제목 : 유니티 엔진을 사용한 런게임 제작

개요

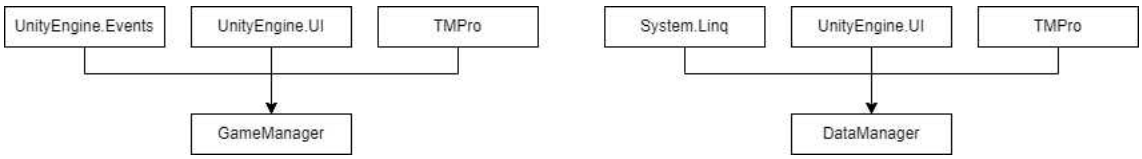
해당 게임은 플레이어가 돌을 피해 계속해서 점프하여 최고 점수를 달성하는 무한 런 게임임. 게임은 시작할 때 타이틀 상태(Ready)에서 시작하며, 게임 시작 버튼 클릭시 준비 상태(Ready)에서 플레이 상태(Play)로 전환됨.

게임 플레이 상태에서는 장애물인 돌이 생성되며 플레이어는 점프 버튼을 눌러 피해야함, 만약 돌과 플레이어가 충돌했을 시 게임 오버 상태로 전환 됨.

Rigidbody, Colider를 사용하지 않고 점프 및 충돌을 구현하기 위해 중력 가속도와 수직으로 움직이는 방향을 계산하여 자연스러운 점프를 구현하였고, ABB(Axis-Aligned Bounding Box) 충돌 검사를 사용하여를 응용하여 플레이어와 장애물의 충돌을 구현함.

게임 오버 상태에서는 해당 게임에 기록된 플레이어와 점수를 1위부터 10위까지 표시되며 재시작 버튼으로 타이틀 상태로 되돌아가 다시 게임을 플레이 할 수 있음.

