삽질부터 시작하는 유니티

참고도서 : 개정판 유니티 4 게임 개발의 정석

유니티 소개

- 씬 뷰 : 게임의 무대로 사용되는 3차원 공간의 설정
- 계층 뷰 : 게임 오브젝트의 관리
- 인스펙터 뷰 : 게임 오브젝트가 갖는 컴포넌트들의 관리
- 프로젝트 뷰 : 게임에 사용되는 에셋들의 관리
- 게임 뷰 : 제작된 콘텐츠가 올바로 동작하는지 점검
- 콘솔 뷰 : 에디터에서 발생하는 각종 문제점에 대한 메시지를 출력

통합 애셋 시스템

게임 제작에 사용되는 다양한 종류의 파일 데이터인 **애셋**

=> 프로젝트 뷰 인터페이스에서 확인 가능.

--> 4장에서 자세히

셰이더 시스템

: 저사양의 모바일부터 데스크탑, 콘솔 기기를 모두 관리할 수 있다고 함.

더 알아봐야 할듯-->5장에서

> https://www.slideshare.net/jungsoopark104/ss-63417653

지형생성기능

도 있다고함...신기신기

우리 게임에서 아직 쓸 일은 별로 없을 것 같지만 혹시나 해서

유니티 오디오 & 사운드

-> 9장 후반부

애니메이션

--> 8장

파티클 시스템

: 게임의 효과를 증폭시키는 이펙트 제작

물리엔진기능

: 피직스 엔진(3차원!)

프로그래밍기능

: 스크립트를 통해 구현할 수 있는 강력한 라이브러리 제공

모노디벨롭(개발도구)

2D 게임 기능

--> 7장

게임 오브젝트와 컴포넌트

게임 오브젝트 : 플레이어와 함께 상호작용하는 고유한 물체의 단위 - 계층 뷰에서 관리

컴포넌트 : 하나의 컴포넌트는 독립된 기능을 수행하며, 더욱 복잡한 역할을 수행하게 됩니다. -> 인스펙터 뷰

물리 엔진

포물선 물리 현상 구현?

=> 유니티에서 제공하는 물리 컴포넌트인 **리지드바디**를 사용해 던지는 방향으로 지정된 힘을 전달.

=> 충돌 영역을 지정해하는 충돌체 컴포넌트를 추가해주면 자동으로 충돌 감지.

- 리지드바디 : 어떠한 힘을 받아도 변하지 않는 단단한 물체 (강체..우와 물리다!)
    - 질량
    - 저항
    - 각저항
    - 속도
- 충돌체
    - 영역 설정
    - 마찰
    - 탄성

트리거 시스템

물체의 충돌 여부만 판단하는 기능

ex) 레이싱 게임에서 자동차 랩 타이 ㅁ기록

RPG 적 출현 등등

7장 스크립팅 시스템

스크립팅을 위해 자바스크립트, C#, 부의 세 종류 언어 제공.

- 자바스크립트

  : 간편한 문법, 코드 복잡해지면 관리하기 힘들다.

- C#

  : 처음 설정이 다소 복잡함. 다양한 키워드를 제공해 구조적이고 정교한 프로그래밍 가능

프레임 함수

게임은 영사기에서 필름을 돌리는 것처럼 프로그램이 종료될 때까지 화면 생산, 모니터에 뿌려줘야 함.

=> 프레임

유니티도 프레임을 기준으로 동작하며, 콘텐츠를 재생하면 유니티 프로그램이 동작하면서 모든 컴포넌트가 초기화를 거친 후, 프레임 단위로 동작.

제작한 스크립트 컴포넌트를 유니티가 실행하려면?

1. 모든 컴포넌트는 MonoBehaviour 클래스로부터 상속받아야 함.
2. 유니티가 지정한 이름의 함수를 구현해야 함.

- Awake : 컴포넌트에서 가장 먼저 호출하는 함수
- Start : 처음 프레임을 그리기 전에 한 번 호출되는 함수
- Update : 프레임마다 호출되는 함수
- LateUpdate : Update 함수에 이어서 호출되는 함수

유니티 스크립트 시스템의 초기화 프로세스

1. 생성자 및 변수 초기화
2. 유니티 엔진 초기화
3. 저장된 값으로 변수 값 변경
4. Awake, Start 함수 실행

입력 관리자

: 해당하는 축의 입력만 신경 쓰면 된다.

2D 게임 제작

스프라이트 컴포넌트

스프라이트 : 정점 4개로 구성한 평면 사각형에 투명한 이미지 배치

유니티 강의

B1. 유니티 에디터 기초

프로젝트

계층구조 : 오브젝트 생성

- q: 뷰 이동
- w : 이동
- e : 회전 (회색으로 자유롭게 회전)
- r : 크기
- t : 사각툴 (2D 모델링 ui)
- 오른쪽 마우스 : 카메라 회전
- alt + 클릭 : 카메라 축 회전

