게임 메인 로직, 다이스와 리롤, 턴 진행

상태	DONE
팀	개발
우선순위	우선순위 1
담당자	김 김휘년
생성자	김 김휘년
# Completed	1
# Goal	1
∑ 진척도	✅

▼ **참고사항**

1. 과정 : 개발/디자인 과정 중, 생각, 기능, 흐름, diagram 등을 기록. **(선택)**
2. 기획 : 해당 Backlog 의 자세한 기획 및 기획 내의 명세 / 요구 사항을 기록. **(선택)**
3. **필요 결과물** : 해당 Backlog 에서 나와야 하는 필요 결과물을 정리 및 완수를 표시.
    a. 완료 한 경우, 그에 대한 증빙을 동영상 / 스크린샷 등으로 만들어 첨부할 것.
    b. 담당자를 멘션하거나, 링크를 첨부하여 확인 받을 수 있도록 할 것.
    c. 만약 해당 결과물을 다른 사람이 사용해야 하는 경우, 📜 Mannual 내에 해당 항목을 작성하고, 이를 증빙에 함께 첨부할 것.

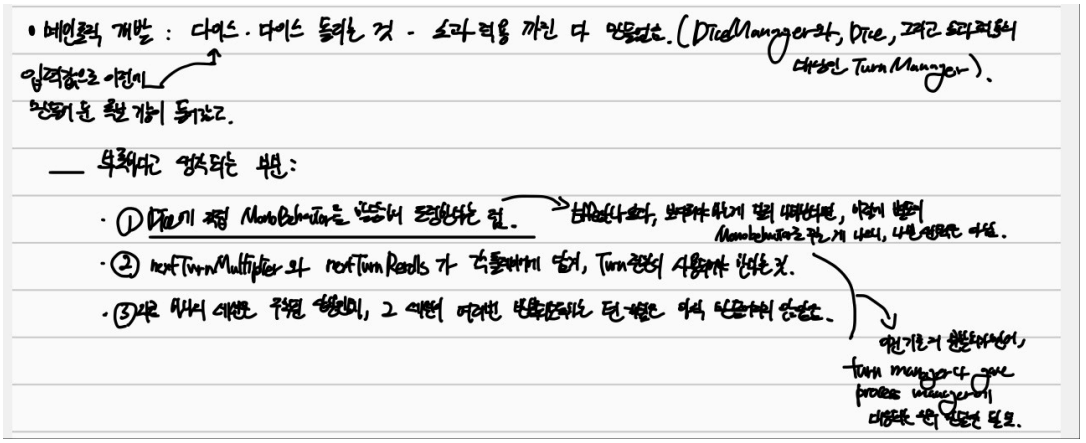
1. 기획 (전략)

1. 에 나와 있는 내용 중, 아이템을 제외한 나머지 **효과적용**까지 모두 만드는 것을 염두해 둠.
    a. 이렇게 생각하면, 여기에 턴 진행까지 포함될 것 같긴 한데. 리롤과 게임 메인 흐름이랑. 턴 전환이 약간 별개이긴 해서. 일단 구분해 둠. 만약 합치는 게 용이하다 싶으면 합치자.

2. 과정

1 - 리롤 및 다이스 관련 기능 제작

1. 시각화는 다르게 일어나겠지만, 결국 Dice 굴림의 규칙 자체는 동일. 이에 따라, 하나의 DiceManager (굴림, 고정, 효과 파악 등을 갖고 있는) 만을 갖고 있으면 됨.
2. **단, 다음 턴에 효과가 전이 되는 Multiplier 나, Rerolls는 플레이어가 갖고 있도록 해야. DiceManager 는 턴이 되었을 때의 굴림과, 효과만 지니고 있으면 충분!**
3. 현재 만든 결과 :
    a. Input 이 족보 데이터와 다이스.
    b. 다이스 Roll, Lock, 효과 적용
    c. Output 이 효과 적용인데, 그 효과 적용 대상은 Character
    - 현재 코드 상으로 보면, DiceManager, Dice, Character를 추가로 제작 완료 함.
4. 현재 마음에 안 드는 부분 개선 :



- 결국 지금 제작한 내용이, 한 턴의 다이스 굴림 및 적용만 제작되어서 그런 거고. 턴 개념까지 적용하면, 위의 불만은 모두 해결.
- 해결 전, 효과 적용에 사용되는, 저주 관련 로직을 먼저 만들고 넘어가는 게 깔끔할 것 같음.

