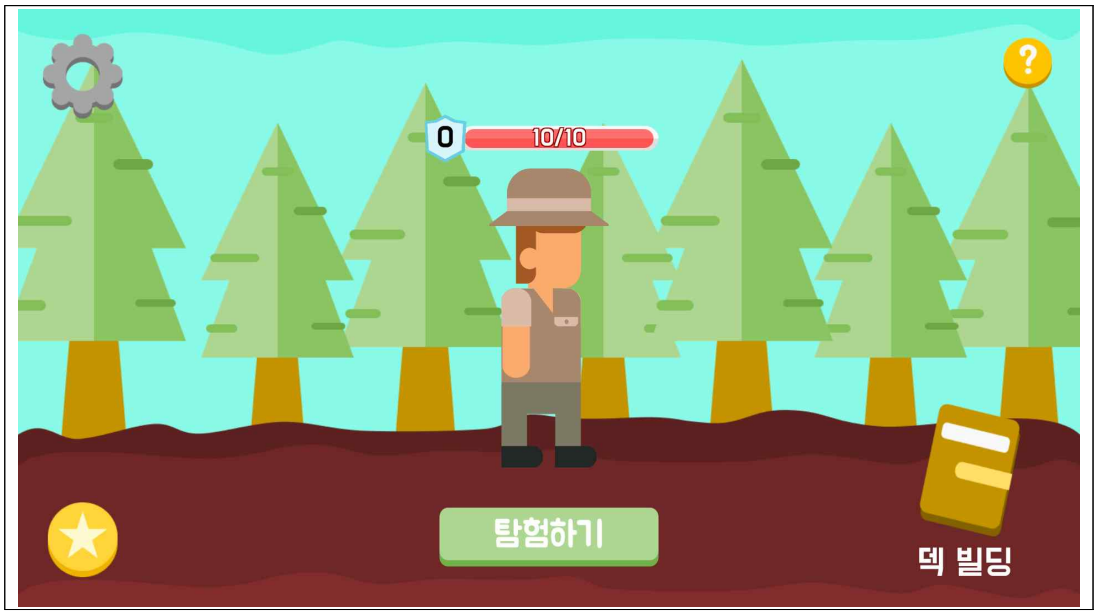
[표지]

2019 선린 모바일 콘텐츠 개발대회
[게임(디자인)]분야
출품작 설명서

작품명	Draft The Fate



참 가 분 야	게임(디자인)
팀 명	불타는 버스
팀대표	학번 : 30414 / 이름 : 송영범
팀대표 핸드폰	
팀 원(학번/이름)	학번 : 30420 / 이름 : 이민용
	학번 : 30622 / 이름 : 한지윤

[본문]

1. 작품 개요

가. 개발 동기 및 기대효과

2019년 1월 'Slay the Spire'가 스팀에 출시되었고 로그라이크와 카드 게임 형식의 참신함이 재미를 불러일으켜 스팀 내에서 압도적으로 긍정적인 평가를 받고 메타크리틱 스코어 89점을 기록했다. 또한 다키 스톤이라는 명칭으로 스트리머들 사이에서도 유행하여 주목을 받았으며, 새로운 장르인 로그라이크식 덱 빌딩 카드 게임 즉 'STS' 장르를 새로 개척한 게임으로 많은 찬사를 받았다. 우리에게도 굉장히 흥미로운 게임이었고 운에 따른 요소가 강하면서도 덱 빌딩을 통해 이를 통제할 수 있다는 점이 마음에 들었다. 또한 보드게임 '미니빌'에서도 많은 영감을 얻었다. 미니빌은 간단한 구성물과 주사위 2개로만 이루어져 있는 게임임에도 불구하고 주사위라는 변칙적인 요소로 다양한 플레이를 즐길 수 있는 점이 매우 인상 깊었다. 우리는 이 두게임을 조합하여 운에 의존하되 덱을 어떻게 짜는지에 따라 게임의 양상이 달라지는 STS 게임을 만들고 싶어 'DraftTheFate'을 기획 개발하게 되었다.

나. 프로그램 개발 환경 및 사용 환경

개발환경	OS	Windows 10 Home
	플랫폼(cpu/ram)	Intel Core I7-7700HQ / 16GB
	프로그래밍 제작툴	Visual Studio 2017
	그래픽 프로그램	Microsoft PowerPoint
	기타 도구	Unity 2018.2.19f1

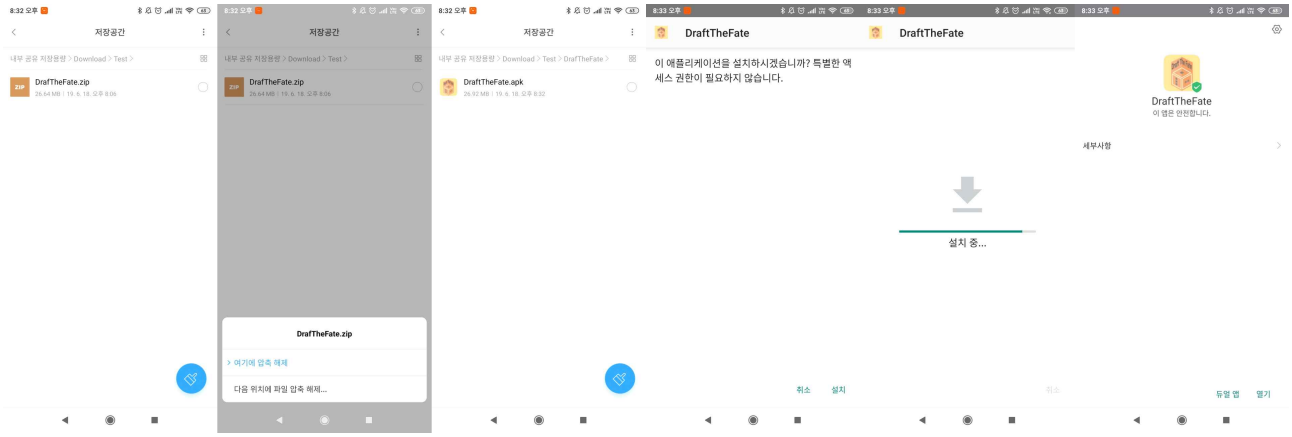
사용환경	OS	Android 8.0 Oreo Android 9.0 Pie
	플랫폼(cpu/ram)	Samsung Exynos 7880 Soc / 3GB Qualcomm Snapdragon 636 SDM636 Platform / 4GB
	기타	Samsung Galaxy A5 (2017) / SM-A520 Xiaomi Redmi Note 5 / M1803E7SG

다. 제작과정

	날짜 또는 기간	제작과정, 협업내역	사용 프로그램
개발일정	6.5 ~ 6.6	게임 기획 재설정, 기획서 작성	
	6.7 ~ 6.8	아웃게임, 인게임 UI 디자인 및 적용	Microsoft PowerPoint
	6.9 ~ 6.10	화살표, 주사위 등 기타 오브젝트 제작	
	6.11 ~ 6.13	카드 움직임, 카드 효과 제작, 카드 디자인	
	6.14	플레이어 및 적 캐릭터 디자인	
	6.15	게임 매니저 제작	Visual Studio 2017 Unity 2018.2.19f1
	6.16 ~ 6.17	아웃게임 이펙트 제작	
	6.18	카드 종류 추가	
	6.19	빌드 테스트 및 기획서 작성	
	6.20	카드 및 몬스터 추가	
	6.21	기획서 마감	

