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

Latest release

v2.0.1

459aa57

Compare

Cradle 2.0.1

daterre released this on Jun 21, 2017 · 17 commits to master since this release

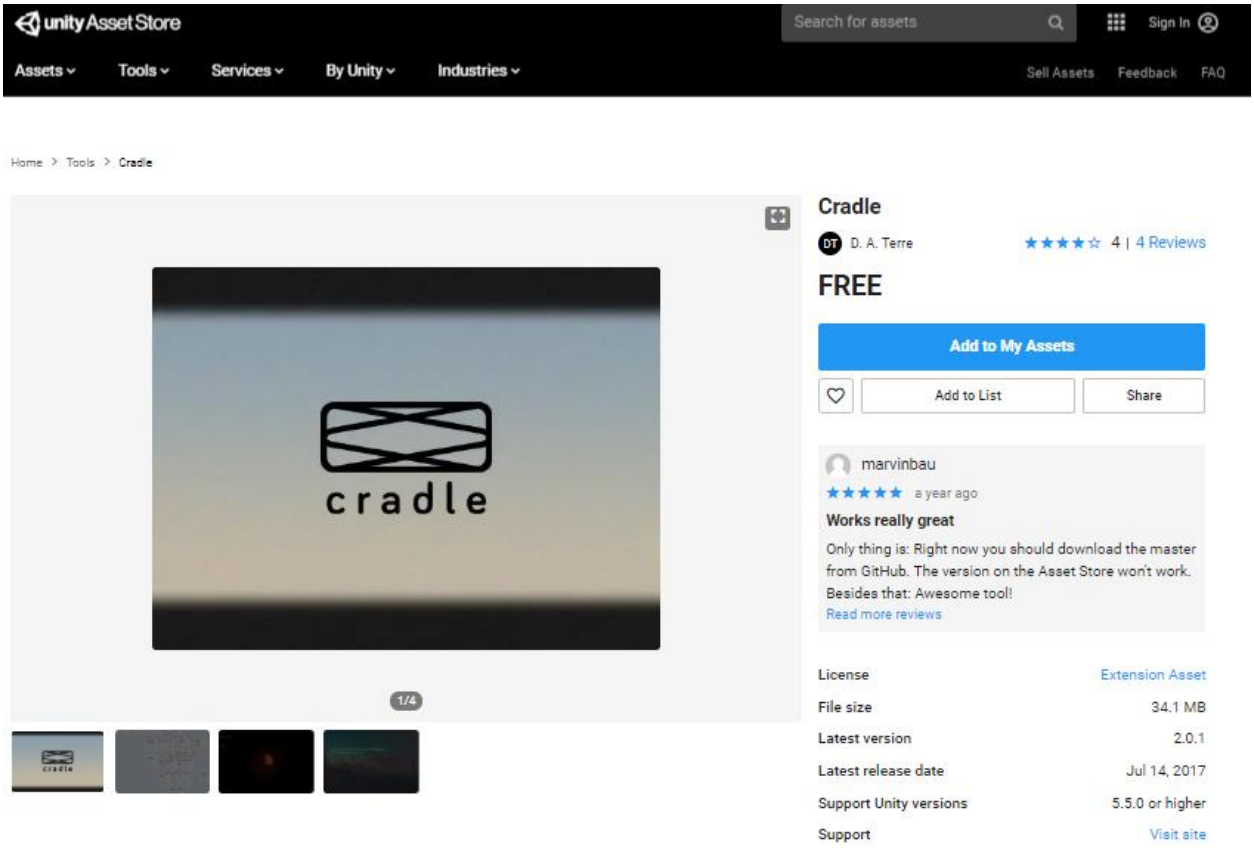
- Renamed the project to Cradle
- Support for published HTML files, including Twine 2
- Story formats: added Harlowe, improved Sugarcane/Sugarcube
- Complete rewrite of the asset importer:
  - Extensible, supports multiple story formats
  - Error checking and graceful failures that don't break the project
- Complete rewrite of the variable system to allow support for many different types (arrays, datasets, etc.)
- Added support for passage fragments to allow deferred code execution and output generation (Harlowe makes extensive use of this feature)
- Source code now compiles to standlone assemblies

Assets 3

Cradle.v2.0.1.unitypackage	33.7 MB
Source code (zip)	
Source code (tar.gz)	

2. 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 ב twine 1 וגם ב twine 2 .

