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

Introduction to Game Design 2ed

05 장 Bartok

유니티 엔진 버전 2019.4.20f1

서진석 jsseo@deu.ac.kr

2021-3-9

섹션 01 : 씬 생성

1. 04 장 Prospector Solitaire 에서 만든 프로젝트 열기
2. __Prospector_Scene_0 복제
   - 이름 변경 : __Bartok_Scene_0
   - 더블 클릭하여 __Bartok_Scene_0 열기
3. 하이어라키 창 / Canvas 아래에 있는 _Scoreboard 와 Highscore 삭제
4. 하이어라키 창 / Canvas 아래에 있는 GameOver 와 RoundResult 비활성화
5. MainCamera 선택
   - Prospector, Score Manager, Layout Prospector 컴포넌트 제거
6. 하이어라키 창 / ProspectorBackground 게임 오브젝트 선택
   - 이름 변경 : BartokBackground
   - 위치 : 0, 0, 1
   - 프로젝트 창 / 머티리얼 생성 : BartokBackground Mat
     - BartokBackground 게임 오브젝트에 적용
7. 하이어라키 창 / Lisht / Directional Light 추가
   - 위치 : -100, -100, 0
8. 게임 창
   - 화면 크기 : 1024 x 768

섹션 02 : Bartok 게임을 위한 Card 클래스

1. __Scripts 폴더에 스크립트 생성 : CardBartok

```
public enum CBState
{
    toDrawpile,
    drawPile,
    toHand,
    hand,
    toTarget,
    target,
    discard,
    to,
    idle
}

public class CardBartok : Card
{
    static public float MOVE_DURATION = 0.5f;
    static public string MOVE_EASING = Easing.InOut;
    static public float CARD_HEIGHT = 3.5f;
    static public float CARD_WIDTH = 2f;

    [Header("Set Dynamically: CardBartok")]
    public CBState state = CBState.drawPile;
    public List<Vector3> bezierPts;
    public List<Quaternion> bezierRots;
    public float timeStart, timeDuration;

    public GameObject reportFinishTo = null;

    public void MoveTo(Vector3 ePos, Quaternion eRot)
    {
        bezierPts = new List<Vector3>();
        bezierPts.Add(transform.localPosition);
        bezierPts.Add(ePos);

        bezierRots = new List<Quaternion>();
        bezierRots.Add(transform.rotation);
        bezierRots.Add(eRot);

        if (timeStart == 0)
        {
            timeStart = Time.time;
        }
```

```
        timeDuration = MOVE_DURATION;
        state = CBState.to;
}

    public void MoveTo(Vector3 ePos)
    {
        MoveTo(ePos, Quaternion.identity);
    }

    void Update ()
    {
        switch (state)
        {
        case CBState.toHand:
        case CBState.toTarget:
        case CBState.toDrawpile:
        case CBState.to:
            float u = (Time.time - timeStart) / timeDuration;
            float uC = Easing.Ease(u, MOVE_EASING);

            if (u < 0)
            {
                transform.localPosition = bezierPts[0];
                transform.rotation = bezierRots[0];
                return;
            }
            else if (u >= 1)
            {
                uC = 1;
                if (state == CBState.toHand) state = CBState.hand;
                if (state == CBState.toTarget) state = CBState.target;
                if (state == CBState.toDrawpile) state = CBState.drawPile;
                if (state == CBState.to) state = CBState.idle;

                transform.localPosition = bezierPts[bezierPts.Count - 1];
                transform.rotation = bezierRots[bezierRots.Count - 1];

                timeStart = 0;

                if (reportFinishTo != null)
                {
                    reportFinishTo.SendMessage("CBCallback", this);
                    reportFinishTo = null;
                }
                else
                {
```

```
                Vector3 pos = Utils.Bezier(uC, bezierPts);
                transform.localPosition = pos;
                Quaternion rotQ = Utils.Bezier(uC, bezierRots);
                transform.rotation = rotQ;
            }
            break;
        }
    }
}
```