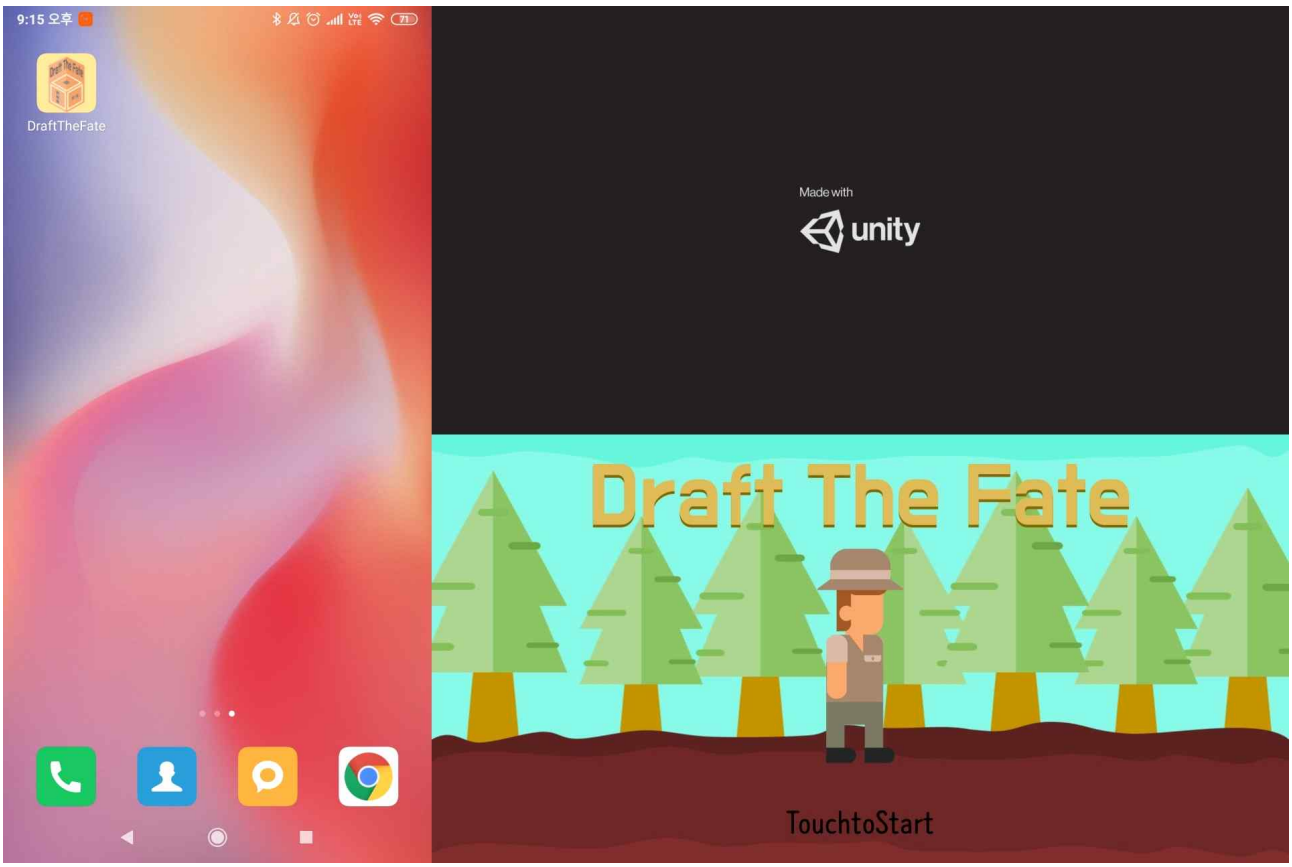
2. 설치 및 실행 방법

가. 프로그램 설치 방법



본 게임의 압축을 풀고 'DraftTheFate.apk'를 설치한다.

나. 프로그램 실행 방법

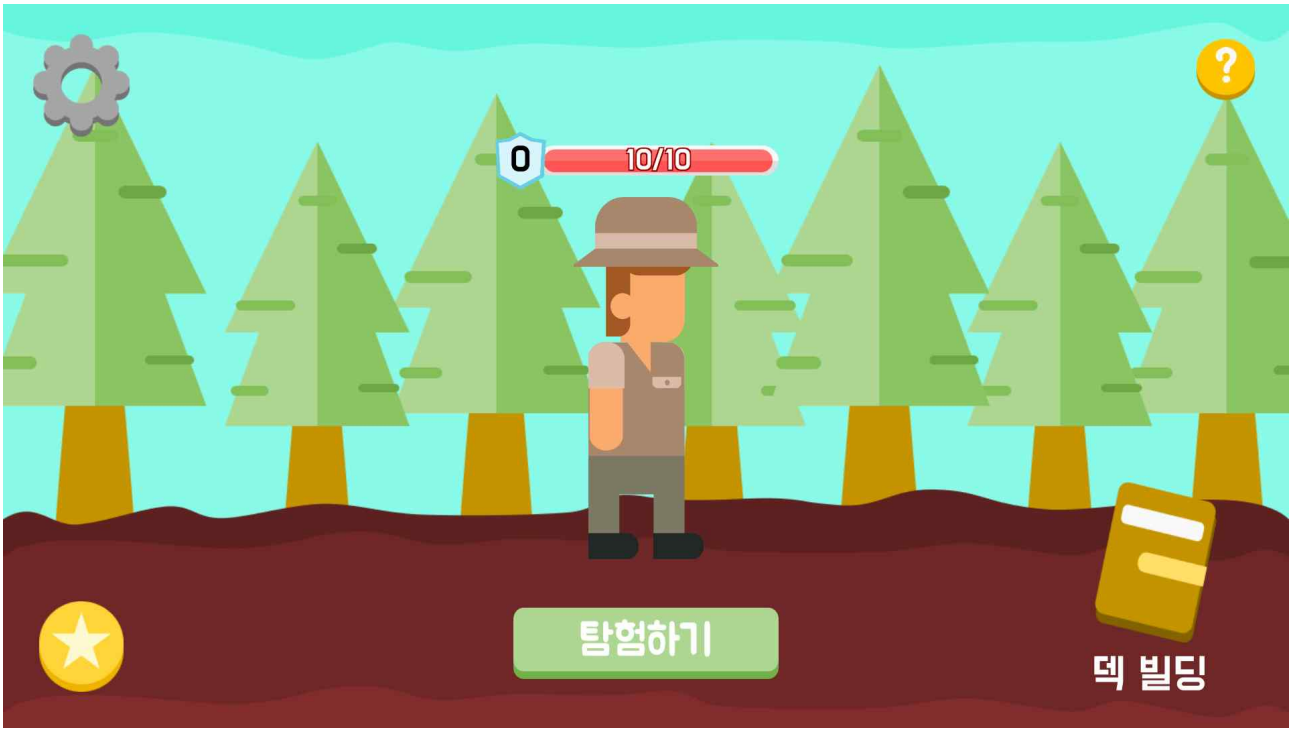


설치된 'DraftTheFate'를 실행한다.

