עד השלב הזה של בניית המשחק השגנו כלים בסיסיים ליצירת חפיסת קלפים שיכולה לשמש אותנו למשחק שונים.
החל מעכשיו נתעסק במשחק הספציפי שאנחנו רוצים לבנות.

במהלך הקובץ הקרוב, נראה איך לבנות משחק סוליטר ייחודי *Prospector* המבוסס על משחק הסוליטר המוכר כ- *Tri-Peack*.
למי שלא מכיר את החוקים להלן סרטון קצר המסביר בצורה הטובה ביותר את חוקי המשחק:

https://www.youtube.com/watch?time_continue=3&v=sqpykUtzpWg&feature=emb_logo

החוקים זהים למעט שני דברים:

- הנחת היסוד של *Prospector* היא שהשחקן חופר בשביל להשיג "זהב"
- מטרת המשחק כאן, בניגוד לסוליטר רגיל ששם המטרה לנקות את השולחן מקלפים, היא להרוויח כמה שיותר נקודות ע"י שרשרת של רצף קלפים כמה שיותר ארוכה של קלפים, ובכל קלף זהב בשרשרת מכפיל את הערך של כל השרשרת.

להמחשה הנה סרטון שיציג את התוצאה הסופית של המשחק שלנו:

https://www.youtube.com/watch?v=sTRW3ad3J9E

לפני שתתחילו לבנות את המשחק ,כדאי לכם לנסות לשחק אותו בעצמכם- קחו חפיסה וסדרו אותה כמתואר.
אם אתם מתעצלים לעשות את זה (משום מה), יש גם אלטרנטיבה לזה:

/http://mhgamedev.com/ProspectorSolitaire

ייקח למשחק כמה שניות להטען, ואח"כ תוכלו להתנסות במשחק.

## יישום המשחק בקוד-

אם התנסתם במשחק כפי שראיתם המשחק *Prospector* הוא משחק דיי פשוט, וגם דיי מהנה. נוכל אח"כ להוסיף למשחק אפילו עוד אלמנטים כדי להפוך אותו למהנה אפילו יותר. אך לפני כן, בואו נתחיל בבניית בסיס למשחק.

### הגדרת מחלקת Layout-

1. ב-unity , פתחו את *Layout.xml* בתיקיית resource כדי לראות את המידע אודות פריסת העמוד.
יש לשים לב שבקבציי html-ו xml הערות נכתבות בצורה הזאת: <!-- ... --> (השלוש נקודות באמצע הן מה שכתוב

בהערה, ולא חלק מהמבנה של כתיבת הערה).

```
<xml>
        <!-- This file holds info for laying out the Prospector card game. -->

        <!-- The multiplier is multiplied by the x and y attributes below. -->
        <!-- This determines how loose or tight the layout is. -->
        <multiplier x="1.25" y="1.5" />

        <!-- In the XML below, id is the number of the card -->
        <!-- x and y set position -->
        <!-- faceup is 1 if the card is face-up -->
        <!-- layer sets the depth layer so cards overlap properly -->
        <!-- hiddenby is the ids of cards that keep a card face-down -->

        <!-- Layer0, the deepest cards. -->
        <slot id="0" x="-6" y="-5" faceup="0" layer="0" hiddenby="3,4" />
        <slot id="1" x="0"  y="-5" faceup="0" layer="0" hiddenby="5,6" />
        <slot id="2" x="6"  y="-5" faceup="0" layer="0" hiddenby="7,8" />

        <!-- Layer1, the next level. -->
        <slot id="3" x="-7" y="-4" faceup="0" layer="1" hiddenby="9,10" />
        <slot id="4" x="-5" y="-4" faceup="0" layer="1" hiddenby="10,11" />
        <slot id="5" x="-1" y="-4" faceup="0" layer="1" hiddenby="12,13" />
        <slot id="6" x="1"  y="-4" faceup="0" layer="1" hiddenby="13,14" />
        <slot id="7" x="5"  y="-4" faceup="0" layer="1" hiddenby="15,16" />
        <slot id="8" x="7"  y="-4" faceup="0" layer="1" hiddenby="16,17" />


        <!-- Layer2, the next level. -->
        <slot id="9"  x="-8" y="-3" faceup="0" layer="2" hiddenby="18,19" />
        <slot id="10" x="-6" y="-3" faceup="0" layer="2" hiddenby="19,20" />
        <slot id="11" x="-4" y="-3" faceup="0" layer="2" hiddenby="20,21" />
        <slot id="12" x="-2" y="-3" faceup="0" layer="2" hiddenby="21,22" />
        <slot id="13" x="0"  y="-3" faceup="0" layer="2" hiddenby="22,23" />
        <slot id="14" x="2"  y="-3" faceup="0" layer="2" hiddenby="23,24" />
        <slot id="15" x="4"  y="-3" faceup="0" layer="2" hiddenby="24,25" />
        <slot id="16" x="6"  y="-3" faceup="0" layer="2" hiddenby="25,26" />
        <slot id="17" x="8"  y="-3" faceup="0" layer="2" hiddenby="26,27" />


        <!-- Layer3, the top level. -->
        <slot id="18faceup="1" layer="3" />
        <slot id="19" x="-7" y="-2"" x="-9" y="-2"  faceup="1" layer="3" />
        <slot id="20" x="-5" y="-2" faceup="1" layer="3" />
        <slot id="21" x="-3" y="-2" faceup="1" layer="3" />
        <slot id="22" x="-1" y="-2" faceup="1" layer="3" />
        <slot id="23" x="1"  y="-2" faceup="1" layer="3" />
        <slot id="24" x="3"  y="-2" faceup="1" layer="3" />
        <slot id="25" x="5"  y="-2" faceup="1" layer="3" />
        <slot id="26" x="7"  y="-2" faceup="1" layer="3" />
        <slot id="27" x="9"  y="-2" faceup="1" layer="3" />

        <!-- This positions the draw pile and staggers it -->
        <slot type="drawpile" x="6" y="4" xstagger="0.15" layer="4"/>

        <!-- This positions the discard pile and target card -->
        <slot type="discardpile" x="0" y="1" layer="5"/>

<xml/>
```

כפי שניתן לראות , יש כאן מידע על הפריסה של כל אחד מהקלפים במבנה המשחק(שבנויים מ- <slot>-ים בלי תכונה type) למעט שני slotים שלהם כן יש סוג: drawpile ו-discardpile .

2. בואו נכתוב קוד כדי לפרסר את ה-LayoutXML לכדי מידע שימושי. צרו סקריפט חדש עם השם *Layout* בתיקייה היעודית לכך, והזינו אתת הקוד הבא:

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

//The SlotDef is not a subclass of MonoBehavior' so it doesn't need
//a separate C# file

[System.Serializable]
public class SlotDef
{
    public float x;
    public float y;
    public bool faceUp = false;
    public string layerName = "Default";
    public int LayerID = 0;
    public int id;
    public List<int> hiddenBy = new List<int>();
    public string type = "slot";
    public Vector2 stagger;
}

public class Layout : MonoBehaviour
{
    public PT_XMLReader xmlr;//Just like Deck, this has a PT_XMLReader
    public PT_XMLHashtable xml;//This vairable is for faster xml access
    public Vector2 multiplier;//The offset of the tableau's center
    //SlotDef references
    public List<SlotDef> slotDefs;
    public SlotDef drawPile;
    public SlotDef discardPile;
    //This holds all the possible names for the layers set by layerID
    public string[] sortingLayerNames = new string[] { "Row0", "Row1", "Row2", "Row3", "Discard", "Draw" };

    //This function is called to read in the LayoutXML.xml file
    public void ReadLayout(string xmlText)
    {
        xmlr = new PT_XMLReader();
        xmlr.Parse(xmlText);
        xml = xmlr.xml["xml"][0];//xml is set as a shortcut to the XML

        //read the multiplier, which sets card specing
        multiplier.x = float.Parse(xml["multiplier"][0].att("x"));
        multiplier.y = float.Parse(xml["multiplier"][0].att("y"));

        //Read in slots
        SlotDef tSD;
        //slotsX is used as a shortcut to all the <slots>s
        PT_XMLHashList slotsX = xml["slot"];
        print("slotsX count is: "+slotsX.Count);
        for (int i = 0; i < slotsX.Count; i++)
        {
            tSD = new SlotDef();
            if (slotsX[i].HasAtt("type"))
            {
                //If this slot has a type attribute parse it
                tSD.type = slotsX[i].att("type");
            }
            else
            {
                //If not, set its tyoe to "slot"; it's a card in the row
                tSD.type = "slot";
            }
            //Various attributes are parsed into numerical values
            tSD.x = float.Parse(slotsX[i].att("x"));
            tSD.y = float.Parse(slotsX[i].att("y"));
            tSD.LayerID = int.Parse(slotsX[i].att("layer"));
            //This convert the number of the layerID into a text layerName
            tSD.layerName = sortingLayerNames[tSD.LayerID];

            switch (tSD.type)
            {
                //pull additional attribute based on the type of this <slot>
```