장면

인스펙터 : 오브젝트 속성

B3. 게임 구조와 프로그래밍

C# 스크립트 파일

B4. C# 기초 문법

```
//NewBehaviourScript.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//MonoBehaviour : 유니티 게임 오브젝트 클래스

public class NewBehaviourScript : MonoBehaviour
{
    //전역 변수 선언 (필수 정보)
    int health = 30;
    int level = 5;
    float strength = 15.5f;
    string playerName = "김현수";
    bool isFullLevel = false;
```

```
void Start()
{
    //지역 변수 선언
    Debug.Log("이름");
    Debug.Log(playerName);
    Debug.Log("레벨");
    Debug.Log(level);
    Debug.Log("힘");
    Debug.Log(strength);
    Debug.Log("렙");
    Debug.Log(isFullLevel);

    //그룹형 변수(배열)
    string[] monsters = { "슬라임", "사막뱀", "악마" };
    int[] monsterLevel = { 1, 6, 20 };

    Debug.Log("레벨");
    Debug.Log(monsterLevel[0]);

    //Generic *새로운 문법*
    // 데이터 삭제 가능
    List<string> items = new List<string>();
    items.Add("생명물약30");

    Debug.Log("가진 아이템");
    Debug.Log(items[0]);


    //데이터 삭제
    items.RemoveAt(0);

    //문자열 연산
    string title = "전설의";
    Debug.Log("용사의 이름은?");
    Debug.Log(title + " " + playerName);

    //불 연산 이용
    int fullLevel = 99;
    isFullLevel = level == fullLevel;
    Debug.Log("용사는 만렙입니까?" + isFullLevel);

    bool isEndTutorial = level > 20;
    Debug.Log("튜토리얼 끝?" + isEndTutorial);

    int health = 30, mana = 25;
    bool isBadCondition;
    isBadCondition = health <= 50 && mana <= 20;

    //삼항 연산자
    string condition = isBadCondition ? "나쁨" : "좋음";
    Debug.Log("뒤짐" + isBadCondition);

    //조건문
    if(condition == "나쁨")
    {
        Debug.Log("나쁨");
    }
```

```
        else
        {
            Debug.Log("좋음");
        }

        if(isBadCondition && items[0] == "생명물약30")
        {
            items.RemoveAt(0);
            health += 30;
            Debug.Log("생명포션 사용");
        }

        switch (monsters[0])
        {
            case "슬라임":
                Debug.Log("소형");
                break;
            case "사막뱀":
                Debug.Log("중형");
                break;
            case "악마":
                Debug.Log("대형");
                break;
            default:
                Debug.Log("???");
                break;
        }

        //반복문
        /*while(health > 0)
        {
            health--;
            Debug.Log("독데ㅐㅁ" + health);
        }
        */

        for(int count=0; count<10; count++)
        {
            health++;
            Debug.Log("붕대로 치료" + health);
        }

        //Length : 배열변수 길이, Count : 리스트변수 길이
        for(int index = 0; index < monsters.Length; index++)
        {
            Debug.Log("지역 몬스터 : " + monsters[index]);
        }

        //새로운 문법
        foreach(string monster in monsters)
        {
            Debug.Log("이 지역에 있는 몬스터 : " + monster);
        }

        //함수 사용
        health = Heal(health);
        Heal2();
```

```
        //함수와 반복문의 조합
        for (int index = 0; index < monsters.Length; index++)
        {
            Debug.Log("용사는" + monsters[index] + "에게"
                + Battle(monsterLevel[index]));
        }

        //클래스 : 하나의 사물(오브젝트)와 대응하는 로직
        //유니티에서는 보통 하나의 클래스에 하나의 오브젝트 사용
        Player player = new Player();

        //접근자를 통해 접근 권한 달라짐.
        //아무것도 안치면 private로 외부 클래스에서 접근 불가능!
        player.id = 0;
        player.name = "나법사";
        player.title = "현명한";
        player.strength = 2.4f;
        player.weapon = "나무 지팡이";
        Debug.Log(player.Talk());
        Debug.Log(player.HasWeapon());

        player.LevelUp();
        Debug.Log(player.name + "의 레벨은 " + player.level + "입니다.");
    }

    //함수(메소드) 정의
    int Heal(int health)
    {
        health += 10;
        Debug.Log("힐 받음" + health);
        return health;
    }

    /*health 지역 변수이므로 사용 불가*/

    void Heal2()
    {
        health += 10;
        Debug.Log("힐 받음" + health);
    }

    string Battle(int monsterLevel)
    {
        string result;
        if (level >= monsterLevel)
            result = "이겼습니다.";
        else
            result = "졌습니다.";
        return result;
    }
}
```

```
//player.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
//Actor 에 있었던 함수를 그대로 상속받을 수 있다.
//Actor : 부모클래스, Player : 자식 클래스 => 상속관계

public class Player : Actor
{
    public string move()
    {
        return "플레이어는 움직입니다.";
    }
}
```

```
//actor.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Actor
{
    public int id;
    public string name;
    public string title;
    public string weapon;
    public float strength;
    public int level;

    public string Talk()
    {
        return "대화를 걸었습니다.";
    }

    public string HasWeapon()
    {
        return weapon;
    }

    public void LevelUp()
    {
        level += 1;
    }


}
```

B5. 오브젝트 라이프사이클

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{
    //초기화 영역
```

```
    //Awake : 게임 오브젝트 생성 시, 최초 실행

    private void Awake()
    {
        Debug.Log("플레이어 데이터가 준비됨");
    }

    //Start : 업데이트 시작 전, 최초 실행
    private void Start()
    {
        Debug.Log("사냥 장비를 챙겼다.");
    }

    //오브젝트 활성화(초기화영역~물리연산 영역 사이에 위치)
    //켜고 끌때마다 실행
    private void OnEnable()
    {
        Debug.Log("플레이어가 로그인했다.");
    }


    //물리연산 영역
    //FixedUpdate : 물리연산업데이트
    private void FixedUpdate()
    {
        Debug.Log("이동~");
    }

    //게임 로직 업데이트
    //pc 환경에 따라 실행 주기가 떨어질 수 있음.
    private void Update()
    {
        Debug.Log("몬스터 사냥!!");
    }

    //LateUpdate : 모든 업데이트 끝난 후
    //카메라나 로직의 후처리
    private void LateUpdate()
    {
        Debug.Log("경험치 획득");
    }

    //오브젝트 비활성화(게임로직~해체 영역)
    private void OnDisable()
    {
        Debug.Log("플레이어가 로그아웃");
    }


    //해체 영역
    //OnDestroy : 게임 오브젝트가 삭제될 때
    private void OnDestroy()
    {
        Debug.Log("플레이어 데이터를 해제하였습니다.");
    }
}
```

B6. 키보드, 마우스로 오브젝트 이동시키기

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{

    //게임 로직 업데이트
    //pc 환경에 따라 실행 주기가 떨어질 수 있음.
    void Update()
    {
        //Input : 게임 내 입력을 관리하는 클래스

        //아무 입력을 최초로 받을 때 true(불값 return)
        if (Input.anyKeyDown)
            Debug.Log("사용자의 입력");

        //아무 입력을 받고 있을 때 true(불값 return)
        if (Input.anyKey)
            Debug.Log("사용자가 입력 중");

    }

}
```