Playback

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

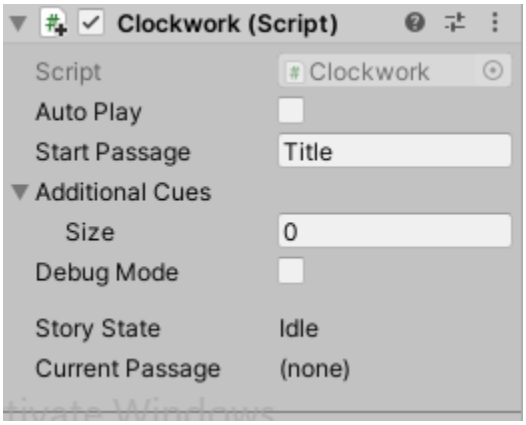
כל סקריפטי הסיפורים כוללים את מאפייני העורך הבאים:

AutoPlay: כשהפרמטר הזה הוא true אז הוא מתחיל לנגן את הסיפור באופן 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.

```
0 references
void Start () {
    if (!Application.isPlaying)
        return;

    LinkTemplate.gameObject.SetActive(false);
    ((RectTransform)LinkTemplate.transform).SetParent(null);
    LinkTemplate.transform.hideFlags = HideFlags.HideInHierarchy;

    WordTemplate.gameObject.SetActive(false);
    WordTemplate.rectTransform.SetParent(null);
    WordTemplate.rectTransform.hideFlags = HideFlags.HideInHierarchy;

    LineBreakTemplate.gameObject.SetActive(false);
    LineBreakTemplate.SetParent(null);
    LineBreakTemplate.hideFlags = HideFlags.HideInHierarchy;

    if (this.Story == null)
        this.Story = this.GetComponent<Story>();
    if (this.Story == null)
    {
        Debug.LogError("Text player does not have a story to play. Add a story
        return;
    }

    this.Story.OnPassageEnter += Story_OnPassageEnter;
    this.Story.OnOutput += Story_OnOutput;
    this.Story.OnOutputRemoved += Story_OnOutputRemoved;

    if (StartStory)
        this.Story.Begin();
}
```

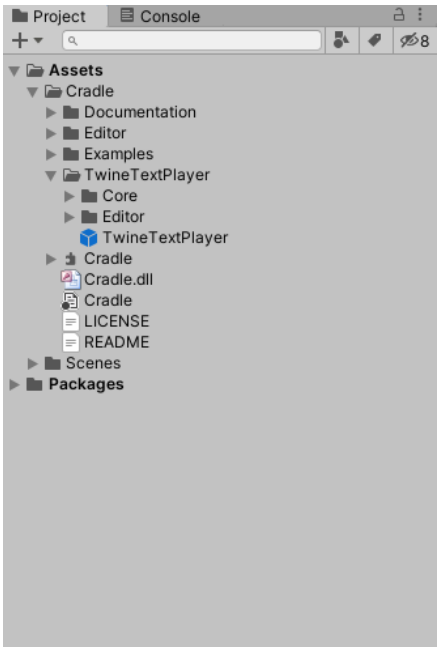
סקריפט המפעיל את הסיפור, מוציא ממנו מידע ומציג אותו על ה Canvas.

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 ה container 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 הAssets .

התיקייה TwineTextPlayer מכילה prefab הכולל בתוכו script מובן 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:

```
public Story story;

void Start() {
        story.Begin();
}

void Update() {
        // You'd want your script to check a few things before doing this, but hey this is an example
        if (Input.GetMouseButtonDown(0))
                story.DoLink("myLink");
}
```

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:

```
public Story story;

void Start() {
        story.OnOutput += story_OnOutput;
        story.Begin();
}

void story_OnOutput(StoryOutput output) {
        // Do something with the output here
        Debug.Log(output.Text);
}
```

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 באשר 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:

```
[[Visit your grandmother|grandma]]
```

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:

```
[[visitGrandma = Visit your grandmother|grandma]]
```

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:

```
public Story story;

void Update() {
        if (story.Vars["ammo"] > 10) {
                Debug.Log("Ammo limited to 10");
                story.Vars["ammo"] = 10;
        }
}
```

או דרך המשתנה שנוצר ישירות:

```
public JimsAdventure story; // generated class name from the file JimsAdventure.twee

void Update() {
        if (story.Vars.ammo > 10) {
                Debug.Log("Ammo limited to 10");
                story.Vars.ammo = 10;
        }
}
```