2 - 상대에게 주는 족보 기반 효과 먼저 완결

1. 족보 관련 효과 자체가 각각의 개별 로직으로 구현되다 보니, 바로 스크립트 단에서 분리될 것. 이에 따라 SO로 나누기 보단, enum으로 나누고, 이에 따른 효과를 바로 구분하는 것이 나을 것으로 보임.
2. 지금 구현이 안 되어 있는 것들을 살펴보면 :
    a. ~~Item~~
    b. reroll & multiplier : 이건 Player 한테 값을 더하거나 해야 할 듯.
        - 이것도 상대턴에 적용되는 개념이므로, Turn 을 먼저 만들고 처리하는 게 나을 듯.
    c. ~~Normal Curse 와 Powerful Curse 는 저주 효과를 만들어서~~, Process Curse 를 구현해야 할 것으로 보임.
        - 그냥 Enum으로 만들려고 생각해보니깐, Description 이 있긴 있어야 되네 ⇒ 근데 SO나 enum을 포함하는 다른 class 를 만든다 쳤을 때, 확실히 좀 애매하긴 함! ⇒ 그냥 enum 으로 해놨다가, 나중에 UI나 Descrption 필요할 때, Localization 으로 UI 단에서 따로 불러오도록 처리하거나 하는 게 나을 듯.
        - 아이템 사용 로직도 위의 저주 부분 고려한 것과 비슷해서, enum으로 두고 체크하는 게 낫긴 할 듯.
        - Process Curse는, 각 Curse 보다 다르긴 하지만, Process 개념이 적용되어야 하므로, 지금 만들기엔 무리가 있음.

3 - 턴 기반 동작

1. 현재 _nextTurnExtraRerolls 와, _nextTurnMultiplier 는 바로 다음 턴에 적용되는데, 플레이어와 적의 턴이 번갈아 진행되므로, 이를 자신의 턴에 적용되도록 만들어야 함.
2. 지금 만들어져 있지 않은 것은, 첫 게임 시작 / (이후 플레이어가 UI 를 통해 조작하는 과정이 계속 나오고 ⇒ UI를 통한 조작은 지금 없지만, DiceManager 가 여기 UI 와 상호작용 하는 UI 일 것.) ⇒ 이 부분이 끝나기 까지 기다려야 함. / 이후 모든 게 끝나 확정되면, 그 확정 족보 기반으로 효과 적용하고(DiceManager의 CheckCurrentCombination 과, ApplyEffects 를 이용하여), 효과 적용이 끝나고 나면 (이것도 나중에 Animation 쓰거나 하면, 그거 끝나는 타이밍 까지 좀 기다려야 할 듯) 상대의 턴으로 넘어가는 방식으로.
    - 또한, Player 의 경우 기다리는데, 만약에 AI 의 턴이면, AI가 턴을 시작하는 것을 기다려야 함.

- 기존에 만들어놨던 TurnManager 가 있긴 해서, 이걸 사용할 수는 없는지 한번 체크.
  - ⇒ 어느 정도 차용해서 만드는 것이 가능할 듯.