키보드 입력받기

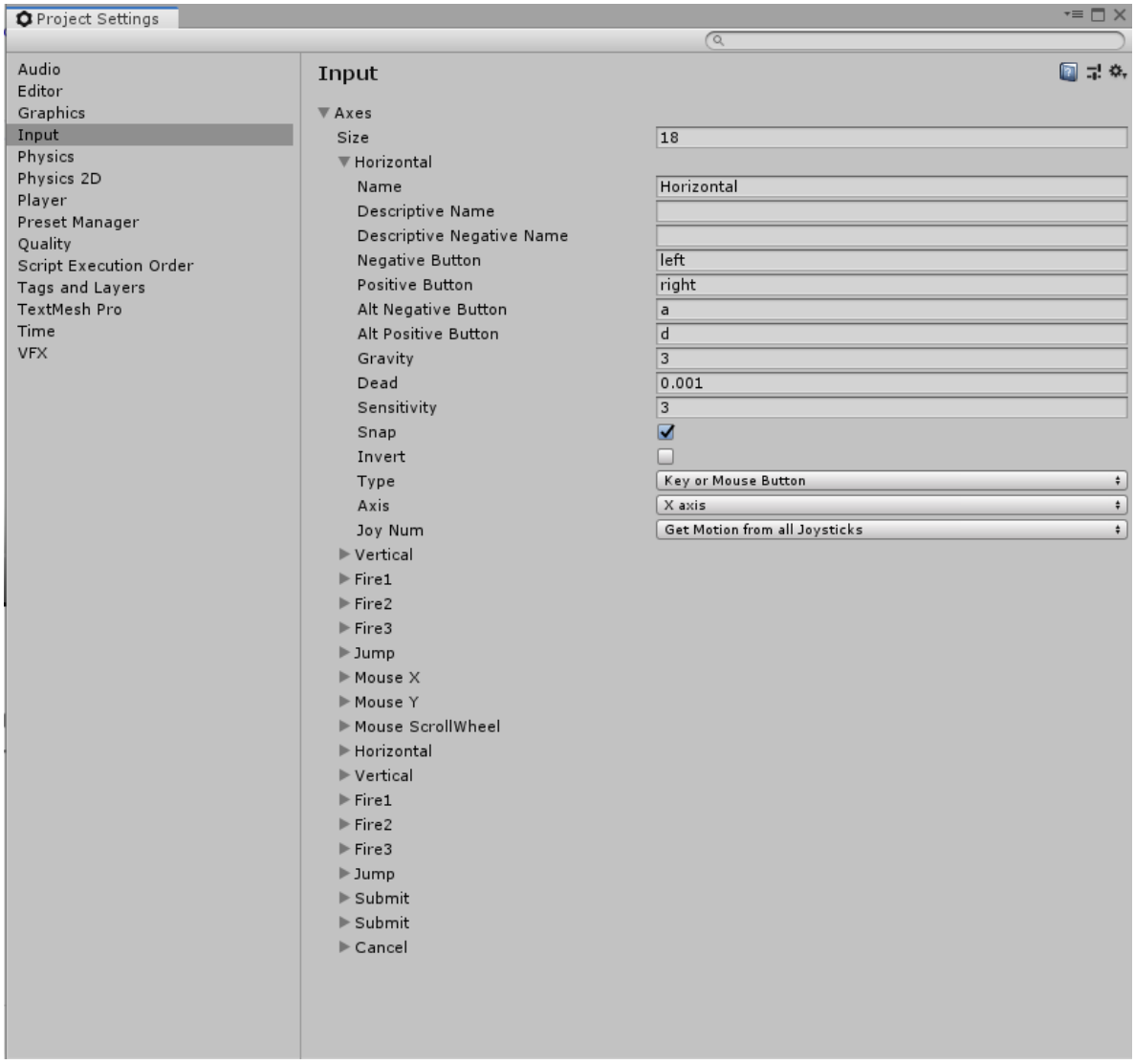
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{

    //게임 로직 업데이트
    //pc 환경에 따라 실행 주기가 떨어질 수 있음.
    void Update()
    {
        //Input : 게임 내 입력을 관리하는 클래스

        //아무 입력을 최초로 받을 때 true(불값 return)
        if (Input.anyKeyDown)
            Debug.Log("사용자의 입력");

        //각 입력함수는 세 가지로 구분한다.
        /*
         Down : 눌렸을 때
```

```
         Stay : 누르고 있을 때
         Up : 뗐을 때
        */

        //GetKey : 키보드버튼 입력을 받으면 true
        //enter == Return
        //esc == Escape
        //나머지 키보드버튼은 그대로

        if (Input.GetKeyDown(KeyCode.Return))
            Debug.Log("아이템을 구입하였습니다.");
        if (Input.GetKey(KeyCode.LeftArrow))
            Debug.Log("왼쪽으로 이동 중");
        if (Input.GetKeyUp(KeyCode.RightArrow))
            Debug.Log("오른쪽 이동을 멈추었습니다.");
    }
}
```

마우스 입력받기

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{

    //게임 로직 업데이트
    //pc 환경에 따라 실행 주기가 떨어질 수 있음.
    void Update()
    {
        //Input : 게임 내 입력을 관리하는 클래스

        //아무 입력을 최초로 받을 때 true(불값 return)
        if (Input.anyKeyDown)
            Debug.Log("사용자의 입력");

        /* 0 : 마우스 왼쪽 버튼
           1 : 마우스 오른쪽 버튼*/
        if (Input.GetMouseButtonDown(0))
            Debug.Log("미사일 발사!");
        if (Input.GetMouseButton(0))
            Debug.Log("미사일 모으는 중..");
        if (Input.GetMouseButtonUp(0))
            Debug.Log("슈퍼 미사일 발사!");
    }
}
```

Project settings > Input Manager 에서 Button 설정 가능하다.