3. 작품 소개

가. 사용 설명서

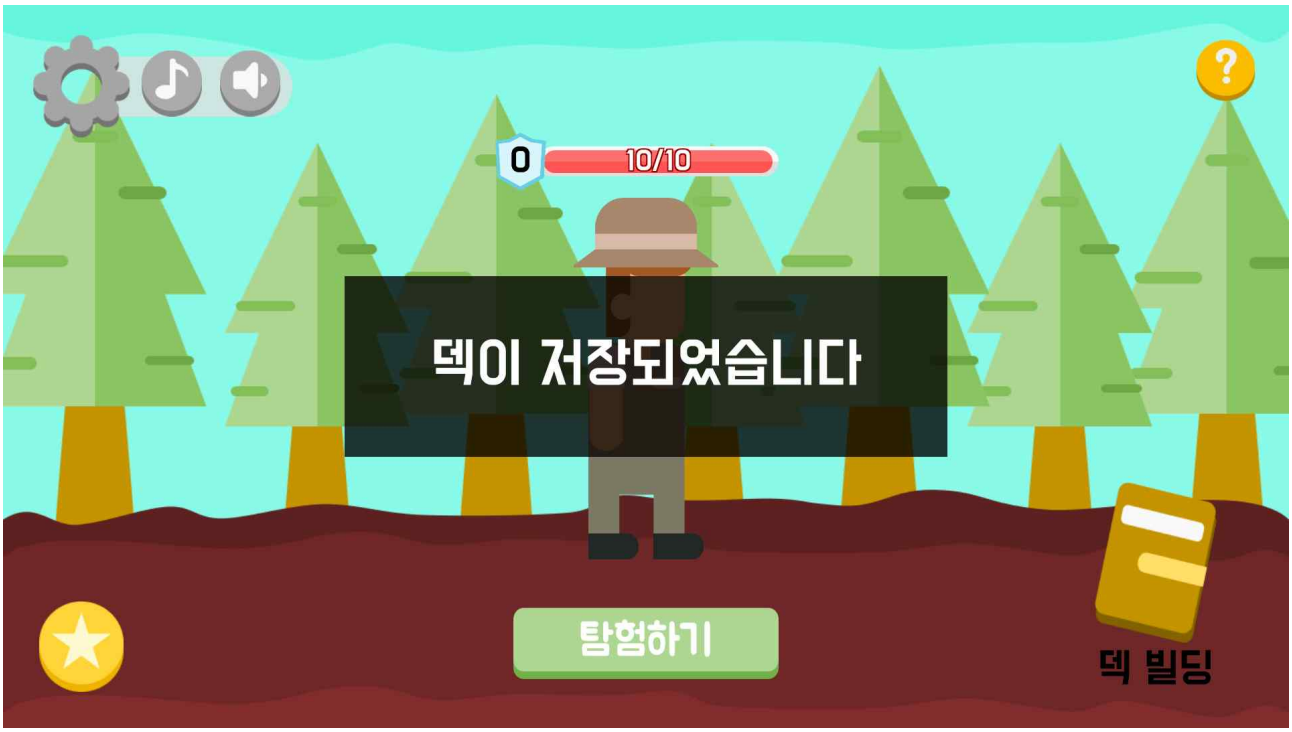
게임의 인트로 화면이다. 화면을 터치하여 메인 화면으로 진입한다.



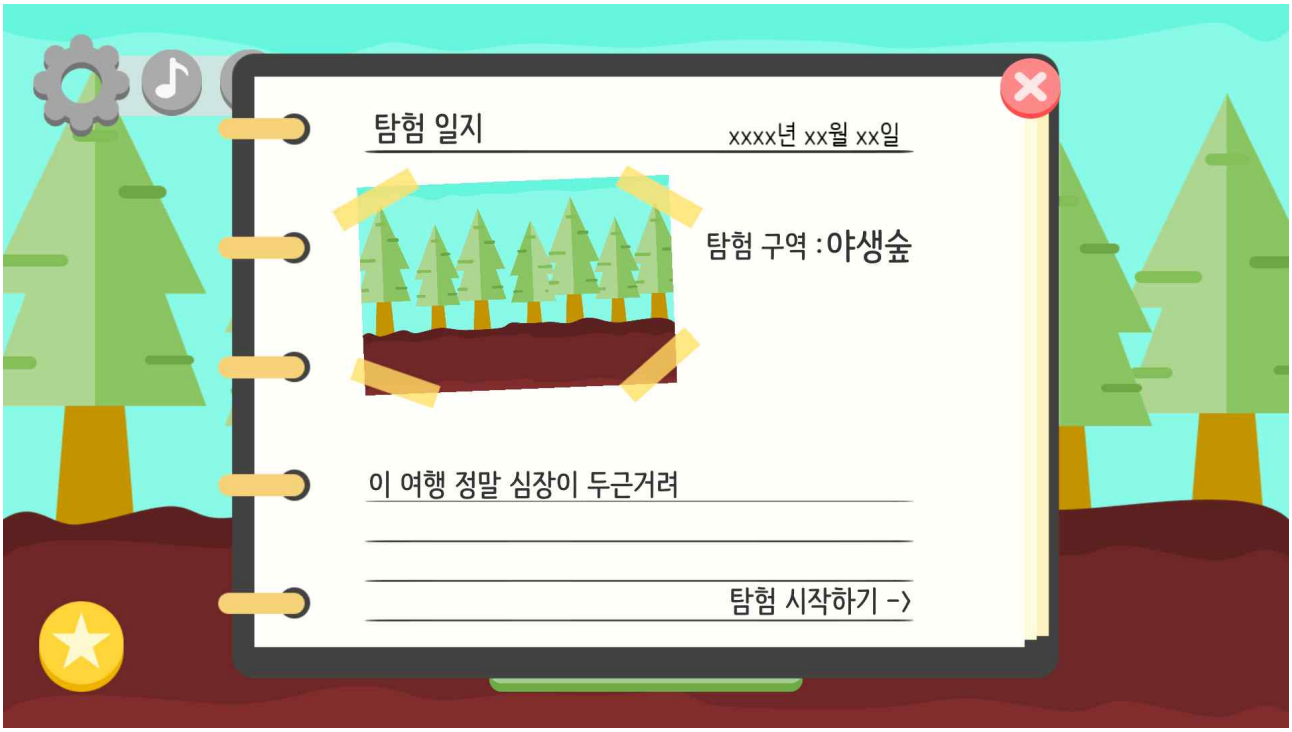
게임의 메인 화면이다. 캐릭터를 선택할 수 있고 모험하기를 통해 게임을 시작할 수 있다.
게임 환경 설정 메뉴와 도움말 및 조작법 메뉴, 덱 빌딩 메뉴가 있다.

덱 빌딩 메뉴에서는 카드를 클릭해 덱에 카드를 추가해 덱을 자신의 전략에 맞게 구성할 수 있다.
최소 5장, 최대 12장의 카드를 소지할 수 있다.

