출처 : Introduction to Game Design, Prototyping and Development 2ed, Jeremy Gibson BOND

Introduction to Game Design 2ed

04 장 Prospector Solitaire

유니티 엔진 버전 2019.4.20f1

서진석 jsseo@deu.ac.kr

2021-4-15

섹션 01 : 프로젝트 생성 및 애셋 준비

1. 새로운 3D 프로젝트 생성 : Prospector
2. 리소스 다운로드 : 04_Prospector_Starter.zip
   - 압축 해제
   - 04_Prospector_Starter.unitypackage 파일 임포트 (더블 클릭)
   - __Prospector_Scene_0 씬 더블 클릭
   - 게임 창
     - 화면 : Fixed Resolution (1024 x 768)
3. 이미지 애셋을 스프라이트로
   - 프로젝트 창 / _Sprites 폴더로 이동
   - 모든 이미지 선택 / 인스펙터 창
     - Texture Type : Sprite 로 변경
     - Apply 버튼 클릭
   - Letters 이미지 선택
     - Sprite Mode : Multiple 로 변경, Apply 버튼 클릭
     - Sprite Editor 버튼 클릭
     - Sprite Editor 가 열리지 않는 경우 (패키지 설치 필요)
       - 메뉴 / Window / Package Manager
         - Unity Registry 선택
         - 2D Sprite 검색 / Install
     - Slice 버튼 클릭
       - Type : Grid By Cell Size
       - Pixel Size : 32, 32
       - Slice 버튼 클릭
       - Apply 버튼 클릭

섹션 02 : XML 이용하기

1. _Scripts 폴더에 스크립트 생성 : Card

```
public class Card : MonoBehaviour
{

}
```

```
[System.Serializable]
public class Decorator
{
    public string type;
    public Vector3 loc;
    public bool flip = false;
    public float scale = 1f;
}

[System.Serializable]
public class CardDefinition
{
    public string face;
    public int rank;
    public List<Decorator> pips = new List<Decorator>();
}
```

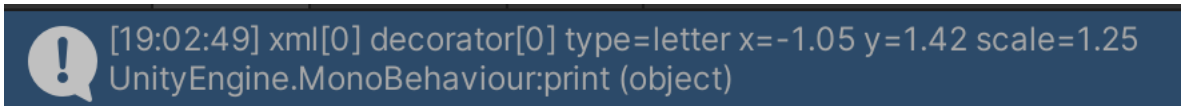
2. _Scripts 폴더에 스크립트 생성 : Deck

```
public class Deck : MonoBehaviour
{
    [Header("Set Dynamically")]
    public PT_XMLReader                 xmlr;

    public void InitDeck(string deckXMLText)
    {
        ReadDeck(deckXMLText);
    }

    public void ReadDeck(string deckXMLText)
    {
        xmlr = new PT_XMLReader ();
        xmlr.Parse (deckXMLText);

        string s = "xml[0] decorator[0] ";
        s += "type="+xmlr.xml["xml"][0]["decorator"][0].att("type");
        s += " x="+xmlr.xml["xml"][0]["decorator"][0].att("x");
        s += " y="+xmlr.xml["xml"][0]["decorator"][0].att("y");
        s += " scale="+xmlr.xml["xml"][0]["decorator"][0].att("scale");
        print(s);
    }
```

3. _Scripts 폴더에 스크립트 생성 : Prospector

```
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class Prospector : MonoBehaviour
{
    static public Prospector S;

    [Header("Set in Inspector")]
    public TextAsset deckXML;

    [Header("Set Dynamically")]
    public Deck deck;

    void Awake()
    {
        S = this;
    }

    void Start ()
    {
        deck = GetComponent<Deck>();
        deck.InitDeck(deckXML.text);
    }
}
```

4. Prospector 스크립트와 Deck 스크립트를 _MainCamera 게임 오브젝트에 어태치
5. _MainCamera 게임 오브젝트 선택 / 인스펙터 창
   - Prospector 컴포넌트 / Deck XML : DeckXML 로 설정
6. 플레이
   - 콘솔 창에 다음 메시지가 출력되는지 확인



섹션 03 : Deck XML 데이터 파싱(Parsing)

1. Deck 스크립트 수정
   - 변수 추가

```
public PT_XMLReader                 xmlr;
public List<string>                 cardNames;
public List<Card>                   cards;
public List<Decorator>              decorators;
public List<CardDefinition>         cardDefs;
public Transform                    deckAnchor;
public Dictionary<string, Sprite>   dictSuites;
```

   - ReadDeck 메서드 수정

```
public void ReadDeck(string deckXMLText)
{
    xmlr = new PT_XMLReader();
    xmlr.Parse(deckXMLText);

    string s = "xml[0] decorator[0] ";
    s += "type="+xmlr.xml["xml"][0]["decorator"][0].att("type");
    s += " x="+xmlr.xml["xml"][0]["decorator"][0].att("x");
    s += " y="+xmlr.xml["xml"][0]["decorator"][0].att("y");
    s += " scale="+xmlr.xml["xml"][0]["decorator"][0].att("scale");
    //print(s);

    decorators = new List<Decorator>();
    PT_XMLHashList xDeocs = xmlr.xml["xml"][0]["decorator"];
    Decorator deco;
    for (int i = 0; i < xDeocs.Count; i++)
    {
        deco = new Decorator();
        deco.type = xDeocs[i].att("type");
        deco.flip = (xDeocs[i].att("flip") == "1");
        deco.scale = float.Parse(xDeocs[i].att("scale"));
        deco.loc.x = float.Parse(xDeocs[i].att("x"));
        deco.loc.y = float.Parse(xDeocs[i].att("y"));
        deco.loc.z = float.Parse(xDeocs[i].att("z"));
        decorators.Add(deco);
    }

    cardDefs = new List<CardDefinition> ();
    PT_XMLHashList xCardDefs = xmlr.xml ["xml"] [0] ["card"];
    for (int i = 0; i < xCardDefs.Count; i++)
    {
        CardDefinition cDef = new CardDefinition ();
        cDef.rank = int.Parse (xCardDefs [i].att ("rank"));

        PT_XMLHashList xPips = xCardDefs [i] ["pip"];
```

```
        if (xPips != null)
        {
            for (int j = 0; j < xPips.Count; j++)
            {
                deco = new Decorator();
                deco.type = "pip";
                deco.flip = (xPips[j].att("flip") == "1");
                deco.loc.x = float.Parse(xPips [j].att("x"));
                deco.loc.y = float.Parse(xPips [j].att("y"));
                deco.loc.z = float.Parse(xPips [j].att("z"));
                if (xPips [j].HasAtt("scale"))
                {
                    deco.scale = float.Parse(xPips[j].att("scale"));
                }
                cDef.pips.Add(deco);
            }
        }

        if (xCardDefs [i].HasAtt ("face"))
        {
            cDef.face = xCardDefs [i].att("face");
        }
        cardDefs.Add(cDef);
    }
}
```