horizontal : 횡 이동

vertical : 종 이동

특정 Input 으로 입력받기

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{
    void Update()
    {
        //GetButton : 해당하는 input의 버튼 입력을 받으면 True
        //현재는 "Jump" Input의 기본값인 up, s로 설정되어 있다.
        if (Input.anyKeyDown)
            Debug.Log("플레이어가 아무 키를 누름.");
        if (Input.GetButton("Jump"))
            Debug.Log("Jump charging..");
        if (Input.GetButtonUp("Jump"))
            Debug.Log("Super Jump!");
    }
}
```

횡 이동

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{
    void Update()
    {
        //GetButton : 해당하는 input의 버튼 입력을 받으면 True
        //현재는 기본값인 w, s로 설정되어 있다.

        if (Input.anyKeyDown)
            Debug.Log("플레이어가 아무 키를 눌렀다.");

        if (Input.GetButton("Horizontal"))
        {
            Debug.Log("횡 이동 중..." + Input.GetAxis("Horizontal"));
            //GetAxis : 수평, 수직 버튼 입력을 받으면 float 값 반환
        }
    }
}
```

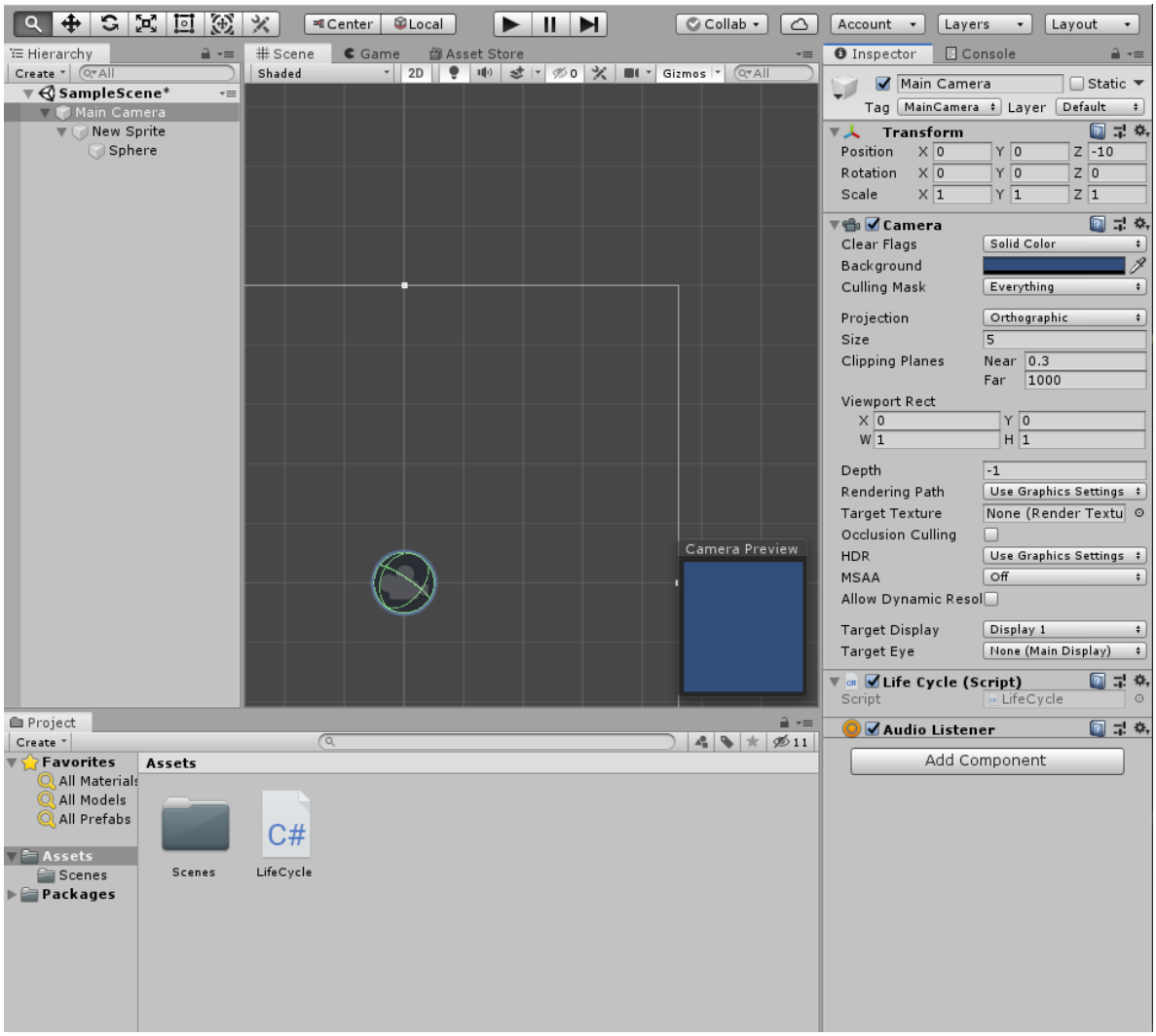
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{
    void Update()
    {
        //GetButton : 해당하는 input의 버튼 입력을 받으면 True
        //현재는 기본값인 w, s로 설정되어 있다.

        if (Input.anyKeyDown)
            Debug.Log("플레이어가 아무 키를 눌렀다.");

        if (Input.GetButton("Horizontal"))
        {
            Debug.Log("횡 이동 중..." + Input.GetAxisRaw("Horizontal"));
            //GetAxis : 수평, 수직 버튼 입력을 받으면 float 값 반환 (가중치 O)
            //GetAxisRaw : 가중치 없이 일정하게 가게 함
        }
    }
}
```

벡터 값 이동

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class LifeCycle : MonoBehaviour
{
    //실제 오브젝트 이동시키기
    //Transform : 오브젝트 형태에 대한 기본 컴포넌트
    /*게임 오브젝트는 Transform과 일대일 대응관계이기 때문에
      따로 변수 만들 필요 X*/
    void Start()
    {
        //Translate : 벡터 값을 현재 위치에 더하는 함수
        //Vector2 : 2차원, Vector3 : 3차원
        Vector3 vec = new Vector3(1, 0, 0);
        transform.Translate(vec);
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{
    //update 안에 있으므로 지속적으로 0.1씩 움직이는 것을 확인할 수 있다.
    void Update()
    {
        Vector3 vec = new Vector3(0.1f, 0, 0);
        transform.Translate(vec);
    }
}
```



메인 카메라에 같이 이동하는 스크립트를 사용하면 오브젝트에 카메라가 고정된 효과를 낼 수 있다.