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:

```
public Story story;

void Start() {
        story.OnStateChanged += story_OnStateChanged;
        story.Begin();
}

void story_OnStateChanged() {
        if (story.State == StoryState.Idle) {
                // Interaction methods can be called now
                story.DoLink("enterTheCastle");
        }
}
```

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:

```
publicStory story;
public Sprite blackOverlay;

const float fadeInTime = 2f;

IEnumerator castle_Enter() {
        story.Pause();

        blackOverlay.color = new Color(0f, 0f, 0f, 1f);

        for (float t = 0; t <= fadeInTime; t+=Time.deltaTime) {
                // Update the alpha of the sprite
                float alpha = 1f - Mathf.Clamp(t/fadeInTime, 0f, 1f);
                blackOverlay.color = new Color(0f, 0f, 0f, alpha);

                // Wait a frame
                yield return null;
        }

        story.Resume();
}
```

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:

```
bool shieldsUp;

void Attack_Enter()
{
        Camera.main.backgroundColor = Color.blue;
}

void Defend_Enter()
{
        Camera.main.backgroundColor = Color.red;
        shieldsUp = true;
}

void Defend_Update()
{
        // Runs every frame like a normal Update method,
        // but only when the current passage is Defend
}

void Defend_Exit()
{
        shieldsUp = false;
}
```

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:

```
using Cradle;

// Specifies that this method is an Enter cue for the passage named "A large empty yard"
[StoryCue("A large empty yard", "Enter")]
void enterYardCutscene()
{
        // ...
}
```

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.

```
IEnumerator spaceship_Enter() {
        Debug.Log("Wait for it...");
        yield return new WaitForSeconds(3f);
        Debug.Log("Go!")
}
```

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 types שלא 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:

```
using UnityEngine;

public SoundEffectsMacros: Cradle.RuntimeMacros
{
        [RuntimeMacro]
        public void sfxPlay(string soundName)
        {
                GameObject.Find(soundName).GetComponent<AudioSource>().Play();
        }

        [RuntimeMacro]
        public void sfxStop(string soundName)
        {
                GameObject.Find(soundName).GetComponent<AudioSource>().Stop();
        }
}
```

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 Sugarcane / SugarCube עשויים לזרוק שגיאה אם נתקלו 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):

```
window.sfxPlay = function() {};
window.sfxStop = function() {};
```

מדריך צעד אחר צעד:

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

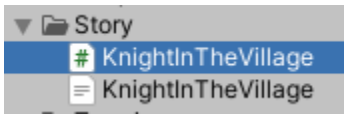
לצורך ההדגמה אנחנו משתמשים בסיפור twine שנקרא Night In The Village.

קישור לסקריפט של הסיפור twine בגיטהאב:

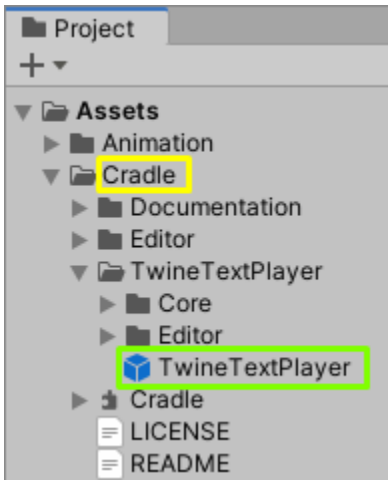
https://github.com/ChenOst/twine-to-unity/blob/master/Class%20Project/Assets/Story/KnightInTheVillage.cs

```
1   /*
2   ----------------------------------------------
3   enerated by Cradle 2.0.1.0
4   ttps://github.com/daterre/Cradle
5
6   riginal file: KnightInTheVillage.html
7   tory format: Harlowe
8   ----------------------------------------------
9   */
10
11  sing System.Collections;
12  sing System.Collections.Generic;
13  sing UnityEngine;
14  sing Cradle;
15  sing IStoryThread = System.Collections.Generic.IEnumerable<Cradle.StoryOutput>;
16  sing Cradle.StoryFormats.Harlowe;
17
    reference
18  ublic partial class @KnightInTheVillage: Cradle.StoryFormats.Harlowe.HarloweStory
19
20      Variables
48
49      Initialization
87
88      // ------------
89      // #1: Ask for help
90
        1 reference
91      void passage1_Init()
92      {
93          this.Passages[@"Ask for help"] = new StoryPassage(@"Ask for help", new string[]{  }, passage1_Main);
94      }
95
        1 reference
96      IStoryThread passage1_Main()
97      {
98          yield return text("Officer officer please help!!!");
99          yield return lineBreak();
100         yield return link("Calm down, what happened?", "Tell what happened", null);
101         yield return lineBreak();
102         yield return link("I can't help I'm busy ", "Leave the farmer", null);
103         yield return lineBreak();
104         Vars.playersMissionIsComplete  = false;
105         yield break;
106     }
```

