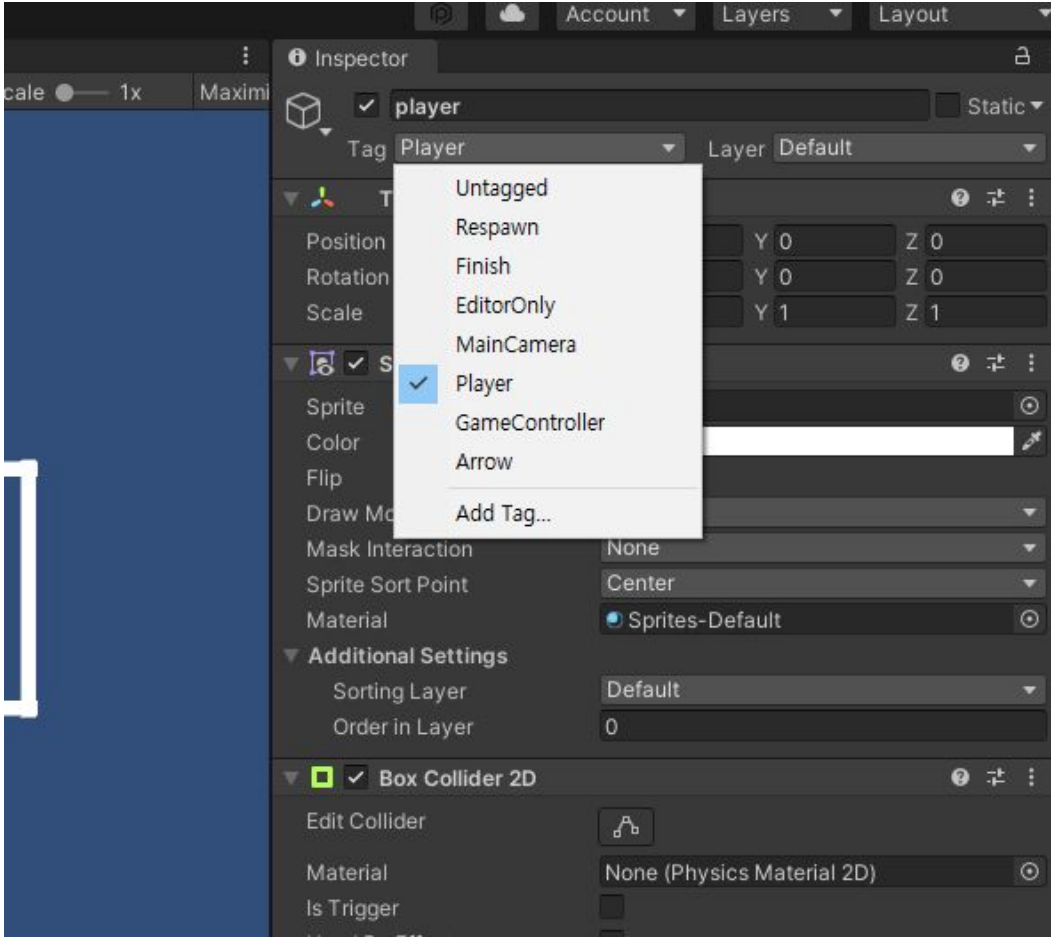
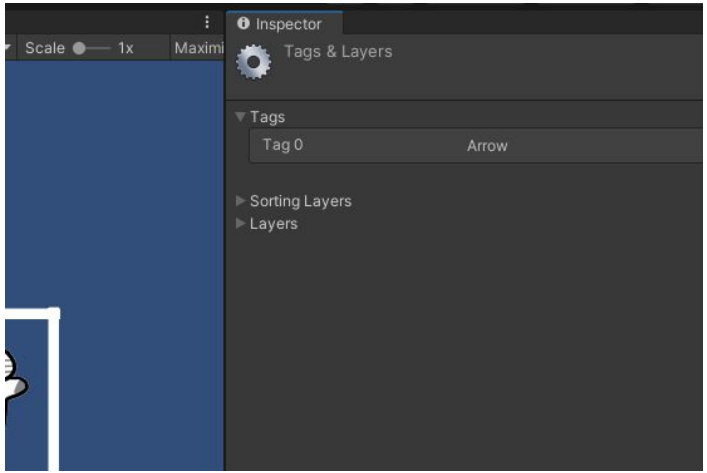
Tag