    ▼ 현재 코드 전문

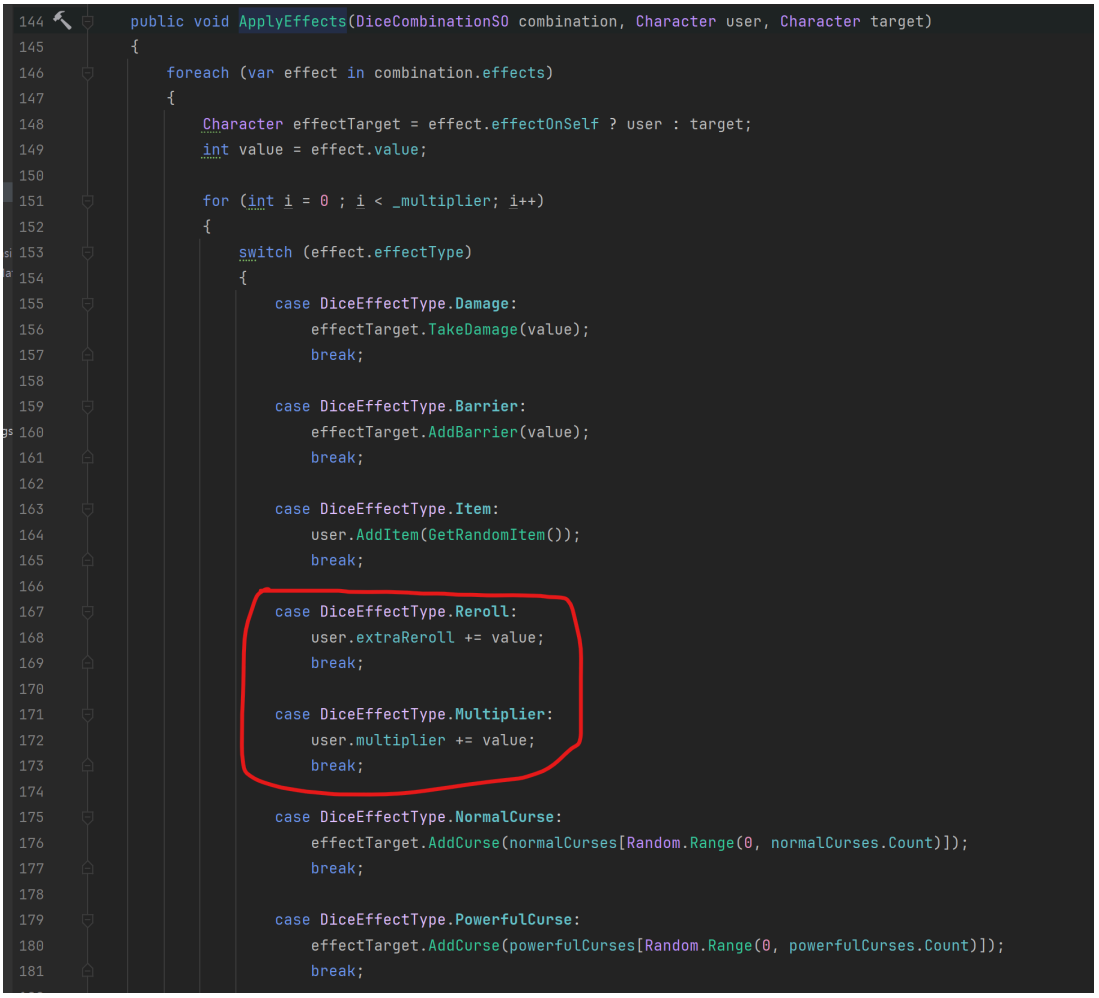
```
using UnityEngine;
using UnityEngine.Events;

public class TurnManager : MonoBehaviour
{
    public Character player;
    public Character enemy;
    public Character CurrentCharacter { get; private set; }
    public DiceManager diceManager;

    public enum TurnPhase
    {
        Start,        // 턴 시작 - 주사위 굴리기 전
        Rolling,      // 첫 번째 주사위 굴린 후 부터, 마지막 주사위 굴릴 때까지
        Resolution,   // 주사위 끝까지 굴린 후, 족보 효과 적용 중.
        End           // 턴 종료 - 족보 효과 적용 후
    }

    public TurnPhase CurrentPhase { get; private set; }
    public UnityEvent<Character> onTurnStart = new UnityEvent<Cl
    public UnityEvent<Character> onTurnEnd = new UnityEvent<Ch
    public UnityEvent onWin = new UnityEvent();
    public UnityEvent onLose = new UnityEvent();

    public void StartGame()
    {
        CurrentCharacter = player; // Player goes first
        StartTurn();
    }

    public void StartTurn()
    {
```

```
        CurrentPhase = TurnPhase.Start;
        CurrentCharacter.ProcessCurses();
        diceManager.StartNewTurn();
        onTurnStart.Invoke(CurrentCharacter);
    }

    public void ProcessDiceCombination()
    {
        var combination = diceManager.CheckCurrentCombination();
        diceManager.ApplyEffects(combination, CurrentCharacter, Cu

        if (IsGameOver())
        {
            EndGame();
            return;
        }

        EndTurn();
    }

    private bool IsGameOver()
    {
        return player.currentHealth <= 0 || enemy.currentHealth <= 0;
    }

    private void EndGame()
    {
        // Handle game over logic
        Debug.Log($"Game Over! Winner: {(player.currentHealth > 0
        if (player.currentHealth > 0)
            onWin.Invoke();
        else
            onLose.Invoke();
    }

    public void EndTurn()
    {
        CurrentPhase = TurnPhase.End;
```

```
        onTurnEnd.Invoke(CurrentCharacter);

        CurrentCharacter = (CurrentCharacter == player) ? enemy : p
        StartTurn();
    }
}
```

