

***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 פריסת עמוד-

1. ב-unity , פתחו את *Layout.xml* בתיקיית resource כדי לראות את המידע אודות פריס 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.