2. 플레이
   - _MainCamera 게임 오브젝트 선택 / 인스펙터 창
   - Deck 컴포넌트의 Decorators 와 Card Defs 확인

섹션 04 : 스프라이트 할당

1. Deck 스크립트 수정

```
public class Deck : MonoBehaviour
{
    [Header("Set In Inspector")]
    public Sprite                       suitClub;
    public Sprite                       suitDiamond;
    public Sprite                       suitHeart;
    public Sprite                       suitSpade;

    public Sprite[]                     faceSprites;
    public Sprite[]                     rankSprites;

    public Sprite                       cardBack;
    public Sprite                       cardFront;

    public GameObject                   prefabCard;
    public GameObject                   prefabSprite;
```

2. _MianCamera 게임 오브젝트 / 인스펙터 창 (잠금) / Deck 컴포넌트
   - Suit Club, Suit Diamond, Suit Heart, Suit Spade 에 Club, Diamond, Heart, Spade 스프라이트 설정
   - Face Sprites 에 FaceCard 로 시작하는 이름의 12 개 스프라이트 설정
   - Rank Sprites 에 Letters_0 ~ Letters_15 스프라이트 설정
   - Card Back, Card Front 에 Card_Back, Card_Front 스프라이트 설정

섹션 05 : Card 프리팹 생성

1. 하이어라키 창 / 2D Object / Sprite / Square 생성
   - 이름 변경 : PrefabCard
   - Sprite Renderer / Sprite : Card_Front 스프라이트 설정
   - Card 스크립트 어태치
   - Box Collider 컴포넌트 어태치
   - PrefabCard 게임 오브젝트로 프로젝트 창의 _Prefabs 폴더로 드래그하여 프리팹 생성
   - 하이어라키 창에 있는 PrefabCard 게임 오브젝트 삭제
2. _MainCamera 게임 오브젝트 선택 / 인스펙터 창 / Deck 컴포넌트
   - Prefab Card 와 Prefab Sprite 설정

섹션 06 : 코드로 카드 생성 - 데코레이터

1. Card 스크립트 수정

```
public class Card : MonoBehaviour
{
    [Header("Set Dynamically")]
    public string           suit;
    public int              rank;
    public Color            color = Color.black;
    public string           colS = "Black";

    public List<GameObject> decoGOs = new List<GameObject>();
    public List<GameObject> pipGos = new List<GameObject>();

    public GameObject       back;

    public CardDefinition   def;
}
```

2. Deck 스크립트 수정
   - InitDeck 메서드에 코드 추가

```
    public void InitDeck(string deckXMLText)
    {
        if (GameObject.Find("_Deck") == null)
        {
            GameObject anchorGO = new GameObject("_Deck");
            deckAnchor = anchorGO.transform;
        }

        dictSuites = new Dictionary<string, Sprite>()
        {
            {"C", suitClub},
            {"D", suitDiamond},
            {"H", suitHeart},
            {"S", suitSpade}
        };

        ReadDeck(deckXMLText);

        MakeCards();
    }
```

(다음 페이지에서 계속)

- GetCardDefinitionByRank 메서드 추가

```
public CardDefinition GetCardDefinitionByRank(int rnk)
{
    foreach (var cd in cardDefs)
    {
        if (cd.rank == rnk
        {
            return cd;
        }
    }
    return null;
}
```

- MakeCards 메서드 추가

```
public void MakeCards()
{
    cardNames = new List<string>();
    string[] letters = new string[]{"C", "D", "H", "S"};
    foreach (var s in letters)
    {
        for (int i = 0; i < 13; i++)
        {
            cardNames.Add(s + (i+1));
        }
    }

    cards = new List<Card>();

    for (int i = 0; i < cardNames.Count; i++)
    {
        cards.Add(MakeCard(i));
    }
}
```

(다음 페이지에서 계속)

- MakeCard 메서드 수정

```
private Card MakeCard(int cNum)
{
    GameObject cgo = Instantiate(prefabCard) as GameObject;
    cgo.transform.parent = deckAnchor;
    Card card = cgo.GetComponent<Card>();

    cgo.transform.localPosition = new Vector3((cNum % 13) * 3, cNum / 13 * 4, 0);

    card.name = cardNames[cNum];
    card.suit = card.name[0].ToString();
    card.rank = int.Parse(card.name.Substring(1));

    if (card.suit == "D" || card.suit == "H")
    {
        card.colS = "Red";
        card.color = Color.red;
    }

    card.def = GetCardDefinitionByRank(card.rank);

    AddDecorators(card);

    return card;
}
```

(다음 페이지에서 계속)

- 두 개의 변수와 AddDecorators 메서드 추가

```
private GameObject _tGO = null;
private SpriteRenderer _tSR = null;

private void AddDecorators(Card card)
{
    foreach (var deco in decorators)
    {
        if (deco.type == "suit")
        {
            _tGO = Instantiate(prefabSprite) as GameObject;
            _tSR = _tGO.GetComponent<SpriteRenderer>();
            _tSR.sprite = dictSuites[card.suit];
        }
        else
        {
            _tGO = Instantiate(prefabSprite) as GameObject;
            _tSR = _tGO.GetComponent<SpriteRenderer>();
            _tSR.sprite = rankSprites[card.rank];
            _tSR.color = card.color;
        }

        _tSR.sortingOrder = 1;
        _tGO.transform.parent = card.transform;
        _tGO.transform.localPosition = deco.loc;

        if (deco.flip)
        {
            _tGO.transform.rotation = Quaternion.Euler(0, 0, 180);
        }

        if (deco.scale != 1)
        {
            _tGO.transform.localScale = Vector3.one * deco.scale;
        }

        _tGO.name = deco.type;
        card.decoGOs.Add(_tGO);
    }
}
```