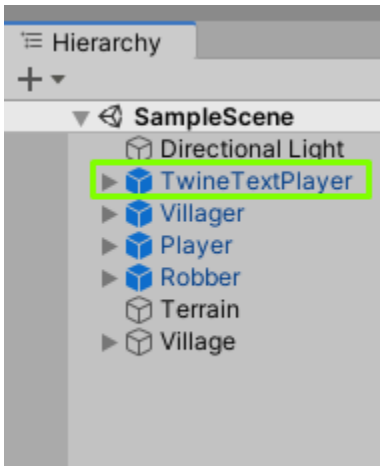
1. מורידים את תוסף cradle כפי שהוסבר בהתחלה.
2. לאחר ההורדה תופיע לנו תיקיית Cradle בתוך ה Assets.
3. מייבאים את סיפור ה Twine שאנו רוצים להשתמש בו.



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

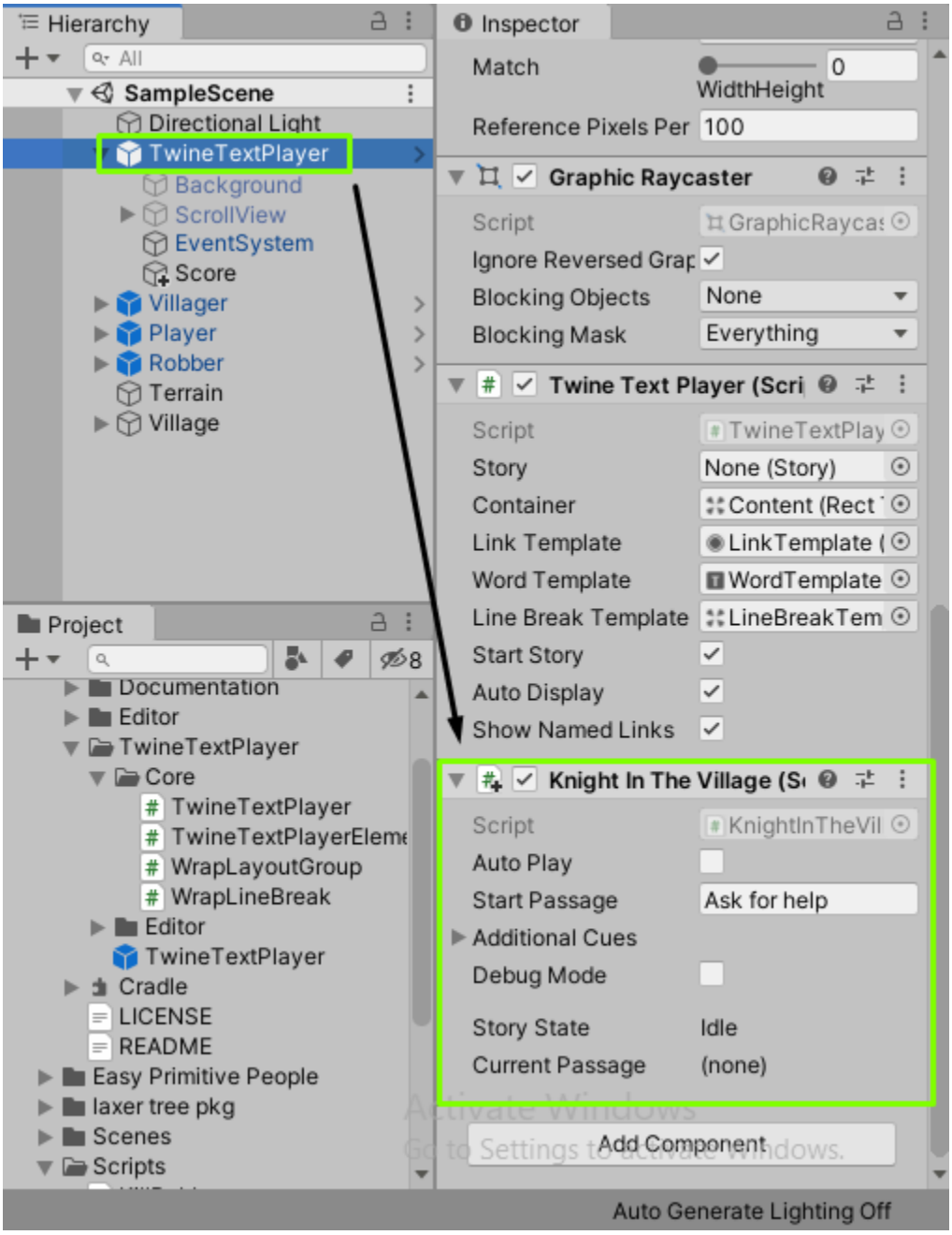