소스코드 구조

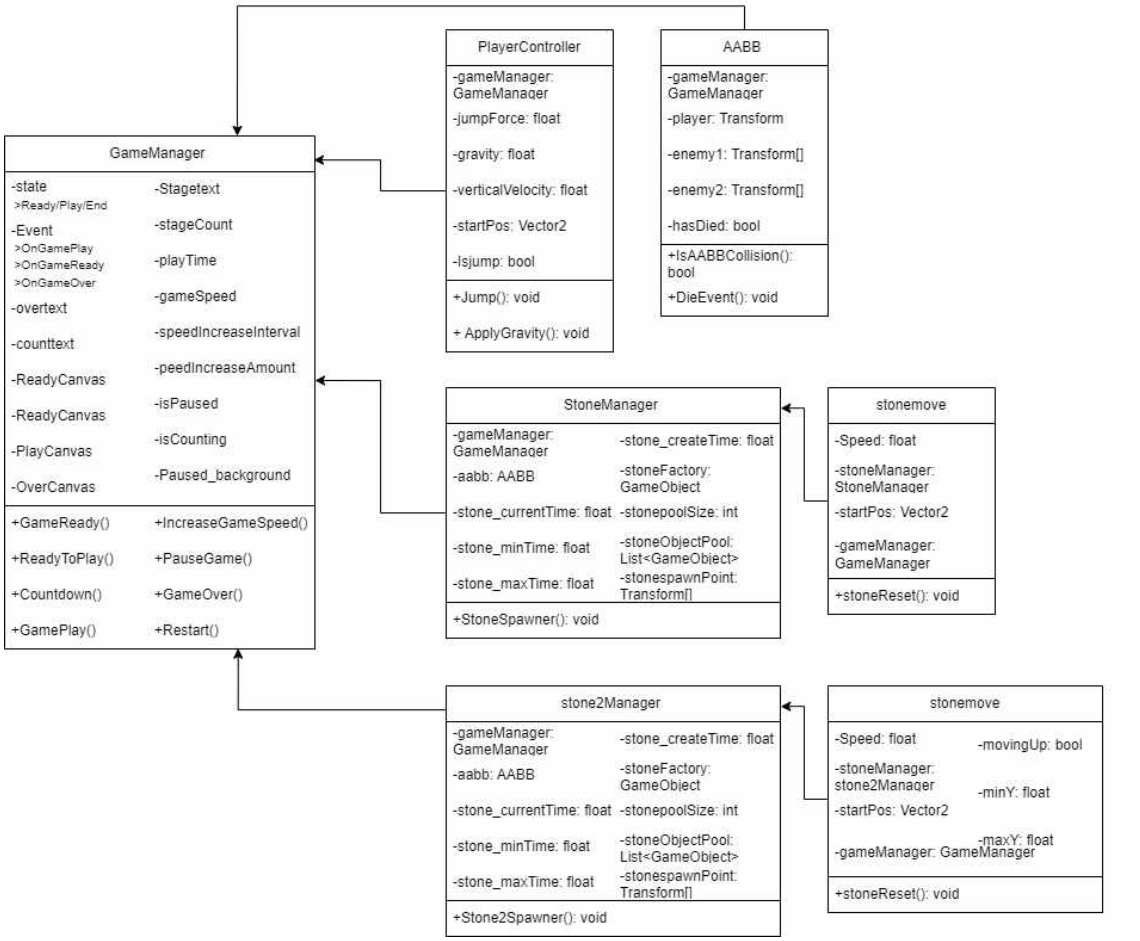


GameManager에서 현재 상태에 따른 이벤트를 관리 및 호출하기 위해 <UnityEngine.Events>을 사용하였고,

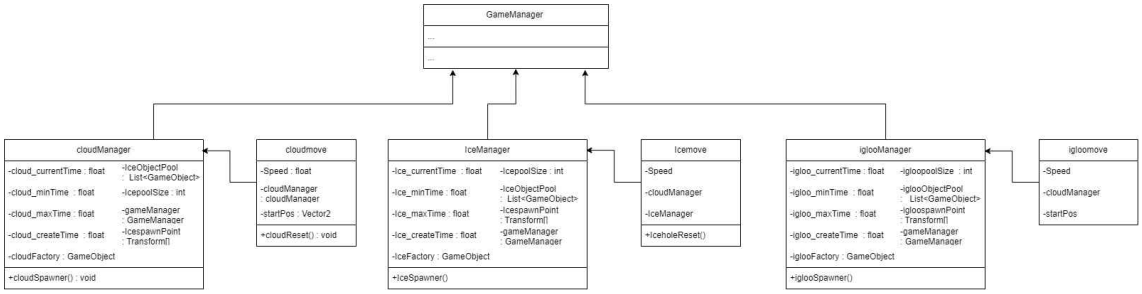
소스코드 파일 설명

Gamemanager.cs	전체적인 게임의 상태 관리와 화면(Canvas)를 관리함
UnityEngine.Events	게임 오브젝트 간의 상호작용, 사용자 입력, 애니메이션 상태 변경 등 다양한 상황에서 사용. 게임 상태 변환 및 상황에 따른 함수 관리 호출에 씀
UnityEngine.UI	텍스트와 입력 필드 등의 UI 요소를 생성하고 조작하기 위해 사용. 텍스트와 입력 칸을 사용하는데 씀
TMPro	TextMesh Pro를 조작하기 위해 사용.
Datamanager.cs	사용자의 이름, 점수를 관리하며 점수 관련 화면 출력을 관리함
System.Linq	데이터를 쉽게 조작하기 위한 소스코드로, 데이터 정렬과 관련된 작업을 수행하기 위해 사용
UnityEngine.UI	텍스트와 입력 필드 등의 UI 요소를 생성하고 조작하기 위해 사용. 점수 및 랭킹 관련 텍스트를 사용하는데 씀
TMPro	TextMesh Pro를 조작하기 위해 사용.

제작한 클래스별 설명



플레이어 및 장애물 관련		
헤더파일	클래스명	설명
AABB.cs	AABB	AABB로 플레이어와 장애물들의 충돌 여부를 계산
PlayerController.cs	PlayerController	플레이어의 점프 동작을 담당하며, 원할한 점프와 현재 자리에서 벗어나지 않도록 함
StoneManager.cs	StoneManager	장애물1의 생성 관련한 로직을 담당하며, 오프젝트풀을 진행
stonemove.cs	stonemove	장애물1의 움직임을 담당하며, 특정 범위를 벗어나면 오브젝트 비활성화와 처음 위치로 옮김
stone2Manager.cs	stone2Manager	장애물2의 생성과 관련한 로직을 담당하며, 오브젝트풀을 진행
stone2move.cs	stone2move	장애물2의 움직임을 담당하며, 특정 범위를 벗어나면 오브젝트 비활성화와 처음 위치로 옮김



