Jackrabbit). הוא יורה בצב ומשמיד אותו. הצב נעלם מהמשחק, אולם בעוד כמה רגעים במרחב נגיע לאזור חדש ושוב צב חדש יופיע. השכפול הזה נקרא `instantiate` ובשיעור זה נלמד אודות אלמנט זה.

אותה דוגמא תקפה גם ליהלומים שתיארנו. נאסוף אחד, אולם בהמשך יהיו כמה מאותו הסוג... גם במקרה זה, האובייקט עליו נלמד השיעור נותן את המענה.

נדגים את השימוש באובייקט הinstantiate באמצעות משחק של איסוף חפצים נופלים. ראו את ההסבר בסרטון המצורף מטה.

עלינו להבין את שורת הקוד אותה נטמיע בעת התנגשות, כפי שמופיע בסרטון:

האובייקט instantiate מכיל 3 חלקים עיקריים:

1. מיהו הprefab שלנו? כלומר, נתבקש לגרור פנימה בחלון הinspector את האובייקט מתוך הASSET שלנו שאותו נרצה להציג מחדש. הprefab יהיה מסוג של GameObject לרוב.
2. את המיקום של האובייקט החדש. בסרטון למשל, הצגנו מיקום רנדומלי... ניתן להציג את המיקום באמצעות האובייקט Vector3 שמכיל 3 צירים - X , Y , Z
3. Quaternion.identity - סיבוב תנועתיות האובייקט החדש. נותיר לרוב ב0f.

לאחר כל אלו, נסו לגרום לאובייקט שלכם להתחיל מחדש בנקודה מסוימת על גבי המסך בכל פעם שתאספו אותו. נסו לבנות בעצמכם את הפרויקט של איסוף החפצים הנופלים או לחלופין לשדרג את פרויקט הארנב שאוסף גזרים, כך שבכל פעם יתאתחל גזר חדש. בהצלחה!

```
Instantiate(coin, new Vector3(5f, 5f, 0), Quaternion.identity);
```