***Cradle***

# מה זה Cradle ?

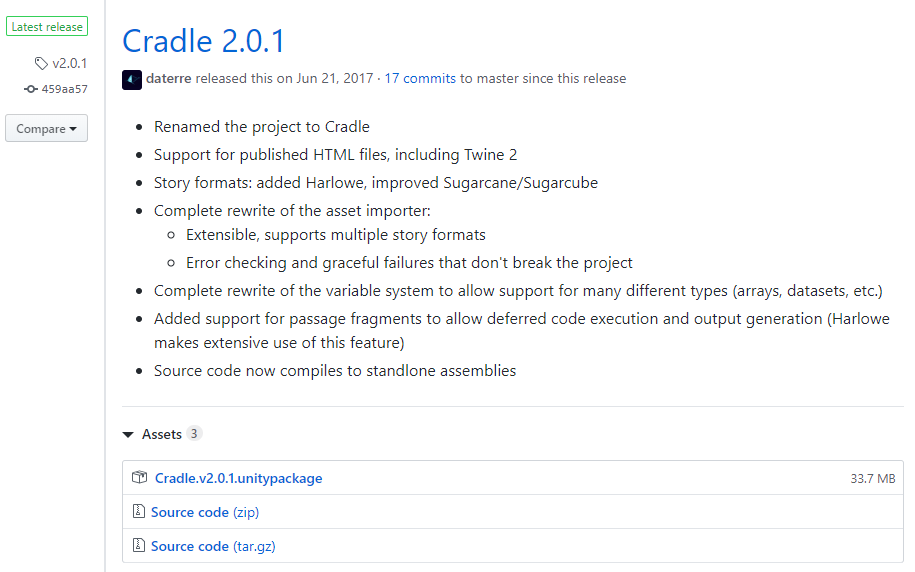
Cradle בעברו היה נקרא UnityTwine הוא תוסף לUnity המאפשר למפתח גישה לכל האפשריות של story-telling במשחק. הוא מייבא סיפורי Twine, מריץ אותם ומקל על הוספות אינטראקטיביות מותאמת אישית באמצעות סקריפטים.

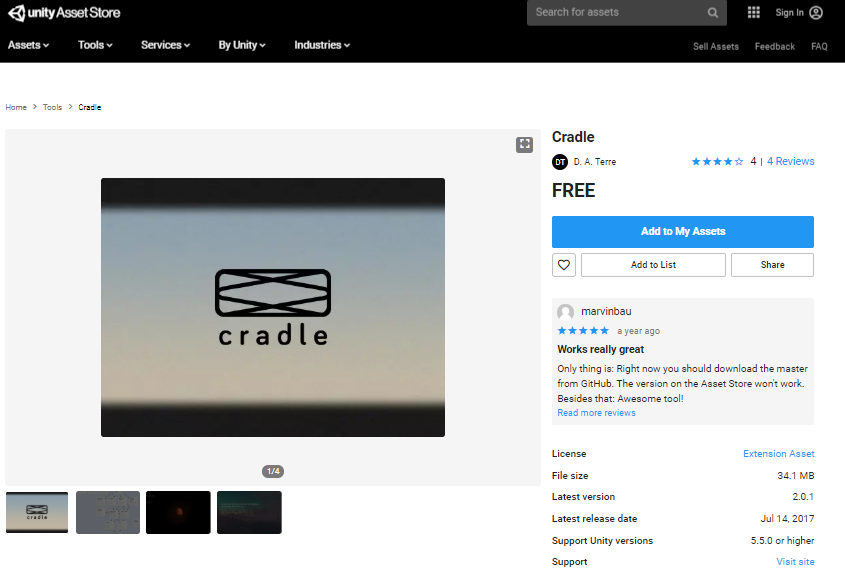
כותבים יכולים לעצב ולבחון את סיפוריהם באופן עצמאי כמו שניתן באמצאות Twine, מתכנתים ואמנים יכולים לפתח את האינטראקציה והתצוגה מבלי לדאוג לבקרת זרימה. כמורידים את Cradle לUnity הוא מאחד בין שני העולמות ומקל על המשתמש.

# הורדה

ישנן שתי דרכים להוריד את התוסף על מנת שנוכל להשתמש בו בUnity:

1. ניתן להוריד Cradle דרך הgithub של התוסף <https://github.com/daterre/Cradle/releases>.