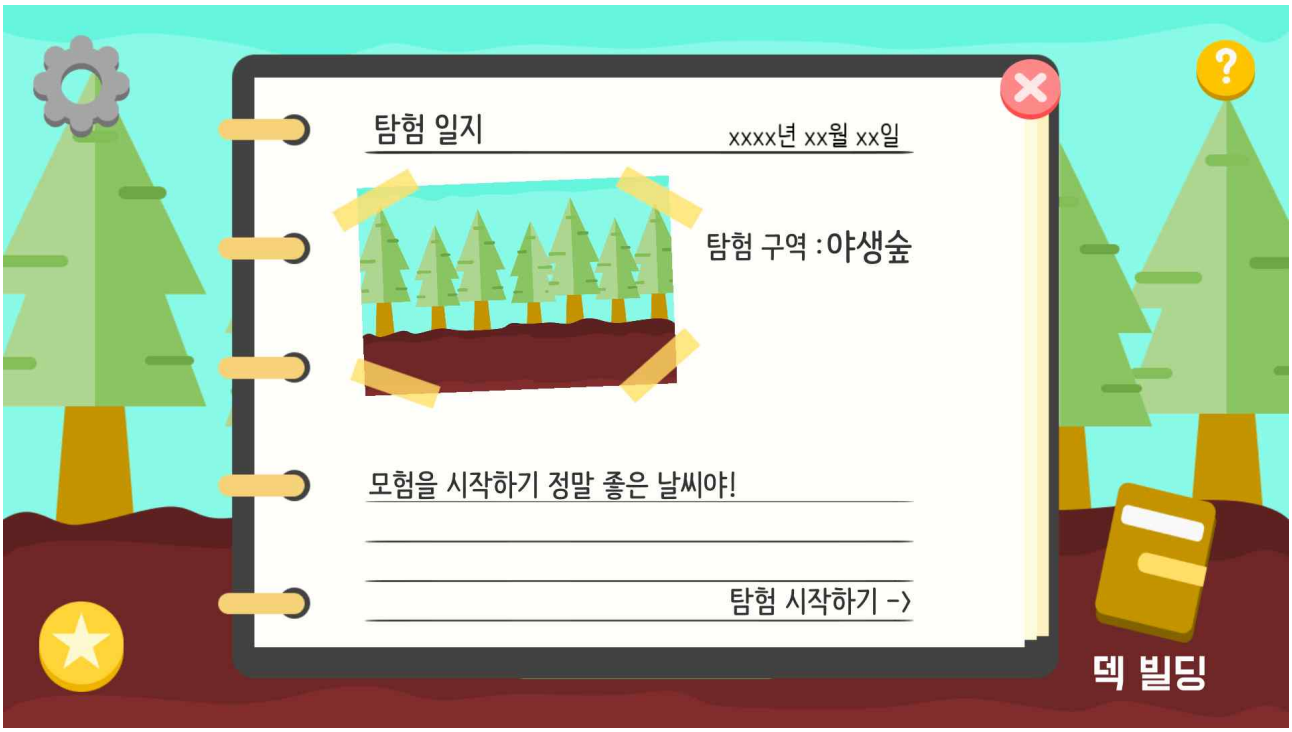
5장 미만의 카드로 덱을 구성할 경우 알림이 뜨며 덱 빌딩 창을 종료할 수 없다.



덱 빌딩을 마치고 창을 나가면 덱이 저장됐다는 알림이 표시된다

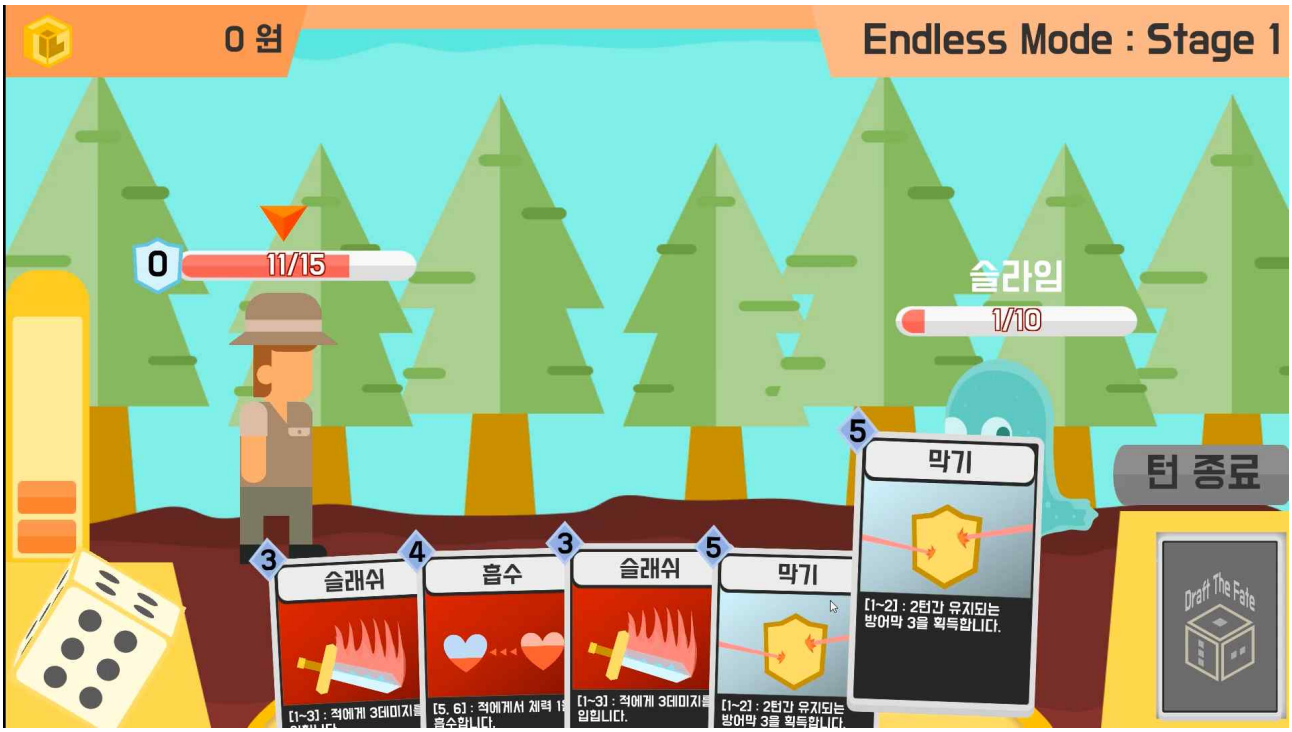


모험하기 버튼을 누르면 표시되는 탐험 일지이다.