배경 관련		
헤더파일	클래스명	설명
iglooManager.cs	iglooManager	배경1의 생성 관련한 로직을 담당
igloomove.cs	igloomove	배경2의 생성 관련한 로직을 담당
IceManager.cs	IceManager	배경3의 생성 관련한 로직을 담당
Icemove.cs	Icemove	배경1의 움직임을 담당
cloudManager.cs	cloudmove	배경1의 움직임을 담당
cloudmove.cs	cloudmove	배경1의 움직임을 담당

제작한 클래스별 설명

1. GameManager

a) 멤버변수 설명

접근지정자	이름	설명
public	enum GameState	Ready, Play, End로 게임의 상태를 나타냄
public	UnityEvent OnGameReady	게임 준비(타이틀) 상태에서 호출되는 이벤트로 게임의 초기화를 진행함
public	UnityEvent OnGamePlay	게임 플레이 중 상태에서 호출되는 이벤트로 장애물/배경 관련 함수를 호출함
public	UnityEvent OnGameOve	게임 오버 상태에서 호출되는 이벤트로 랭킹 함수를 호출함
public	TextMeshProUGUI overtext	게임 오버 시 표시되는 텍스트
public	TextMeshProUGUI counttext	카운트다운 텍스트를 표시하는 텍스트
public	GameObject ReadyCanvas	플레이 버튼을 가지고 있는 캔버스며 게임 준비 상태에서 표시됨
public	GameObject PlayCanvas	현재 점수, 스테이지, 일시정지 버튼을 가지고 있는 캔버스며 게임 플레이 중에 표시됨
public	GameObject	랭킹 출력 텍스트, 재시작 버튼을 가지고

	OverCanvas	있는 캔버스며 게임 오버 상태에서 표시됨
public	TextMeshProUGUI Stagetext	현재 스테이지를 나타내는 텍스트
private	int stageCount	현재 스테이지의 숫자를 저장함
private	float playTime	현재 게임 플레이 시간을 저장함
private	float gameSpeed	현재 게임 속도를 저장하는 변수이며 해당 변수가 게임의 속도가 됨
private	float speedIncreaseInterval	게임 속도를 증가시키는 간격을 저장하는 변수이며 playTime이 해당 변수 간격을 넘을때마다 gameSpeed을 증가시킴
private	float speedIncreaseAmount	게임 속도를 증가시키는 양을 저장하는 변수
private	bool isPaused	게임이 일시 정지되었는지를 나타내며, 일시 정지 시 오브젝트들이 움직이거나 플레이어가 점프하는 것을 방지하기 위한 구분으로 씀
public	bool isCounting	카운트다운이 진행 중인지를 나타냄
public	GameObject Paused_background;	일시정지 및 카운트다운 진행 시 사용되며 현재 화면 상태를 구분하기 위한 오브젝트

b) 멤버 함수

* GameReady() : 게임 준비 상태를 설정하고, 관련된 UI 요소들을 활성화 및 비활성화함

* ReadyToPlay() : 게임 시작 버튼을 눌렀을 때 카운트다운을 진행하는 코루틴을 실행함

* Countdown() : 3부터 1까지 1초 간격으로 카운트다운을 함. 카운트다운 중에는 게임 속도를 일시정지하고, 카운트다운이 끝나면 게임 속도를 복구시킴

* GamePlay() : GameState가 Play일때 호출되며 OnGamePlay를 실행시킴

* IncreaseGameSpeed() : 게임 속도를 증가시키는 함수. 게임 플레이 시간에 따라 일정 간격으로 호출되며, 게임 속도와 스테이지 수를 증가시킴

* PauseGame() : 게임 일시 정지를 처리하는 함수로 게임이 일시 정지되었을 때는 게임 속도를 0으로 설정하고 다시 게임을 진행할 때는 카운트다운을 시작함

* GameOver() : 게임 오버 상태를 설정하고, 관련된 UI 요소들을 활성화함. 게임 속도를 0으로 설정하여 게임을 멈춤

* Restart() : 게임을 다시 시작하기 위해 필요한 초기화 작업을 수행하는 함수. 게임 요소를 초기화 시킨 후 GameReady()를 호출해 다시 게임을 진행 시킴