1. או פשוט להוריד דרך ישירות דרך הUnity בasset store ולצרף לפרוייקט <https://assetstore.unity.com/packages/tools/cradle-93606>  
   

# ייבוא סיפור חדש מ- twine

ה-cradle מקשיב אם ישנן קבצי .html או .twee ששמנו בפרוייקט שלנו וממשיך לפעול על מנת לייבא אותם. הוא מתייחס לtwine כאל לוגיקה ומתרגם אותו לסקריפט של html עם מבנה דומה ואותו שם.  
כל סיפור יכול להיות מייובא מספר רב של פעמים וכל פעם התוסף יידע לשנות את הסקריפטים בהתאם (מה שמקל על המתכנת).  
  
**איך מייבאים את הסיפורים מ-Twine?**

Twine   
1. צריך לבחור publish to file מהתפריט של ההסיפור  
2. לשמור את הקובץ במיקום של הפרוייקט של יוניטי.

**פורמטי סיפור נתמכים על ידי Cradle:**

1. Harlowe: הפורמט הדיפולטיבי של Twine 2 והוא הכי מומלץ לשימוש על ידי Cradle.
2. Sugarcane: הפורמט הדיפולטיבי של Twine 1 (גרסה ישנה ובגלל זה לא נדבר עליה).
3. SugarCube: גרסה ה-"עשירה" יותר התומכת גם בtwine 1 וגם ב twine 2 .

**Playback**

ברגע שאנו מייבאים את הסיפור ונוצר לנו סקריפט – ניתן להוסיף אותו לאובייקט בסצנה כמו שאנו מוסיפים סקריפט רגיל.

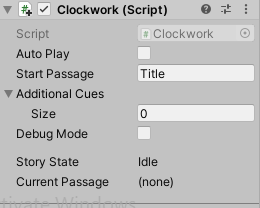
כל סקריפטי הסיפורים כוללים את מאפייני העורך הבאים:

AutoPlay: כשהפרמטר הזה הוא true אז הוא מתחיל לנגן את הסיפור באופו אוטומטי בתחילת המשחק.

StartPassage: מציין מאיזה מעבר להתחיל לנגן.

AdditionalCues: אובייקטים נוספים עליהם צריך לחפש cues. (נרחיב על cues בהמשך)

OutputStyleTags: הסיפור יפיק תגיות המציינות מידע על תוכן הסיפור.



**TwineTextPlayer**:

בתוך Cradle נכללת אופציה של prefab וסקריפט שניתן להשתמש בהם כדי להציג ולנהל אינטראקציה עם סיפור בצורה טקסטואלית בלבד. ה-prefab הזה בנוי עם רכיבי ממשק המשתמש של יוניטי (4.6+) לטיפול בטקסט וב-layout.

ניתן להשתמש בצורה הבאה:  
1. ליצור סצנה חדשה.

2.לגרור את הprefab של TwineTextPlayer לתוך הסצנה.

3.לייבא את סיפור ה-Twine ולגרור את הסקריפט שנוצר לתוך הTwineTextPlayer.



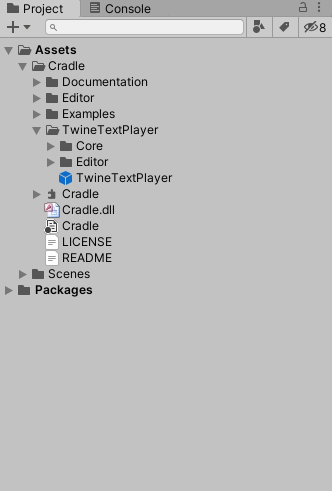
סקריפט המפעיל את הסיפור, מוציא ממנו מידע ומציג אותו על ה Canvas.

בהתחלה הופך את כל האובייקטים ללא פעילים, כל אובייקט יהפוך לפעיל אם יכנסו אליו ערכים.

בכל כניסה לעמוד חדש שיצרנו הוא מוחק את כל האובייקטים בתוך הcontainer שהיו שייכים לעמוד הקודם.

לאחר מכן, הוא עובר על הערכים שהוא מוציא מהסיפור ומוסיף אותם לcontainer לפי הגדרתם.

בסוף, הוא עובר על הערכים של הקטע ומוחק אותם.



(!) לאחר ההורדה תופיע לנו תיקיית Cradle בתוך   
ה Assets.

התיקייה TwineTextPlayer מכילה prefab הכולל בתוכו script מוכן שלוקח את הסיפור Twine שלנו ומוציא ממנו מידע.

כלומר, לא צריך לכתוב שום קוד במיוחד על מנת ליצור שיחה פשוטה בין דמויות ! נשאר לעצב רק את ה prefab בהתאם לרצוננו.