```
                case "slot":
                    tSD.faceUp = (slotsX[i].att("faceup") == "1");
                    tSD.id = int.Parse(slotsX[i].att("id"));
                    if (slotsX[i].HasAtt("hiddenby"))
                    {
                        string[] hiding = slotsX[i].att("hiddenby").Split(',');
                        foreach (string s in hiding)
                        {
                            tSD.hiddenBy.Add(int.Parse(s));
                        }
                    }
                    slotDefs.Add(tSD);
                    break;

                case "drawpile":
                    tSD.stagger.x = float.Parse(slotsX[i].att("xstagger"));
                    drawPile = tSD;
                    break;

                case "discardpile":
                    discardPile = tSD;
                    break;
            }
        }
    }
}
```

השדה layerName של SlotDef נועד כדי לוודא שהקלפים הנכונים נמצאים מעל לאחרים. בדו-ממד למעשה כל ה-assets נמצאים באותו הגובה על ציר ה-z, לכן אנחנו משתמשים בשכבות במקום כדי לבדיל בין מיקום של אובייקטים במרחב.
בשלב זה, רוב הסינטקס שהשתמשנו בו לא אמור להיות זר לנו לגמרי. המחלקה SlotDef נוצרה כדי לאחסן מידע מה XML בצורה יותר נגישה. ואז הגדרנו את המחלקה Layout והמתודה ReadLayout() משמשת כדי לקחת מחרוזות מה-XML ולאחסן אותן בסדרה של SlotDefs.

3. פתחו את המחלקה Prospector ושנו\ הוסיפו את השורות הבאות (מה שהדגשנו):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;//will be used later
using UnityEngine.UI; //will be used later

public class Prospector : MonoBehaviour
{
    static public Prospector S;
    [Header("Set in inspector")]
    public TextAsset deckXML;
    public TextAsset layoutXML;
    [Header("Set Dynamically")]
    public Deck deck;
    public Layout layout;

    void Awake()
    {
        S = this;//set up a singleton for prospector
    }
    void Start()
    {
        deck = GetComponent<Deck>();// get the Deck
        deck.InitDeck(deckXML.text);// pass DeckXML to it
        Deck.Shuffle(ref deck.cards);//this shuffles the deck by reference
        //Card c;
        //for(int cNum=0; cNum<deck.cards.Count; cNum++)
        //{
        //    c = deck.cards[cNum];
        //    c.transform.localPosition = new Vector3((cNum % 13) * 3, cNum / 13 * 4, 0);
        //}
        layout = GetComponent<Layout>();//Get the Layout component
        layout.ReadLayout(layoutXML.text);//Pass LayouutXML to it
```

```
    }
}
```

4. שמרו על כל הסקריפטים שעשינו וחזרו ל-unity.
5. ב-unity בחרו באובייקט המצלמה הראשית בחלון ההיררכיה. גררו את הסקריפט של Layout מחלון הפרויקט לאינספקטור של המצלמה. אתם אמורים להיות מסוגלים עכשיו לגלול למטה בחלון ולראות את הרכיב *Layout(Script)* .
6. חפשו את הרכיב *Prospector(Script)* באותו אינספקטור. עתה אנחנו אמורים לראות את השדות הציבוריים: layoutXML ו- layout . מחלון הפרוייקט, בתיקייה Resources גררו את LayoutXML למקום המתאים באינספקטור של המצלמה(ברכיב layoutXML היכן שה- TextAsset )
7. שמרו את הסצנה, והריצו את המשחק. אם תבחרו את המצלה, גללו למטה באינספקטור שלה עד שתגיעו לרכיב Layout נוכל לבדוק עכשיו את כל הערכים בשדה slotDefs ע"י לחיצה על המשולש הקטן מימן לשדה. אם נפתח אותו בזמן ריצת המשחק אנחנו אמורים לראות שכל ה-<slot>ים פורססו מה-XML.

## שימוש ב CardProspector–

לפני שנמקם את הקלפים על הלוח במבנה הייחודי למשחק, נצטרך להוסיף כמה פיצ'רים חדשים למחלקת Card שספציפיים למשחק שלנו. היות והמחלקות Card ו- Deck שיצרנו נוצרו תוך מחשבה לשימוש במשחקי קלפים עתידיים , אנחנו ניצור מחלקת CardProspector כתת מחלקה של Card ולא נשנה את המחלקה המקורית.

1. צרו סקריפט חדש בתיקייה _Scripts עם השם CardProspector והזינו לה את הקוד הבא:

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


//An enum defines a variable type with a few prenames values
public enum eCardState
{
    drawpile,
    tableau,
    target,
    discard
}
//Make sure CardProspector extends Card
public class CardProspector : Card
{
    [Header("Set Dynamically: CardProspector")]
    //This is how you use the enum eCardState
    public eCardState state = eCardState.drawpile;
    //The hiddenBy list stores which other cards will keep this one face down
    public List<CardProspector> hiddenBy = new List<CardProspector>();
    //The layoutID matches this card to the tableau card
    public int layoutID;
    //The SlotDef class stores information pulled in form the LayoutXML <slot>
    public SlotDef SlotDef;
}
```

זהו enum, שמגדיר את סוג של משתנה שיש לו רק כמה שמות ערכים אפשריים. המשתנה eCardState מכיל אחד מתוך ארבעה ערכים אפשריים: drawpile,tableau,target ו- discard שעוזרים לאינסטנסים של CarsProspector לעקוב היכן הם אמורים להיות במשחק.

הירושה ממחלקת Card במחלקה החדשה שיצרנו מאפשרת לנו לטפל בדברים שלא יכולנו לטפל בהם קודם לכן, כמו ארבעת המיקומים שבהם יכול קלף מסוים להיות במהלך המשחק- החפיסה שממנה מושכים (drawpile), אחד מ-28

הקלפים שב-"מכרה" (tableau), הקלפים המושלחים שהשתמשנו בהם כבר(discard) והקלף הפעיל שמעל לקלפים המשומשים(target).
מידע על ה-(slotDef) layout ומידע שקובע אם קלף אמור להיות עם הפנים למעלה או למטה(hiddenBy ו-layoutID).

עתה כתת המחלקה זמינה, אנחנו צריכים להמיר את הקלפים בחפיסה מ-Cards ל-CardProspector .

2. בשביל לעשות את זה הוסיפו את הקוד הבא למחלקת Prospector(מה שמודגש):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;//will be used later
using UnityEngine.UI; //will be used later

public class Prospector : MonoBehaviour
{
    static public Prospector S;
    [Header("Set in inspector")]
    public TextAsset deckXML;
    public TextAsset layoutXML;
    [Header("Set Dynamically")]
    public Deck deck;
    public Layout layout;
    public List<CardProspector> drawPile;

    void Awake()
    {
        S = this;//set up a singleton for prospector
    }
    void Start()
    {
        deck = GetComponent<Deck>();// get the Deck
        deck.InitDeck(deckXML.text);// pass DeckXML to it
        Deck.Shuffle(ref deck.cards);//this shuffles the deck by reference
        //Card c;
        //for(int cNum=0; cNum<deck.cards.Count; cNum++)
        //{
        //    c = deck.cards[cNum];
        //    c.transform.localPosition = new Vector3((cNum % 13) * 3, cNum / 13 * 4, 0);
        //}
        layout = GetComponent<Layout>();//Get the Layout component
        layout.ReadLayout(layoutXML.text);//Pass LayouutXML to it
        drawPile = Convert_List_Card_To_List_CradProspector(deck.cards);
    }

    List<CardProspector> Convert_List_Card_To_List_CradProspector(List<Card> lcd)
    {
        List<CardProspector> lcp = new List<CardProspector>();
        CardProspector tCP;
        foreach (Card tCD in lcd)
        {
            tCP = tCD as CardProspector;
            lcp.Add(tCP);
        }
        return lcp;
    }
}
```

המילה השמורה as מנסה להמיר את סוג האובייקט ל-CardProspector.
3. שמרו את הסקריפטים וחזרו ל-unity
4.הריצו את המשחק והסתכלו על השדה drawPile באינספקטור של המצלמה(ברכיב Prospector).

שימו לב שעכשיו כל הקלפים ב-drawPile הם null, זאת משום שמתי שאנחנו מנסים להתיחס ל Card tCD כאובייקט CardProspector המילה השמורה as מחזירה null במקום אובייקט Card שאומר ל-CardProspector . ככה מונחה עצמים

עובד ב-C# . הסיבה לכך היא די מובנת, אנחנו יכולים להתייחס לקלפי פרוספקטור כקלפים, שכן כל קלפי הפרוספקטור הם סוג של קלף , אבל אנחנו לא יכולים להתייחס לקלף כסוג של "קלף פרוספקטור" ,כי לא בטוח שהקלף הוא באמת קלף פרוספקטור, שכן לא כל הקלפים הם קלפים פרוספקטור. אם כך כיצד נוכל לתקן את הבעיה?

כדי לפתור את הבעיה נצטרך לוודא שה-CardProspector היה צמיד מאותו הסוג שהוא רק "מעמיד פנים" שהוא Card בכל הקוד במחלקת Deck.

5. בחרו ב-prefabCard בחלון הפרויקט. הוא מופיע באינספקטור עם רכיב Card(Script). גררו את הסקריפט CardProspector לחלון הפרוספקטור שלו כנדרש.
6. כדי למחוק את הרכיב Card(Script) הישן לחצו על האייקון הקטן של הגלגל שיניים מצד ימין למעלה ברכיב ובחרו *Remove component* מהתפריט.
7. בחר מצלמה והרץ את המשחק. עתה אנחנו אמורים לראות שכל השדות ב-drawPile מלאים ולא null כמו מקודם.