5. ניקח את ה prefab ונגרור אותו ל Hierarchy של המשחק שלנו.



6. זהו כמעט יש לנו הכל על מנת להתחיל לבנות את המשחק, יש לנו את הסיפור ויש לנו את הסקריפט של twineTextPlayer.

7. ב-TwineTextPlayer מוסיפים את הסקריפט שנוצר של הסיפור שלנו.



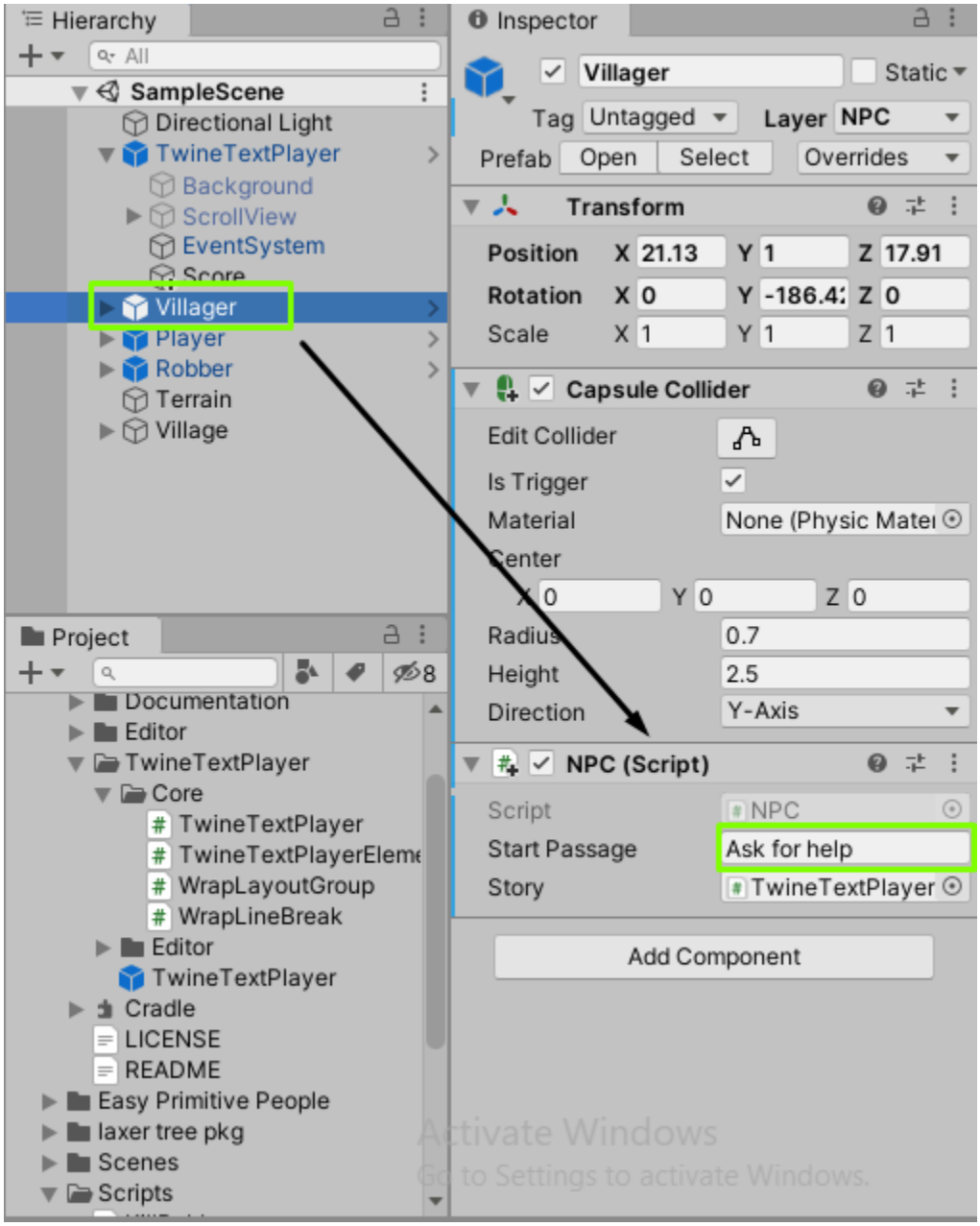
8. לצורך ההדגמה בנינו סקריפט NPC ששולט באיזה קטע אנו נמצאים בו.