2. Bartok 스크립트 생성

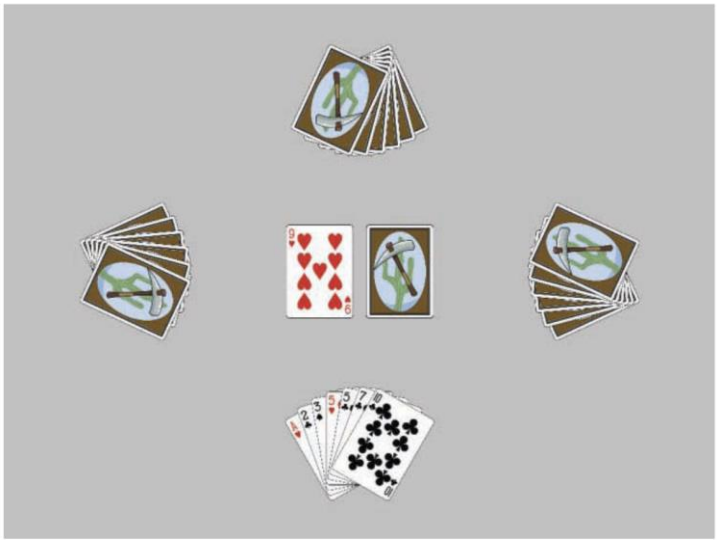
```
using UnityEngine.SceneManagement;

public class Bartok : MonoBehaviour
{
    static public Bartok S;

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

    [Header("Set Dynamically")]
    public Deck deck;
    public List<CardBartok> drawPile;
    public List<CardBartok> discardPile;

    void Awake()
    {
        S = this;
    }

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

3. 프로젝트 창 / PrefabCardProspector 복제
   - 이름 변경 : PrefabCardBartok
   - Box Collider : Is Trigger 체크
   - Box Collider / Size / Z : 0.1
   - CardProspector 컴포넌트 제거
   - CardBartok 컴포넌트 추가
   - Rigidbody 컴포넌트 추가

        - Use Gravity : 체크 해제
        - Is Kinematic : 체크
4. MainCamera 게임 오브젝트 선택
    - Deck 컴포넌트
        - Prefab Card : PrefabCardBartok 으로 교체
        - Start Face Up : 체크
    - Bartok 컴포넌트 추가
        - Deck XML : DeckXML 로 설정
5. 플레이
    - 정상적으로 카드가 생성되고 출력되는지 확인

섹션 03 : Bartok 게임 레이아웃

1. 구현할 레이아웃



2. 프로젝트 창 / Resources 폴더 / LayoutXML 복제
   - 이름 변경 : BartokLayoutXML

```
<xml>
    <multiplier x="1" y="1"/>

    <slot type="drawpile" x="1.5" y="0" xstagger="0.05" layer="1"/>

    <slot type="discardpile" x="-1.5" y="0" layer="2"/>

    <slot type="target" x="-1.5" y="0" layer="4"/>

    <slot type="hand" x="0" y="-8" rot="0" player="1" layer="3"/>
    <slot type="hand" x="-10" y="0" rot="270" player="2" layer="3"/>
    <slot type="hand" x="0" y="8" rot="180" player="3" layer="3"/>
    <slot type="hand" x="10" y="0" rot="90" player="4" layer="3"/>
</xml>
```

3. 스크립트 생성 : LayoutBartok

```
[System.Serializable]
public class SlotDefBartok
{
    public float x;
    public float y;
    public bool faceUp = false;
    public string layerName = "Default";
    public int layerID = 0;
    public int id;
    public float rot;
    public string type = "slot";
```

```
    public Vector2 stagger;
    public int player;
    public Vector3 pos;
}

public class LayoutBartok : MonoBehaviour
{

}
```

4. LayoutBartok 스크립트에 코드 추가

```
public class LayoutBartok : MonoBehaviour
{
    [Header("Set Dynamically")]
    public PT_XMLReader xMLReader;
    public PT_XMLHashtable xml;
    public Vector2 multiplier;

    public List<SlotDefBartok> slotDefs;
    public SlotDefBartok drawPile;
    public SlotDefBartok discardPile;
    public SlotDefBartok target;