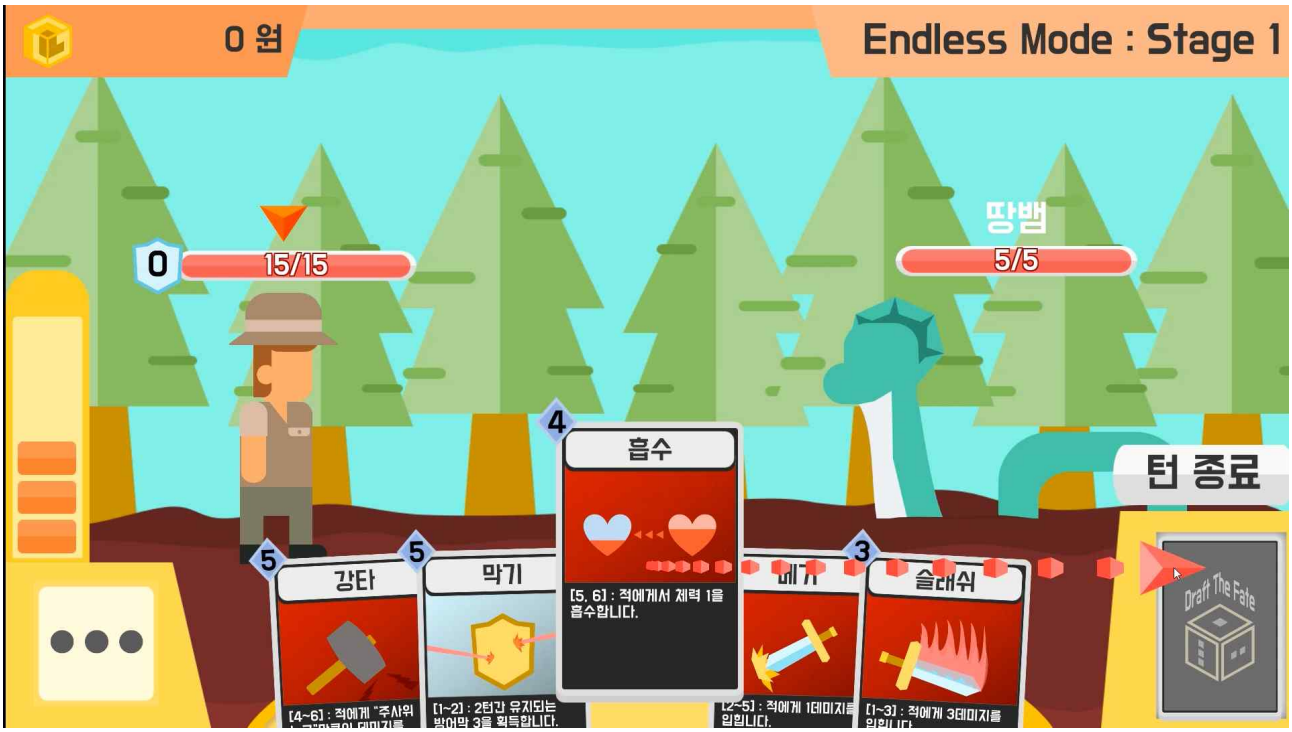
탐험 일지는 매번 내용이 바뀌며 탐험 시작하기 버튼을 눌러 게임이 시작된다.

게임 진행 화면이다. 플레이어가 왼쪽, 몬스터가 오른쪽에 위치한다.

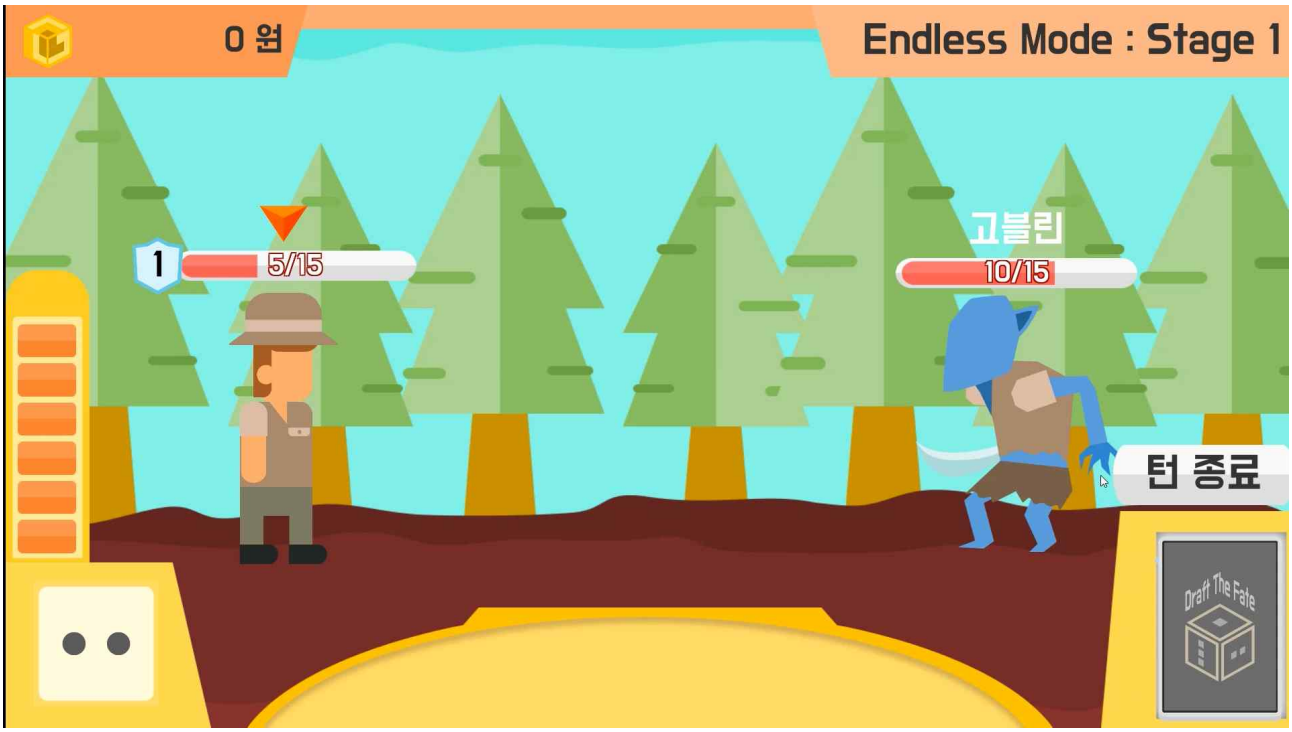


주사위를 터치해 주사위를 굴러 게임을 진행한다.
주사위를 굴려 나오는 값에 따라 우측 코스트가 충전된다.

상대턴에는 몬스터가 공격을 한다. 대미지는 일정 범위내에서 랜덤으로 결정된다.
플레이어가 공격을 받으면 방어력 -> 체력 순으로 감소된다.