קישור לסקריפט:

https://github.com/ChenOst/twine-to-unity/blob/master/Class%20Project/Assets/Scripts/NPC.cs

```
using Cradle;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

2 references
public class NPC : MonoBehaviour
{

    [SerializeField]
    private string startPassage;
    private string currentPassage;
    [SerializeField]
    private Story story;
    bool active = false;

    // Start is called before the first frame update
    0 references
    void Start()
    {
        currentPassage = startPassage;
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        if (active)
        {
            currentPassage = story.CurrentPassage.Name;
        }
    }
    1 reference
    public void GoToPassage()
    {
        story.GoTo(currentPassage);
    }
    1 reference
    public void Active()
    {
        active = true;
    }
    1 reference
    public void ShotDown()
    {
        active = false;
    }
}
```

9. מוסיפים את הסקריפט לאובייקט שאנו רוצים שיפעיל את הסיפור ומחליטים מאיזה קטע להתחיל את הסיפור.



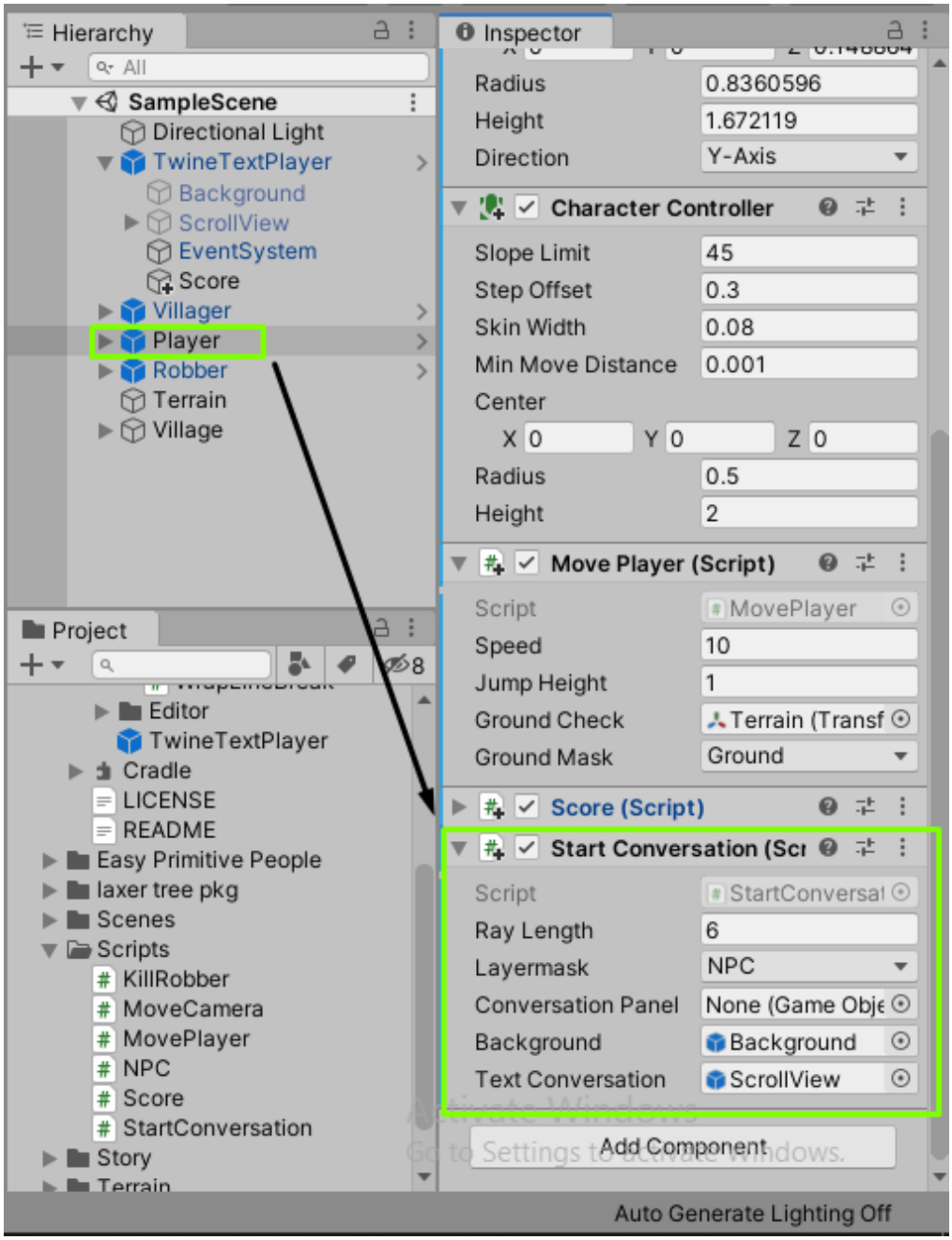
10. בנינו סקריפט של startConversation שמחובר ל-player ובודק במי הוא פוגע (כלומר במי הוא לוחץ עם העכבר כאשר הוא בקירבתו). מחשב את המרחק בין האובייקט לשחקן (כדי לבדוק אם הוא לא התרחק). מתחיל את השיחה וכאשר מתרחק הוא סוגר את השיחה.