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

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

        SlotDefBartok tSD;
        PT_XMLHashList slotX = xml["slot"];

        for (int i = 0; i < slotX.Count; i++)
        {
            tSD = new SlotDefBartok();
            if (slotX[i].HasAtt("type"))
            {
                tSD.type = slotX[i].att("type");
            }
            else
            {
                tSD.type = "slot";
            }

            tSD.x = float.Parse(slotX[i].att("x"));
```

```
            tSD.y = float.Parse(slotX[i].att("y"));
            tSD.pos = new Vector3(tSD.x * multiplier.x, tSD.y * multiplier.y, 0);

            tSD.layerID = int.Parse(slotX[i].att("layer"));
            tSD.layerName = tSD.layerID.ToString();

            switch (tSD.type)
            {
            case "slot":
                break;
            case "drawpile":
                tSD.stagger.x = float.Parse(slotX[i].att("xstagger"));
                drawPile = tSD;
                break;
            case "discardpile":
                discardPile = tSD;
                break;
            case "target":
                target = tSD;
                break;
            case "hand":
                tSD.player = int.Parse(slotX[i].att("player"));
                tSD.rot = float.Parse(slotX[i].att("rot"));
                slotDefs.Add(tSD);
                break;
            }
        }
    }
}
```

- MainCamera 게임 오브젝트에 어태치
- MainCamera 게임 오브젝트 / 인스펙터 창 / Bartok 컴포넌트
  - ■ Layout XML : BartkLayoutXML 로 설정

5. Bartok 스크립트 수정
   - 변수 추가

```
    public List<CardBartok> discardPile;

    private LayoutBartok layout;
    private Transform layoutAnchor;
```

(다음 페이지에서 계속)

- UpgradeCardList 메서드 추가

```
List<CardBartok> UpgradeCardList(List<Card> lCD)
{
    List<CardBartok> lCB = new List<CardBartok>();
    foreach (var tCD in lCD)
    {
        lCB.Add(tCD as CardBartok);
    }
    return lCB;
}
```

- Start 메서드에 코드 추가

```
    Deck.Shuffle(ref deck.cards);

    layout = GetComponent<LayoutBartok>();
    layout.ReadLayout(layoutXML.text);

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

6. 플레이
   - MainCamera 게임 오브젝트 / 인스펙터 창 / Bartok 컴포넌트
     - Draw Pile 확인
   - Layout Bartok 컴포넌트 내용 확인

섹션 04 : 플레이어 클래스

1. 스크립트 생성 : Player

```
using System.Linq;

public enum PlayerType
{
    human,
    ai
}

[System.Serializable]
public class Player
{
    public PlayerType type = PlayerType.ai;
    public int playerNum;
    public BartokSlotDef handSlotDef;
    public List<CardBartok> hand;

    public CardBartok AddCard(CardBartok eCB)
    {
        if (hand == null) hand = new List<CardBartok>();

        hand.Add(eCB);

        return eCB;
    }

    public CardBartok RemoveCard(CardBartok cardBartok)
    {
        if (hand == null || !hand.Contains(cardBartok)) return null;
        hand.Remove(cardBartok);
        return cardBartok;
    }
}
```

2. Bartok 스크립트 수정
   - 변수 추가 (Set in Inspector 부분)

```
    public Vector3 layoutCenter = Vector3.zero;
    public float handFanDegrees = 10f;
```

   - 변수 추가 (Set Dynamically 부분)

```
    public List<CardBartok> discardPile;
    public List<Player> players;
    public CardBartok targetCard;
```

- Start 메서드 마지막에 코드 추가

```
        LayoutGame();
    }
```

- 다음 메서드들 추가

```
    public void ArrangeDrawPile()
    {
        CardBartok tCB;

        for (int i = 0; i < drawPile.Count; i++)
        {
            tCB = drawPile[i];
            tCB.transform.SetParent(layoutAnchor);
            tCB.transform.localPosition = layout.drawPile.pos;

            tCB.faceUp = false;
            tCB.SetSortingLayerName(layout.drawPile.layerName);
            tCB.SetSortOrder(-i * 4);
            tCB.state = CBState.drawPile;
        }
    }

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

        ArrangeDrawPile();

        Player player;
        players = new List<Player>();
        foreach (var tSD in layout.slotDefs)
        {
            player = new Player();
            player.handSlotDef = tSD;
            players.Add(player);
            player.playerNum = tSD.player;
        }
        players[0].type = PlayerType.human;
    }
```

(다음 페이지에서 계속)

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

void Update()
{
    if (Input.GetKeyDown(KeyCode.Alpha1))
    {
        players[0].AddCard(Draw());
    }
    if (Input.GetKeyDown(KeyCode.Alpha2))
    {
        players[1].AddCard(Draw());
    }
    if (Input.GetKeyDown(KeyCode.Alpha3))
    {
        players[2].AddCard(Draw());
    }
    if (Input.GetKeyDown(KeyCode.Alpha4))
    {
        players[3].AddCard(Draw());
    }
}
```

3. 플레이
   - MainCamera 게임 오브젝트 / 인스펙터 창 / Bartok 컴포넌트
     - 1 ~ 4 키를 눌러 Players / Hand 필드 확인

섹션 05 : 카드 펼치기

1. Player 스크립트 수정
   - FanHand 메서드 추가