3. 플레이
   - 52 장의 카드가 생성됨
   - 카드 모서리의 데코레이터만 생성됨

섹션 07 : 코드로 카드 생성 – Pip 와 Face

1. Deck 스크립트 수정
   - MakeCard 메서드에 코드 추가

```
        AddDecorators(card);
        AddPips(card);
        AddFace(card);

        return card;
```

   - AddPips 메서드 추가

```
    private void AddPips(Card card)
    {
        foreach (var pip in card.def.pips)
        {
            _tGO = Instantiate(prefabSprite);
            _tGO.transform.parent = card.transform;
            _tGO.transform.localPosition = pip.loc;

            if (pip.flip) {
                _tGO.transform.rotation = Quaternion.Euler(0, 0, 180);
            }

            if (pip.scale != 1) {
                _tGO.transform.localScale = Vector3.one * pip.scale;
            }

            _tGO.name = "pip";
            _tSR = _tGO.GetComponent<SpriteRenderer>();
            _tSR.sprite = dictSuites[card.suit];
            _tSR.sortingOrder = 1;
            card.pipGos.Add(_tGO);
        }
    }
```

(다음 페이지에서 계속)

- AddFace, GatFace 메서드 추가

```
private void AddFace(Card card)
{
    if (card.def.face == "")
    {
        return;
    }

    _tGO = Instantiate(prefabSprite);
    _tSR = _tGO.GetComponent<SpriteRenderer>();
    _tSp = GetFace(card.def.face + card.suit);
    _tSR.sprite = _tSp;
    _tSR.sortingOrder = 1;
    _tGO.transform.parent = card.transform;
    _tGO.transform.localPosition = Vector3.zero;
    _tGO.name = "face";
}

private Sprite GetFace(string faceS)
{
    foreach (var _tSP in faceSprites)
    {
        if (_tSP.name == faceS)
        {
            return _tSP;
        }
    }
    return null;
}
```

2. 플레이
   - 카드 앞면이 모두 그려짐

섹션 08 : 코드로 카드 생성 – 카드 뒷면

1. Card 스크립트에 코드 추가

```
public CardDefinition   def;

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

2. Deck 스크립트 수정
   - 변수 추가

```
[Header("Set In Inspector")]
public bool                         startFaceUp = false;

public Sprite                       suitClub;
```

   - AddBack 메서드 추가

```
private void AddBack(Card card)
{
    _tGO = Instantiate(prefabSprite);
    _tSR = _tGO.GetComponent<SpriteRenderer>();
    _tSR.sprite = cardBack;
    _tGO.transform.parent = card.transform;
    _tGO.transform.localPosition = Vector3.zero;
    _tSR.sortingOrder = 2;
    _tGO.name = "back";
    card.back = _tGO;
    card.faceUp = startFaceUp;
}
```

   - MakeCard 메서드에 코드 추가

```
    AddFace(card);
    AddBack(card);
```

3. 플레이
   - 카드 뒷면이 그려짐
4. _MainCamera 게임 오브젝트 / 인스펙터 창 / Deck 컴포넌트
   - Start Face Up : 체크

섹션 09 : 카드 섞기

1. Deck 스크립트에 메서드 추가

```
static public void Shuffle(ref List<Card> oCards)
{
    List<Card> tCards = new List<Card>();

    int ndx;
    while (oCards.Count > 0)
    {
        ndx = Random.Range(0, oCards.Count);
        tCards.Add(oCards [ndx]);
        oCards.RemoveAt(ndx);
    }
    oCards = tCards;
}
```

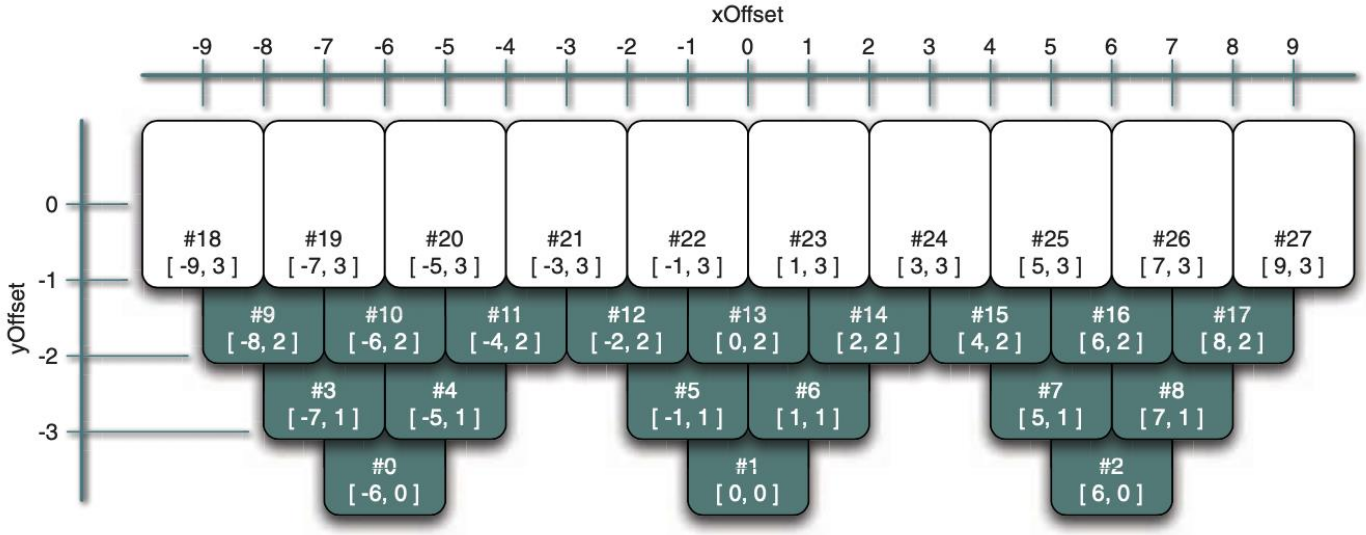
2. Prospector 스크립트의 Start 메서드에 코드 추가

```
void Start ()
{
    deck = GetComponent<Deck>();
    deck.InitDeck(deckXML.text);
    Deck.Shuffle (ref deck.cards);