קישור לסקריפט:

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

```
0 references
private void Update()
{
    if (Input.GetMouseButtonDown(0)){
        RaycastHit hit;
        // Does the ray intersect any objects excluding the player layer
        if (Physics.Raycast(transform.position, transform.TransformDirection(Vector3.forward), out hit, rayLength, layermask) && !isEnter)
        {
            otherObject = GameObject.Find(hit.collider.gameObject.name).GetComponent<NPC>();
            Debug.Log("hit " + otherObject.name);

            // Calculate the distance between Player and NPC
            distance = Vector3.Distance(otherObject.transform.position, transform.position);

            //Open conversation
            isEnter = true;
            background.SetActive(true);
            TextConversation.SetActive(true);
            otherObject.GoToPassage();
            otherObject.Active();
        }
    }

    if (isEnter)
    {
        distance = Vector3.Distance(otherObject.transform.position, transform.position);
        // If the player is too far away, close conversation
        if (distance > rayLength)
        {
            background.SetActive(false);
            TextConversation.SetActive(false);
            otherObject.ShotDown();
            isEnter = false;
        }
    }
}
```

11. מחברים את הסקריפט הזה לשחקן שלנו.



12. וזהו מפה ניתן רק להוסיף ולשחק עם האופציות עד שמסופקים מהתוצאה.