```
public void FanHand()
{
    float startRot = 0;
    startRot = handSlotDef.rot;
    if (hand.Count > 1)
    {
        startRot += Bartok.S.handFanDegrees * (hand.Count - 1) / 2;
    }

    Vector3 pos;
    float rot;
    Quaternion rotQ;
    for (int i = 0; i < hand.Count; i++)
    {
        rot = startRot - Bartok.S.handFanDegrees * i;
        rotQ = Quaternion.Euler(0, 0, rot);

        pos = Vector3.up * CardBartok.CARD_HEIGHT / 2f;
        pos = rotQ * pos;

        pos += handSlotDef.pos;
        pos.z = -0.5f * i;

        hand[i].transform.localPosition = pos;
        hand[i].transform.rotation = rotQ;
        hand[i].state = CBState.hand;

        hand[i].faceUp = (type == PlayerType.human);
        hand[i].SetSortOrder(i * 4);
    }
}
```

   - AddCard 메서드에 코드 추가

```
hand.Add(eCB);
FanHand();
return eCB;
```

   - RemoveCard 메서드에 코드 추가

```
hand.Remove(cardBartok);
FanHand();
return cardBartok;
```

2. 플레이 : 1~4 키를 눌러 카드가 전달되는지 확인

섹션 06 : LINQ 를 이용한 카드 정렬

1. Player 스크립트의 AddCard 메서드 수정

```
        hand.Add(eCB);

        if (type == PlayerType.human)
        {
            CardBartok[] cards = hand.ToArray();
            cards = cards.OrderBy(cd => cd.rank).ToArray();
            hand = new List<CardBartok>(cards);
        }

        FanHand();
```

2. 플레이
   - 플레이어의 카드가 자동으로 정렬됨

섹션 07 : 카드 이동 애니메이션

1. Player 스크립트의 FanHnad 메서드 수정

```
            pos += handSlotDef.pos;
            pos.z = -0.5f * i;

            hand[i].MoveTo(pos, rotQ);
            hand[i].state = CBState.toHand;

            // hand[i].transform.localPosition = pos;
            // hand[i].transform.rotation = rotQ;
            // hand[i].state = CBState.hand;

            hand[i].faceUp = (type == PlayerType.human);
            hand[i].SetSortOrder(i * 4);
```

2. 플레이

섹션 08 : 카드 나누어 주기

1. Bartok 스크립트 수정
   - 변수 추가

```
public float handFanDegrees = 10f;
public int numStartingCards = 7;
public float drawTimeStagger = 0.1f;
```

   - LayoutGame 메서드 수정

```
    players[0].type = PlayerType.human;

    CardBartok tCB;
    for (int i = 0; i < numStartingCards; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            tCB = Draw();
            tCB.timeStart = Time.time + drawTimeStagger * (i * 4 + j);
            players[(j + 1) % 4].AddCard(tCB);
        }
    }
    Invoke("DrawFirstTarget", drawTimeStagger * (numStartingCards * 4 + 4));
}
```

   - DrawFirstTarget, MoveToTarget 메서드 추가

```
public void DrawFirstTarget()
{
    CardBartok tCB = MoveToTarget(Draw());
}

public CardBartok MoveToTarget(CardBartok tCB)
{
    tCB.timeStart = 0;
    tCB.MoveTo(layout.discardPile.pos + Vector3.back);
    tCB.state = CBState.toTarget;
    tCB.faceUp = true;

    targetCard = tCB;

    return tCB;
}
```

2. 플레이
   - 스프라이트들 간의 정렬문제가 있음

섹션 09 : 2D 스프라이트 정렬

1. 메뉴 / Edit / Project Settings...
   - Tags & Layers 섹션
     - ■ Soring Layers 에 레이어 추가 : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 (기존 Prospector 게임을 위한 Sorting Layers 는 유지)
2. CardBartok 스크립트 수정
   - 변수 추가

```
public float timeStart, timeDuration;
public int eventualSortOrder;
public string eventualSortLayer;
```

   - Update 메서드의 case CBState.to:의 마지막 else 블록에 코드 추가