ניקח את ה prefab ונגרור אותו ל Hirarchy של המשחק שלנו.

**Scripting**

כל קטע בסיפור שמיובא הופך לפונקציה שה-output שלה הוא טקסט או קישורים. סקריפטים מותאמים אישית יכולים להאזין לפלט שנוצר, להציג אותו לפי הצורך ולשלוט באילו קישורים משמשים לקידום הסיפור.

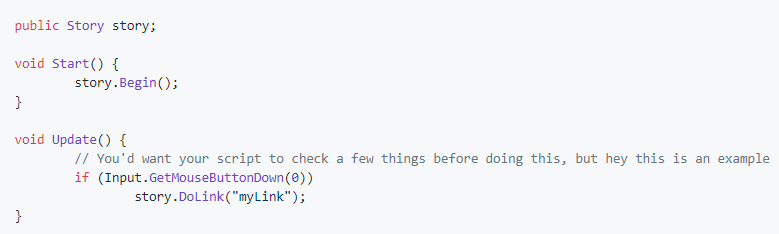
כדי להבין סקריפטים עם Cradle יש צורך להכיר קודם את Story class, שממנו יורשים כל הסיפורים המיובאים.

**אינטראקציה עם הסיפור:**

ה- story class הוא בעצם ה"לב" של Cradle והוא מכיל את תוכן הסיפור וכולל מספר מטודות המאפשרות סקריפטים אחרים לשחק ולקיים אינטראקציה עם סיפור רץ.

* Begin() – מתחיל את הסיפור על ידי נגינת הקטע שהוגדר על ידי StartPassage.
* DoLink(string linkName) – עוקב אחר הקישור עם השם שצוין.
* GoTo(string passageName) – קופץ למעבר שצוין ומנגן את הסיפור משם.

דוגמא לסקריפט:

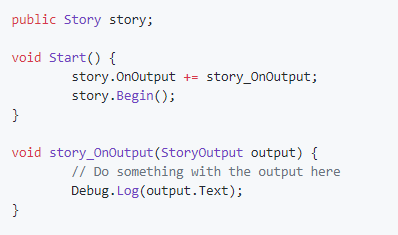


**קריאת תוכן הסיפור:**

לאחר שהגענו לחלק מסויים בסיפור, ניתן לבחון את ה-output שלו דרך הסקריפט.

* Output – רשימה של כל הפלטים של הקטע.
* GetCurrentText() – תת רשימה של הפלט – אך רק מציג את הטקסט.
* GetCurrentLinks() – תת רשימה של הפלט – אך רק מציג את הלינקים.
* Tags – כל התגיות של הקטע.
* Vars – הערכים העדכניים של הפרמטרים הגלובליים בסיפור.
* CurrentPassageName – השם של הקטע הנוכחי שמבוצע.
* PassageHistory – רשימה של כל הקטעים שהסיפור ביקר שהם בצורה כרונולוגית.

ניתן לעצור פלט של קטע גם בזמן שהוא מבוצע באמצעות cues או עם אירוע OnOutput בצורה הבאה:



**לינקים**

כפורמט מבוסס web, Twine בנוי סביב מושג הקישורים. לחיצה על קישורים היא הדרך העיקרית בה משחקי Twine מתנגנים והדרך בה מתקדם הסיפור.

ב-cradle הלינקים מיוצגים על ידי StoryLink class ומבצעים אחד מהשני הפעולות הבאות כאשר יש להן trigger עם Story.DoLink():

* הולך לקטע אחר.
* מבצע action שזהו מקטע של קטע שלא הוצג עם הכניסה לקטע.

אם צוינו גם action וגם שם מעבר, הפעולה מבוצעת תחילה, ורק כאשר היא נעשית הסיפור מתקדם לקטע הבא.

**לינקים עם שמות:**

נתייחס לדוגמא הבאה:   


על מנת להפעיל ולהיכנס לקטע “grandma” צריך לקרוא ל- Story.DoLink("Visit your grandmother") בתוך הסקריפט.

אבל מה קורה אם הכותבים מחליטים לשנות את השם של הלינק ל- "Go to your grandmother's house"?

יהיה צריך לשנות ולעדכן את הסקריפט בתוך יוניטי על מנת שהקריאה לא תיכשל.