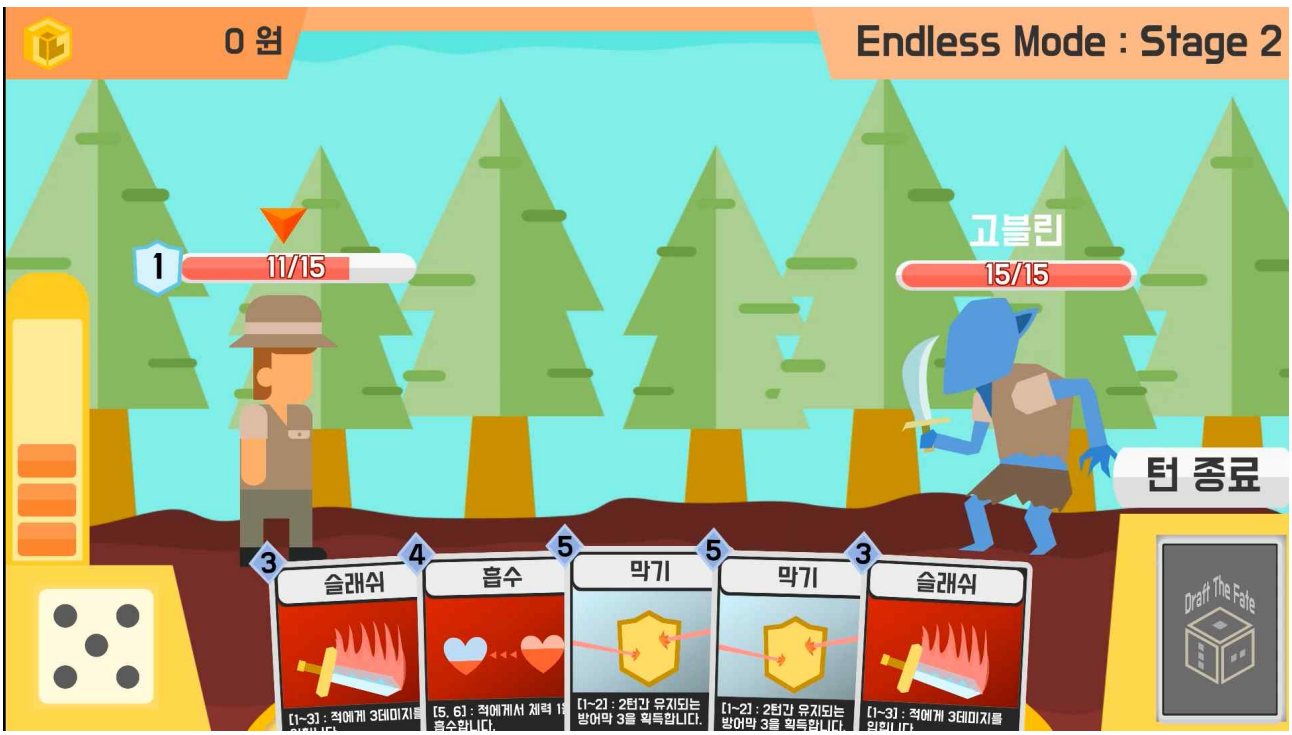


카드를 드래그 드롭으로 다시 덱에 넣을 수 있으며,
해당 카드의 코스트 만큼 플레이어의 코스트가 충전다.

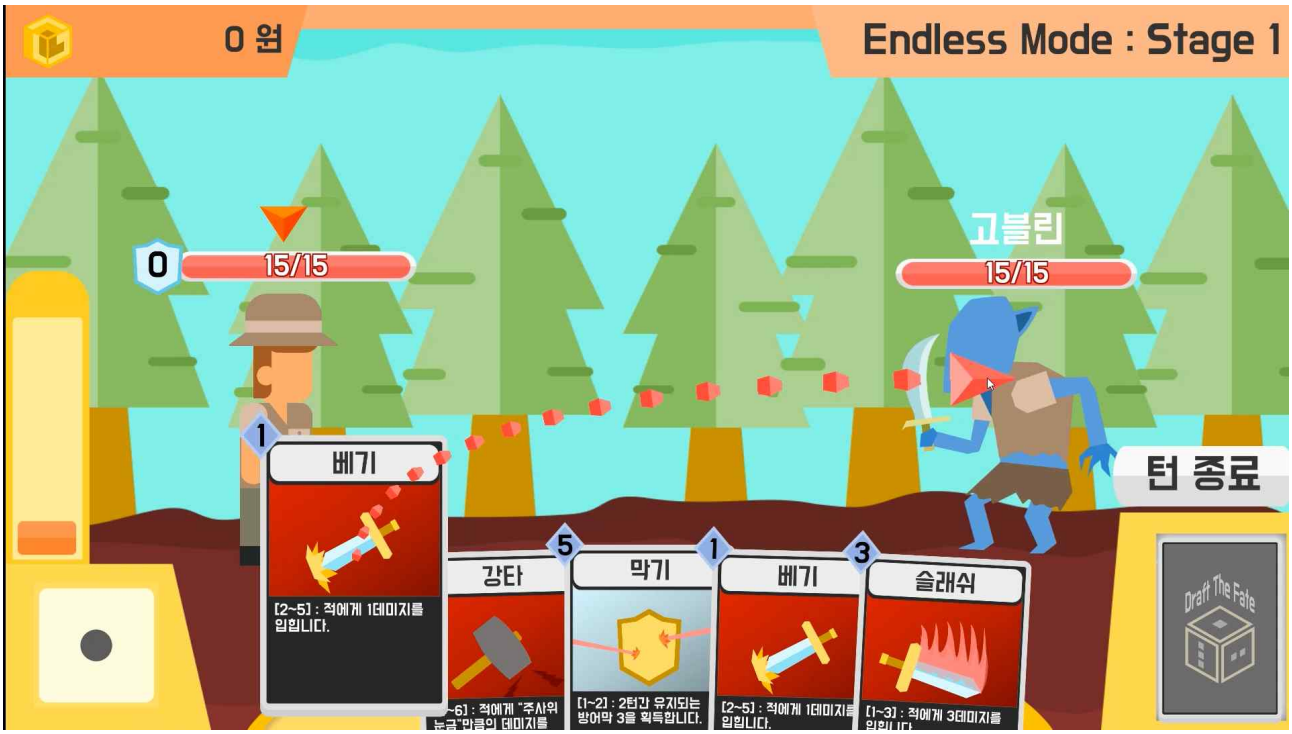


모든 카드를 소모하거나 버릴 수 있으며, 다음 턴에 다시 손안의 카드가 5장이 카드를 보충한다.

적 몬스터의 체력이 0이되면 몬스터는 죽으며 다음 스테이지로 이동한다.

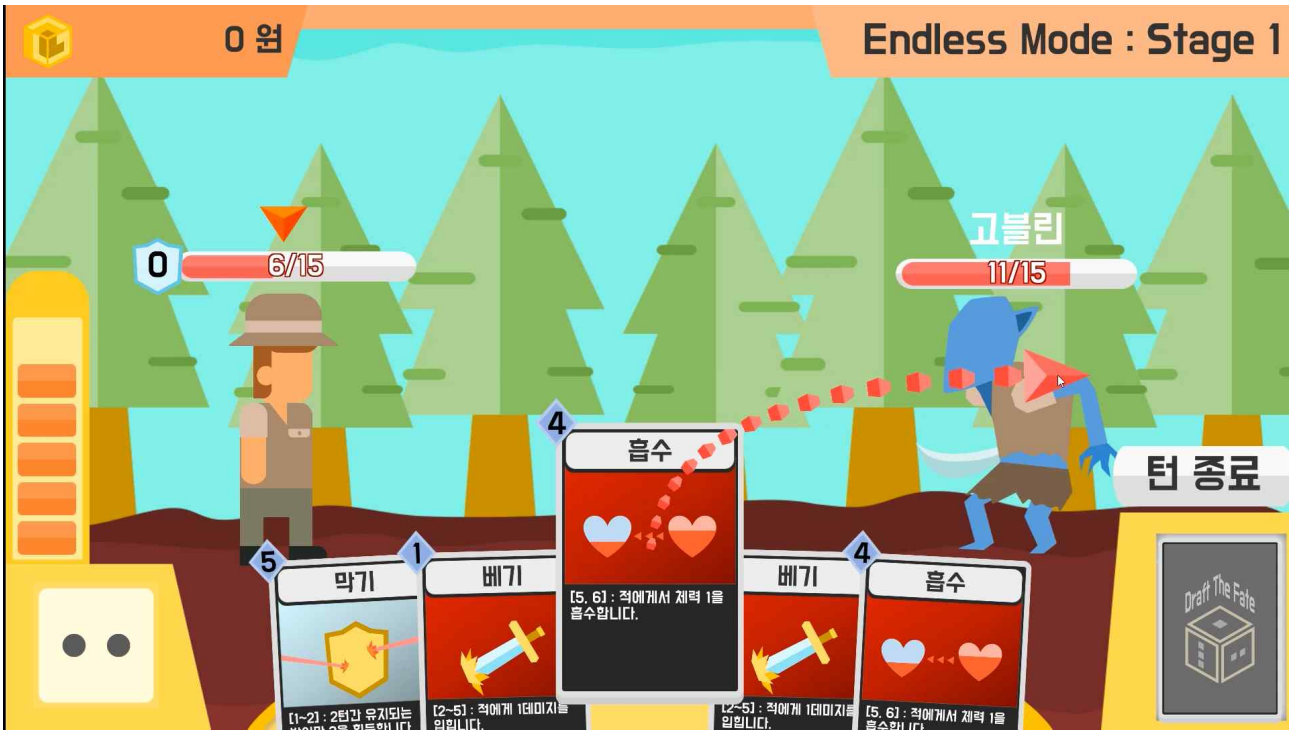


다음 스테이지로 이동하면 다른 몬스터가 랜덤으로 스폰된다.



