

Unity2D 개인프로젝트 기술문서

이동준

```
[Serializable]
public class GameData
{
    public bool[] Stage;
    public bool[] npcQuest;
    public bool storyClear = false;
}
```

```
static DataController _instance;
참조 20개
public static DataController instance
{
    get
    {
        if (!_instance)
        {
            container = new GameObject();
            container.name = "DataController";
            _instance = container.AddComponent(typeof(DataController)) as DataController;
            DontDestroyOnLoad(container);
        }
        return _instance;
    }
}
public string GameDataFileName = "ClearData.json";
```

```
string FromJsonData = File.ReadAllText(filePath);
_gameData = JsonUtility.FromJson<GameData>(FromJsonData);
}
else
{
    //새 파일 생성
    _gameData = new GameData();
    Debug.Log("파일생성완료");
}
참조 2개
public void SaveGameData()
{
    string ToJsonData = JsonUtility.ToJson(gameData);
    string filePath = Application.persistentDataPath + GameDataFileName;
    File.WriteAllText(filePath, ToJsonData);
    Debug.Log("Save Success");
}
@ Unity 메시지 | 참조 0개
private void Awake()
{
    LoadGameData();
}
@ Unity 메시지 | 참조 0개
private void OnApplicationQuit()
```

게임 저장

데이터는 스테이지 클리어 유무, 퀘스트 클리어 유무, 스토리 클리어 유무를 직렬화를 시켜 json 파일로 저장하게 했다.

스테이지 클리어, 퀘스트 클리어, 스토리 클리어 시 자동으로 저장이 되며

게임이 시작될 때 자동으로 저장 데이터가 로드 된다.

```

void Update()
{
    if(worldTime > 6 && worldTime < 17)
    {
        Am = true;
        transform.GetChild(0).GetComponent<SpriteRenderer>().sprite = afternoonSky;
        transform.GetChild(1).GetComponent<Light2D>().intensity = 1;
        for (int i = 0; i < npc.Length; i++)
        {
            npc[i].gameObject.SetActive(true);
        }
    }
    else if(worldTime >= 17 && worldTime < 20)
    {
        Am = true;
        transform.GetChild(0).GetComponent<SpriteRenderer>().sprite = eveningSky;
        transform.GetChild(1).GetComponent<Light2D>().intensity = 0.7f;
        for (int i = 0; i < npc.Length; i++)
        {
            npc[i].gameObject.SetActive(true);
        }
    }
    else if (worldTime >= 20 || worldTime <= 6)
    {
        Am = false;
        transform.GetChild(0).GetComponent<SpriteRenderer>().sprite = nightSky;
        transform.GetChild(1).GetComponent<Light2D>().intensity = 0.5f;
        for(int i = 0; i < npc.Length; i++)
        {
            npc[i].gameObject.SetActive(false);
        }
    }
    if(worldTime > 24)
    {
        worldTime -= 24;
    }
}

```

```

[System.Serializable]
public class SaveData
{
    public int worldTime;

    public SaveData(int _time)
    {
        worldTime = _time;
    }
}

```

게임 내 시간

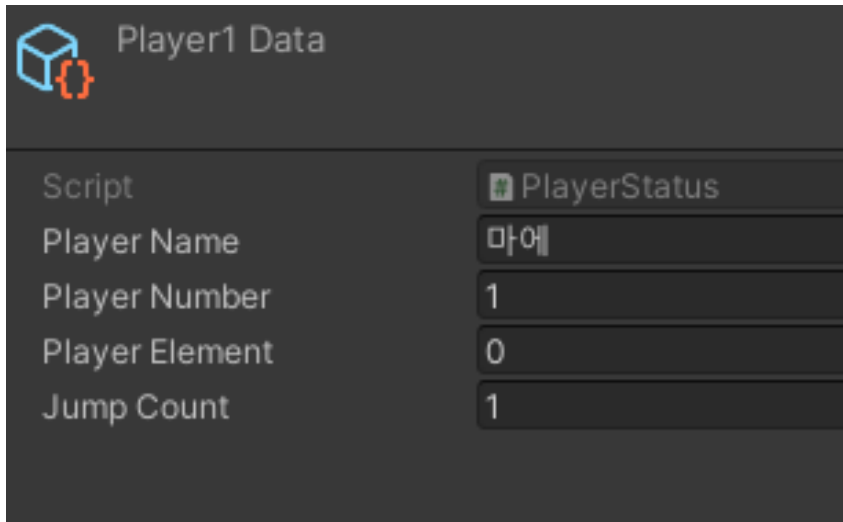
GameController 스크립트에

Int형으로 worldTime 이라는 변수를 선언해 주었고 worldTime 이 일정 값으로 바뀔 때 마다 하늘 배경의 스프라이트와 빛의 값을 바꾸어주었다.