    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);
    }
}
```

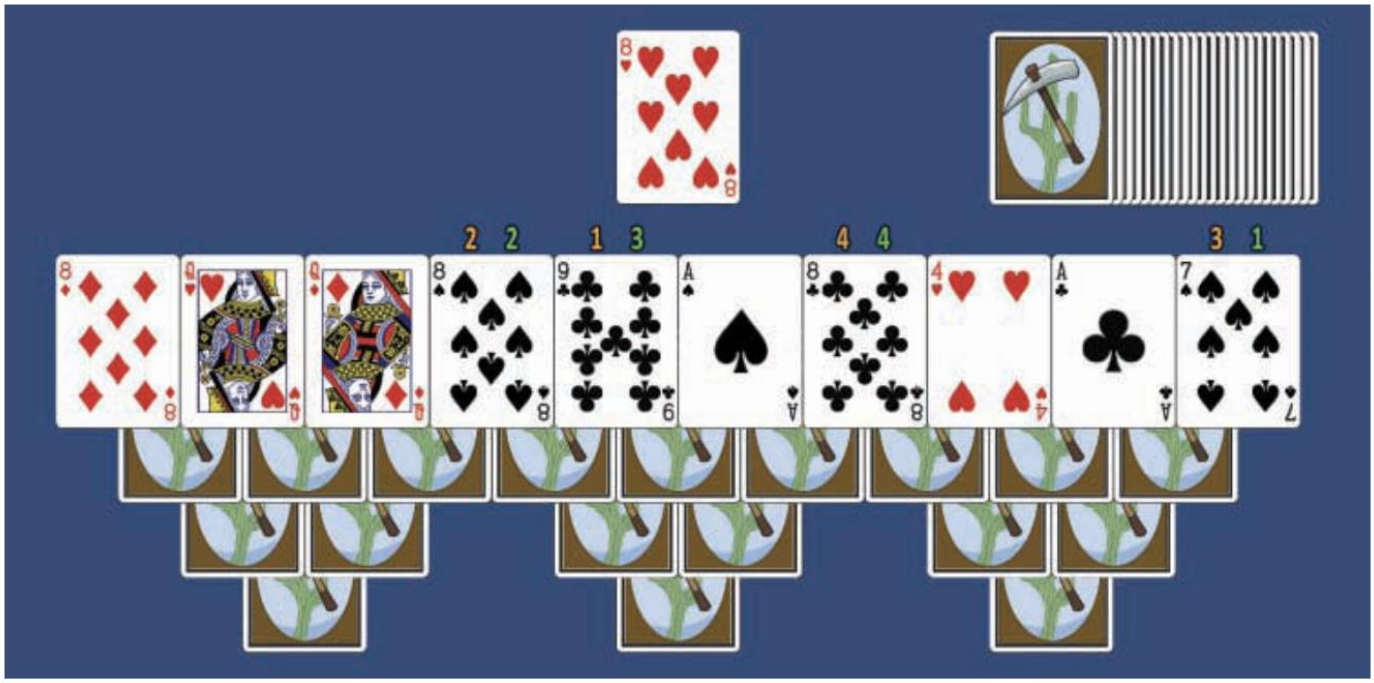
3. 플레이
   - 카드가 랜덤하게 섞여 있음

섹션 10 : 타블로 레이아웃 XML

1. 타블로 구성



2. 레이아웃 완성 그림



3. 프로젝트 창 / Resources / LayoutXML.xml 파일 관찰

(다음 페이지에서 계속)

4. 새로운 스크립트 생성 : LayoutProspector
   - _MainCamera 게임 오브젝트에 어태치

```
[System.Serializable]
public class SlotDefProspector
{
    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 LayoutProspector : MonoBehaviour
{
    public PT_XMLReader         xmlr;
    public PT_XMLHashtable      xml;
    public Vector2              multiplier;

    public List<SlotDefProspector>  slotDefs;
    public SlotDefProspector    drawPile;
    public SlotDefProspector    discardPile;

    public string[] sortingLayerNames = new string[] {
        "Row0", "Row1", "Row2", "Row3", "Draw", "Discard"
    };

    public void ReadLayout(string xmlText)
    {
        xmlr = new PT_XMLReader();
        xmlr.Parse(xmlText);
        xml = xmlr.xml["xml"][0];

        multiplier.x = float.Parse(xml["multiplier"][0].att("x"));
        multiplier.y = float.Parse(xml["multiplier"][0].att("y"));

        SlotDefProspector tSD;
        PT_XMLHashList slotsX = xml["slot"];

        for (int i = 0; i < slotsX.Count; i++)
        {
            tSD = new SlotDefProspector();

            if (slotsX[i].HasAtt("type"))
            {
```

```
                tSD.type = slotsX[i].att("type");
            }
            else
            {
                tSD.type = "slot";
            }

            tSD.x = float.Parse(slotsX[i].att("x"));
            tSD.y = float.Parse(slotsX[i].att("y"));
            tSD.layerID = int.Parse(slotsX[i].att("layer"));
            tSD.layerName = sortingLayerNames[tSD.layerID];

            switch (tSD.type)
            {
            case "slot":
                tSD.faceUp = (slotsX[i].att("faceup") == "1");
                tSD.id = int.Parse(slotsX [i].att ("id"));
                if (slotsX[i].HasAtt("hiddenby"))
                {
                    string[] hidings = slotsX[i].att("hiddenby").Split(',');
                    foreach (var s in hidings)
                    {
                        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;
            }
        }
    }
}
```

(다음 페이지에서 계속)

5. Prospector 스크립트 수정
   - 변수 추가

```
[Header("Set in Inspector")]
public TextAsset deckXML;
public TextAsset layoutXML;

[Header("Set Dynamically")]
public Deck deck;
public LayoutProspector layout;
```

   - Start 메서드 수정 (기존의 for 블록은 주석 처리)

```
void Start ()
{
    deck = GetComponent<Deck>();
    deck.InitDeck(deckXML.text);
    Deck.Shuffle (ref deck.cards);

    // 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<LayoutProspector>();
    layout.ReadLayout(layoutXML.text);
}
```

6. _MainCamera 게임 오브젝트 선택 / 인스펙터 창 / Prospector 컴포넌트
   - Layout XML 에 LayoutXML.xml 애셋 설정
7. 플레이
   - _MainCamera 게임 오브젝트 / 인스펙터 창 / LayoutProspector 컴포넌트
     - Slot Defs 확인

섹션 11 : Card 의 서브 클래스 - CardProspector

1. 스크립트 생성 : CardProspector