כאשר Deck מאתחל אובייקט PrefabCard ומקבל את הרכיב Card שלו, זה עובד מצויין משום ש CardProspector הוא סוג של Card. ומתי שאנחנו משתמשים במתודה שממירה את הקלפים לקלפי פרוספקטור, המתודה בעצם מורידה את המעטפת של האובייקט tCD מ-Card לאובייקט שהוא היה בהתחלה: CardProspector.

## מיקום הקלפים ב-Tableau -

עכשיו שהכל מוכן, זה הזמן להוסיף קצת קוד כדי לפרוס את העמוד כמו שצריך.
לכו למחלקת Prospector והוסיפו את הקוד הבא:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;//will be used later
using UnityEngine.UI; //will be used later

public class Prospector : MonoBehaviour
{
    static public Prospector S;
    [Header("Set in inspector")]
    public TextAsset deckXML;
    public TextAsset layoutXML;
    public float xoffset = 3;
    public float yoffset = -2.5f;
    public Vector3 layoutCenter;

    [Header("Set Dynamically")]
    public Deck deck;
    public Layout layout;
    public List<CardProspector> drawPile;
    public Transform layoutAnchor;
    public CardProspector target;
    public List<CardProspector> tableau;
    public List<CardProspector> discardPile;
    void Awake()
    {
        S = this;//set up a singleton for prospector
    }
    void Start()
    {
        deck = GetComponent<Deck>();// get the Deck
        deck.InitDeck(deckXML.text);// pass DeckXML to it
        Deck.Shuffle(ref deck.cards);//this shuffles the deck by reference
        //Card c;
        //for(int cNum=0; cNum<deck.cards.Count; cNum++)
        //{
        //    c = deck.cards[cNum];
        //    c.transform.localPosition = new Vector3((cNum % 13) * 3, cNum / 13 * 4, 0);
        //}
        layout = GetComponent<Layout>();//Get the Layout component
```

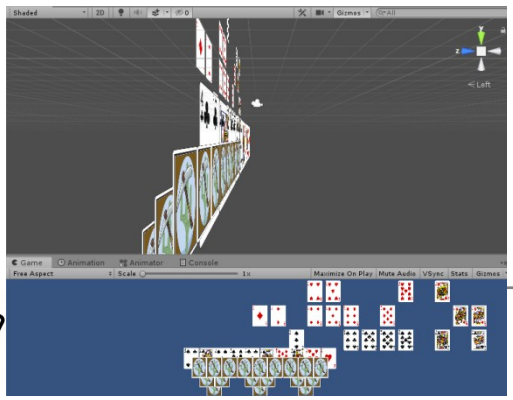
```
        layout.ReadLayout(layoutXML.text);//Pass LayouutXML to it
        drawPile = Convert_List_Card_To_List_CradProspector(deck.cards);
        LayoutGame();
    }

    List<CardProspector> Convert_List_Card_To_List_CradProspector(List<Card> lcd)
    {
        List<CardProspector> lcp = new List<CardProspector>();
        CardProspector tCP;
        foreach (Card tCD in lcd)
        {
            tCP = tCD as CardProspector;
            lcp.Add(tCP);
        }
        return lcp;
    }

    //The Draw function will pull a single card from the drawPile
    CardProspector Draw()
    {
        CardProspector cd = drawPile[0];
        drawPile.RemoveAt(0);
        return (cd);
    }

    //LayoutGame() positions the initial tableau of card, a.k.a the "mine"
    void LayoutGame()
    {
        //Create an empty object to serve as an anchor for the tableau
        if (layoutAnchor == null)
        {
            GameObject tGO = new GameObject("layoutAnchor");
            layoutAnchor = tGO.transform;
            layoutAnchor.transform.position = layoutCenter;
        }
        CardProspector cp;
        //Follow the layout
        foreach(SlotDef tSD in layout.slotDefs)
        {
            cp = Draw();
            cp.faceUp = tSD.faceUp;
            cp.transform.parent = layoutAnchor;
            //This replaces the previous parent; deck.deckAnchor which
            //appears as _Deck in the Hierarchy when the the scene is playing
            cp.transform.localPosition = new Vector3(layout.multiplier.x * tSD.x,
layout.multiplier.y * tSD.y, -tSD.LayerID);
            //Set the localPosition of the card based on slotDef
            cp.layoutID = tSD.id;
            cp.SlotDef = tSD;
            //CrdProsperctor in the tableau have the state CardState.tableau
            cp.state = eCardState.tableau;
            tableau.Add(cp);//Add this CradProspector to the List<> tableau
        }
    }
}
```

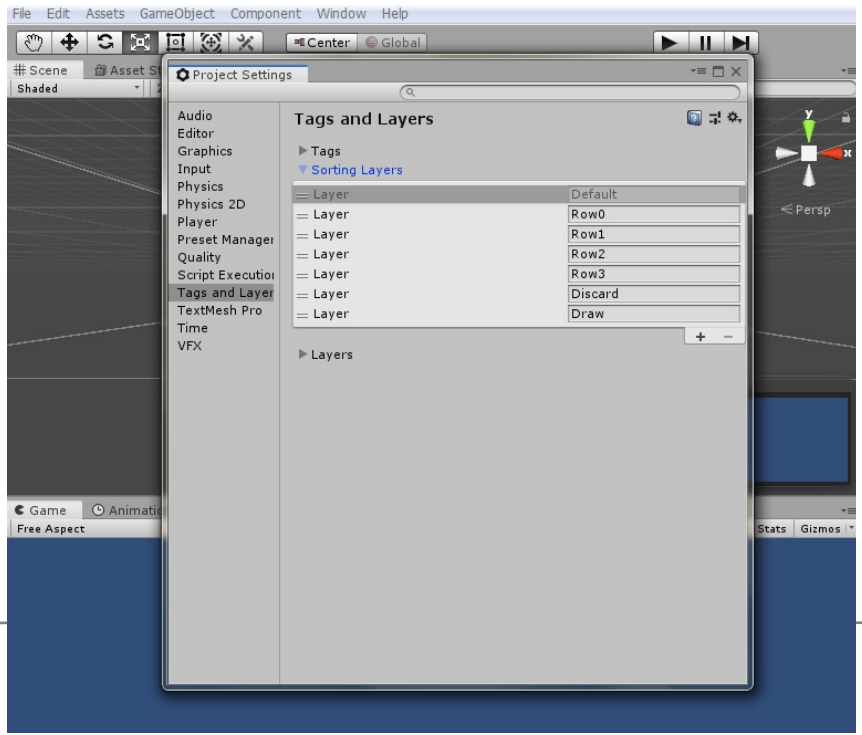
שמרו את הסקריפט וחיזרו ל-unity. מתי שאתם מריצים את המשחק , תראו כי הקלפים אכן עומדים במבנה ה-tableau כפי שמתואר ב-Layput.xm , אבל יש בעיה רצינית עם השכבות:



תוך כדי שהמשחק רץ , לחצו על Alt וחץ למטה (או option וחץ למטה) כדי לעשות זום אין לתוך התמונה בחלון הסצנה, ניתן גם עם לחיצה על Alt ומקש ימני בעכבר, בעזרת הזזת העכבר ניתן לסרוק את הסצנה מכל מיני זוויות שונות.
אם נשתמש בכפתור לראייה דו ממדית של הסצנה (איפה שכתוב 2D מעל לחלון הסצנה) המרחק של האובייקט 2D למצלמה, לא מושפע מהמיון עומק של האובייקט (כלומר , איזה אובייקט מרונדר מעל איזה אובייקט).
מוקדם יותר במהלך בניית המשחק, היה לנו קצת מזל עם בניית הקלפים, היות ובנינו אותם מהחלק האחורי לקדמי( back to front) כך שכל ה-pips וה-Decorators נראו מעל הקלף . אולם, כאן אנחנו צריכים להיות יותר ערניים לגבי זה עם פריסת האובייקטים במשחק כדי להימנע מהבעיה שראינו כרגע.
למזלנו ל-unity יש שתי שיטות להתמודד עם מיון עומק:

- Sorting Layer: משמש כדי לקבץ אובייקטי דו-ממד. כל דבר שאמור להיות שבכבה נמוכה יותר ,מרונדר מאחורי כל דבר שנמצא בשכבה גבוהה יותר. כל רכיב SpriteRenderer יש משתנה מחרוזת sortingLayerName שיכול לערוך את השכבה של אותו אובייקט.
- Spr
Spri
sor

ting order: לכל רכיב teRenderer יש גם משתנה tingOrder . הוא נועד כדי למקם אלמנטים בשכבות מיון שונות יחסיות אחת לשנייה.




בהיעדר מיון שכבות ו-sortingOrder , ספרייטים לעיתים מתרנדרים מלפנים החוצה.
הפסיקו את הרצת המשחק לפני שנמשיך.

מיון השכבות-
כדי לערוך את ה-sorting layer עקבו אחר הצעדים הבאים:
1. מהתפריט הראשי בחרו Edit->

Project Setting ->Tags and Layer

2. פתחו את המשולש ליד ה-Sorting layer וכתבו את השכבות כפי שהן מופיעות בתמונה למטה .

כדי להוסיף שכבה חדשה לחצו על הפלוס שנמצא בצד ימין למטה. באינספקטור הנ"ל השורה האחרונה (Draw) מעל לכל שאר השכבות.

משום ש SpriteRenderer ומיון עומק הם דברים שיהיו שימושיים בכל משחק קלפים, כדאי להכניס את הקוד הבא למחלקת Card (ולא ל-CardProspector שהוא ספציפי למשחק שלנו).

פתחו את הסקריפט Card והוסיפו את זה לקוד:

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

public class Card : MonoBehaviour
{
    [Header("Set Dynamically")]
    public string suit;//(C,D,H,or S)
    public int rank;//(1-14)
    public Color color = Color.black;//color to tint pips
    public string colS = "Black";//or Red. Name thr color

    //this list hold all of thre Decorator GameObjects
    public List<GameObject> decoGOs = new List<GameObject>();
    //this list holds all thr pip GameObject
    public List<GameObject> pipGOs = new List<GameObject>();
    public GameObject back;//the GameObject of the back of the card
    public CardDefinition def;//parsed from DeckXML.xml
    //List of the SpriteRenderer Components of this GameObject
    public SpriteRenderer[] spriteRenderers;

     void Start()
    {
        SetSortOrder(0);
```

```
    }

    //If spriteRenderers is not yet define, this function defins it
    public void PopulateSpriteRenderers()
    {
        // If spriteRenderers is null or empty
        if(spriteRenderers==null||spriteRenderers.Length==0)
        {
            //Get SpriteRenderer Components of this GameObject and its children
            spriteRenderers = GetComponentsInChildren<SpriteRenderer>();
        }
    }

    //Sets the sortingLayerName on all SpriteRenderer Components
    public void SetSortingLayerName(string tSLN)
    {
        PopulateSpriteRenderers();
        foreach (SpriteRenderer tSR in spriteRenderers)
        {
            tSR.sortingLayerName = tSLN;
        }
    }

    //Sets the sortingOreder of all SpriteRenderer Components
    public void SetSortOrder(int sOrd)
    {
        PopulateSpriteRenderers();

        //Iterate through all the spriteRenderers as tSR
        foreach(SpriteRenderer tSR in spriteRenderers)
        {
            //If the gmeObject is this.gameObject,it's the backgroud
            if (tSR.gameObject == this.gameObject)
            {
                tSR.sortingOrder = sOrd;//Set it's order to sOrd
                continue;//And continue to the next iteration of the loop
            }
           //Each of the children of this GameObject are named
           //switch based on the names
           switch(tSR.gameObject.name)
            {
                case "back":
                    //Set it to the highest layer to cover the other sprites
                    tSR.sortingOrder = sOrd + 2;
                    break;
                case "face":// if the name is "face"
                default:// or if it's anything else
                    //Set it to the middle layer to be above the background
                    tSR.sortingOrder = sOrd + 1;
                    break;
            }
        }
    }



    public bool faceUp
    {

     .   .   .


    }

}
```