על מנת שנמנע משבירת לינקים בצורה כזאת – cradle מרחיב את הסינטקס הסטנדרטי ונותן לקרוא ללינק בצורה הבאה:



כעט ניתן לקרוא ל- Story.DoLink("visitGrandma") וזה יעבוד.   
כל עוד הכותבים שומרים על השם של הלינק – שאר הטקסט אינו משפיע וניתן לערוך אותו.

**למה לא להשתמש פשוט בקטע המטרה כשם הלינק?**

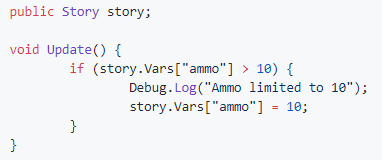
יש שתי תשובות לכך:

1. לפעמים ללינקים אין קטע אלא רק action שאותו הוא מבצע.
2. לפעמים יותר מלינק אחד מוביל לאותו קטע אבל לכל אחד יהיה action אחר.

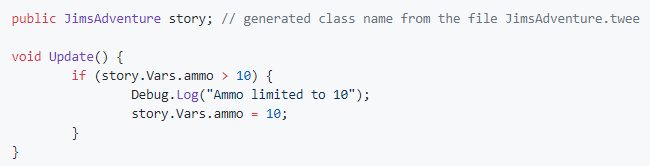
**משתנים**

סיפורים משתמשים לרוב ב-marcos כדי לאחסן ערכים במשתנים, לקרוא אותם מאוחר יותר על מנת לבדוק תנאים (if), להציג אותם (print) ועוד. בסקריפטים ניתן לגשת למשתנים אלה בשתי דרכים:

להשתמש עם getters or setters, לדוגמא:



או דרך המשתנה שנוצר ישירות:



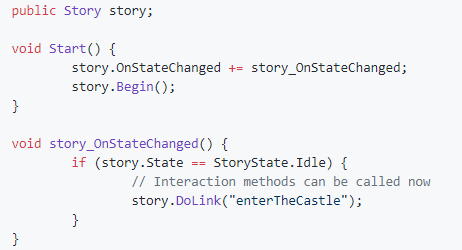
הערה: משתנים הם כולם מהסוג StoryVar, שהוא סוג ערך דינאמי שיכול לייצג מחרוזת, מספר, משתנה בוליאני או כל סוג אחר הנתמך על ידי פורמט הסיפור בשימוש.

**מצב סיפור**

כאשר סיפור מתנגן, הוא יכול להיות באחת מכמה מצבים. ניתן להגיע למצב הסיפור מהמאפיין Story.State.

* Idle - הסיפור לא התחיל או סיים את ביצוע הקטע או התת קטע. על מנת לבדוק ניתן ללכת למאפיין הפלט של הסיפור כדי לראות מה הוא פלט ואז לקרוא ל- DoLink() כדי להמשיך.
* Playing - הסיפור מבצע/מנגן כעת קטע, מטודות אינטראקציה לא יעבדו.
* Paused - הסיפור מבצע כעת קטע, אך הושהה באמצע, מטודות אינטראקציה לא יעבדו, צריך לקרוא ל- resume() על מנת להמשיך.

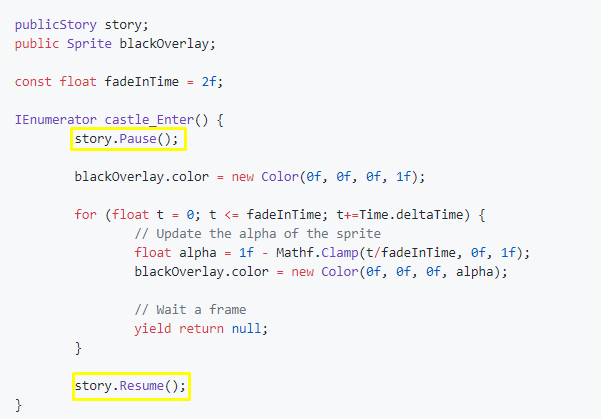
על מנת לאבחן מתי המצב של הסיפור השתנה צריך להשתמש בevent של OnStateChanged, לדוגמא:



**Pause and Resume**:

ניתן להשהות את הסיפור על מנת לבצע משימות הגוזלות זמן, כמו לחכות לסיום האנימציות או לטעינת סצנה, לפני שנוצר פלט סיפור נוסף. השהיה נחוצה רק כאשר הסיפור במצב play, אם זה Idle, אז אין מה להשהות.