```
public enum eCardState
{
    drawpile,
    tableau,
    target,
    discard
}

public class CardProspector : Card
{
    [Header("Set Dynamically: CardProspector")]
    public eCardState           state = eCardState.drawpile;
    public List<CardProspector> hiddenBy = new List<CardProspector>();
    public int                  layoutID;
    public SlotDefProspector    slotDef;
}
```

2. Prospector 스크립트 수정
   - 변수 추가

```
    public LayoutProspector layout;
    public List<CardProspector> drawPile;
```

   - 메서드 추가

```
    List<CardProspector> ConvertListCardsToListCardProspector(List<Card> lCD)
    {
        List<CardProspector> lCP = new List<CardProspector>();
        CardProspector tCP;
        foreach (var tCD in lCD) {
            tCP = tCD as CardProspector;
            lCP.Add(tCP);
        }
        return lCP;
    }
```

   - Start 메서드에 코드 추가

```
        layout.ReadLayout(layoutXML.text);

        drawPile = ConvertListCardsToListCardProspector(deck.cards);
    }
```

3. 플레이
   - _MainCamera 게임 오브젝트 / Prospector 컴포넌트 / DrawPile 관찰
   - 모두 None 임
4. 프로젝트 창 / PrefabCard 프리팹 선택 / 복제
   - 이름 변경 : PrefabCardProspector

- Card 컴포넌트 삭제
- CardProspector 스크립트 컴포넌트 어태치

5. _MainCamera 게임 오브젝트 / Deck 컴포넌트
   - Prefab Card : PrefabCardProspector 로 변경
6. 플레이
   - _MainCamera 게임 오브젝트 / Prospector 컴포넌트 / DrawPile 관찰

섹션 12 : 타블로에 카드 배치

1. Prospector 스크립트 수정
   - 변수 추가

```
[Header("Set in Inspector")]
public TextAsset deckXML;
public TextAsset layoutXML;
public Vector3 layoutCenter;

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

   - Start 메서드에 코드 추가

```
        drawPile = ConvertListCardsToListCardProspector(deck.cards);
        LayoutGame();
    }
```

   - Draw 메서드 추가

```
    CardProspector Draw()
    {
        CardProspector cd = drawPile[0];
        drawPile.RemoveAt(0);
        return cd;
    }
```

(다음 페이지에서 계속)

- LayoutGame 메서드 추가

```
void LayoutGame ()
{
    if (layoutAnchor == null)
    {
        GameObject tGO = new GameObject("_LayoutAnchor");
        layoutAnchor = tGO.transform;
        layoutAnchor.transform.position = layoutCenter;
    }

    CardProspector cp;
    foreach (var tSD in layout.slotDefs)
    {
        cp = Draw ();
        cp.faceUp = tSD.faceUp;
        cp.transform.parent = layoutAnchor;
        cp.transform.localPosition = new Vector3(
            layout.multiplier.x * tSD.x,
            layout.multiplier.y * tSD.y,
            -tSD.layerID);
        cp.layoutID = tSD.id;
        cp.slotDef = tSD;
        cp.state = eCardState.tableau;

        tableau.Add(cp);
    }
}
```

2. 플레이
   - 카드 사이의 Z-order 문제가 있음

섹션 13 : Sorting Layer 설정

1. 메뉴 / Edit / Project Settings / Tags and Layers / Sorting Layers
   - Layer 추가 : Row0, Row1, Row2, Row3, Discard, Draw
2. Card 스크립트 수정

```
public CardDefinition   def;

public SpriteRenderer[] spriteRenderers;

void Start()
{
    SetSortOrder(0);
}

public void PopulateSpriteRenderers()
{
    if (spriteRenderers == null || spriteRenderers.Length == 0)
    {
        spriteRenderers = GetComponentsInChildren<SpriteRenderer>();
    }
}

public void SetSortingLayerName(string tSLN)
{
    PopulateSpriteRenderers();

    foreach (var tSR in spriteRenderers)
    {
        tSR.sortingLayerName = tSLN;
    }
}
```

(다음 페이지에서 계속)

```
public void SetSortOrder(int sOrd)
{
    PopulateSpriteRenderers ();

    foreach (var tSR in spriteRenderers)
    {
        if (tSR.gameObject == this.gameObject)
        {
            tSR.sortingOrder = sOrd;
            continue;
        }

        switch (tSR.gameObject.name)
        {
        case "back":
            tSR.sortingOrder = sOrd + 2;
            break;
        case "face":
        default:
            tSR.sortingOrder = sOrd + 1;
            break;
        }
    }
}

public bool faceUp
```

3. Prospector 스크립트 LayoutGame 메서드의 마지막 부분 수정

```
        cp.state = eCardState.tableau;
        cp.SetSortingLayerName(tSD.layerName);

        tableau.Add(cp);
    }
}
```

4. 플레이

섹션 14 : 클릭 가능한 카드 - Draw Pile

1. Card 스크립트에 메서드 추가

```
virtual public void OnMouseUpAsButton()
{
    print(name);
}
```

2. 플레이
   - 카드를 클릭하면 카드 이름이 출력됨
3. CardProspector 에 메서드 추가

```
override public void OnMouseUpAsButton()
{
    Prospector.S.CardClicked(this);
    base.OnMouseUpAsButton();
}
```

4. Prospector 스크립트 수정
   - MoveToDiscard 메서드 추가

```
void MoveToDiscard(CardProspector cd)
{
    cd.state = eCardState.discard;
    discardPile.Add(cd);
    cd.transform.parent = layoutAnchor;

    cd.transform.localPosition = new Vector3(
        layout.multiplier.x * layout.discardPile.x,
        layout.multiplier.y * layout.discardPile.y,
        -layout.discardPile.layerID + 0.5f);
    cd.faceUp = false;
    cd.SetSortingLayerName(layout.discardPile.layerName);
    cd.SetSortOrder(-100 + discardPile.Count);
}
```

(다음 페이지에서 계속)