הרקע הלבן של הקלף מופיע בתחתית (sOrd).
מעל לזה יהיו ה-pips,decorators,face ועוד (sOrd+1).
גב הקלף מופיע מעל לכל, ומתי הוא "נראה" הוא מכסה את כל השאר (sOrd+2).
4.ב- *Prospector* צריך להוסיף שורה אחת ליד הסוף של המתודה LayoutGame() כדי לוודא שהקלפים יאותחלו לשכבה הראשית:

```
public class Prospector : MonoBehaviour
```

```
{
 ...
void LayoutGame()
    {
        . . .
        foreach(SlotDef tSD in layout.slotDefs)
        {
                . . .

            //CrdProsperctor in the tableau have the state CardState.tableau
            cp.state = eCardState.tableau;
            cp.SetSortingLayerName(tSD.layerName);//Set the sorting layers
            tableau.Add(cp);//Add this CradProspector to the List<> tableau
        }
    }
}
```

5.שמרו את הסקריפטים וחזרו ל-unity והריצו את הסצנה.
הקלפים אמורים להיות מסודרים כראוי ב-tableau. עדיין לא סידרנו את הקלפים שנשארו, אבל גם לזה נגיע.

## יישום לוגיקת משחק-

לפני שנמשיך ליישום החפיסה הראשית, ממנה מושכים קלפים , בואו נתחיל בתחימת האפשרויות שיכולות לקרות במהלך המשחק:

A. אם הקלף המוצג (the target card) מוחלף קלף אחר, הקלף שהחלפנו מושלך לחפיסת המולכים או ,בשם אחר, חפיסת "הזבל"(discard pile).

B. קלף יכול לזוז מהחפיסה הראשית (draw pile) ולהיות ה-target card .

C. קלף אחד מתחת/ מעל(מבחינת ערך (rank ל-target card יכול להפך ל-target card.

D. אם רק הגב של קלף מוצג (face-down), ואין יותר קלפים מעליו שמסתירים אותו הוא יסובב (face-up).

E. המשחק נגמר כאשר המכרה (ה-tableau) ריק ואז זה ניצחון, או שהחפיסה ממנה מושכים קלפים ריקה ואז זה הפסד.

שימו לב ש-C ו-B הם מקרים של מהלכים אפשריים במשחק, ושאר המקרים הם תוצאות אפשריות של אותם מהלכים.

### לאפשר הקלקה על הקלפים(Clickable)-

היו וכל הפעולות הללו מתאפשרות ע"י על אחד הקלפים, המטלה הראשונה שלנו תהיה לגרום לקלפים להיות ניתנים להקלקה:

1. נשים לב שכאן אנחנו צריכים לאפשר לכל קלף את האפשרות להקליק עליו לכן הוסיפו את המתודה הבאה לקראת סוף מחלקת Card:

```
public class Card : MonoBehaviour
{
...
public bool faceUp
    {
        get
        {
            return (!back.activeSelf);
        }
        set
        {
            back.SetActive(!value);
        }
    }

    virtual public void OnMouseUpAsButton()
    {
```

```
        print(name);//When clicked, this outputs the  card name
    }
}
```

שימו לב שהשתמשנו במילה השמורה virtual , כפי שראינו כבר ,או שאנחנו מכירים מ-c++, המילה השמורה virtual מסמנת לקומפיילר שזו מתודה שניתן לדרוס אותה במהלך ירושה מהמחלקה.
שמרו את הסקריפט וחיזרו ל-unity. הריצו את המשחק. עתה מתי שאנחנו לוחצים על הקלפים אנחנו אמורים לראות אותם בצד של חלון המשחק.

2. אולם, במשחק prospector, לחיצה על הקלף אמורה לעשות יותר מזה, אז הוסיפו  את התודה הבאה לסוך מחלקת CardProspector:

```
public class CardProspector : Card
{
...
public override void OnMouseUpAsButton()
    {
        //Call the CardClicked method on the Prospector singleton
        Prospector.S.CardClick();

        base.OnMouseUpAsButton();
    }
}
```

משום שזאת פונקציה שאנחנו דורסים ממחלקת האב Card  אנחנו צריכים להוסיף לחתימת הפונקציה override, כמו כם אנחנו קוראים למתודה של פונקצית האב ולכן מציינים גם את השורה base....
בנתיים המתודה CardClick  לא מופיע במחלקת Prospector. אנחנו עדיין צריכים להוסיף אותה, אבל קודם נוסיף למחלקת Prospector כמה מתודות עזר
3. היכנסו ל-Prospector והכניסו את המתודות הבאות לסוף המחלקה:

```
public class Prospector : MonoBehaviour
{
...
void LayoutGame()
    {
     .   .   .
}

    //Move the current target to the dicard pile
    void MoveToDiscard(CardProspector cd)
    {
        //Change the state of the card to discard
        cd.state = eCardState.discard;
        discardPile.Add(cd);
        cd.transform.parent = layoutAnchor;

        //Position this card on the discardPile
        cd.transform.localPosition = new Vector3(layout.multiplier.x * layout.discardPile.x,
layout.multiplier.y * layout.discardPile.y, -layout.discardPile.LayerID + 0.5f);
        cd.faceUp = true;
        //Place it on top of the pile for deph sorting
        cd.SetSortingLayerName(layout.discardPile.layerName);
        cd.SetSortOrder(-100 + discardPile.Count);
    }

    //Make cd the new target card
    void MoveToTarget(CardProspector cd)
    {
        //If there is currently a target card, move it to discard pile
        if (target != null) MoveToDiscard(target);
        target = cd;
        cd.state = eCardState.target;
        cd.transform.parent = layoutAnchor;
        //Move to the target position
```

```
        cd.transform.localPosition = new Vector3(layout.multiplier.x * layout.discardPile.x,
layout.multiplier.y * layout.discardPile.y, -layout.discardPile.LayerID);

        cd.faceUp = true;//Make it faceUP
        //Set the depth sorting
        cd.SetSortingLayerName(layout.discardPile.layerName);
        cd.SetSortOrder(0);
    }

    //Arrange all the cards of the draw pile to show how many are left
    void UpdateDrawPile()
    {
        CardProspector cd;
        for (int i = 0; i < drawPile.Count; i++)
        {
            cd = drawPile[i];
            cd.transform.parent = layoutAnchor;

            //Position it correctly with the layout.drawPile.stagger
            Vector2 dpStagger = layout.drawPile.stagger;
            cd.transform.localPosition = new Vector3(layout.multiplier.x * (layout.drawPile.x + i *
dpStagger.x)
                , layout.multiplier.y * (layout.drawPile.y + i * dpStagger.y)
                , -layout.drawPile.LayerID + 0.1f * i);

            cd.faceUp = false;//Make theme all face-down
            cd.state = eCardState.drawpile;
            //Set depth sorting
            cd.SetSortingLayerName(layout.drawPile.layerName);
            cd.SetSortOrder(-10 * i);
        }
    }

}
```

4. הוסיפו את הקוד הבא לסוף Prospector.LayoutGame() כדי למשוך את הקלף ההתחלתי ולסדר את החפיסה הראשית.