> 와 신기하다!

버튼 입력을 통한 오브젝트 이동

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeCycle : MonoBehaviour
{
    void Update()
    {
        Vector3 vec = new Vector3
            (Input.GetAxis("Horizontal"),
            Input.GetAxis("Vertical"),
            0);
        transform.Translate(vec);
    }
}
```

B7. 목표 지점으로 이동시키기

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    Vector3 target = new Vector3(8, 1.5f, 0);
    void Update()
    {
        // MoveTowards : 등속 이동
        // (현재위치, 목표위치, 속도)로 구성
        // 마지막 매개변수에 비례하여 속도 증가
        transform.position =
            Vector3.MoveTowards(transform.position
                                , target, 0.5f);
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    Vector3 target = new Vector3(8, 1.5f, 0);
    void Update()
    {
        //SmoothDamp
        //(현재위치, 목표위치, 참조 속도, 속도)
        // 마지막 매개변수에 반비례하여 속도 증가
        // 참조 속도 : 참조 접근 -> 실시간으로 바뀌는 값 적용 가능

        Vector3 velo = Vector3.zero;

        transform.position =
            Vector3.SmoothDamp(transform.position,
                               target, ref velo, 1f);
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
```

```
{
    Vector3 target = new Vector3(8, 1.5f, 0);
    void Update()
    {
        //SmoothDamp
        //(현재위치, 목표위치, 참조 속도, 속도)
        Vector3 velo = Vector3.up * 2;

        transform.position =
            Vector3.SmoothDamp(transform.position,
                               target, ref velo, 1f);
    }
}
```

하지만 참조 속도에 방향값을 줘버리면 목표위치에 도달 X하기 때문에 보통은 zero로 두고 쓴다.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    Vector3 target = new Vector3(8, 1.5f, 0);
    void Update()
    {
        //Lerp
        //선형 보간, SmoothDamp보다 감속 시간이 김.
        //마지막 매개변수에 비례해 속도 증가
        transform.position =
            Vector3.Lerp(transform.position
                               , target, 2f);
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    Vector3 target = new Vector3(8, 1.5f, 0);
    void Update()
    {
        //Lerp
        //구면 선형 보간, 호를 그리며 이동.
        transform.position =
            Vector3.Slerp(transform.position
                               , target, 2f);
    }
}
```

B8. DeltaTime(델타 타임)

Time.deltaTime 사용하는 방법

Translate : 벡터에 곱하기

transform.Translate(Vec * Time.deltaTime);

Vector : 시간 매개변수에 곱하기

Vector3.Lerp(Vec1, Vec2, T * Time.deltaTime);

성능마다 fps의 차이.

==> deltaTime을 이용.

deltaTime 값은 프레임이 적으면 크고, 많으면 작음.

B13. 2D 프로젝트 실행

SpriteRenderer

스프라이트를 보여주는 컴포넌트

Camera > Size를 조절해 오브젝트 줌인, 줌아웃 가능.

Camera > Orthographic : 원근법이 벗는 정사영 투시

SpriteRenderer : 이미지 색상 변경 가능

2D 오브젝트가 겹칠 때

1. Z축 활용
2. Order in Layer 값이 높을수록 위로 얹어지는 것을 활용.

폴더 만든 다음 그래픽 불러오기.

2D 프로젝트에서는 자동으로 스프라이트 사용.

sprite 클릭 > Filtermode > 번지지 않게 Point로 설정

압축 방식에 따라 색깔이 변화할 수 있음.

크기가 작은 도트 그래픽의 경우 압축 none 설정.

sprite mode > pixels per unit (한 칸에 픽셀이 몇 개나 들어가는지)

> 이미지 크기에 따른 설정

2D 물리엔진 적용

Add Component > **Box Collider 2D**

> **Rigidbody 2D**

낙하효과 적용 가능.

> 그래픽이 너무 작을 경우 오차 생길 수 있음.

오차 줄이기

Project Settings > Physics 2D > Default Contact Offset 에서 조정

B14. 2D 아틀라스와 애니메이션

아틀라스 생성 및 스프라이트 분리.

여러 스프라이트 끌어다놓으면 애니메이션 생성 가능!

B15. 2D 플레이어 이동 구현하기

이동 스크립트

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    Rigidbody2D rigid;

    private void Awake()
    {
        rigid = GetComponent<Rigidbody2D>();
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");

        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.
```

```
            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }
    }
}
```

마찰력 조정

Physics Material 2D

> 마찰력, 탄성력으로 구성

--> Box Colider 2D에 삽입

저항 조정

Rigidbody 2D > Linear Drag

> 공기 저항, 이동 시 속도를 느리게 해줌.

멈추기 (문워크 방지)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    Rigidbody2D rigid;

    private void Awake()
    {
        rigid = GetComponent<Rigidbody2D>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
```

```
        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }
    }
}
```

누르는 키에 따라 이미지 Flip

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    Rigidbody2D rigid;
    SpriteRenderer spriteRenderer;
```

```
    private void Awake()
    {
        //초기화
        rigid = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }

        //방향전환
        if(Input.GetButtonDown("Horizontal"))
            spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }
    }
}
```

애니메이터 매개변수

상태를 바꿀 때 필요한 변수

매개변수 설정 후 Conditions 에서 활용.

Has Exit Time : 애니메이션이 끝날 때까지 상태 유지

==> 쌍방향으로 만들어야 함에 유의!

다음 스크립트에서 만들기.