- MoveToTarget 메서드 추가

```
void MoveToTarget(CardProspector cd)
{
    if (target != null) MoveToDiscard(target);
    target = cd;
    cd.state = eCardState.target;
    cd.transform.parent = layoutAnchor;

    cd.transform.localPosition = new Vector3(
        layout.multiplier.x * layout.discardPile.x,
        layout.multiplier.y * layout.discardPile.y,
        -layout.discardPile.layerID);
    cd.faceUp = true;
    cd.SetSortingLayerName(layout.discardPile.layerName);
    cd.SetSortOrder(0);
}
```

- UpdateDrawPile 메서드 추가

```
void UpdateDrawPile()
{
    CardProspector cd;

    for (int i = 0; i < drawPile.Count; i++)
    {
        cd = drawPile[i];
        cd.transform.parent = layoutAnchor;

        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;
        cd.state = eCardState.drawpile;

        cd.SetSortingLayerName(layout.drawPile.layerName);
        cd.SetSortOrder(-10 * i);
    }
}
```

(다음 페이지에서 계속)

- LayoutGame 메서드의 마지막 부분에 코드 추가

```
            tableau.Add(cp);
        }

        MoveToTarget(Draw());
        UpdateDrawPile();
    }
```

- CardClicked 메서드 추가

```
    public void CardClicked(CardProspector cd)
    {
        switch (cd.state)
        {
        case eCardState.target:
            break;
        case eCardState.drawpile:
            MoveToTarget(Draw());
            UpdateDrawPile();
            break;
        case eCardState.tableau:
            break;
        }
    }
```

5. 플레이
   - Draw Pile 에 있는 카드 클릭 가능
   - 클릭하면 새로운 Target 이 됨

섹션 15 : 클릭 가능한 카드 - 타블로

1. Prospector 스크립트 수정
   - CardClicked 메서드 수정 (case eCardState.tableau: 블록)

```
case eCardState.tableau:
    bool validMatch = true;
    if (!cd.faceUp)
    {
        validMatch = false;
    }
    if (!AdjacentRank(cd, target))
    {
        validMatch = false;
    }
    if (!validMatch) return;

    tableau.Remove(cd);
    MoveToTarget(cd);
    break;
```

   - AdjacentRank 메서드 추가

```
public bool AdjacentRank(CardProspector c0, CardProspector c1)
{
    if (!c0.faceUp || !c1.faceUp) return false;

    int diff = Mathf.Abs(c0.rank - c1.rank);
    if (diff == 1 || diff == 12) return true;

    return false;
}
```

2. 플레이
   - 타블로의 첫 번째 줄 카드만 버리기 가능
3. Prospector 스크립트 수정
   - LayoutGame 메서드에 코드 추가 (끝 부분)

```
    foreach (var tCP in tableau)
    {
        foreach (var hid in tCP.slotDef.hiddenBy)
        {
            cp = FindCardByLayoutID(hid);
            tCP.hiddenBy.Add(cp);
        }
    }

    MoveToTarget(Draw());
    UpdateDrawPile();
}
```

- FindCardByLayoutID 메서드와 SetTableauFaces 메서드 추가

```
CardProspector FindCardByLayoutID(int layoutID)
{
    foreach (var tCP in tableau)
    {
        if (tCP.layoutID == layoutID) return tCP;
    }
    return null;
}

void SetTableauFaces()
{
    foreach (var cd in tableau)
    {
        bool faceUp = true;
        foreach (var cover in cd.hiddenBy)
        {
            if (cover.state == eCardState.tableau) faceUp = false;
        }
        cd.faceUp = faceUp;
    }
}
```

- CardClicked 메서드 수정 (끝부분)

```
            MoveToTarget(cd);
            SetTableauFaces();
            break;
        }
```

4. 플레이
   - 타블로에 있는 모든 카드를 버릴 수 있음

섹션 16 : 게임 오버 판정

1. Prospector 스크립트 수정
   - CardClicked 메서드 수정 (끝부분)

```
            SetTableauFaces();
            break;
        }
        CheckForGameOver();
    }
```

   - CheckForGameOver 메서드와 GameOver 메서드 추가

```
    void CheckForGameOver()
    {
        if (tableau.Count == 0)
        {
            GameOver(true);
            return;
        }

        if (drawPile.Count > 0) return;

        foreach (var cd in tableau)
        {
            if (AdjacentRank(cd, target)) return;
        }
        GameOver(false);
    }

    void GameOver(bool won)
    {
        if (won)
        {
            print ("Game Over. You won!");
        }
        else
        {
            print ("Game Over. You Lost!");
        }
        SceneManager.LoadScene("__Prospector_Scene_0");
    }
```

2. 플레이

섹션 17 : 점수 계산

1. 스크립트 생성 : ScoreManager
   - _MainCamera 게임 오브젝트에 어태치

```
public enum eScoreEvent
{
    draw,
    mine,
    gameWin,
    gameLoss
}

public class ScoreManager : MonoBehaviour
{
    static private ScoreManager S;

    static public int   SCORE_FROM_PREV_ROUND = 0;
    static public int   HIGH_SCORE = 0;

    [Header("Set Dynamically")]
    public int  chain = 0;
    public int  scoreRun = 0;
    public int  score = 0;

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

        if (PlayerPrefs.HasKey("ProspectorHighScore"))
        {
            HIGH_SCORE = PlayerPrefs.GetInt("ProspectorHighScore");
        }

        score += SCORE_FROM_PREV_ROUND;
        SCORE_FROM_PREV_ROUND = 0;
    }
```

(다음 페이지에서 계속)

```
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)
    {
    case eScoreEvent.draw:
    case eScoreEvent.gameWin:
    case eScoreEvent.gameLoss:
        chain = 0;
        score += scoreRun;
        scoreRun = 0;
        break;

    case eScoreEvent.mine:
        chain++;
        scoreRun += chain;
        break;
    }

    switch (evt)
    {
    case eScoreEvent.gameWin:
        SCORE_FROM_PREV_ROUND = score;
        print("You won this round! Round score: " + score);
        break;
    case eScoreEvent.gameLoss:
        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 + "  chian: " + 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; }
    }
}
```