```
            else
            {
                Vector3 pos = Utils.Bezier(uC, bezierPts);
                transform.localPosition = pos;
                Quaternion rotQ = Utils.Bezier(uC, bezierRots);
                transform.rotation = rotQ;

                if (u > 0.5f)
                {
                    SpriteRenderer sRend = spriteRenderers[0];
                    if (sRend.sortingOrder != eventualSortOrder)
                    {
                        SetSortOrder(eventualSortOrder);
                    }
                    if (sRend.sortingLayerName != eventualSortLayer)
                    {
                        SetSortingLayerName(eventualSortLayer);
                    }
                }
            }
            break;
```

3. Bartok 스크립트 수정
   - MoveToTaregt 메서드 수정

```
        tCB.faceUp = true;

        tCB.SetSortingLayerName("10");
        tCB.eventualSortLayer = layout.target.layerName;
        if (targetCard != null)
        {
            MoveToDiscard(targetCard);
        }

        targetCard = tCB;
```

- MoveToDiscard 메서드 추가

```
public CardBartok MoveToDiscard(CardBartok tCB)
{
    tCB.state = CBState.discard;
    discardPile.Add(tCB);
    tCB.SetSortingLayerName(layout.discardPile.layerName);
    tCB.SetSortOrder(discardPile.Count * 4);
    tCB.transform.localPosition = layout.discardPile.pos + Vector3.back / 2;

    return tCB;
}
```

4. Player 스크립트 수정
   - AddCard 메서드 수정

```
        eCB.SetSortingLayerName("10");
        eCB.eventualSortLayer = handSlotDef.layerName;

        FanHand();
        return eCB;
```

   - FanHand 메서드 수정 (마지막 부분)

```
            hand[i].faceUp = (type == PlayerType.human);
            hand[i].eventualSortOrder = i * 4;
            //hand[i].SetSortOrder(i * 4);
        }
```

5. 플레이
   - 스프라이트 정렬 문제가 해결됨

섹션 10 : 턴 처리

1. Bartok 스크립트 수정
   - 클래스 선언 위에 열거형 데이터 타입 선언

```
using UnityEngine.SceneManagement;

public enum TurnPhase
{
    idle,
    pre,
    waiting,
    post,
    gameOver
}

public class Bartok : MonoBehaviour
```

   - 정적 변수 추가

```
public class Bartok : MonoBehaviour
{
    static public Bartok S;
    static public Player CURRENT_PLAYER;
```

   - Set Dynamically 부분에 변수 추가

```
    public CardBartok targetCard;
    public TurnPhase phase = TurnPhase.idle;
```

   - DrawFirstTarget 메서드 수정

```
        CardBartok tCB = MoveToTarget(Draw());
        tCB.reportFinishTo = this.gameObject;
    }
```

   - CBCallback, StartGame 메서드 추가

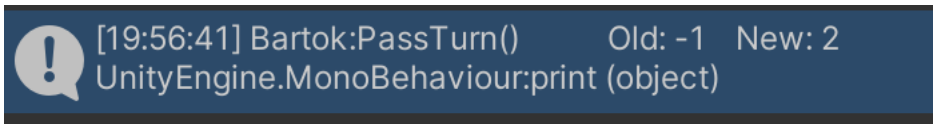
```
    public void CBCallback(CardBartok cardBartok)
    {
        Utils.tr("Bartok:CBCallback()", cardBartok.name);
        StartGame();
    }

    public void StartGame()
    {
        PassTurn(1);
    }
```

(다음 페이지에서 계속)

- PassTurn, ValidPlay 메서드 추가

```
public void PassTurn(int num = -1)
{
    if (num == -1)
    {
        int ndx = players.IndexOf(CURRENT_PLAYER);
        num = (ndx + 1) % 4;
    }
    int lastPlayerNum = -1;
    if (CURRENT_PLAYER != null)
    {
        lastPlayerNum = CURRENT_PLAYER.playerNum;
    }
    CURRENT_PLAYER = players[num];
    phase = TurnPhase.pre;

    // CURRENT_PLAYER.TakeTurn();