דוגמא:

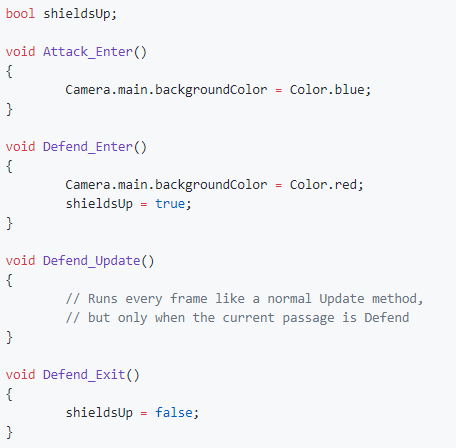


**Cues**

ב-cradle בנוסף יש מערכת של cues מאוד חזקה אשר מאפשרת לסקריפטים לרוץ במקביל לקטע המתנגן.

הערה: לפני שנקראו ,cues בגרסה 2.0 היו נקראים 'hooks', זה שונה כדי למנוע בלבול עם המונח hook כפי שהוא משמש בתבנית הסיפור של Harlowe.

דוגמא פשוטה לשימוש:   
נגיד יש לנו שני קטעים אשר נקראים attack ו-defend. הינה סקריפט עם cues אשר ישנה את הקרע של המצלמה כדי להתאים לקטע:



**איך מגדירים סקריפט של Cue?**

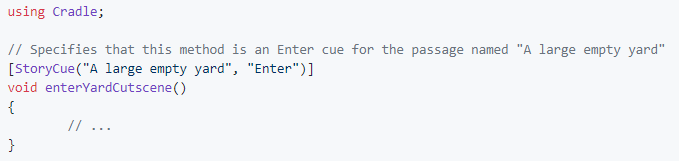
1. יוצרים סקריפט חדש
2. הוסף אותו לאובייקט המשחק הזה שמכיל את תסריט הסיפור שלך, או ...
3. ... הוסף אותו לכל אובייקט משחק בסצנה שלך והוסף את אובייקט המשחק הזה לרשימת AdditionalCues list

**סוגים של cues:**

(מחליפים את “passage” בשם של הקטע)

1. passage\_Enter() - נקרא ברגע שנכנסים לקטע. זה אומר שאחרי Begin,DoLink or GoTo וכל תת קטע שקודד.
2. passage\_Exit() – נקרא על מקטעים נוכחיים רגע לפני שנכנסים לקטע מרכזי חדש בעזרת DoLink or GoTo. סגירת קטע נקראת קודם על תתי קטעים בצורת LIFO.
3. passage\_Done() - נקרא כאשר הקטע מסיים את הביצוע והסיפור נכנס למצב Idle. כל פלט המעבר זמין.
4. passage\_link\_Done() - נקרא לאחר השלמת פעולת הקישור ולפני הכניסה לקטע הבא (אם צוין).
5. passage\_Update() - כאשר הסיפור במצב Idle, Cue זה נקרא פעם אחת בכל פריים.
6. passage\_Output(StoryOutput output) - בכל פעם שקטע מייצר פלט (טקסט, קישורים וכו '), cue זה מקבל אותו.

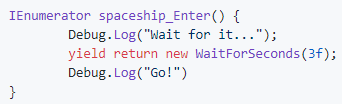
אם ברצונך לצרף cue לקטע עם שם המכיל רווחים או תווים אחרים שאינם מורשים ב- C #, אתה יכול לקשט את המטודה שלך עם attribute בצורה הבאה:



הערות:  
- ניתן שיהיה מספר תכונות של StoryCues באותה מטודה.  
- התכונה של StoryCue מקבלת עדיפות על שם המטודה, כך שאם קיימת תכונה עם אותו שם, מתעלמים משם של המטודה, גם אם זה נראה כמו שם של cue חוקי.

**Coroutine cues:**

Cue היא מטודה מסוג enumerated (מחזירה את IEnumerator ב- C # או כולל yield statement ב- UnityScript) הוא משמש להפעלת קורוטין. Cues של קורוטין מתנהגים ממש כמו קורוטין רגיל ביוניטי.



**הערות:**- כל ה-cues יכולים להיות corutines חוץ מ- update cues שצריחות להחזיר void .  
- אחרי ה-yeild הראשון הסיפור יהיה במצב idle וכל הפלט של הסיפור יהיה זמין. זה משום הקטע ממשיך לרוץ אחרי קריאת ה-cue אז עד שה-coroutine סיים לחכות – הקטע נגמר. על מנת להשהות ריצה עד שה-coroutine סיים יש להשתמש ב- Pause() and Resume().