בהמשך הקוד גם נכתוב גרסה התחלתית של המתודה CardClicked(). בנתיים CardClicked() תשמש רק בהזזת קלף מהחפיסה הראשית ל target אבל נרחיב אותה בהמשך:

```
public class Prospector : MonoBehaviour
{
...
void LayoutGame()
    {
        . . .
    tableau.Add(cp);//Add this CradProspector to the List<> tableau
     }
        //Set up initial target card
        MoveToTarget(Draw());

        //Set up the Draw pile
        UpdateDrawPile();
    }

        . . .
 void UpdateDrawPile()
    {
        . . .
    }

    //CardClicked is called any time a card in the game is clicked
    public void CardClicked(CardProspector cd)
    {
        //The reaction is determind the by state of the thhe clicked card
        switch(cd.state)
        {
            case eCardState.target:
                //Does nothing
                break;
            case eCardState.drawpile:
                //Clicking any card in the drawPile will draw the next pile
                MoveToDiscard(target);
```

```
                MoveToTarget(Draw());
                UpdateDrawPile();
                break;
            case eCardState.tableau:
                //Will check if it valid play
                break;
        }
    }

}
```

5. שמרו את הסקריפט, חיזרו ל-unity והריצו את המשחק.
אנחנו אמורים לראות את החפיסה הראשית ממנה מושכים קלפים, ואת המכרה שבנוי כפירמידה משולשת.
אם נלחץ על קלף מהחפיסה הוא ימשך לנו לחבילה של ה-target.

## התאמת הקלפים מהמכרה למשחק-

כדי לגרום לקלפים מהמכרה לעבוד, נצטרך להוסיף קוד שבודק שהקלף שלחצנו עליו הוא הוא רמה מעל\מתחת לקלף ה-target(וכמובן מקרי קצה כמו אס ומלך).

1. היכנסו למתודה CardClicked() של מחלקת *Prospector* והוסיפו לה את הקוד הבא:

```
public class Prospector : MonoBehaviour
{
 . . .

public void CardClicked(CardProspector cd)
    {
        //The reaction is determind the by state of the thhe clicked card
        switch (cd.state)
        {
            case eCardState.target:
                //Does nothing
                break;
            case eCardState.drawpile:
                //Clicking any card in the drawPile will draw the next pile
                MoveToDiscard(target);
                MoveToTarget(Draw());
                UpdateDrawPile();
                break;
            case eCardState.tableau:
                  bool validMatch = true;
                  if (!cd.faceUp)
                  {
                      //If the card is face down
                      validMatch = false;
                  }
                  if (!AdjacentRank(cd, target))
                  {
                      //If it is not an adjacent rank, it is not valid
                      validMatch = false;
                  }
                  if (!validMatch)
                      return;

                  //If we got here, then:Yay! it's a valiid card
                  tableau.Remove(cd);//Remive it from the tableau List
                  MoveToTarget(cd);//Make it target card

                break;
        }
    }


//Return true if the two cards are adjacent in rank
    public bool AdjacentRank(CardProspector c0,CardProspector c1)
    {
        if (Mathf.Abs((c0.rank % 13) - (c1.rank % 13)) == 1)
        {
```

```
            return true;
        }
        //For the king and the queen
        if ((c0.rank == 13 && c1.rank == 12) || (c0.rank == 12 && c1.rank == 13))
            return true;
        else return false;
    }

}
```

2. שמרו את הסקריפט וחזרו ל-unity
הריצו את המשחק, עכשיו אנחנו יכולים לשחק (בצורה חלקית לפחות). אך, אם תשימו לב, הקלפים ההפוכים אינם מסתובבים. בשביל זה השדה List<CardProspector> CardProspector.hiddenBy. המידע על איזה קלף מסתיר אחרים נמצא ב
List<int>SlotDef.hiddenBy, אבל אנחנו צריכים להיות מסוגלים להמיר מהאינטג'ר IDs ב- SlotDef.hiddenBy לאבייקט CardProspector שיש לו את אותו ה-ID .

3. הוסיפו את הקוד הבא ל-Prospectror כדי לעשות זאת(שוב, מה שבולט):

```
public class Prospector : MonoBehaviour
{
...

//LayoutGame() positions the initial tableau of card, a.k.a the "mine"
    void LayoutGame()
    {
        .  .  .

foreach (SlotDef tSD in layout.slotDefs)
        {
           .  .  .

tableau.Add(cp);//Add this CradProspector to the List<> tableau
        }


//Set which cards are hiding others
         foreach (CardProspector tCP in tableau)
         {
             foreach (int hid in tCP.slotDef.hiddenBy)
             {
                 cp = FindCardByLayoutID(hid);
                 tCP.hiddenBy.Add(cp);
             }
         }
        //Set up initial target card
        MoveToTarget(Draw());

        //Set up the Draw pile
        UpdateDrawPile();
    }


//Convert from layoutID int to the CardProspector with that ID
    CardProspector FindCardByLayoutID(int layoutID)
    {
         foreach (CardProspector tCP in tableau)
         {
             //Search throught all cards in the tableau List<>
             if (tCP.layoutID == layoutID)
             {
                 return tCP;
             }
```

```
        }
        //If it's not found, return null
        return null;
    }

//this turns cards in the mind face-up or down
    void SetTableauFaces()
    {
        foreach(CardProspector cd in tableau)
        {
            bool faceUp = true;//Assume this card will be face-up
            foreach (CardProspector cover in cd.hiddenBy)
            {
                //If either of the converting cards are in the tableau
                if (cover.state == eCardState.tableau)
                {
                    faceUp = false;//then this card is face down
                }
            }
            cd.faceUp = faceUp;//Set the value on this card
        }
    }


. . .

//CardClicked is called any time a card in the game is clicked
    public void CardClicked(CardProspector cd)
    {
        //The reaction is determind the by state of the thhe clicked card
        switch (cd.state)
        {
            . . .
case eCardState.tableau:
                bool validMatch = true;
                if (!cd.faceUp)
                {
                    //If the card is face down
                    validMatch = false;
                }
                if (!AdjacentRank(cd, target))
                {
                    //If it is not an adjacent rank, it is not valid
                    validMatch = false;
                }
                if (!validMatch)
                    return;

                //If we got here, then:Yay! it's a valiid card
                tableau.Remove(cd);//Remive it from the tableau List
                MoveToTarget(cd);//Make it target card
                SetTableauFaces();//Update tableau card face-ups
                break;
        }
    }


//Return true if the two cards are adjacent in rank
    public bool AdjacentRank(CardProspector c0,CardProspector c1)
    {
        . . .
    }

}
```

עכשיו, לאחר שתשמרו את הסקריפט ותחזרו ל-unity, נוכל לשחק סבב שלם של המשחק!

4. השלב הבא הוא לסיים את המשחק כאשר נגמרים לנו הקלפים(בחפיסה הראשית או במכרה)
הוסיפו את הקוד הבא למתודה CardClicked() של מחלקת Prospector:

```
public class Prospector : MonoBehaviour
{

...

//CardClicked is called any time a card in the game is clicked
    public void CardClicked(CardProspector cd)
    {
        //The reaction is determind the by state of the thhe clicked card
        switch (cd.state)
        {
          .   .   .
        }

        //Check to see whether the is over or not
         CheckForGameOver();
    }

    //Test whether the game is over
    void CheckForGameOver()
    {
        //if the tableau is empty then the game is over
        if (tableau.Count == 0)
        {
            //Call GameOver() you won
            GameOver(true);
            return;
        }

        //if there are still cards in the draw pile, the game is not over
        if (drawPile.Count > 0)
        {
            return;
        }

        //Check for remaining valid playes
        foreach (CardProspector cd in tableau)
        {
            if (AdjacentRank(cd, target))
            {
                return;//if there is a valid play, the game is not over
            }
        }

        //Since there are no valid palys, the game is over
        //Call GameOver() you lost
        GameOver(false);
    }



    void GameOver(bool won)
    {
        if (won)
        {
            print("Game Over. You won!  :-)");
        }
        else
        {
            print("Game Over. you Lost! :-(");
        }

        //Reload the scene, resetting the game/
        SceneManager.LoadScene("__Prospector_Scene_0");
    }

public bool AdjacentRank(CardProspector c0,CardProspector c1)
```

```
    {
     .   .   .
    }

}
```

5. שמרו את הסקריפטים וחזרו ל-unity כדי לבחון את המשחק עד כה.

כעת ניתן לשחק ממש במשחק והוא חוזר על עצמו לאחר הספד או ניצחון. כדי לבחון הפסד ניתן פשוט לשחק במשחק ורק למשוך מהחפיסה עד שהיא תתרוקן. הסצנה אמורה להטען מחדש לאחר ההפסד. לבחון מה קורה כמנצחים במשחק עלול לקחת קצת יותר זמן ;-)

בשלב הבא נעבוד על ניקוד

## להוסיף ניקוד למשחק-

במשחק המקורי אין מנגנון של ניקוד- או שהשחקן ניצח או שהוא הפסיד. אבל כמשחק דיגיטאלי, שמירה על ניקוד והצגת טבלת ניקוד כך שלשחקנים יהיה תמריץ להמשיך לשחק (לעבור את הניקוד המקסימלי שלהם) זה ממש שימושי.