1. DataManager

a) 멤버변수 설명

접근지정자	이름	설명
private	GameManager gameManager	게임 매니저 오브젝트에 대한 참조를 저장함

public	int currentScore	현재 점수를 저장함
private	int bestScore	최고 점수를 저장함
public	TextMeshProUGUI scoreText	게임 플레이 중에 점수를 표시하기 위한 텍스트로 프리팹에 BestScore_으로 저장됨
public	TextMeshProUGUI bestScoreText	게임 종료 후 최고 점수를 표시하기 위한 텍스트로 프리팹에 저장된 BestScore_를 불러옴
public	TextMeshProUGUI currentScoreText	게임 종료 후 현재 점수를 표시하기 위한 텍스트
public	TMP_InputField nickname	플레이어의 닉네임을 입력하기 위한 입력 필드로 입력된 내용은 nowName에 저장됨
public	string nowName	게임 시작할 때 입력된 플레이어 이름을 저장하는 변수로 프리팹에 PlayerName으로 저장됨
public	TextMeshProUGUI RankNames	순위를 표시하는 이름을 나타내는 텍스트로, 랭크를 저장하는 리스트 내용 중 Name을 불러와 출력함
public	TextMeshProUGUI RankScores	순위를 표시하는 점수 타내는 텍스트로, 랭크를 저장하는 리스트 내용 wnd Score를 불러와 출력함
private	int playCount	플레이 횟수를 저장하는 변수로, 프리팹에 저장함. 랭크에서 순위권에 기록된 만큼 출력하기 위해 사용함
public	static DataManager dataManager	데이터 매니저의 인스턴스를 저장
private	float playTime	플레이 시간을 저장하며, 현재 점수로 사용됨. 점수로 출력될 때 소수점은 모두 버림

b) 멤버 함수

* IncreasePlayCount() : 플레이 횟수를 증가시키는 함수로 플레이 횟수를 10회까지 증가시킬 수 있음. PlayerPrefs에 플레이 횟수를 저장하여 랭크를 출력할 때 사용함

* RankReset() : 순위를 초기화하는 함수로 rankList를 비우고 PlayerPrefs를 삭제하여 순위와 플레이 횟수를 초기화 시킴

* ShowScoreRank() : 순위를 표시하는 함수로, PlayerPrefs에서 플레이어 이름과 해당하는 최고 점수를 가져와 rankList에 추가시킴. 그리고 rankList를 점수에 따라 내림차순으로 정렬한 후, 최대 10개의 순위를 텍스트로 표시함.

* UpdatePlayerNameList() : 플레이어 이름 리스트를 업데이트하는 함수로 PlayerPrefs에서 저장된 플레이어 이름을 가져와 nowName이 중복되지 않고 빈 값이 아니면 리스트에 추가하고 PlayerPrefs에 저장함. 이를 통해 중복되지 않는 플레이어 이름 리스트를 유지함

C) 클래스 : ScoreEntry

* public string Name { get; private set; }: 이름을 나타내는 속성

* public int Score { get; private set; }: 점수를 나타내는 속성타내는 속성

C-1) 클래스 함수

* public ScoreEntry(string name, int score): 이름과 점수를 매개변수로 받아서 ScoreEntry 객체를 초기화하는 생성자로 매개변수로 전달된 이름과 점수를 사용하여 멤버 변수인 Name과 Score를 초기화함. 게임 오버 상태에서 랭킹을 출력하는데 사용 함

3. PlayerController

a) 멤버변수 설명

접근지정자	이름	설명
private	GameManager gameManager	게임 매니저 오브젝트에 대한 참조를 저장함. 현재 게임 상태에 따른 동작을 제어하는데 사용
public	float jumpForce	점프할 때 캐릭터에 가해지는 힘의 크기를 결정함
public	float gravity	중력 가속도를 나타내며 플레이어에게 작용하는 중력의 세기를 결정함
private	float verticalVelocity	캐릭터의 수직 속도를 저장하며, 중력과 점프로 인한 이동을 저장하여 플레이어 이동에 적용 시킴
private	Vector2 startPos	캐릭터의 시작 위치를 저장하며, 게임 재시작 시 캐릭터를 원래 위치로 이동시키기 위해 사용
private	bool Isjump	캐릭터가 점프할 수 있는 상태인지를 나타냄, 중복 점프를 방지하기 위해 사용

b) 멤버 함수

* Jump() : 캐릭터를 점프시키는 함수로 gameManager.isCounting 상태가 아닐 때만 점프가 가능하며, 캐릭터에 jumpForce만큼의 수직 속도를 부여함