    Utils.tr("Bartok:PassTurn()", "Old: " + lastPlayerNum,
             "New: " + CURRENT_PLAYER.playerNum);
}

public bool ValidPlay(CardBartok cardBartok)
{
    if (cardBartok.rank == targetCard.rank) return true;
    if (cardBartok.suit == targetCard.suit) return true;
    return false;
}
```

2. 플레이
   - 다음과 같은 메시지가 콘솔창에 출력되는지 확인



3. Bartok 스크립트의 Update 메서드는 주석 처리

섹션 11 : 턴 전환 피드백을 위한 라이트 효과

1. 하이어라키 창 / Light / Point Light 추가
   - 이름 변경 : TurnLight
   - 위치 : 0, 0, -3
   - Intensity : 2
2. 스크립트 생성 : TurnLight
   - 1 에서 만든 TurnLight 게임 오브젝트에 어태치

```
void Update ()
{
    transform.position = Vector3.back * 3;
    if (Bartok.CURRENT_PLAYER == null)
    {
        return;
    }
    transform.position += Bartok.CURRENT_PLAYER.handSlotDef.pos;
}
```

3. 플레이

섹션 12 : Bartok AI

1. Bartok 스크립트 수정
   - PassTurn 메서드에서 주석 처리된 "CURRENT_PLAYER.TakeTurn()" 문장 주석 해제
2. Player 스크립트 수정
   - FanHand 메서드의 for 블록 수정

```
        pos += handSlotDef.pos;
        pos.z = -0.5f * i;

        if (Bartok.S.phase != TurnPhase.idle)
        {
            hand[i].timeStart = 0;
        }

        hand[i].MoveTo(pos, rotQ);
        hand[i].state = CBState.toHand;
```

   - TakeTurn, CBCallback 메서드 추가

```
public void TakeTurn()
{
    Utils.tr("Player.TakeTurn");

    if (type == PlayerType.human) return;

    Bartok.S.phase = TurnPhase.waiting;

    CardBartok cb;

    List<CardBartok> validCards = new List<CardBartok>();
    foreach (var tCB in hand)
    {
        if (Bartok.S.ValidPlay(tCB))
        {
            validCards.Add(tCB);
        }
    }

    if (validCards.Count == 0)
    {
        cb = AddCard(Bartok.S.Draw());
        cb.callbackPlayer = this;
        return;
    }

    cb = validCards[Random.Range(0, validCards.Count)];
    RemoveCard(cb);
    Bartok.S.MoveToTarget(cb);
```

```
        cb.callbackPlayer = this;
    }

    public void CBCallback(CardBartok tCB)
    {
        Utils.tr("Player.CBCallback()", tCB.name, "Player " + playerNum);
        Bartok.S.PassTurn();
    }
```

3. CardBartok 스크립트 수정
   - 변수 추가

```
    public GameObject reportFinishTo = null;
    [System.NonSerialized]
    public Player callbackPlayer = null;
```

   - Update 메서드의 case CBState.to 블록 수정

```
                if (reportFinishTo != null)
                {
                    reportFinishTo.SendMessage("CBCallback", this);
                    reportFinishTo = null;
                }
                else if (callbackPlayer != null)
                {
                    callbackPlayer.CBCallback(this);
                    callbackPlayer = null;
                }
                else
                {

                }
```

4. 플레이
   - 컴퓨터 플레이어 턴이 진행됨

섹션 13 : 휴먼 플레이어 처리

1. CardBartok 스크립트에 OnMouseUpAsButton 메서드 추가

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

2. Bartok 스크립트에 CardClicked 메서드 추가

```
public void CardClicked(CardBartok tCB)
{
    if (CURRENT_PLAYER.type != PlayerType.human) return;
    if (phase == TurnPhase.waiting) return;

    switch (tCB.state)
    {
    case CBState.drawPile:
        CardBartok cardBartok = CURRENT_PLAYER.AddCard(Draw());
        cardBartok.callbackPlayer = CURRENT_PLAYER;
        Utils.tr("Bartok:CardClicked()", "Draw", cardBartok.name);
        phase = TurnPhase.waiting;
        break;
    case CBState.hand:
        if (ValidPlay(tCB))
        {
            CURRENT_PLAYER.RemoveCard(tCB);
            MoveToTarget(tCB);
            tCB.callbackPlayer = CURRENT_PLAYER;
            Utils.tr("Bartok:CardClicked()", "Play", tCB.name,
                        targetCard.name + " is target");
            phase = TurnPhase.waiting;
        }
        else
        {
            Utils.tr("Bartok:CardClicked()", "Attempted to Play",
                        tCB.name, targetCard.name + " is target");
        }
        break;
    }
}
```

3. 플레이
   - 플레이어 처리 가능

섹션 14 : Draw Pile 카드 소진 처리

1. CardBartok 스크립트의 Draw 메서드 수정