### דרכים להרוויח ניקוד במשחק:

אנחנו ניישם כמה דרכים להרוויח ניקוד במהלך המשחק:

א. הזזה של קלף מהמכרה(tableau) ל-target מזכה בנקודה אחת.
ב. כל רצף של הזזת קלפים מהמכרה מבלי למשוך מהחפיסה הראשית(Draw pile) באמצע מעלה בנקודה את הזכייה על הזזת קלף מהמכרה, כלומר אם הצלחנו להזיז חמישה קלפים ברצף, על הקלף הראשון נרוויח נקודה ,על השני שתי נקודות, על השלישי שלוש, רביעי ארבע וחמישי חמש ,סה"כ חמש עשרה.(15=1+2+3+4+5)
ג. אם שחקן מנצח בסיבוב היא/הוא לוקח את הניקוד איתו גם לסיבוב הבא. אם מפסידים אז מתאפס הניקוד.
ד. בכל רצף הזזות של קלפים מהמכרה, הרווח מוכפל ע"י הזזת קלף "מוזהב", כך למשל אם לקחנו שני קלפים מוזהבים מהמכרה ברצף הזהה לרצף מסעיף ב', נקבל שהניקוד שלנו יהיה: 15 * 2 * 2= 60 .

המחלקה *Prospector* תטפל בזה כי היא מודעת לכל המצבים האפשריים שיכולים לתרום לניקוד.
כמו כן ניצור סקריפט מיוחד *Scoreboard* כדי לטפל בכל האלמנטים הוויזואליים שמציגים את הניקוד לשחקן.

### יצירת ניקוד לרצף הזזת קלפים מהמכרה:

כדי לעקוב אחרי הניקוד במשחק, נצטרך ליצור אובייקט *ScoreManager* ולהצמיד אותו לאובייקט המצלמה.
היות ואנחנו מאפשרים משיכה של כמה קלפים מהמכרה ברצף ואפשרות להכפיל את הניקוד במהלך הרצף, היוגי יהיה לאחסן את הניקוד לחוד ואת הניקוד הסופי(כולל הוספת ניקוד על רצף) בנפרד (הניקוד מתקבל לאחר שמשכנו קלף מהחפיסה הראשית, כי אז נגמר הרצף).

1. צרו סקריפט חדש בתיקייה הייעודית לסקריפטים, עם השם *ScoreManager* .

2. הצמידו אותו לאובייקט המצלמה (*MainCamera_*)

3. היכנסו לסקריפט של ה-*ScoreManager* והכניסו לו את הקוד הבא:

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



// an enum to handle all to possible scoring event
public enum eScoreEvent
{
  draw,
  mine,
  mineGold,
  gameWin,
  gameLoss
}


//Score Manager handles all of the scoring
public class ScoreManager : MonoBehaviour

{
    static private ScoreManager S;//singleton
    static public int SCORE_FROM_PREV_ROUND = 0;
    static public int HIGH_SCORE = 0;

    [Header("Set Dynamically")]
    //Fields to track score info
    public int chain = 0;
    public int scoreRun = 0;
    public int score = 0;

    void Awake()
    {
        if (S == null)
        {
            S = this;//Set the private singleton
        }
        else
        {
            Debug.LogError("ERROR: ScoreManager.Awake() : S is already set!");
        }

        //Check for a high score in PlayPrefs
        if (PlayerPrefs.HasKey("ProspectorHighScore"))
        {
            HIGH_SCORE = PlayerPrefs.GetInt("ProspectorHighScore");
            /**
             *PlayerPrefs is a simple way to store and retrieve values for use within your project.
             * It is well known that as it isn't encrypted you shouldn't store anything sensitive in there.
             * However, it can be very useful for storing other values,
             * such as user preferences or configuration.
             * */
        }

        //Add the score from the last round, which will be >0 if it was a win
        score += SCORE_FROM_PREV_ROUND;
        //And reset the score form prev round
        SCORE_FROM_PREV_ROUND = 0;
    }

    static public void EVENT(eScoreEvent evt)
    {
        try
        {
            S.Event(evt);
        }
        catch (System.NullReferenceException nre)
        {
            Debug.LogError("ScoreManager: EVENT() called while S=null\n" + nre);
        }
    }

    void Event(eScoreEvent evt)
    {
        switch (evt)
        {
```

```
            //Thing that should happen wheter it's a draw, a win, or a loss
            case eScoreEvent.draw:
            case eScoreEvent.gameWin:
            case eScoreEvent.gameLoss:
                chain = 0;// reset the score chain
                score += scoreRun;//add scoreRun to total score
                scoreRun = 0;//reset scoreRun
                break;

            case eScoreEvent.mine://Remove a mine card
                chain++;//increase the score chain
                scoreRun += chain;//add score for this card to scoreRun
                break;
        }

        //This second switch statement handles round wins and losses
        switch (evt)
        {
            case eScoreEvent.gameWin:
                //if this is a win, add the score to the next round
                //static fields are not by SceneManager.LoadScene()
                SCORE_FROM_PREV_ROUND = score;
                print("You won this round! Round score: " + score);
                break;

            case eScoreEvent.gameLoss:
                //If it a loss' check against the high score
                if (HIGH_SCORE <= score)
                {
                    print("You got the high score! High score: " + score);
                    HIGH_SCORE = score;
                    PlayerPrefs.SetInt("ProspectorHighScore", score);
                }
                else
                {
                    print("Your final score for the game was: " + score);
                }
                break;

            default:
                print("score: " + score + "  ,scoreRun: " + scoreRun + "  ,chain: " + chain);
                break;
        }
    }

    static public int CHAIN { get { return S.chain; } }
    static public int SCORE { get { return S.score; } }
    static public int SCORE_RUN { get { return S.scoreRun; } }

}
```

א. אם תשימו לב השתמשנו בתבנית עיצוב סינגלטון כדי לדאוג שלא יהיה ניתן ליצור עוד אובייקט מסוג המחלקה. הפעם הגדרנו את האינסטנס ש המחלקה(האובייקט מסוג המחלקה ששמור כמשתנה עצם) כ-private , זאת כדי להוסיף לו עוד "הגנה" כך שרק ה-*ScoreManager* יוכל לגשת לאובייקט.

ב. הגירסה ה-static public EVENT() מאפשרת למחלקות אחרות לגשת למתודה (למשל למחלקה -*Prospector*) ולשלוח לה eScoreEvent, מתי שהן עושות את זה,EVENT() קוראת למתודה הלא סטאטית Event() ע"י האינסטנס של המחלקה S, אם S עדיין לא הוגדר המתודה תזרוק שגיאה.

ג. במהלך הקוד השתמשנו באובייקט מערכת של unity שלא ראינו לפני כן- PlayerPrefs.

PlayerPrefs הוא כמין מיני דאטה בייס שמשמש לאחסון נתונים בין משחקים. החיסרון העיקרי שלו היא הקלות שבה ניתן לשנות נתונים ולהוסיף ניקוד ידנית כי הטקסט לא מוצפן.

למידע נוסף על המחלקה:

https://www.youtube.com/watch?v=donIirlj074

כמו כן בערוץ Brackeys יש מדריך מעולה כיצד ניתן להתגבר על החיסרון העיקרי של המחלקה ע"י יצירת קוד בניארי:

https://www.youtube.com/watch?v=XOjd_qU2Ido

4. הוסיפו את השורות הבאות למתודות CardClicked() ול- GameOver() של מחלקת Prospector כדי שנוכל להשתמש ב-ScoreManager:

```
public void CardClicked(CardProspector cd)
    {
        //The reaction is determind the by state of the thhe clicked card
        switch (cd.state)
        {
            case eCardState.target:
                //Does nothing
                break;
            case eCardState.drawpile:
                //Clicking any card in the drawPile will draw the next pile
                MoveToDiscard(target);
                MoveToTarget(Draw());
                UpdateDrawPile();
                  ScoreManager.EVENT(eScoreEvent.draw);
                break;
            case eCardState.tableau:
                 .   .   .

                tableau.Remove(cd);//Remive it from the tableau List
                MoveToTarget(cd);//Make it target card
                SetTableauFaces();//Update tableau card face-ups
                ScoreManager.EVENT(eScoreEvent.mine);
                break;
        }

        //Check to see whether the is over or not
        CheckForGameOver();
            }


        . . .

void GameOver(bool won)
    {
        if (won)
        {
            print("Game Over. You won!  :-)");
            ScoreManager.EVENT(eScoreEvent.gameWin);
        }
        else
        {
            print("Game Over. you Lost! :-(");
            ScoreManager.EVENT(eScoreEvent.gameLoss);
        }