**הרחבה**

ניתן להרחיב את cradle על מנת שיכלול macros ו- var typesשלא קיימים בפורמט המקורי.

**Runtime macros:**

Runtime macros הם ההרחבה הכי פשוטה שקיימת בcradle, היא פשוט פונקציה שניתן לקרוא לה בתוך קטע. זה לא יכול לייצר פלט סיפור נוסף או להשפיע על שטף הקטעים, אבל זה יכול לעורר טריגר פונקציונליות ספציפית ליוניטי בנקודות מדויקות בסיפור שלך.

איך עושים זאת?

1. יוצרים סקריפט של C#.
2. במקום לירוש את MonoBehaviour אנו צריכים לירוש את Cradle.RuntimeMacros.
3. כדי לחשוף שיטה כמאקרו בזמן ריצה, פשוט מקשטים אותה בתכונה [RuntimeMacro]. אם רוצים ששמו של המאקרו כפי שנכתב ב- Twine יהיה שונה משם השיטה C #, פשוט מוסיפים את השם לתכונה: [RuntimeMacro ("sfx")]
4. מייבאים את הסיפור.

דוגמא לשימוש עם audioSource:



הערות:

* כדי לגשת לרכיב ה- Story מתוך מאקרו, השתמש פשוט this.Story.
* אם ברצונך להוסיף מאפיינים שניתן להקצות מהעורך, מומלץ להעביר את הקריאה לסקריפט רגיל של MonoBehaviour המצורף לאותו GameObject כמו רכיב ה- Story שלך.   
  לדוגמה, this.Story.SendMessage("PlaySound", soundName) יעביר את המאקרו לכל סקריפט המצורף לאותו GameObject, שם ניתן להגדיר / להקצות ערכים.
* מופע של class זו נוצר פעם אחת בכל סיפור. אז כל משתני member יתקיימו לאורך כל חיי רכיב הסיפור שלך.
* כשמפעלים בדפדפן, פורמטי הסיפור של SugarCube / Sugarcane עשויים לזרוק שגיאה אם ​​נתקלu בפונקציה לא מזוהה. הדרך הקלה ביותר להימנע מכך היא ליצור פונקציית JavaScript מותאמת אישית של דמה שתמנע את השגיאה. דוגמה (הוסף את זה ב- JavaScript של הסיפור שלך):



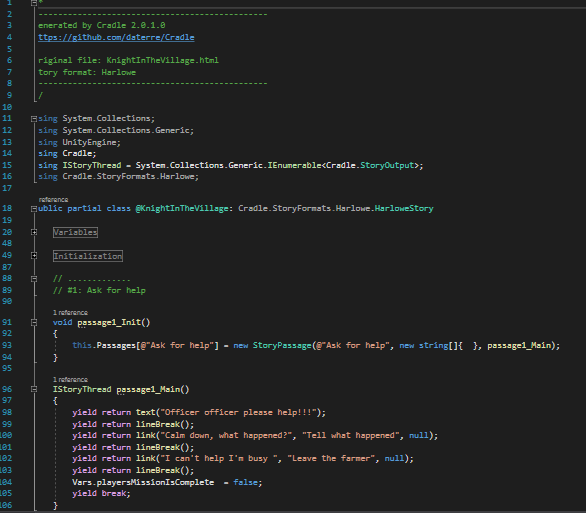
**מדריך צעד אחר צעד:**

**אחרי שהבנו את כל המושגים, זהו מדריך מקוצר איך להשתמש בפועל בתוסף.**

**לצורך ההדגמה אנחנו משתמשים בסיפור twine שנקרא Night In The Village.**

**קישור לסקריפט של הסיפור twine בגיטהאב:**

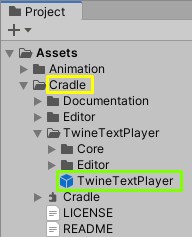
<https://github.com/ChenOst/twine-to-unity/blob/master/Class%20Project/Assets/Story/KnightInTheVillage.cs>



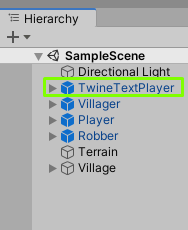
1. מורידים את תוסף cradle כפי שהוסבר בהתחלה.
2. לאחר ההורדה תופיע לנו תיקיית Cradle בתוך ה Assets.
3. מייבאים את סיפור הTwine שאנו רוצים להשתמש בו.