특정 값이 되면 마을에 정해진 NPC도 등장을 한다.

시간은 직렬화를 이용하여 자동으로 저장/불러오기 된다.

```
[CreateAssetMenu(fileName = "Player Data", menuName = "Scriptable
Object/Player Data", order = int.MaxValue)]
public class PlayerStatus : ScriptableObject
{
    [SerializeField]
    string playerName;
    public string PlayerName { get { return playerName; }}
    [SerializeField]
    int playerNumber;
    public int PlayerNumber { get { return playerNumber; }}
    [SerializeField]
    int playerElement; // 0 : 땅 , 1 : 바람 , 2 : 물 , 3 : 불
    public int PlayerElement { get { return playerElement; }}
    [SerializeField]
    int jumpCount;
    public int JumpCount { get { return jumpCount; }}
}
```



Player1 Data	
Script	PlayerStatus
Player Name	마에
Player Number	1
Player Element	0
Jump Count	1

캐릭터 정보

스크립터블오브젝트를 사용하여 캐릭터 정보를 관리하였다.

캐릭터이름

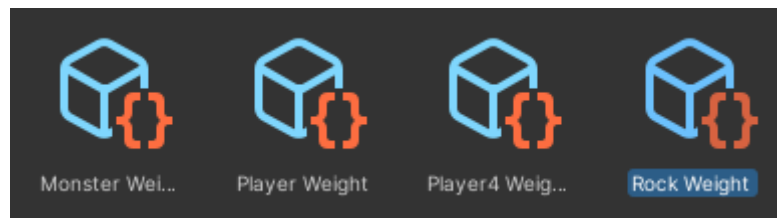
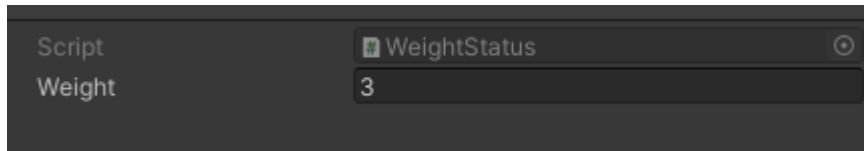
캐릭터넘버

캐릭터속성

캐릭터의 최대 점프 가능 횟수를

저장하는데 사용.

```
[CreateAssetMenu(fileName = "Player Data", menuName = "Scriptable  
Object/Weight Data", order = int.MaxValue)]  
public class WeightStatus : ScriptableObject  
{  
    [SerializeField]  
    int weight;  
    public int Weight { get { return weight; } }  
}
```



무게

스크립터블오브젝트를 사용하여 오브젝트마다 각각 무게를 부여했다.

BGM

빈 오브젝트에 BGM 매니저 스크립트를 만들어 넣었으며

DontDestroyOnLoad 함수를 사용하여 씬이 바뀌어도 유지되게 하였고

Dictionary 를 사용하여 중복 방지 코드를 작성하였다.

public int 형으로 id 값을 받게 하였고 BGM을 달라지게 할 씬은 id값을 다르게 하여서 id 값을 비교 했을 때 기존과 다를 경우 기존에 있던 BGM매니저를 Destroy 시키게 만들었다.