        //Reload the scene, resetting the game/
        SceneManager.LoadScene("__Prospector_Scene_0");
            }
```

5. שמרו את הסקריפטים, חזרו ל-unity והריצו את המשחק.

עכשיו, כשנשחק במשחק, נשים לב להערות מהקונסול, הוא אומר לנו את הניקוד.
בנוסף, אם נבחר באובייקט המצלמה תוך כדי משחק, נראה כי באינספקטור, ברכיב ה-*Score Manager(Script)* אם אנחנו מנצחים בסיבוב, אנחנו שומרים על הניקוד במעבר לשלב הבא. זה מעולה לנו כמתכנני המשחק, אבל מבחינת משחק יש לנו עדיין מה להשתפר בנראות. בואו נשפר את החוויה של המשחק ע"י הוספת ניקוד ויזואלי בזמן ריצת המשחק.

## הצגת הניקוד של השחקן –

למשחק הזה, ניצור גם זוג רכיבים שיכולים לשמש אותנו להצגת הניקוד של השחק.
הראשון יהיה מחלקת *ScoreBoard* כדי לטפל בהצגת הניקוד.

והשני יהיה *FloatingScore*

1. צרו סקריפט חדש בתיקייה _Scripts עם השם: *FloatingScore* והכניסו לו את הקוד הבא:

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

public enum eFSSstate
{
    idle,
    pre,
    active,
    post
}

public class FloatingScore : MonoBehaviour
{
    [Header("Set Dynamically")]
    public eFSSstate state = eFSSstate.idle;

    [SerializeField]
    protected int _score = 0;
    public string scoreString;

    //The score property sets both _score and scoreString
    public int score
    {
        get
        {
            return (_score);
        }
        set
        {
            _score = value;
            scoreString = _score.ToString();
            GetComponent<Text>().text = scoreString;
        }
    }

    public List<Vector2> bezierPts;//bezier points for movment
    public List<float> fontSizes;//Bezier points for font scaling
    public float timeStart = -1f;
    public float timeDuration = 1f;
    public string easingCurve = Easing.InOut;//Use Easing in Utils.cs

    //The gameObject that will receive the SendMessage when this is done moving
    public GameObject reportFinishTo = null;
    private RectTransform rectTrans;
    private Text txt;

    //Set up the FloatingScore and movment
    //Note the use of parameter defaults for eTimesS & eTimeD
    public void init(List<Vector2> ePts, float eTimeS = 0, float eTimeD = 1)
    {
        rectTrans = GetComponent<RectTransform>();
        rectTrans.anchoredPosition = Vector2.zero;
        txt = GetComponent<Text>();
        bezierPts = new List<Vector2>(ePts);

        if (ePts.Count == 1)
        {
            //..Then just go there
            transform.position = ePts[0];
            return;
        }
        //If eTimeS is the default, just start at the current time
        if (eTimeS == 0)
        {
            eTimeS = Time.time;
        }
        timeStart = eTimeS;
        timeDuration = eTimeD;

        //Set it ti the pre state' ready to start time
```

```
        state = eFSSstate.pre;
    }

    public void FScallback(FloatingScore fs)
    {
        //When this callback is called by SendMessage,
        //add the score from the calling FloatungScore
        score += fs.score;
    }

    //Update is called once per frame
    void Update()
    {
         //If this is not moving, just return
        if (state == eFSSstate.idle) return;

        //Get u from the current tim duration
        //u range from 0 to 1
        float u = (Time.time - timeStart) / timeDuration;
        // Use Easing class from Utils to curv the u value.
        float uC = Easing.Ease(u, easingCurve);
        if (u < 0)
        {
            //If u<0, then we shouldn't move yet
            state = eFSSstate.pre;
            txt.enabled = false;//Hide the initially
        }
        else
        {
            if (u >= 1)
            {
                //If u>=1, we're done moving
                uC = 1;//Set uC=1 so we don't overshoot
                state = eFSSstate.post;
                if (reportFinishTo != null)
                {
                    //If there's a callback GameObject
                    //Use SendMessageto call the FScallback method
                    //with this parameter.
                    reportFinishTo.SendMessage("FScallback", this);
                    //Now that the message has been sent
                    // Destroy this GameObject
                    Destroy(gameObject);
                }
                else
                {
                    //If there is nothing to callback
                    //.. then dont destroy this, just let it stay still
                    state = eFSSstate.idle;
                }
            }
            else
            {
                //0<=u<1,which means that ths is active and moving
                state = eFSSstate.active;
                txt.enabled = true;//Show the score once more
            }
                //Use bezier curve to move it to the right point
                Vector2 pos = Utils.Bezier(uC, bezierPts);
                //RecTransform anchors can be used to position UI object relative
                //to total size of the screen
                rectTrans.anchorMin = rectTrans.anchorMax = pos;
                if(fontSizes!= null&& fontSizes.Count>0)
            {
                //If fotSizes hes  values in it
                //...then adjust the fontSize if this GUIText
                int size = Mathf.RoundToInt(Utils.Bezier(uC, fontSizes));
                GetComponent<Text>().fontSize = size;
            }

        }
    }
}
```

2. צרו סקריפט חדש בתיקייה _Script עם השם *Scoreboard* והכניסו לו את הקוד הבא:

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

// The Scoreboard manages showing the scire to the player
public class Scoreboard : MonoBehaviour
{
    public static Scoreboard S;//The singleton for Scorebord

    [Header("Set in inspector")]
    public GameObject prefabFloatingScore;
    [Header("Set Dynammicllay")]
    [SerializeField]
    private int _score = 0;
    [SerializeField]
    private string _scoreString;

    private Transform canvasTrans;

    //The score property also sets the scoreString
    public int score
    {
        get
        {
            return _score;
        }
        set
        {
            _score = value;
            _scoreString = _score.ToString();
        }
    }

    //The scoreString property also sets the Text.text
    public string scoreString
    {
        get
        {
            return _scoreString;
        }
        set
        {
            _scoreString = value;
            GetComponent<Text>().text = _scoreString;
        }
    }

    private void Awake()
    {
        if (S == null)
        {
            S = this;
        }
        else
        {
            Debug.LogError("ERROR:Scoreboard.Awake(): Sis already set!");
        }
        canvasTrans = transform.parent;
    }

    //When called by SendMessage, this adds the fs.score
    public void FScallback(FloatingScore fs)
    {
        score += fs.score;
        S.GetComponent<Text>().text = _scoreString;
    }

    //This will Insatiate a new FloatingScore GameObject and initialize it.
    //it also returns a pointer to the FloatingScore created so that the
    // calling function can di more with it(like Set fontSize, and so on)
    public FloatingScore CreateFloatingScore(int amt, List<Vector2> pts)
    {
```