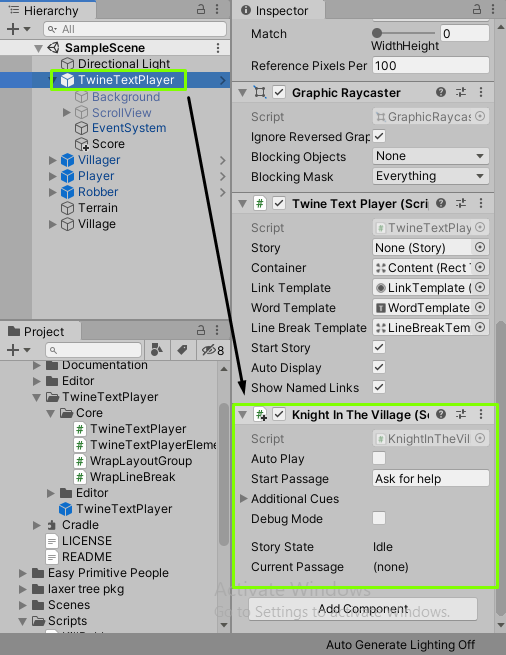
1. אחרי שיש לנו את הכל – ניתן לראות שיש לנו את ה-prefab TwineTextPlayer.

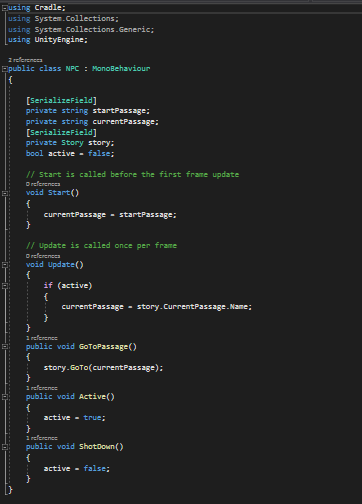
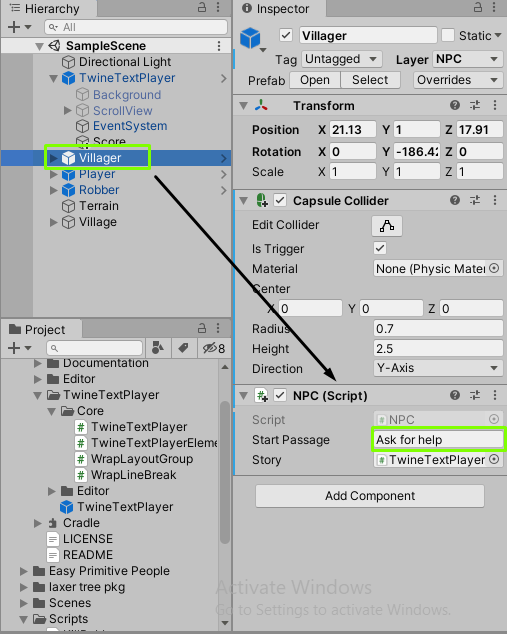


1. ניקח את ה prefab ונגרור אותו ל Hierarchy של המשחק שלנו.

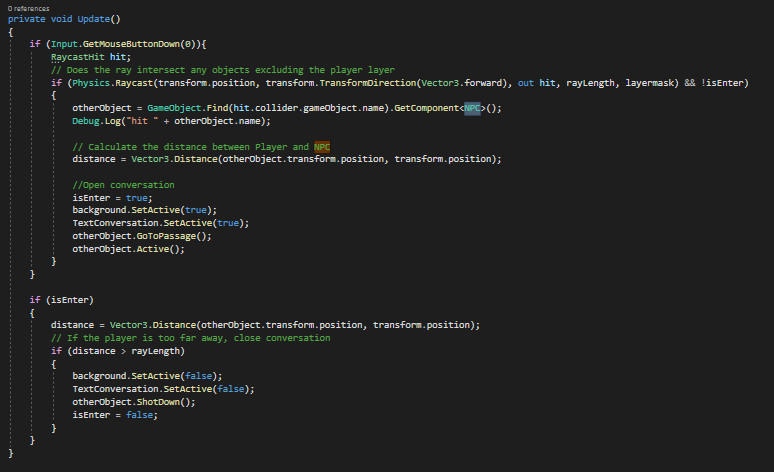


1. זהו כמעט יש לנו הכל על מנת להתחיל לבנות את המשחק, יש לנו את הסיפור ויש לנו את הסקריפט של twineTextPlayer.
2. ב-TwineTextPlayer מוסיפים את הסקריפט שנוצר של הסיפור שלנו.