* JApplyGravity() : 중력을 적용하여 캐릭터를 점프 시키는 함수. 또한 플레이어의 위치가 특정 높이 이하로 내려가면 위치를 처음 위치로 초기화시킴으로 화면 밖으로 이탈하는 것을 방지함

4. (~) move

ston, ston2, igloo, Ice, cloud에 동일하게 적용되며 (~) 부분에 오브젝트 이름이 들어감

a) 멤버변수 설명

접근지정자	이름	설명
private	float Speed	오브젝트의 이동 속도를 나타냄
private	(~)Manager (~)Manager	오브젝트에 해당하는 Manager 오브젝트를 참조함. 오브젝트를 리셋시킬때 ObjectPool 리스트에 다시 추가시킴
private	Vector2 startPos	초기 위치를 저장하며 리셋시킬때 오브젝트를 초기 위치로 옮기기 위해 사용함

private	GameManager gameManager	GameManager 오브젝트에 대한 참조를 저장. 현재 게임 상태에 따른 동작을 제어하는데 사용

b) 멤버 함수

* (~)Reset() : 오브젝트를 초기화 시키는 함수, 오브젝트를 비활성화하고, 초기 위치로 이동시킴. 오브젝트를 다시 (~)ObjectPool에 추가함. 게임 상태가 Ready상태거나 gameManager.isCounting가 아니고 화면 밖으로 이탈할 때 호출함