'배기'카드는 코스트 1을 소모하여 적에게 데미지 1을 입힌다.

'슬래시'카드는 코스트 3을 소모하여 적에게 데미지 3을 입힌다.



'흡수'카드는 코스트 4를 소모하여 적에게 체력 1 흡수한다.

'막기'카드는 코스트 5를 소모하여 2턴간 유지되는 방어력 3을 획득한다.
2턴이 지나면 이 카드를 통해 얻은 방어력은 사라지게된다.

'강타'카드는 코스트 5를 소모하여 5데미지를 입힌다.

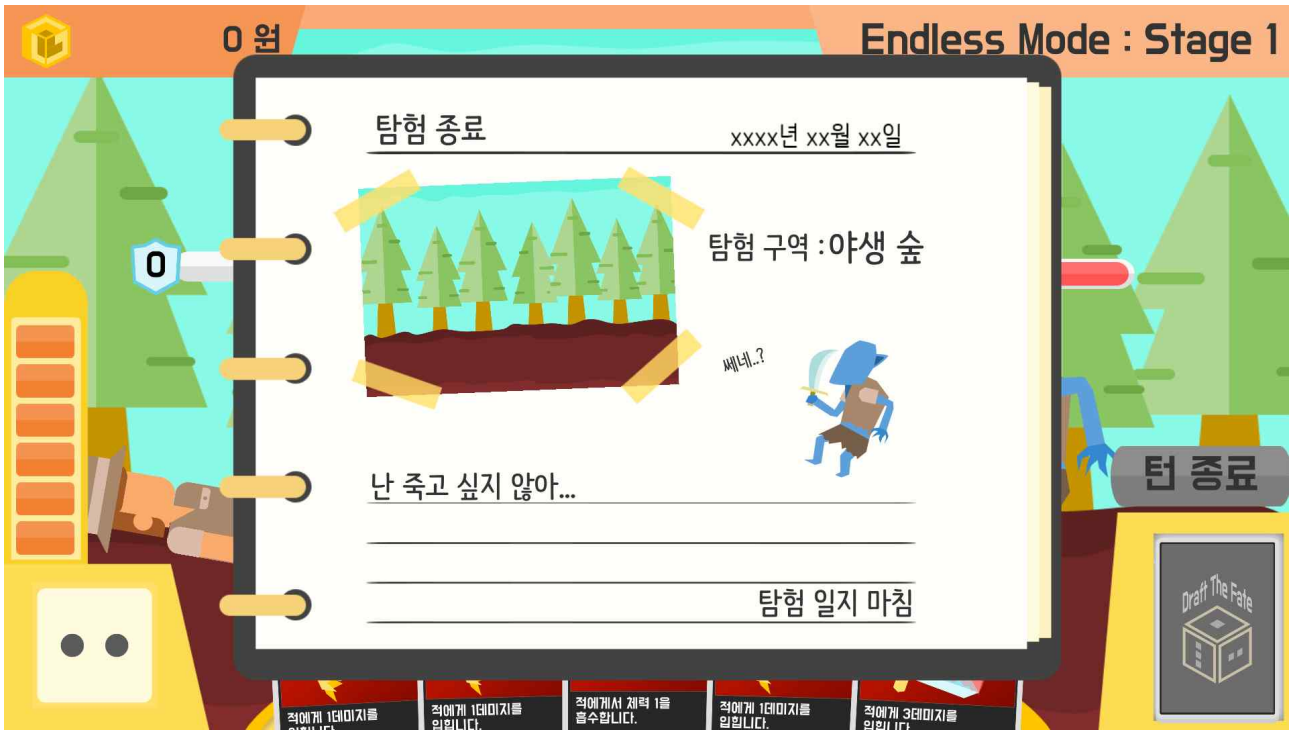
'칼 던지기'카드는 코스트 2를 소모하여 5데미지를 입힌다.
사용과 동시에 다음턴에 아무런 행동을 할 수 없다.

'상처 찌르기' 카드는 적에게 1 데미지를 준다.
카드를 사용할 때 마다 적이 받은 데미지는 1 씩 증가한다.

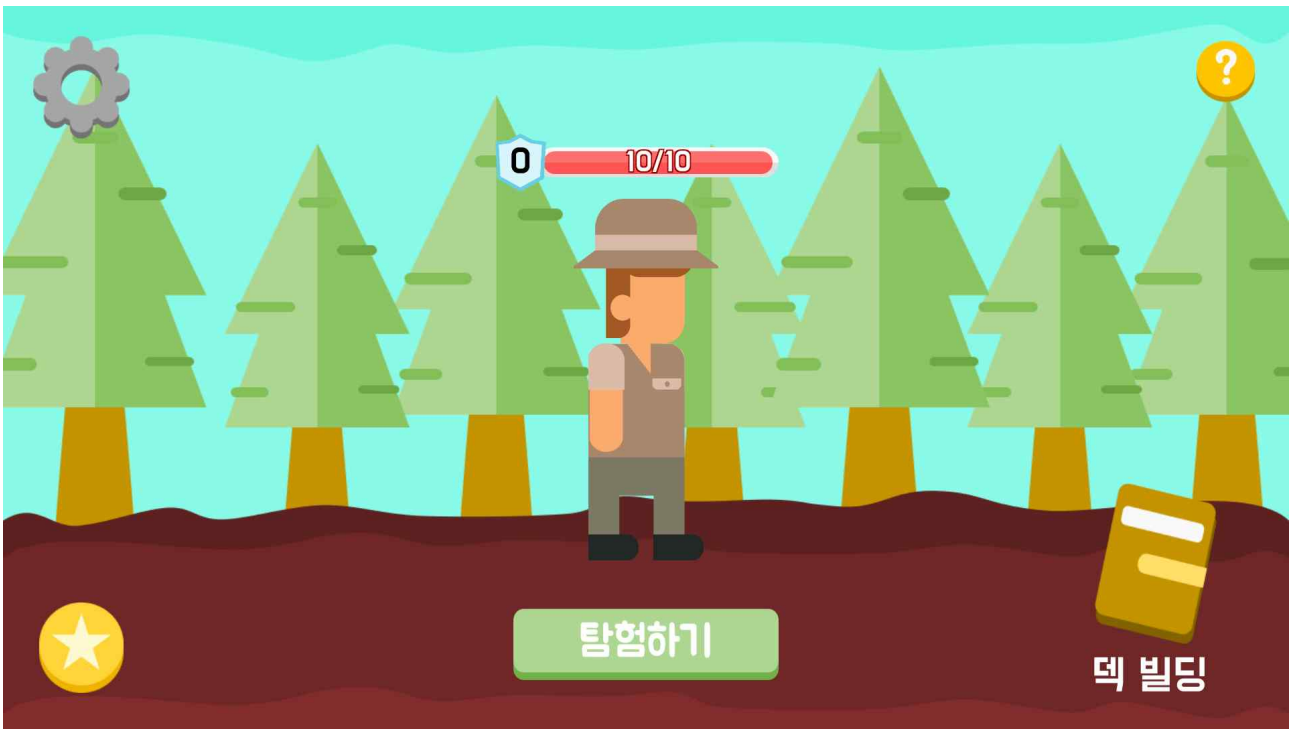
'재빠른 공격'카드는 3코스트를 소모하여 적에게 2데미지를 입히고 카드를 한 장 드로우한다.

