

***סוליטר***

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 xml ו-html 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="18" x="-9" y="-2" faceup="1" layer="3" />

<slot id="19" x="-7" 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 מהXML 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 בCardProspector–

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-layout (slotDef) 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 לCard 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**

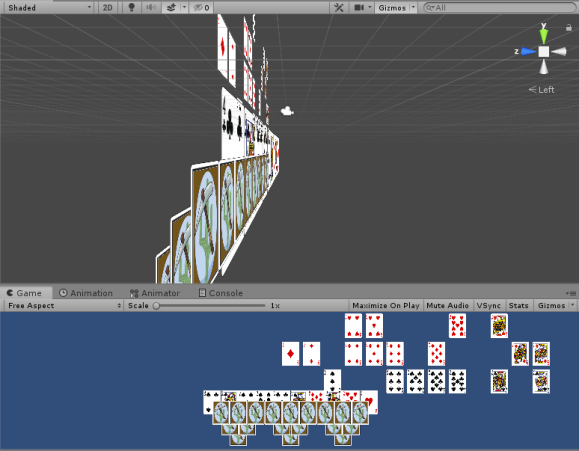
**cp.state = eCardState.tableau;**

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

**}**

}

}

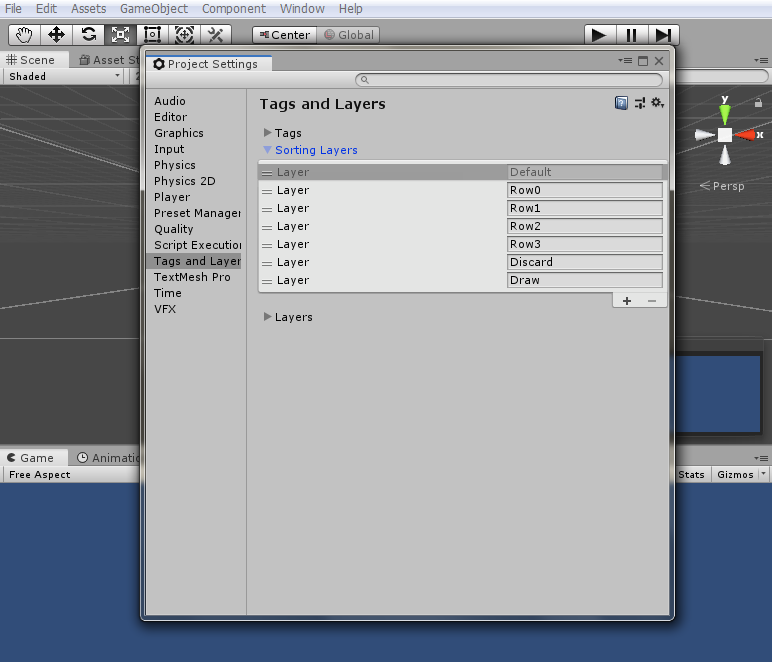
  
  
שמרו את הסקריפט וחיזרו ל-unity. מתי שאתם מריצים את המשחק , תראו כי הקלפים אכן עומדים במבנה ה-tableau כפי שמתואר ב-Layput.xml , אבל יש בעיה רצינית עם השכבות:

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.
* Sprting order: לכל רכיב SpriteRenderer יש גם משתנה sortingOrder . 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. פתתחו 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 שSpriteRenderer 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 לtarget 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;

}

}