5. (~) Manager

ston, ston2, igloo, Ice, cloud에 동일하게 적용되며 (~) 부분에 오브젝트 이름이 들어감

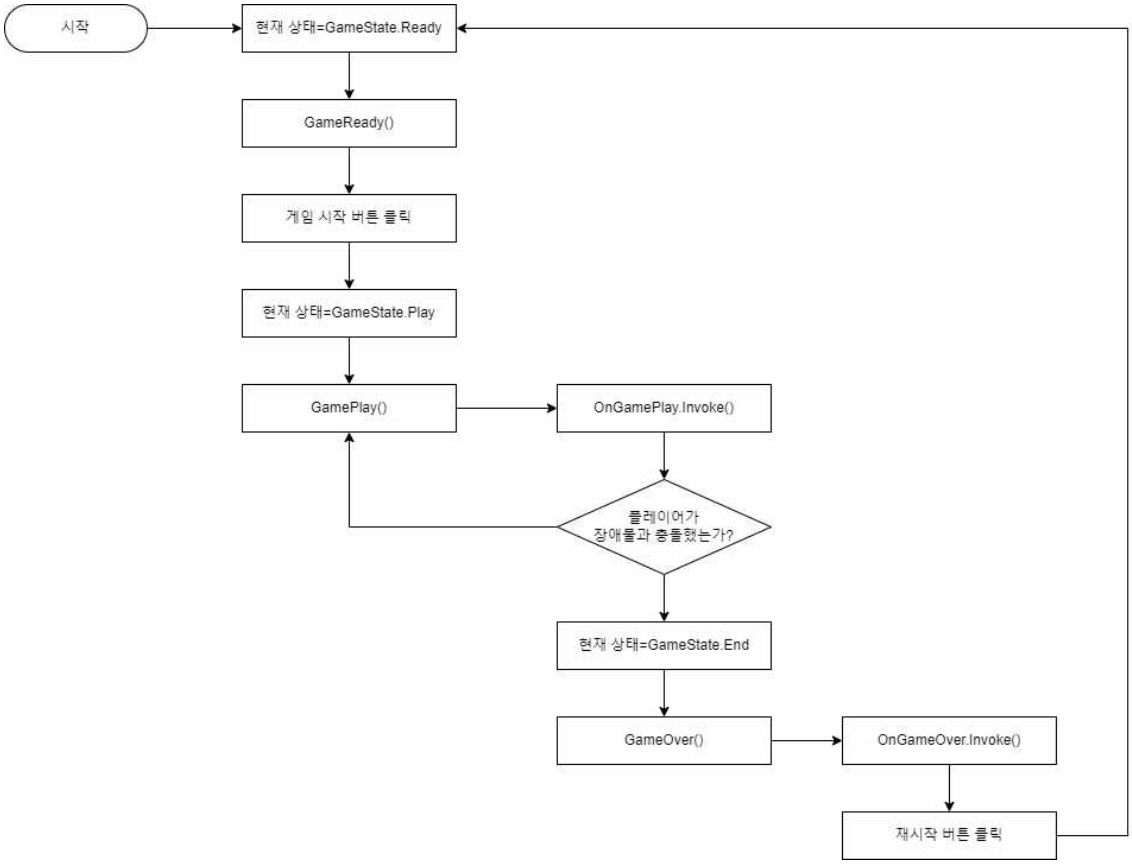
a) 멤버변수 설명

접근지정자	이름	설명
public	GameManager gameManager	GameManager 오브젝트에 대한 참조를 저장. 현재 게임 상태에 따른 동작을 제어하는데 사용
private	AABB aabb	AABB 오브젝트를 참고를 저장. aabb에 오브젝트들의 위치 정보를 넘김
private	float (~)_currentTime	오브젝트의 생성 시간을 측정하는 변수
private	float (~)_minTime	오브젝트의 생성 가능한 최소 시간을 나타냄
private	float (~)_maxTime	오브젝트의 생성 가능한 최대 시간을 나타냄
public	float (~)_createTime	오브젝트의 생성 시간 간격을 조절하며, (~)_minTime~(~)_maxTime 범위 내로 랜덤하게 지정함
public	GameObject (~)Factory	오브젝트 풀을 진행하기위한 오브젝트를 저장함
public	Int (~)poolSize	오브젝트 풀의 크기를 나타냄
public	List<GameObject> (~)ObjectPool	오브젝트 풀을 나타내는 리스트
public	Transform[] (~)spawnPoint	오브젝트를 생성할 수 있는 위치를 나타냄

b) 멤버 함수

* (~)Spawner() : 오브젝트를 생성하는 함수로 gameManager.isCounting가 아닐 때, 오브젝트 생성 시간을 측정하고 일정 시간 간격으로 생성함. 또한 오브젝트 풀에서 비활성화된 오브젝트를 가져와 활성화함.

논리적 구조 설명



동작에 따라 GameManager의 GameState를 바꿔가며 게임 진행을 관리함

시작	시작 시 현재 상태를 GameState.Ready으로 지정하고 GameReady()를 호출해 게임을 초기화 시킴
현재 상태 =GameState.Ready	현재 상태를 Ready로 설정
GameReady()	게임 초기화를 위한 함수 호출. 게임 준비에 해당하는 화면 활성화를 시킴
게임 시작 버튼 클릭	게임 시작 버튼 클릭 시 3초부터 1초 간격으로 줄어드는 카운트다운을 시작 함. 카운트가 0이 되면 게임을 시작함
현재 상태 =GameState.Play	현재 상태를 Play로 지정함
GamePlay()	게임 진행을 위한 함수 호출. 게임 진행에 필요한 화면을 활성화시키고 게임 속도 조절을 위한 플레이 시간을 측정함. 플레이 시간이 15를 넘길때마다 게임 속도를 증가시킴
OnGamePlay.Invoke()	장애물 및 배경 관련 함수를 저장한 OnGamePlay를 실행시킴
플레이어가 장애물과 충돌했는가?	AABB연산을 통해 장애물이 플레이어와 충돌했는지 검사함. 충돌하지 않았을 경우 게임 상태는 변하지 않으므로 실행중이던 GamePlay()를 계속 호출함
True > 현재 상태 =GameState.End	충돌했을 경우 현재 게임 상태를 End로 지정함

GameOver()	게임 오버 화면을 출력시키며 현재 게임 시간을 0으로 변경해 게임을 멈춤
OnGameOver.Invoke()	랭크를 화면에 출력시키기 위해 랭크를 보여주는 함수를 저장한 OnGameOver를 실행시킴
재시작 버튼 클릭	재시작 버튼을 클릭할 시 현재 게임 상태를 Read로 지정해 다시 게임을 진행하도록 함