2. Prospector 스크립트 수정
   - CardClicked 메서드 수정

```
        case eCardState.drawpile:
            MoveToDiscard(target);
            MoveToTarget(Draw());
            UpdateDrawPile();
            ScoreManager.EVENT(eScoreEvent.draw);
            break;
        case eCardState.tableau:
            bool validMatch = true;
            if (!cd.faceUp)
            {
                validMatch = false;
            }
            if (!AdjacentRank(cd, target))
            {
                validMatch = false;
            }
            if (!validMatch) return;

            tableau.Remove(cd);
            MoveToTarget(cd);
            SetTableauFaces();
            ScoreManager.EVENT(eScoreEvent.mine);
            break;
```

- GameOver 메서드 수정

```
void GameOver(bool won)
{
    if (won)
    {
        //print ("Game Over. Youu won!");
        ScoreManager.EVENT(eScoreEvent.gameWin);
    }
    else
    {
        //print ("Game Over. You Lost!");
        ScoreManager.EVENT(eScoreEvent.gameLoss);
    }
    SceneManager.LoadScene("__Prospector_Scene_0");
}
```

3. 플레이

섹션 18 : 점수 UI 를 위한 클래스

1. 스크립트 생성 : FloatingScore

```
using UnityEngine.UI;

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

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

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

    public int score
    {
        get
        {
            return _score;
        }
        set
        {
            _score = value;
            scoreString = _score.ToString("N0");
            GetComponent<Text>().text = scoreString;
        }
    }

    public List<Vector2>    bezierPts;
    public List<float>      fontSizes;
    public float            timeStart = -1f;
    public float            timeDuration = 1f;
    public string           easingCurve = Easing.InOut;

    public GameObject       reportFinishTo = null;

    private RectTransform   rectTrans;
    private Text            uiText;
```

```
    public void Init(List<Vector2> ePts, float eTimeS = 0, float eTimeD = 1)
    {
        rectTrans = GetComponent<RectTransform> ();
        rectTrans.anchoredPosition = Vector3.zero;

        uiText = GetComponent<Text>();

        bezierPts = new List<Vector2>(ePts);

        if (ePts.Count == 1)
        {
            transform.position = ePts[0];
            return;
        }

        if (eTimeS == 0) eTimeS = Time.time;
        timeStart = eTimeS;
        timeDuration = eTimeD;

        state = eFSState.pre;
    }

    public void FSCallback(FloatingScore fs)
    {
        score += fs.score;
    }

    void Update ()
    {
        if (state == eFSState.idle) return;

        float u = (Time.time - timeStart) / timeDuration;
        float uC = Easing.Ease(u, easingCurve);

        if (u < 0)
        {
            state = eFSState.pre;
            uiText.enabled = false;
        }
```

(다음 페이지에서 계속)

```
        else
        {
            if (u >= 1)
            {
                uC = 1;
                state = eFSState.post;
                if (reportFinishTo != null)
                {
                    reportFinishTo.SendMessage ("FSCallback", this);
                    Destroy (gameObject);
                }
                else
                {
                    state = eFSState.idle;
                }
            } else
            {
                state = eFSState.active;
                uiText.enabled = true;
            }

            Vector2 pos = Utils.Bezier(uC, bezierPts);
            rectTrans.anchorMin = rectTrans.anchorMax = pos;
            if (fontSizes != null && fontSizes.Count > 0)
            {
                int size = Mathf.RoundToInt(Utils.Bezier (uC, fontSizes));
                uiText.fontSize = size;
            }
        }
    }
}
```

(다음 페이지에서 계속)

2. 스크립트 생성 : Scoreboard

```
using UnityEngine.UI;

public class Scoreboard : MonoBehaviour
{
    public static Scoreboard S;

    [Header("Set in Inspector")]
    public GameObject prefabFloatingScore;

    [Header("Set Dynamically")]
    [SerializeField] private int _score = 0;
    [SerializeField] private string _scoreString;

    private Transform canvasTrans;

    public int score
    {
        get
        {
            return _score;
        }
        set
        {
            _score = value;
            scoreString = _score.ToString("N0");
        }
    }

    public string scoreString
    {
        get
        {
            return _scoreString;
        }
        set
        {
            _scoreString = value;
            GetComponent<Text> ().text = _scoreString;
        }
    }
```

(다음 페이지에서 계속)

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

    public void FSCallback(FloatingScore fs)
    {
        score += fs.score;
    }

    public FloatingScore CreateFloatingScore(int amt, List<Vector2> pts)
    {
        GameObject go = Instantiate(prefabFloatingScore);
        go.transform.SetParent(canvasTrans, false);
        FloatingScore fs = go.GetComponent<FloatingScore>();
        fs.score = amt;
        fs.reportFinishTo = this.gameObject;
        fs.Init(pts);
        return fs;
    }
}
```

섹션 19 : 점수 UI 출력

1. FloatingScore 프리팹 만들기
   - 메뉴 / GameObject / UI / Text 생성
   - 이름 변경 : PrefabFloatingScore
   - 인스펙터 창
     - Pos X : 0, Pos Y : 0
     - Width : 400, Height : 50
     - Text : 0
     - Font Style : Bold
     - Font Size : 28
     - Alignment : Center, Middle
     - Horizontal Overflow : Overflow, Vertical Overflow : Overflow
     - Color : White
   - FloatingScore 스크립트 어태치
   - PrefabFloatingScore 를 프로젝트 창 / Prefabs 폴더로 드래그하여 프리팹 생성
   - 하이어라키 창에 있는 PrefabFloatingScore 는 삭제
2. Scoreboard 게임 오브젝트 만들기
   - 메뉴 / GameObject / UI / Text 생성
   - 이름 변경 : Scoreboard
   - 인스펙터 창
     - Anchors Min Y : 0.95
     - Anchors Max Y : 0.95
     - Pos X : 0, Pos Y : 0
     - Width : 400, Height : 50
     - Text : 0
     - Font Style : Bold
     - Font Size : 36
     - Alignment : Center, Middle
     - Horizontal Overflow : Overflow, Vertical Overflow : Overflow
     - Color : White
   - Scoreboard 스크립트 어태치
     - Prefab Floating Score : PrefabFloatingScore 설정

(다음 페이지에서 계속)

3. Prospector 스크립트 수정

- Set in Inspector 부분에 변수 추가

```
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);
```

- Set Dynamically 부분에 변수 추가

```
public List<CardProspector> discardPile;
public FloatingScore fsRun;
```

- Start 메서드 수정