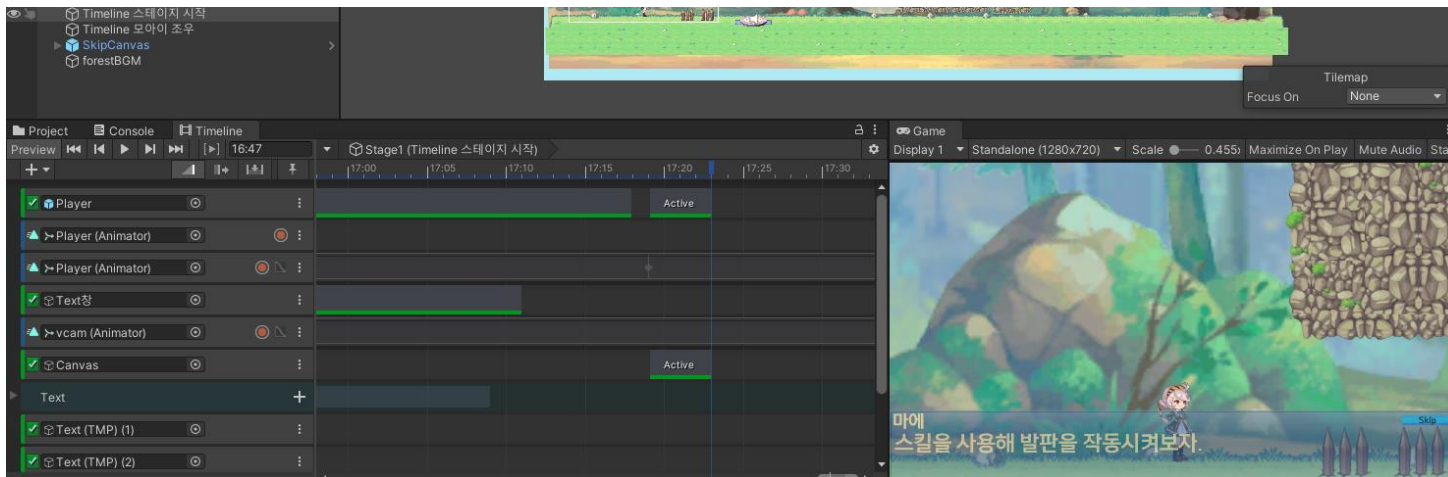
스토리 진행

빈 오브젝트를 만들고

PlayableDirector 를 사용하여

타임라인을 사용하여 스토리를
제작하였다.

씬 시작하자마자 실행되는 타임
라인이 있고 특정 장소에 도달하
였을 때 실행되는 타임라인이 있
다.



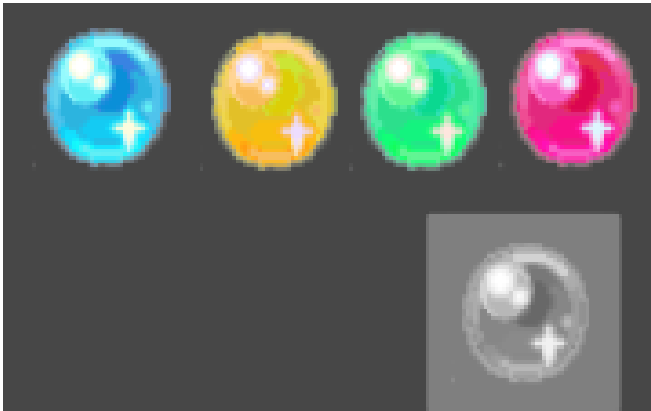
캐릭터 변경

Canvas에 버튼 이미지를 삽입하여 만들었다.

클릭하면 해당 캐릭터로 플레이어가 변경이 되며 해당 캐릭터 교체 쿨타임이 진행된다.



원소구슬



회색 구슬에 위치해 있으며

회색 구슬 클릭 후 원소 Tag가 있는 collider 를 마우스로 클릭 시

해당 원소 Tag에 대응하는 색깔의 구슬로 바뀌게 된다.

색깔이 바뀐 원소 구슬을 한 번더 클릭하게 되면 각각 다른 스킬들이 생성이 된다.

스킬이 사용 되면 15초의 쿨타임이 실행된다.