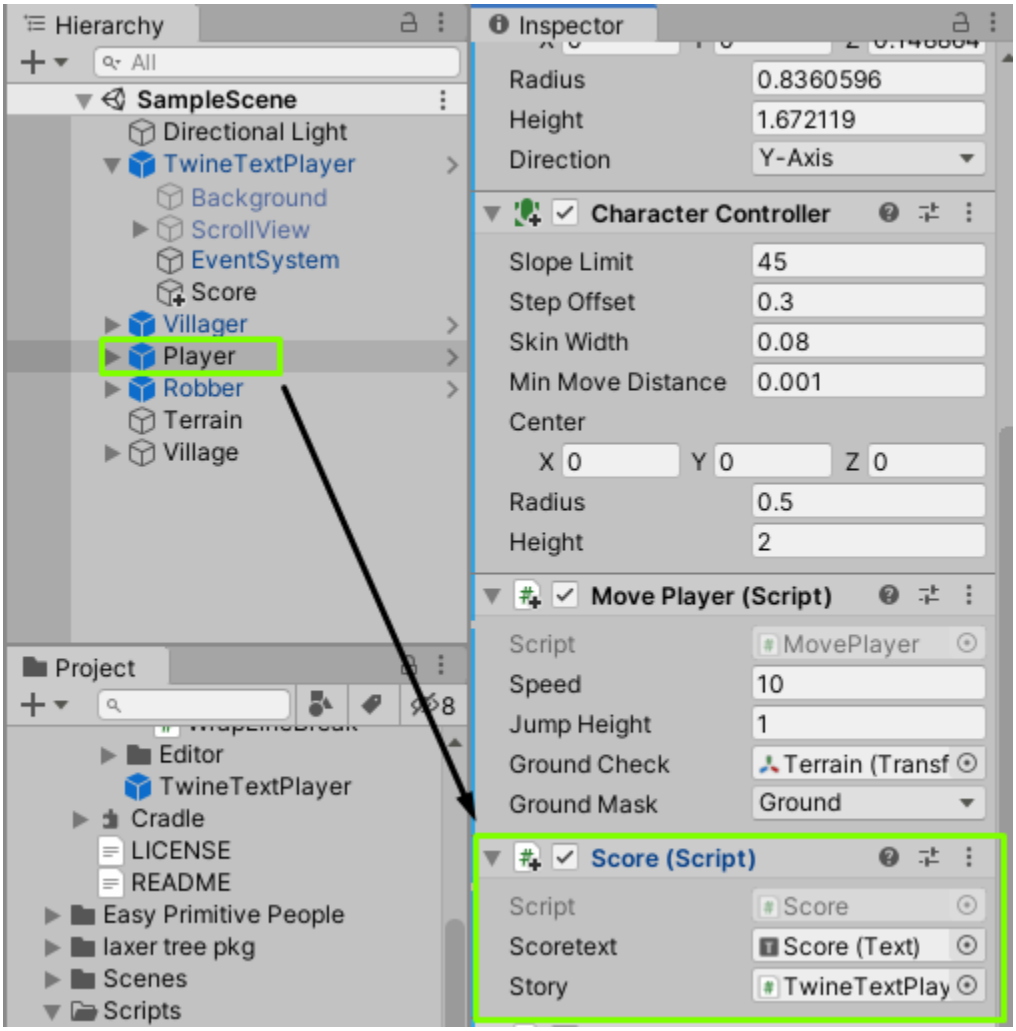
13. עוד דוגמא עם מה שניתן לעשות הוא לקחת משתנים מהסיפור כמו score לפי members מהסיפור ולקבל את הערכים שלהם על מנת להציג אותם בtext על המסך (לדוגמא הסיפוא הנ"ל זה score שמיוצג על ידי הזהב).

קישור לסקריפט:

https://github.com/ChenOst/twine-to-unity/blob/master/Class%20Project/Assets/Scripts/Score.cs

```
private Text Scoretext;
[SerializeField]
private Story story;
int score = 0;
int StolenMoney = 0;
// Start is called before the first frame update
0 references
void Start()
{
    Scoretext.text = score.ToString();
}

// Update is called once per frame
0 references
void Update()
{
    if (story.Vars.GetMember("gold").InnerValue != null){
        if ((int)story.Vars.GetMember("gold").InnerValue > 0)
        {
            score = score + (int)story.Vars.GetMember("gold").InnerValue;
            string myScore = score.ToString();
            Scoretext.text = myScore;
            story.Vars.SetMember("gold", 0);
        }
    }
    if (story.Vars.GetMember("RobberStolengold").InnerValue != null)
    {
        if ((int)story.Vars.GetMember("RobberStolengold").InnerValue > 0)
        {
            score = score + (int)story.Vars.GetMember("RobberStolengold").InnerValue;
            string myScore = score.ToString();
            Scoretext.text = myScore;
            StolenMoney = (int)story.Vars.GetMember("RobberStolengold").InnerValue;
            story.Vars.SetMember("RobberStolengold", 0);
        }
    }
    if (story.Vars.GetMember("ReturnMoney").InnerValue != null)
    {
        if ((bool)story.Vars.GetMember("ReturnMoney").InnerValue)
        {
            score = score - StolenMoney;
            string myScore = score.ToString();
            Scoretext.text = myScore;
            story.Vars.SetMember("ReturnMoney", false);
        }
    }
}
```

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



ביבליוגרפיה

https://github.com/daterre/Cradle