```
void Start ()
{
    Scoreboard.S.score = ScoreManager.SCORE;

    deck = GetComponent<Deck>();
```

- CardClicked 메서드 수정

```
    switch (cd.state)
    {
        case eCardState.target:
            break;
        case eCardState.drawpile:
            MoveToTarget(Draw());
            UpdateDrawPile();
            ScoreManager.EVENT(eScoreEvent.draw);
            FloatingScoreHandler(eScoreEvent.draw);
            break;
        case eCardState.tableau:
            bool validMatch = true;
            if (!cd.faceUp)
            {
                validMatch = false;
            }
            if (!AdjacentRank(cd, target))
            {
                validMatch = false;
            }
            if (!validMatch) return;

            tableau.Remove(cd);
            MoveToTarget(cd);
            SetTableauFaces();
            ScoreManager.EVENT(eScoreEvent.mine);
            FloatingScoreHandler(eScoreEvent.mine);
            break;
    }
```

- FloatingScoreHandler 메서드 추가

```
void FloatingScoreHandler(eScoreEvent evt)
{
    List<Vector2> fsPts = new List<Vector2> ();

    switch (evt) {
    case eScoreEvent.draw:
    case eScoreEvent.gameWin:
    case eScoreEvent.gameLoss:
        if (fsRun != null) {
            fsPts.Add(fsPosRun);
            fsPts.Add(fsPosMid2);
            fsPts.Add(fsPosEnd);
            fsRun.reportFinishTo = Scoreboard.S.gameObject;
            fsRun.Init(fsPts, 0, 1);
            fsRun.fontSizes = new List<float>() { 28, 36, 4 };
            fsRun = null;
        }
        break;
    case eScoreEvent.mine:
        FloatingScore fs;
        Vector2 p0 = Input.mousePosition;
        p0.x /= Screen.width;
        p0.y /= Screen.height;
        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;
    }
}
```

- GameOver 메서드에 코드 추가 (ScoreManager.EVENT 메서드 호출 부분 두 군데)

```
        ScoreManager.EVENT(eScoreEvent.gameWin);
        FloatingScoreHandler(eScoreEvent.gameWin);
    }
    else
    {
        //print ("Game Over. You Lost!");
        ScoreManager.EVENT(eScoreEvent.gameLoss);
        FloatingScoreHandler(eScoreEvent.gameLoss);
    }
```

4. 플레이

섹션 20 : 배경 및 UI 추가

1. 하이어라키 창 / 3D Object / Quad 생성
   - 이름 변경 : ProspectorBackground
   - ProspectorBackground Mat.mat 머티리얼 적용
   - 위치 : 0, 0, 0
   - 크기 변경 : 26.667, 20, 1
   - 플레이
2. Prospector 스크립트 수정
   - 변수 추가

```
public Vector2          fsPosEnd = new Vector2(0.5f, 0.95f);
public float            reloadDelay = 2f;
```

   - GameOver 메서드에 코드 추가

```
    //SceneManager.LoadScene("__Prospector_Scene_0");
    Invoke("ReloadLevel", reloadDelay);
}
```

   - ReloadLevel 메서드 추가

```
void ReloadLevel()
{
    SceneManager.LoadScene("__Prospector_Scene_0");
}
```

3. 플레이
   - 게임이 종료되면 2 초 후에 재시작
4. 메뉴 / GameObject / UI / Text 생성
   - 이름 변경 : GameOver
   - Anchors / Min Y : 0.825
   - Anchors / Max Y : 0.825
   - Pivot Y : 0
   - Pos X : 0, PosY : 0
   - Width : 600, Height 100
   - Text : Game Over
   - Font Style : Bold
   - Font Size : 72
   - Alignment : Center, Bottom
   - Color : White
5. GameOver 복제
   - 이름 변경 : RoundResult
   - Pivot Y : 1
   - Pos X : 0, PosY : 0

- Text : You got the high score! (다음 줄) High score : 124
- Font Size : 36
- Alignment : Center, Top

6. RoundResult 복제
   - 이름 변경 : HighScore
   - Anchors Min X : 0.85, Y : 1
   - Anchors Max X : 0.85, Y : 1
   - Pos X : 0, Pos Y : 0
   - Width : 200, Height : 32
   - Text : High Score: 1,000
   - Font Size : 16
   - Alignment : Center, Middle
   - Horizontal Overfloat : Overflow
   - Vertical Overflow : Overflow
7. Prospector 스크립트 수정
   - 변수 추가

```
public float            reloadDelay = 2f;
public Text             gameOverText, roundResultText, highScoreText;
```

   - SetUpUITexts 메서드 추가

```
void SetUpUITexts()
{
    GameObject go = GameObject.Find("HighScore");
    highScoreText = go.GetComponent<Text>();
    int highScore = ScoreManager.HIGH_SCORE;
    string hScore = $"High Score: {highScore:N0}";
    highScoreText.text = hScore;

    go = GameObject.Find("GameOver");
    gameOverText = go.GetComponent<Text>();

    go = GameObject.Find("RoundResult");
    roundResultText = go.GetComponent<Text>();

    ShowResultsUI(false);
}
```

   - ShowResultsUI 메서드 추가

```
void ShowResultsUI(bool show)
{
    gameOverText.gameObject.SetActive(show);
    roundResultText.gameObject.SetActive(show);
}
```

- Awake 메서드에 코드 추가

```
void Awake()
{
    S = this;
    SetUpUITexts();
}
```

- GameOver 메서드 수정

```
void GameOver(bool won)
{
    int score = ScoreManager.SCORE;
    if (fsRun != null) score += fsRun.score;

    if (won)
    {
        gameOverText.text = "Round Over";
        roundResultText.text = "You won this round!\nRound Score: " + score;
        ShowResultsUI(true);
        //print ("Game Over. Youu won!");
        ScoreManager.EVENT(eScoreEvent.gameWin);
        FloatingScoreHandler(eScoreEvent.gameWin);
    }
    else
    {
        gameOverText.text = "Game Over";
        if (ScoreManager.HIGH_SCORE <= score)
        {
            string str = "You got the high score!\nHigh score: " + score;
            roundResultText.text  = str;
        }
        else
        {
            roundResultText.text = "Your final score was: " + score;
        }
        ShowResultsUI(true);
        //print ("Game Over. You Lost!");
        ScoreManager.EVENT(eScoreEvent.gameLoss);
```

8. 플레이

(04 장 Prospector Solitaire 끝)