'부서진 방패'카드는 6코스트를 소모하여 자신의 방어력을 소진하여 방어력 만큼 적에게 데미지를 준다.

전투 도중 플레이어 캐릭터의 체력이 0이 되면 게임이 종료된다.



탐험 일지가 종료되며, 어떤 몬스터한테 죽었는지와 사망문구가 나오며 게임이 종료된다.



게임이 종료되면 다시 메인 화면으로 돌아간다.

메인화면에서 설정 버튼을 눌러 배경 음악과 효과음의 크기를 조정할 수 있다.

뒤로가기 버튼을 누르면 설정과 종료 창이 나온다.

카드는 '막기', '베기', '흡수', '슬래쉬', '강타', '칼 던지기', '상처 찌르기', '재빠른 공격', '부셔진 방패' 9종류로 구성되있다.

몬스터는 '박쥐토끼', '고블린', '땅뱀', '독버섯돌이', '슬라임' 5종류로 구성되있다.

나. 기술적 난이도

• 화살표 : 화살표는 출발점과 마우스 위치 그리고 중간 지점을 잡은 다음 베지어 곡선 알고리즘을 이용하여 제작하였다.


```
private void Update()
{
    targetPos = UIPosition(Input.mousePosition.x, Input.mousePosition.y);
    originPos = new Vector2(transform.position.x, transform.position.y);
    points[0] = originPos;
    points[2] = targetPos;
    points[1] = new Vector2(originPos.x, targetPos.y);

    for (int i = 0; i < dotCount; i++)
        dots[i] = BezierCalculate(i);

    for (int i = 0; i < dotCount - 1; i++)
    {
        Vector3 pointPos = uiCamera.WorldToScreenPoint(new Vector2(dots[i + 1].x, -dots[i + 1].y));
        pointPos = uiCamera.ScreenToWorldPoint(new Vector3(pointPos.x, pointPos.y, 100));
        arrows[i].position = new Vector2(pointPos.x, -pointPos.y);
    }
    arrows[dotCount - 1].position = targetPos;

    for (int i = 0; i < dotCount - 1; i++)
    {
        Quaternion rot = Quaternion.FromToRotation(Vector3.up, arrows[i + 1].position - arrows[i].position);
        arrows[i].eulerAngles = new Vector3(0, 0, rot.eulerAngles.z);
    }
    arrows[dotCount - 1].eulerAngles = arrows[dotCount - 2].eulerAngles;
}

private Vector3 BezierCalculate(int i)
{
    float t = (float)i / dotCount;
    float s = 1 - t;
    Vector3 b = Vector3.zero;
    for (int p = 0; p < 3; p++)
        b = b + (Pascal(3 - 1, p)) * (Mathf.Pow(s, 3 - 1 - p) * Mathf.Pow(t, p)) * (Vector3)points[p];
    return b;
}
```

• 주사위 : 주사위를 굴릴때 랜덤으로 돌아가되 멈출때 정면을 바라보도록 구현하였고, 해당 주사위가 2번 랜덤으로 돌아간 다음 3번째 경우 의 각도를 90으로 반올림하여 가장 가까운 면이 위로 오도록 하였다.

```
public IEnumerator Roll()
{
    float duration = 0.8f;
    float speed = 40.0f;
    for (int i = 0; i < 2; i++)
    {
        Vector3 torque = new Vector3(Random.Range(1.0f, 2.0f), Random.Range(1.0f, 2.0f), Random.Range(1.0f, 2.0f));
        float nowTime = duration;
        while (nowTime > 0)
        {
            nowTime -= Time.deltaTime;
            speed *= 0.966f;
            transform.Rotate(torque * speed);
            yield return null;
        }
    }

    float targetRotX = (int)(Mathf.Round(transform.localEulerAngles.x / 90) * 90);
    float targetRotY = (int)(Mathf.Round(transform.localEulerAngles.y / 90) * 90);
    float targetRotZ = (int)(Mathf.Round(transform.localEulerAngles.z / 90) * 90);

    float rotX = transform.localEulerAngles.x;
    float rotY = transform.localEulerAngles.y;
    float rotZ = transform.localEulerAngles.z;

    float gapX = targetRotX - rotX;
    float gapY = targetRotY - rotY;
    float gapZ = targetRotZ - rotZ;

    int signX = (int)Mathf.Sign(gapX);
    int signY = (int)Mathf.Sign(gapY);
    int signZ = (int)Mathf.Sign(gapZ);

    speed = 100;
    while (!(((signX > 0 && gapX <= 0) || (signX < 0 && gapX >= 0)) &&
        ((signY > 0 && gapY <= 0) || (signY < 0 && gapY >= 0))&&
        ((signZ > 0 && gapZ <= 0) || (signZ < 0 && gapZ >= 0))
        ))
        {
            ...

            transform.localEulerAngles = new Vector3(rotX, rotY, rotZ);
            yield return null;
        }
    transform.localEulerAngles = new Vector3(targetRotX, targetRotY, targetRotZ);
}
```

• 덱 빌딩 : 플레이어의 덱 정보, 카드 인벤토리등의 데이터를 저장할 때 JsonUtility를 이용하여 Json형태로 변형 후 저장하여 더 수정이 용의하도록 구현하였다.

```
public void SaveDeck()
{
    SaveData(gameData);
}

public void SaveData(GameData data)
{
    CreateJsonFile(JsonUtility.ToJson(data, prettyPrint: true));
}

public void LoadData()
{
    byte[] data = jsonFile.bytes;
    string jsonData = Encoding.UTF8.GetString(data);
    gameData = JsonUtility.FromJson<GameData>(jsonData);
}

void CreateJsonFile(string jsonData)
{
    string path = Application.dataPath + "/Resources/GameData.json";
    if (Application.platform == RuntimePlatform.Android)
        path = Application.persistentDataPath + "/Resources/GameData.json";
    else if (Application.platform == RuntimePlatform.IPhonePlayer)
        path = Application.dataPath + "/Resources/GameData.json";

    FileStream fileStream = new FileStream(path, FileMode.Create);
    byte[] data = Encoding.UTF8.GetBytes(jsonData);
    fileStream.Write(data, 0, data.Length);
    fileStream.Close();
}
```

다. 향후 작품 개선/발전 계획

현재 이 게임의 목표는 덱 빌딩과 플레이어의 전략을 통하여 주사위라는 운적인 요소를 제어할 수 있도록 하는 것이다. 그러기 위해 제작할 예정인 것은 다음과 같다.

1. 주사위에 직접적인 영향을 주는 카드를 제작할 것이다.
예를 들어, 주사위를 자르는 카드로 1,2,3 주사위 하나와 4,5,6 주사위 하나를 각각 굴려 합친 숫자가 주사위눈금이 되게하는 등의 직접적으로 주사위에 영향을 주는 카드를 제작할 예정이다.

2. 시너지를 발휘할 수 있는 카드를 제작할 예정이다.
슬래쉬같은 경우 현재 손에 들고있는 베기의 수만큼 추가 공격을 하도록 수정할 계획이며, 이와 같이 덱 빌딩시 고려할 만한 조건을 늘릴 예정이다.