<매개변수 조정 스크립트>

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    Rigidbody2D rigid;
    SpriteRenderer spriteRenderer;
    Animator anim;
    private void Awake()
    {
        //초기화
        rigid = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }

        //방향전환
        if(Input.GetButtonDown("Horizontal"))
            spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;

        //Walking Animation
        //Mathf : 수학 관련 함수 제공
        if (Mathf.Abs(rigid.velocity.x) < 0.3)
            anim.SetBool("isWalk", false);
```

```
        else
            anim.SetBool("isWalk", true);
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }
    }
}
```

B16. 2D 플레이어 점프 구현하기

물리점프

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    //JumpPower 변수 추가
    public float JumpPower;
    Rigidbody2D rigid;
    SpriteRenderer spriteRenderer;
    Animator anim;
    private void Awake()
    {
```

```
        //초기화
        rigid = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
        //Jump
        if (Input.GetButtonDown("Jump"))
        {
            rigid.AddForce(Vector2.up  * JumpPower, ForceMode2D.Impulse);
        }
        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }

        //방향전환
        if (Input.GetButtonDown("Horizontal"))
        {
            spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;
        }

        //Walking Animation
        //Mathf : 수학 관련 함수 제공
        if (Mathf.Abs(rigid.velocity.x) < 0.3)
            anim.SetBool("isWalk", false);
        else
            anim.SetBool("isWalk", true);
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
```

```
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }
    }
}
```

project settings > physics 2D

에서 전체 프로젝트의 중력 조절 가능.

위와 같은 방식으로 애니메이션 적용.

But, bool값을 어디서 false로 바꿔야 할 지의 문제가 생김.

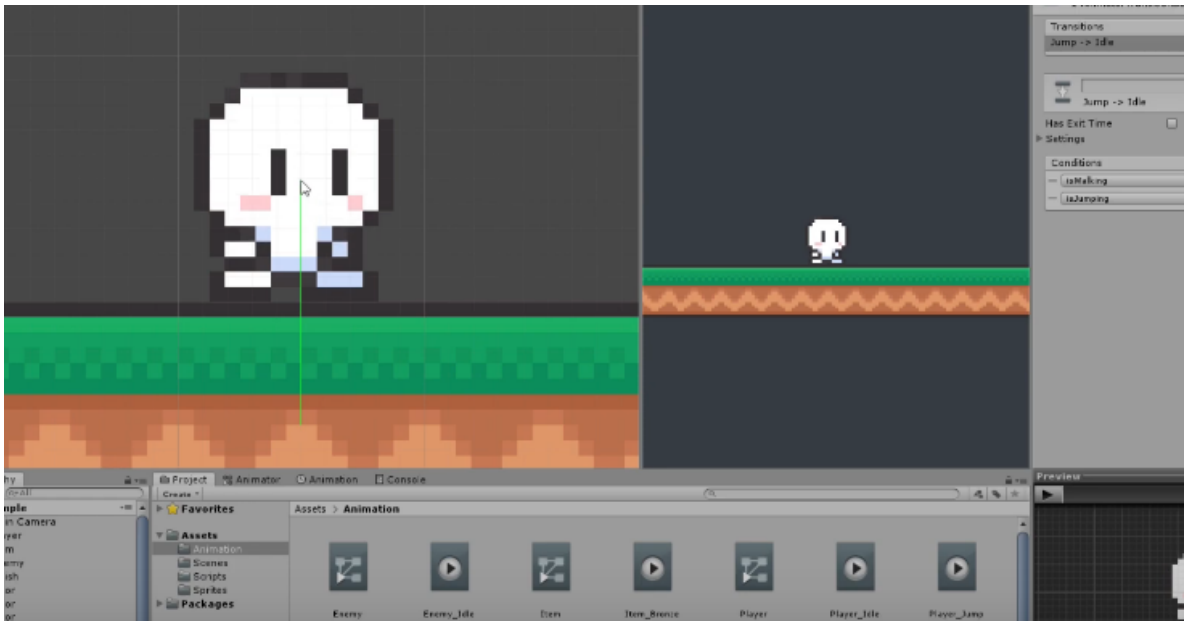
레이 캐스트

: 오브젝트 검색을 위해 Ray를 쏘는 방식

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    public float JumpPower;
    Rigidbody2D rigid;
    SpriteRenderer spriteRenderer;
    Animator anim;
    private void Awake()
    {
        //초기화
        rigid = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
        //Jump
        if (Input.GetButtonDown("Jump"))
        {
            rigid.AddForce(Vector2.up  * JumpPower, ForceMode2D.Impulse);
```

```
            //jump 되었을 때 불값 변경
            anim.SetBool("isJump", true);
        }


        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }

        //방향전환
        if (Input.GetButtonDown("Horizontal"))
        {
            spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;
        }

        //Walking Animation
        //Mathf : 수학 관련 함수 제공
        if (Mathf.Abs(rigid.velocity.x) < 0.3)
            anim.SetBool("isWalk", false);
        else
            anim.SetBool("isWalk", true);
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }

        //RayCast
```

```
        //Landing Platform
        //DrawRay : 에디터 상에서만 Ray를 그려주는 함수
        Debug.DrawRay(rigid.position, Vector3.down, new Color(0, 1, 0));
    }
}
```



> 다음과 같은 녹색 빔이 생긴다(!)

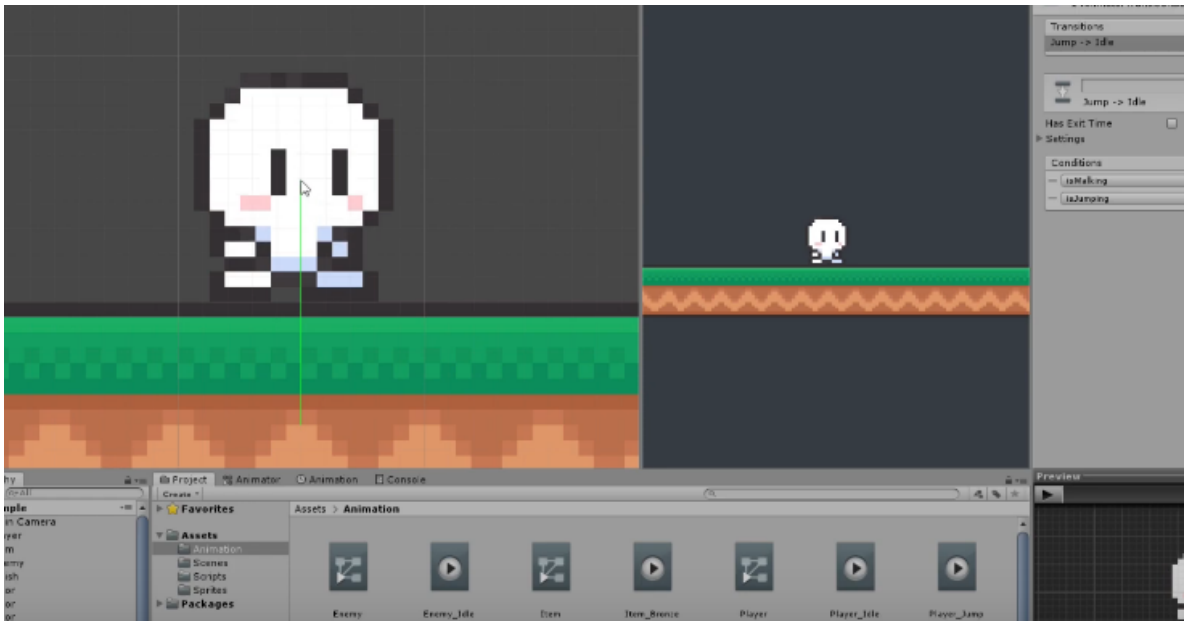
다음과 같은 코드를 통해 ray에 맞은 오브젝트를 밝혀내보자.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    public float JumpPower;
    Rigidbody2D rigid;
    SpriteRenderer spriteRenderer;
    Animator anim;
    private void Awake()
    {
        //초기화
        rigid = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
        //Jump
        if (Input.GetButtonDown("Jump"))
        {
            rigid.AddForce(Vector2.up  * JumpPower, ForceMode2D.Impulse);
            //jump 되었을 때 불값 변경
            anim.SetBool("isJump", true);
```

```
        }


        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }

        //방향전환
        if (Input.GetButtonDown("Horizontal"))
        {
            spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;
        }

        //Walking Animation
        //Mathf : 수학 관련 함수 제공
        if (Mathf.Abs(rigid.velocity.x) < 0.3)
            anim.SetBool("isWalk", false);
        else
            anim.SetBool("isWalk", true);
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }

        //RayCast
        //Landing Platform
        //DrawRay : 에디터 상에서만 Ray를 그려주는 함수
```

```
        Debug.DrawRay(rigid.position, Vector3.down, new Color(0, 1, 0));

        //RayCastHit : Ray에 닿은 오브젝트 정보 저장
        RaycastHit2D RayHit = Physics2D.Raycast(rigid.position, Vector3.down,
1);

        //collider null 값 아닌 경우 내용물이 있는 것.
        if(RayHit.collider != null)
        {
            Debug.Log(RayHit.collider.name);
        }
    }
}
```

ray는 통과 X하기 때문에 처음 부딪힌 플레이어만 판정이 뜸.

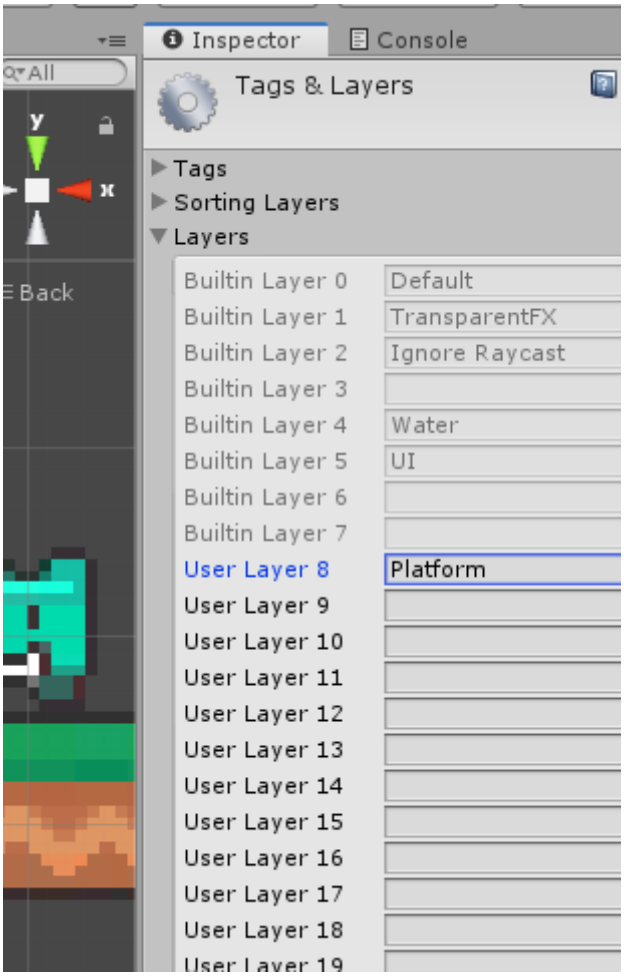


> 다시 확인해보자

그렇다면?

Inspector 상단의 Layer

LayerMask : 물리효과를 구분하는 정수값



바닥 전체에 레이어 설정

GetMask() : 레이어 이름에 해당하는 정수 값을 리턴하는 함수.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    public float JumpPower;
    Rigidbody2D rigid;
    SpriteRenderer spriteRenderer;
    Animator anim;
    private void Awake()
    {
        //초기화
        rigid = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
        //Jump
        if (Input.GetButtonDown("Jump"))
```

```
        {
            rigid.AddForce(Vector2.up  * JumpPower, ForceMode2D.Impulse);
            //jump 되었을 때 불값 변경
            anim.SetBool("isJump", true);
        }


        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }

        //방향전환
        if (Input.GetButtonDown("Horizontal"))
        {
            spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;
        }

        //Walking Animation
        //Mathf : 수학 관련 함수 제공
        if (Mathf.Abs(rigid.velocity.x) < 0.3)
            anim.SetBool("isWalk", false);
        else
            anim.SetBool("isWalk", true);
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }
```

```
        //RayCast
        //Landing Platform
        //DrawRay : 에디터 상에서만 Ray를 그려주는 함수
        Debug.DrawRay(rigid.position, Vector3.down, new Color(0, 1, 0));

        //RayCastHit : Ray에 닿은 오브젝트 정보 저장
        RaycastHit2D RayHit = Physics2D.Raycast(rigid.position, Vector3.down, 1,
LayerMask.GetMask("8"));