1. 이전 1번 항목의 경우, StartNewTurn 이 TurnManager 에서 실행되기 때문에. 여기서 현재 Player 의 정보를 넘겨주면 충분할 것으로 보임.

    ▼ 실제로 그렇게 하는 구조로 만들었음.



```
public void ApplyEffects(DiceCombinationSO combination, Character user, Character target)
{
    foreach (var effect in combination.effects)
    {
        Character effectTarget = effect.effectOnSelf ? user : target;
        int value = effect.value;

        for (int i = 0 ; i < _multiplier; i++)
        {
            switch (effect.effectType)
            {
                case DiceEffectType.Damage:
                    effectTarget.TakeDamage(value);
                    break;

                case DiceEffectType.Barrier:
                    effectTarget.AddBarrier(value);
                    break;

                case DiceEffectType.Item:
                    user.AddItem(GetRandomItem());
                    break;

                case DiceEffectType.Reroll:
                    user.extraReroll += value;
                    break;

                case DiceEffectType.Multiplier:
                    user.multiplier += value;
                    break;

                case DiceEffectType.NormalCurse:
                    effectTarget.AddCurse(normalCurses[Random.Range(0, normalCurses.Count)]);
                    break;

                case DiceEffectType.PowerfulCurse:
                    effectTarget.AddCurse(powerfulCurses[Random.Range(0, powerfulCurses.Count)]);
                    break;
```

2. 현재 TurnManager 를 만들었으므로, 이전 언급한 구조와 비슷하게 동작하도록 제작 완료.
    a. 첫 게임 시작
    b. 플레이어가 UI 를 통해 조작하는 과정이 계속 나오고 ⇒ UI를 통한 조작은 지금 없지만, DiceManager 가 여기 UI 와 상호작용 하는 UI 일 것 ⇒ **이 부분이 끝난 후.**
    c. 확정 족보 기반으로 효과 적용 (DiceManager의 CheckCurrentCombination 과, ApplyEffects 를 이용)
    d. 효과 적용이 끝나고 나면 (이것도 나중에 Animation 쓰거나 하면, 그거 끝나는 타이밍 까지 좀 기다려야 할 듯 ⇒ 이 기다리는 건 아직 안 만들어 지긴 했음!) 상대의 턴으로 넘어가는 방식.

4 - 이제 동작 및 테스팅 진행하려면?

- 위 시스템을 이용하는 두 가지 만들어야 할 듯.
    1. 시각화 UI (간단하게 표현) :

2. 적 AI

- 이후, 별개적으로 사용하는 “아이템, 획득, 사용 + 아이템 사용 AI ” 제작 필요.

⇒ 이건 각 별개 문서에서 제작 진행하는 게 나을 듯.

---

3. 필요 결과물

☑️ ~~지금 내부 시스템 만들고 있어서, 실행 파악만 하고, 별도의 통합 테스팅은 안 거치고 있음. 시각화 UI 까지 만들고 나서 영상으로 촬영 가능할 듯.~~