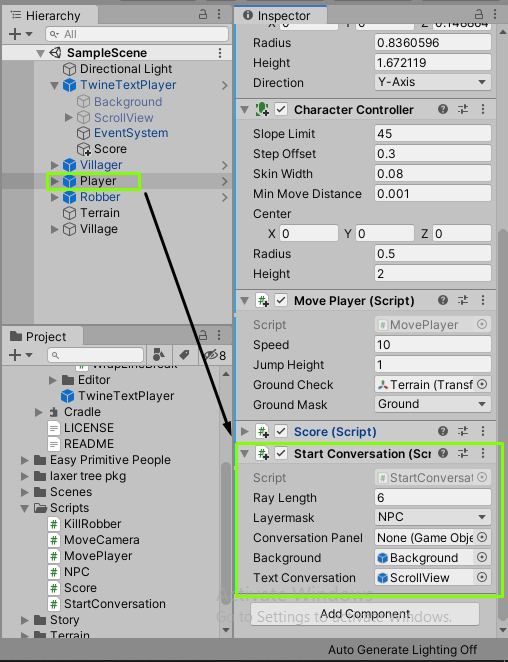


1. לצורך ההדגמה בנינו סקריפט NPC ששולט באיזה קטע אנו נמצאים בו.  
   קישור לסקריפט:  
   <https://github.com/ChenOst/twine-to-unity/blob/master/Class%20Project/Assets/Scripts/NPC.cs>  
    
2. מוסיפים את הסקריפט לאובייקט שאנו רוצים שיפעיל את הסיפור ומחליטים מאיזה קטע להתחיל את הסיפור.
3. בנינו סקריפט של startConversation שמחובר ל-player ובודק במי הוא פוגע (כלומר במי הוא לוחץ עם העכבר כאשר הוא בקירבתו). מחשב את המרחק בין האובייקט לשחקן (כדי לבדוק אם הוא לא התרחק). מתחיל את השיחה וכאשר מתרחק הוא סוגר את השיחה.

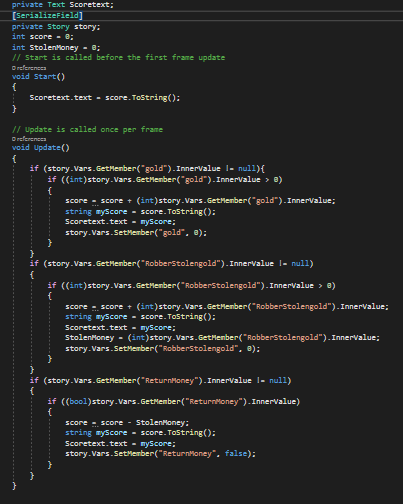
קישור לסקריפט:  
<https://github.com/ChenOst/twine-to-unity/blob/master/Class%20Project/Assets/Scripts/StartConversation.cs>



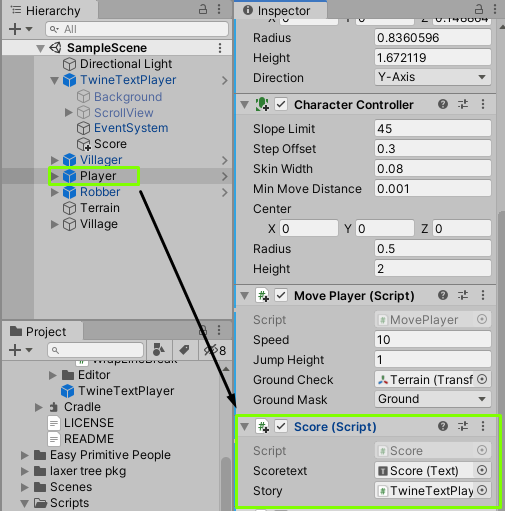
1. מחברים את הסקריפט הזה לשחקן שלנו.



1. וזהו מפה ניתן רק להוסיף ולשחק עם האופציות עד שמסופקים מהתוצאה.
2. עוד דוגמא עם מה שניתן לעשות הוא לקחת משתנים מהסיפור כמו score לפי members מהסיפור ולקבל את הערכים שלהם על מנת להציג אותם בtext על המסך (לדוגמא הסיפוא הנ"ל זה score שמיוצג על ידי הזהב).   
   קישור לסקריפט:  
   <https://github.com/ChenOst/twine-to-unity/blob/master/Class%20Project/Assets/Scripts/Score.cs>



1. אחרי זה פשוט מחברים את הסקריפט לשחקן על מנת שישמור את הscore.



# ביבליוגרפיה

<https://github.com/daterre/Cradle>