```
        GameObject go = Instantiate<GameObject>(prefabFloatingScore);
        go.transform.SetParent(canvasTrans);
        FloatingScore fs = go.GetComponent<FloatingScore>();
        fs.score = amt;
        fs.reportFinishTo = this.gameObject;//Set fs to call back to this
        fs.init(pts);
        return fs;
    }
}
```

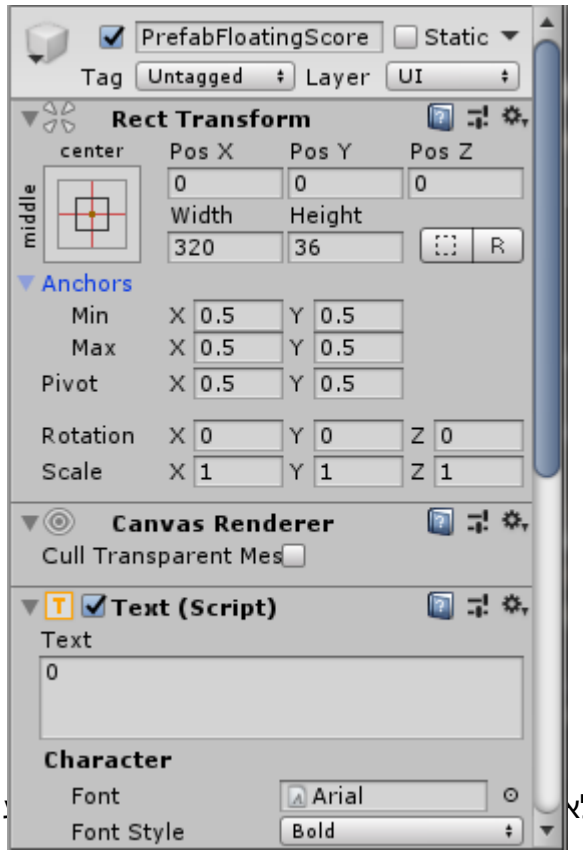
3. שמרו את כל הסקריפטים וחזרו ל-unity.

עכשיו, אנחנו צריכים להכין את ה-GameObject ל-Scoreboard ול- FloatingScore .

יצירת FloatingScore prefab –

כדי ליצור אובייקט FloatingScore:

1. מהתפריט הראשי של unity, בחרו GameObject -> UI ->Text. ושנו את שם האובייקט Text ל-PrefabFloatingScore.
2. לפני שנשנה משהו ב- PrefabFloatingScore, וודאו שהאספקטים (aspects ratio) של חלון המשחק עומדים על *Standalon(1024X768)* או על *iPad Wide(1024x768)*.
3. לכו לאינספקטור של האובייקט ושנו לו את הנתונים הבאים:





לא חלון המשחק.

4. הצמידו את הסקריפט *FloatingScore* ל- *PrefabFloatingScore* ע"י גרירת הסקריפט לאינספקטור של האובייקט.

5. הפכו את האובייקט ל-prefab ע"י גרירה שלו לחולון לתיקייה של _prefabs בחלון הפקרוייקט.

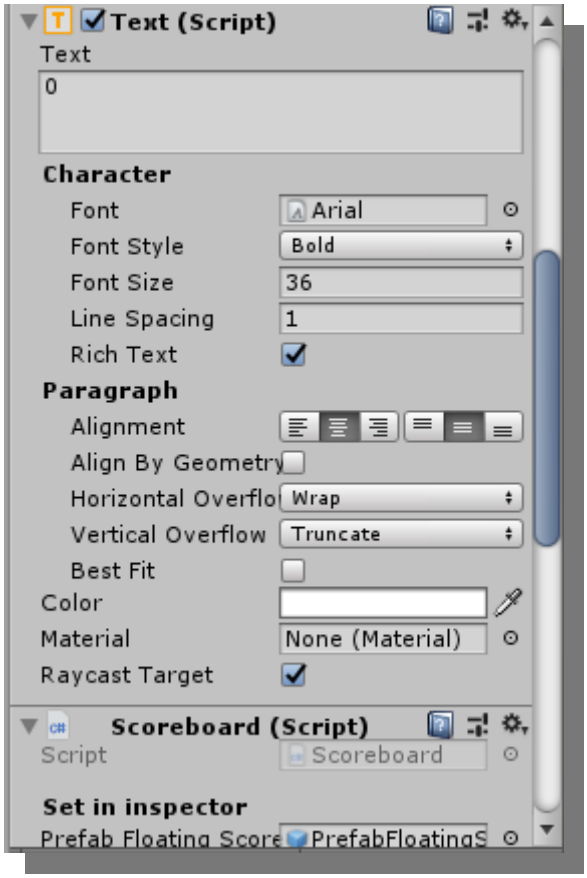
6. מחקו את האובייקט מחלון ההיררכיה.

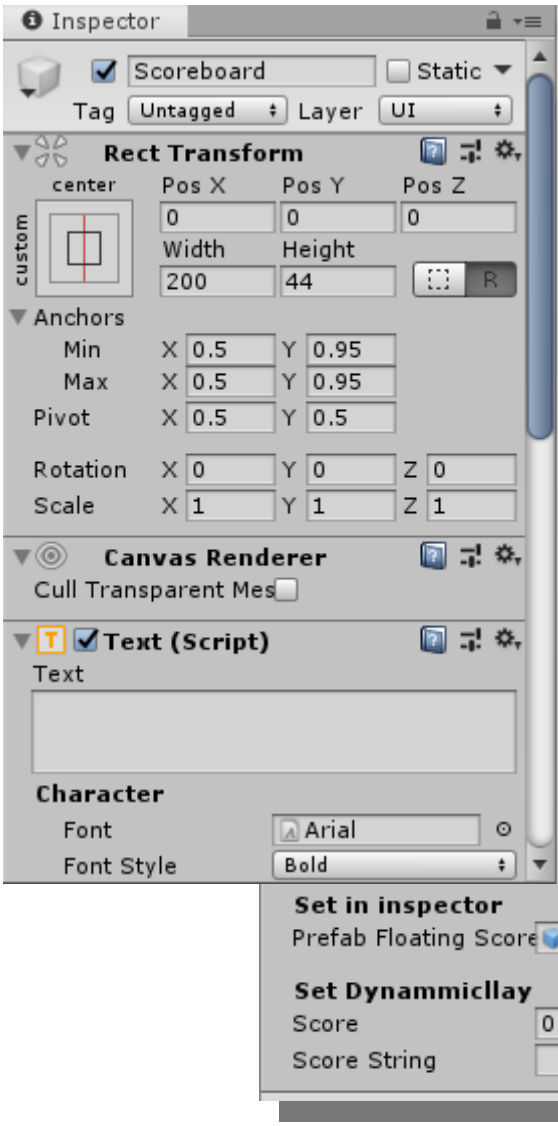
יצירת אובייקט *Scoreboard*-

כדי ליצור אובייקטי Scoreboard :

1. צרו אובייקט Text נוסף בסצנה(באותה דרך שיצרתם את הקודם)

2. שנו את שם האובייקט Text ל-Scoreboard.
3. הצמידו את הסקריפט Scoreboad.cs, גררו לאובייקט *Prefab Floating Score* שברכיב החדש, את האובייקט PrefabFloatingScore שבתיקיה *Prefab_*, ושנו את הנתונים לנתונים הבאים:




4. שמרו את הסצנה.
5. עכשיו נצטרך לבצע כמה שינויים במחלקת Prospector הוסיפו את השורות המובלטות הבאות לקוד:

```
public class Prospector : MonoBehaviour
{
    static public Prospector S;
    [Header("Set in inspector")]
    public TextAsset deckXML;
    public TextAsset layoutXML;
    public float xoffset = 3;
    public float yoffset = -2.5f;
    public Vector3 layoutCenter;
     public Vector2 fsPosMid = new Vector2(0.5f, 0.90f);
     public Vector2 fsPosRun = new Vector2(0.5f, 0.75f);
     public Vector2 fsPosMid2 = new Vector2(0.4f, 1.0f);
     public Vector2 fsPosEnd = new Vector2(0.5f, 0.95f);
```

```
    [Header("Set Dynamically")]
    public Deck deck;
    public Layout layout;
    public List<CardProspector> drawPile;
    public Transform layoutAnchor;
    public CardProspector target;
    public List<CardProspector> tableau;
    public List<CardProspector> discardPile;
     public FloatingScore fsRun;



. . .



void Start()
    {

         Scoreboard.S.score = ScoreManager.SCORE;

        deck = GetComponent<Deck>();// get the Deck

        .   .   .

       LayoutGame();
    }

. . .

//CardClicked is called any time a card in the game is clicked
    public void CardClicked(CardProspector cd)
    {
        //The reaction is determind the by state of the thhe clicked card
        switch (cd.state)
        {
            case eCardState.target:
                //Does nothing
                break;
            case eCardState.drawpile:
                //Clicking any card in the drawPile will draw the next pile
                MoveToDiscard(target);
                MoveToTarget(Draw());
                UpdateDrawPile();
                ScoreManager.EVENT(eScoreEvent.draw);
                   FloatingScoreHandler(eScoreEvent.draw);
                break;
            case eCardState.tableau:
                bool validMatch = true;
                if (!cd.faceUp)
                {
                    //If the card is face down
                    validMatch = false;
                   }
                if (!AdjacentRank(cd, target))
                {
                    //If it is not an adjacent rank, it is not valid
                    validMatch = false;
                }
                if (!validMatch)
                    return;

                //If we got here, then:Yay! it's a valiid card
                tableau.Remove(cd);//Remive it from the tableau List
                MoveToTarget(cd);//Make it target card
                SetTableauFaces();//Update tableau card face-ups
                ScoreManager.EVENT(eScoreEvent.mine);
                   FloatingScoreHandler(eScoreEvent.mine);
                break;
        }

        //Check to see whether the is over or not
        CheckForGameOver();
```

```
    }

.       .       .

void GameOver(bool won)
    {
        if (won)
        {
            print("Game Over. You won!  :-)");
            ScoreManager.EVENT(eScoreEvent.gameWin);
              FloatingScoreHandler(eScoreEvent.gameWin);
         }
        else
        {
            print("Game Over. you Lost! :-(");
            ScoreManager.EVENT(eScoreEvent.gameLoss);
              FloatingScoreHandler(eScoreEvent.gameLoss);
         }

        //Reload the scene, resetting the game/
        SceneManager.LoadScene("__Prospector_Scene_0");
    }

//Return true if the two cards are adjacent in rank
    public bool AdjacentRank(CardProspector c0, CardProspector c1)
    {
       .       .       .
    }

     //Handle FloatingScore movment
     void FloatingScoreHandler(eScoreEvent evt)
     {
         List<Vector2> fsPts;
         switch (evt)
         {
             case eScoreEvent.draw://Drawing a card
             case eScoreEvent.gameWin://Won the round
             case eScoreEvent.gameLoss://Lost the round
                 //Add fsRun to the scoreboard score
                 if (fsRun != null)
                 {
                     //Create points for the bezier curve
                     fsPts = new List<Vector2>();
                     fsPts.Add(fsPosRun);
                     fsPts.Add(fsPosMid2);
                     fsPts.Add(fsPosEnd);
                     fsRun.reportFinishTo = Scoreboard.S.gameObject;
                     fsRun.init(fsPts, 0, 1);
                     //Also adjust the fontSize
                     fsRun.fontSizes = new List<float>(new float[] { 28, 36, 4 });
                     fsRun = null; //Clear fsRun so it's created again
                 }
                 break;
             case eScoreEvent.mine:
                 //Create a FloatingScore for this score
                 FloatingScore fs;
                 //Move it from the mousePosition to fsPosRun
                 Vector2 p0 = Input.mousePosition;
                 p0.x /= Screen.width;
                 p0.y /= Screen.height;
                 fsPts = new List<Vector2>();
                 fsPts.Add(p0);
                 fsPts.Add(fsPosMid);
                 fsPts.Add(fsPosRun);
                 fs = Scoreboard.S.CreateFloatingScore(ScoreManager.CHAIN, fsPts);
                 fs.fontSizes = new List<float>(new float[] { 4, 50, 28 });
                 if (fsRun == null)
                 {
```

```
                fsRun = fs;
                fsRun.reportFinishTo = null;
            }
            else
            {
                fs.reportFinishTo = fsRun.gameObject;
            }
            break;
        }
    }

}
```

6. שמרו את הסקריפטים, חזרו ל-Unity ונסו לשחק.

עכשיו כאשר אנחנו מנסים לשחק במשחק, אנחנו אמורים לראות את הניקוד הזמני "עף" לאובייקט טקסט העליון שמייצג את הניקוד שלנו במשחק ומתחבר אליו. זה דווקא די חשוב כי זה עוזר לשחקן להבין מאיפה הניקוד מגיע ועוזר לחשוך את "המכניזם" של המשחק בזמן המשחק.