```
public CardBartok Draw()
{
    if (drawPile.Count == 0)
    {
        int ndx;
        while (discardPile.Count > 0)
        {
            ndx = Random.Range(0, discardPile.Count);
            drawPile.Add(discardPile[ndx]);
            discardPile.RemoveAt(ndx);
        }
        ArrangeDrawPile();

        float t = Time.time;
        foreach (var tCB in drawPile)
        {
            tCB.transform.localPosition = layout.discardPile.pos;
            tCB.callbackPlayer = null;
            tCB.MoveTo(layout.drawPile.pos);
            tCB.timeStart = t;
            t += 0.02f;
            tCB.state = CBState.toDrawpile;
            tCB.eventualSortLayer = "1";
        }
    }

    CardBartok cd = drawPile[0];
    drawPile.RemoveAt(0);
    return (cd);
}
```

2. 플레이

섹션 15 : UI 추가

1. Canvas 에 있는 GameOver 와 RoundResult 게임 오브젝트 활성화
   - GameOver 게임 오브젝트 편집
     - Anchors Min Y : 0.65
     - Anchors Max Y : 0.65
     - Pivot Y : 0.5
     - Width : 800, Height : 120
     - Pos X : 0, Pos Y : 0
     - Font Size : 100
     - Alignment : Center, Middle
     - Color : Black
   - RoundResult 게임 오브젝트 편집
     - Anchors Min Y : 0.35
     - Anchors Max Y : 0.35
     - Pivot Y : 0.5
     - Width : 800, Height : 80
     - Pos X : 0, Pos Y : 0
     - Font Size : 60
     - Text : Player 2 won
     - Alignment : Center, Middle
     - Color : Black
2. 스크립트 생성 : GameOverUI
   - GameOver 게임 오브젝트에 어태치

```
using UnityEngine.UI;

public class GameOverUI : MonoBehaviour
{
    private Text txt;

    void Awake()
    {
        txt = GetComponent<Text>();
        txt.text = "";
    }
```

(다음 페이지에서 계속)

```
    void Update()
    {
        if (Bartok.S.phase != TurnPhase.gameOver)
        {
            txt.text = "";
            return;
        }
        if (Bartok.CURRENT_PLAYER == null) return;
        if (Bartok.CURRENT_PLAYER.type == PlayerType.human)
        {
            txt.text = "You won!";
        }
        else
        {
            txt.text = "Game over";
        }
    }
}
```

3. 스크립트 생성 : RoundResultUI
   - RoundResult 게임 오브젝트에 어태치

```
using UnityEngine.UI;

public class RoundResultUI : MonoBehaviour
{
    private Text txt;

    void Awake()
    {
        txt = GetComponent<Text>();
        txt.text = "";
    }

    void Update()
    {
        if (Bartok.S.phase != TurnPhase.gameOver)
        {
            txt.text = "";
            return;
        }
```

(다음 페이지에서 계속)

```
        Player cP = Bartok.CURRENT_PLAYER;
        if (cP == null || cP.type == PlayerType.human)
        {
            txt.text = "";
        }
        else
        {
            txt.text = "Player " + (cP.playerNum) + " won";
        }
    }
}
```

4. 플레이
   - 게임을 시작하면 UI 의 텍스트가 사라짐

섹션 16 : 게임 오버 처리

1. Bartok 스크립트 수정
   - PassTurn 메서드 수정

```
if (CURRENT_PLAYER != null)
{
    lastPlayerNum = CURRENT_PLAYER.playerNum;
    if (CheckGameOver())
    {
        return;
    }
}
```

   - CheckGameOver, RestartGame 메서드 추가

```
public bool CheckGameOver()
{
    if (drawPile.Count == 0)
    {
        List<Card> cards = new List<Card>();
        foreach (var cb in discardPile)
        {
            cards.Add(cb);
        }
        discardPile.Clear();
        Deck.Shuffle(ref cards);
        drawPile = UpgradeCardList(cards);
        ArrangeDrawPile();
    }

    if (CURRENT_PLAYER.hand.Count == 0)
    {
        phase = TurnPhase.gameOver;
        Invoke("RestartGame", 1);
        return true;
    }
    return false;
}

public void RestartGame()
{
    CURRENT_PLAYER = null;
    SceneManager.LoadScene("__Bartok_Scene_0");
}
```

2. 플레이

(05 장 Bartok 끝)