        //collider null 값 아닌 경우 내용물이 있는 것.
        if(RayHit.collider != null)
        {
            Debug.Log(RayHit.collider.name);
        }
    }
}
```

//무한 점프 방지 및 점프 애니메이션 총정리

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    public float MaxSpeed;
    public float JumpPower;
    Rigidbody2D rigid;
    SpriteRenderer spriteRenderer;
    Animator anim;
    private void Awake()
    {
        //초기화
        rigid = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    //단발성 키 입력은 FixedUpdate말고 Update에
    private void Update()
    {
        //Jump & 중복 Jump 방지
        if (Input.GetButtonDown("Jump") && !anim.GetBool("isJump"))
        {
            rigid.AddForce(Vector2.up  * JumpPower, ForceMode2D.Impulse);
            //jump 되었을 때 불값 변경
            anim.SetBool("isJump", true);
        }


        //PlayerStop
        //키보드 뗐을 때 속력 급격히 감속.
        if (Input.GetButtonUp("Horizontal"))
        {
```

```
            //nomalized : 벡터 크기를 1로 만든 상태(단위벡터)
            //getaxisraw와 비슷
            rigid.velocity = new Vector2(rigid.velocity.normalized.x * 0.5f,
rigid.velocity.y);
        }

        //방향전환
        if (Input.GetButtonDown("Horizontal"))
        {
            spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;
        }

        //Walking Animation
        //Mathf : 수학 관련 함수 제공
        if (Mathf.Abs(rigid.velocity.x) < 0.3)
            anim.SetBool("isWalk", false);
        else
            anim.SetBool("isWalk", true);
    }

    private void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        //MaxSpeedLimit
        //FixedUpdate에 포함되어 있으므로 꾹 누르면 가속 무한됨.
        rigid.AddForce(Vector2.right * h, ForceMode2D.Impulse);

        //velocity : rigidbody의 현재 속도
        if(rigid.velocity.x > MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(MaxSpeed, rigid.velocity.y);

        }

        //LeftMaxSpeed : 음수 값임에 주의
        else if (rigid.velocity.x < -MaxSpeed)
        {
            //속도 최댓값 이하로 고정.
            //y 속도 0으로 설정하면 그냥 멈춰버림.

            //RightMaxSpeed
            rigid.velocity = new Vector2(-MaxSpeed, rigid.velocity.y);
        }

        //내려가고 있을 때만 ray 투과.
        if (rigid.velocity.y < 0)
        {
            //RayCast
            //Landing Platform
            //DrawRay : 에디터 상에서만 Ray를 그려주는 함수
            Debug.DrawRay(rigid.position, Vector3.down, new Color(0, 1, 0));

            //RayCastHit : Ray에 닿은 오브젝트 정보 저장
```

```
            RaycastHit2D RayHit = Physics2D.Raycast(rigid.position,
Vector3.down, 1, LayerMask.GetMask("Platform"));

            //충돌 판정
            //collider null 값 아닌 경우 내용물이 있는 것.
            if (RayHit.collider != null)
            {
                //플레이어가 바닥에 완전히 착지했는지 여부
                //distance : Ray에 닿았을 때의 거리
                if (RayHit.distance < 0.5f)
                {
                    //충돌 확인 : Debug.Log(RayHit.collider.name);
                    //착지했을 때 불값 변경
                    anim.SetBool("isJump", false);
                }
            }
        }
    }
}
```

B17. 타일맵으로 플랫폼 만들기

Tile Palette 생성 : 타일을 사용하기 위해 모아둔 프리펩

Tile Palette에 스프라이트 삽입 및 기타 등등..

Collider 설정 따로 해줘야 함

==> **Tilemap collider 2D** component 삽입

> Layer 설정도 똑같이!

카메라 설정

Main Camera-> Player(charactor) 폴더 안에 삽입

경사면 타일맵

타일맵 물리 모양은 **Sprite Editor**에서 편집 가능.

sprite editor > generate 통해 편집.

B18. 몬스터 AI 구현하기

```
// 한 방향으로 쭉 움직이기.
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class NewBehaviourScript : MonoBehaviour
{
    //한 방향으로 이동
    Rigidbody2D rigid;

    void Awake()
    {
        rigid = GetComponent<Rigidbody2D>();
    }


    void FixedUpdate()
    {
        rigid.velocity = new Vector2(-1, rigid.velocity.y);
    }
}
```

행동설정

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    //한 방향으로 이동
    Rigidbody2D rigid;
    Animator anim;
    SpriteRenderer spriteRenderer;

    //행동지표 결정할 변수 생성
    public int nextMove;

    void Awake()
    {
        rigid = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        spriteRenderer = GetComponent<SpriteRenderer>();

        //Invoke : 주어진 시간이 지난 뒤, 지정된 함수를 실행하는 함수
        Invoke("Think", 2);
    }


    void FixedUpdate()
    {
        //Move
        rigid.velocity = new Vector2(nextMove, rigid.velocity.y);

        //Platform Check
        Vector2 frontVec = new Vector2(rigid.position.x + nextMove * 0.2f,
rigid.velocity.y);
        Debug.DrawRay(frontVec, Vector3.down, new Color(0, 1, 0));
```

```
        RaycastHit2D RayHit = Physics2D.Raycast(frontVec, Vector3.down, 1,
LayerMask.GetMask("Platform"));

        //장애물 없으면 뒤돌기,
        if (RayHit.collider == null)
        {
            Turn();
        }
    }

    //행동지표를 바꿔줄 함수 하나 생성
    //재귀 함수 : 딜레이 반드시 사용
    void Think()
    {
        //random 클래스 사용
        //Range : 최소 이상 최대 미만의 랜덤 수 생성
        nextMove = Random.Range(-1, 2);

        //스프라이트 애니메이션
        anim.SetInteger("WalkSpeed", nextMove);

        if (nextMove != 0)
            spriteRenderer.flipX = nextMove == 1;
        float NextThinkTime = Random.Range(2f, 5f);
        Invoke("Think", NextThinkTime);
    }

    void Turn()
    {
        nextMove *= -1;
        spriteRenderer.flipX = nextMove == 1;
        //현재 작동 중인 모든 Invoke 함수를 멈춘다.
        CancelInvoke();
        Invoke("Think", 2);
    }
}
```