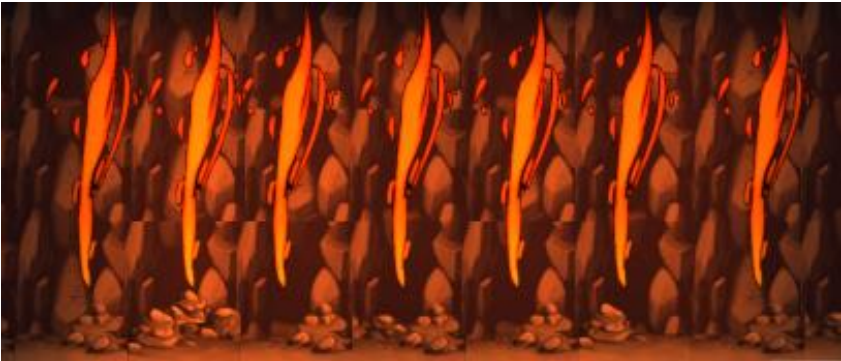


무게발판

발판에 닿은 오브젝트의 무게를 받아와 일정 무게 이상일 경우 함정을 해제하거나 장치를 발동 시킨다.

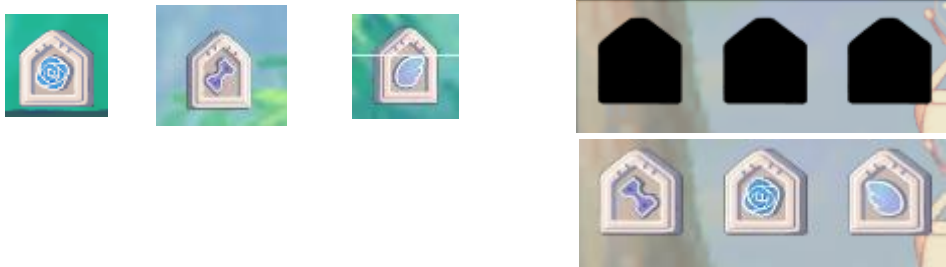


불꽃 함정



플레이어는 지나갈 수 없으며

Water Tag가 달려있는 오브젝트에 닿을 시 소멸하고
20초 뒤에 다시 생성이 된다.



석판

비석을 마우스로 누르게 되면 힌트와 함께 석판 Canvas Panel 이 활성화된다.

맵 곳곳에 석판이 있으며 석판에 플레이어 캐릭터가 닿으면 석판 Panel에 불이 들어온다.

불이 들어온 석판을 마우스 드래그로 타워에 가져다가 붙일 수 있고 올바른 순서로 석판을 붙이게 되면 다음 장소로 가는 포탈이 개방된다.



윈드스톤

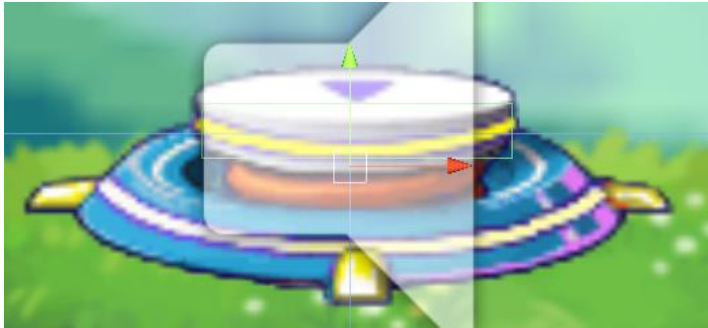
바람속성 원소가 닿으면 상승기류 오브젝트가 생성이 되고 상승기류 오브젝트에 플레이어가 닿을 시 서서히 위로 올라가게된다.



점프 발판

갈색 발판위에서 점프를 하면 정해진 방향으로 캐릭터가 날라가게 되며(AddForce)

스프링 발판위에서 점프를 하면 위로 높이 캐릭터가 점프를 하게 된다.





양초

Fire Tag가 붙은 오브젝트와 충돌하면 불이 켜지고 불에 있는 Playable Director 컴포넌트가 실행이 된다.

Water Tag가 붙은 오브젝트가 충돌하면 불이 꺼진다.



같은그림맞추기

위에있는 그림과 동일한 그림으로 맞추면 Playable Director가 실행이 된다.

밑에 3개의 버튼만 클릭이 되며 총 5개의 그림으로 이루어져있다.



열쇠 개방 포탈

해당 오브젝트에 닿으면 2개의 열쇠가 필요하다는 문구가 출력이 되고 2개의 열쇠를 가지고 있으면 다음 스테이지로 가는 포탈이 개방된다.

열쇠는 총 2개의 스테이지에 각각 하나씩 존재하며

패널을 DontDestroyLoad 로 씬을 넘어가도 Destroy되지 않게 처리했다.

열쇠는 패널에 열쇠가 존재할 경우 해당 씬에있는 열쇠를 Destroy 되게 스크립트를 설계했다.

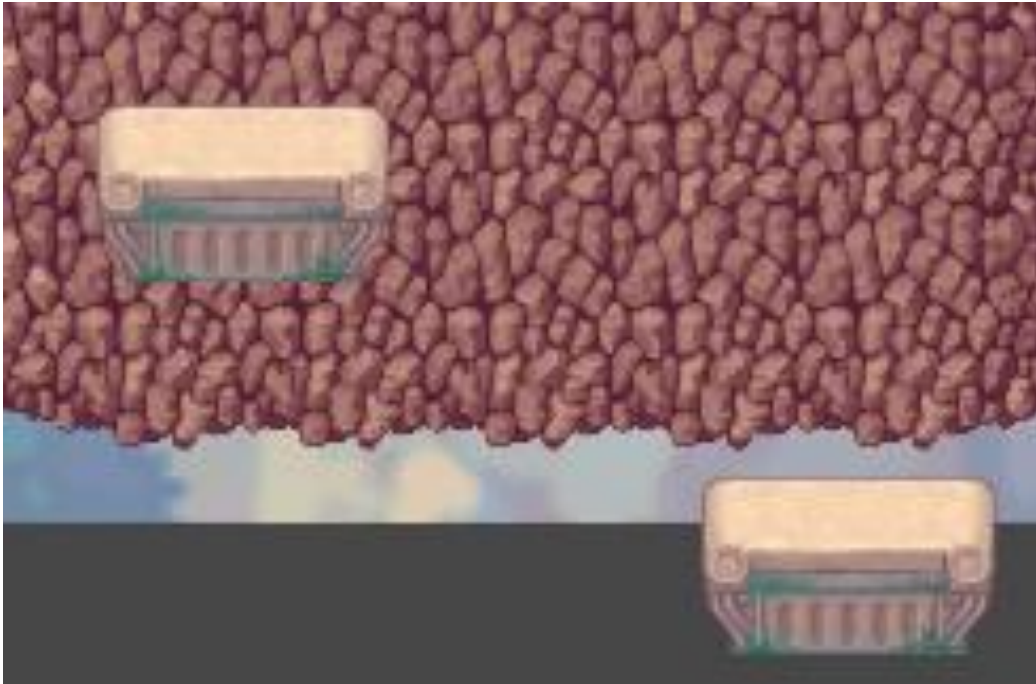
대포



방향키 위 아래 키로 방향을 조절할 수 있고 f키로 포를 발사한다. 발사된 포는 방향에 따라 날라가며 포물선으로 날아가게 스크립트를 만들었다.

삼각함수를 사용하여 대포를 회전시켜 원하는 방향으로 포가 날아가도록 하였다.

움직이는 타일



타일이 특정 방향으로 계속 움직인다.

플레이어가 타일 위에 있으면 플레이어도 타일에 붙은채로 같이 움직인다.

플레이어가 움직이면 타일에서 떨어진다.



원소반응 수정

수정에 맞는 원소를 반응시키면 오른쪽에 있는 나무의 숫자가 1 늘어나고 랜덤하게 다른 색 수정으로 바뀐다.

숫자가 6이 되면 Playable Director 컴포넌트가 실행된다.



구슬 넣기 퍼즐

특정 속성을 가진 플레이어가 석상에 닿으면 석상에 속성의 색깔이 나타나며 위에 있는 원소구슬을 방향키로 움직일 수 있게 된다.

원소구슬을 자신과 같은 색깔에 있는 포탈에 위치시키면 원소구슬이 해당 포탈에 고정되어 움직일 수 없는 상태가 된다.

원소구슬은 다른 원소구슬과 충돌하면 처음 위치로 돌아가게 되며 Reset버튼을 눌러 위치를 전부 초기화 시킬 수 있다.

원소구슬을 전부 알맞은 위치에 위치시키면 스테이지 클리어 포탈이 생성된다.

클리어 포탈

Main 씬으로 돌아가게 되며 해당 스테이지 클리어 데이터가
저장이 된다.