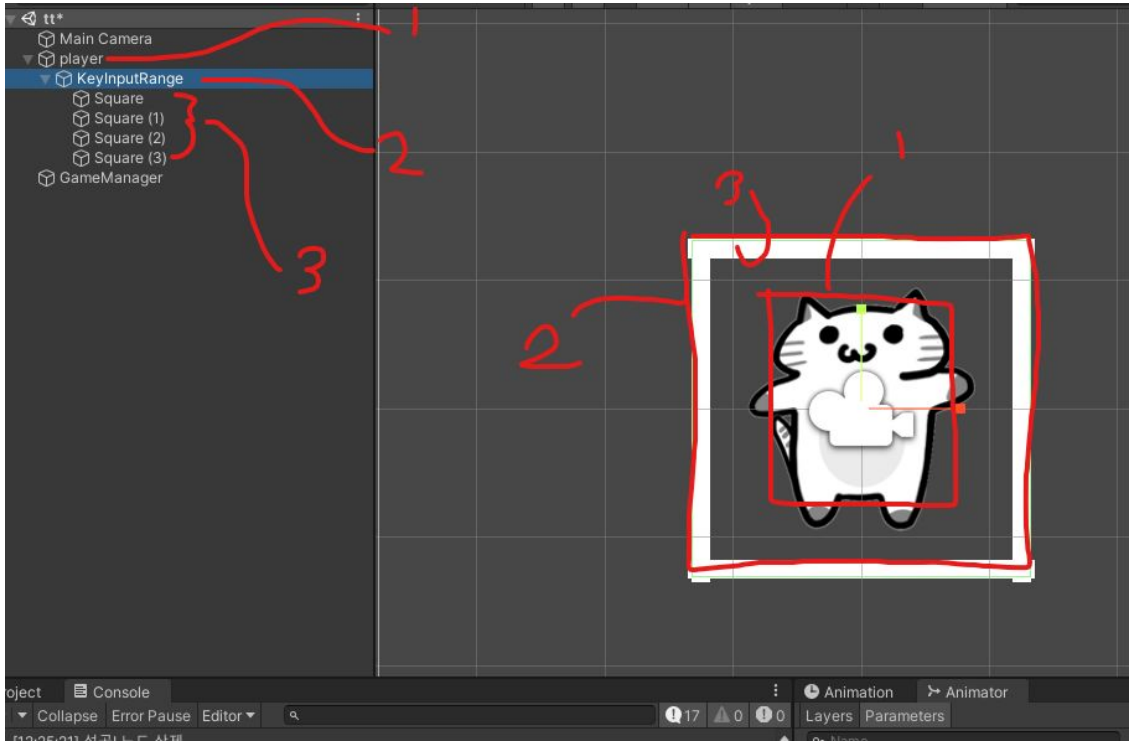
Tag : 오브젝트의 성질을 나타냄(묶음)

Add Tag를 눌러서 Tag를 추가 할 수 있음

Tag를 부여할 뿐 특별한 기능은 없음
충돌판정을 할때 Tag에 맞춰 원하는 값들을 가져올 수있음.
예) 화살표인 녀석들은 모두 Arrow Tag를 붙여서, Arrow라는 녀석들만 연산을 진행한다.



Hierarchy 구성



1 - Player

Arrow에 닿으면 체력이 깍이는 부분

컴포넌트들

1. 이미지
2. 박스콜라이더(2D)
3. 릿지드바디(2D)

해당 박스에 닿으면 체력이 깍임.

2 - KeyInputRange(빈오브젝트로 생성함)

해당 범위 안으로 들어오면 입력됨.

컴포넌트들

1. PlayerCon(스크립트)
2. 박스콜라이더(2D)

박스콜라이더 안으로 들어오면
입력에 따라서 들어온 노드들이 파괴됨.

3 - 프레임(2DObject->Sprite->Square)

아무 능력없음. 유저에게 범위를 보여주기 위한 이미지들.

충돌 판정 스크립트

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

public class PlayerCon : MonoBehaviour
{
    //범위 안으로 들어온 모든 노드들
    public List<GameObject> arrows = new List<GameObject>();

    public void Update()
    {
        //위키를 누르면 위 노드 삭제
        if (Input.GetKeyDown(KeyCode.UpArrow))
        {
            RemoveNode(0);
        }
        //아래키를 누르면 아래 노드 삭제
        else if (Input.GetKeyDown(KeyCode.DownArrow))
        {
            RemoveNode(1);
        }
        //오른쪽키를 누르면 오른쪽 노드 삭제
        else if (Input.GetKeyDown(KeyCode.RightArrow))
        {
            RemoveNode(2);
        }
        //왼쪽키를 누르면 왼쪽 노드 삭제
        else if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            RemoveNode(3);
        }
    }

    public void RemoveNode(int ArrowDir)
    {
        //들어온 모든 노드 검색
        for (int number = 0; number < arrows.Count; number++)
        {
            //노드가 가지고 있는 컴포넌트에서 arrowNumber값과 ArrowDir이 같으면
            if (arrows[number].GetComponent<ArrowMove>().arrowNumber == ArrowDir)
            {
                //파괴 및 삭제
                Destroy(arrows[number]);
                arrows.RemoveAt(number);

                //다음 검색을 위해 한칸 후퇴
                number--;
            }

            Debug.Log("성공! 노드 삭제");
        }
    }

    public void OnTriggerEnter2D(Collider2D collision)
    {
        //Arrow라는 테그인지 확인함
        if(collision.transform.tag == "Arrow")
        {
            //해당 오브젝트가 Arrow인 것을 확인했으면 리스트에 포함
            arrows.Add(collision.gameObject);
        }
    }
}
```

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

public class ArrowMove : MonoBehaviour
{
    float speed = 2f;


    //0 = 위, 1 = 아래, 2 = 오른쪽, 3 = 왼쪽
    public int arrowNumber = 0;

    // Update is called once per frame
    void Update()
    {
        //바라보는 방향을 향해서 나아간다.
        transform.Translate(Vector3.down * speed * Time.deltaTime);
    }

    public void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.transform.tag == "Player")
        {
            PlayerCon pc =
collision.gameObject.GetComponentInChildren<PlayerCon>();
            pc.arrows.RemoveAt(pc.arrows.FindIndex(A => A == gameObject));

            Debug.Log("체력 다운");
            Destroy(gameObject);

        }
    }
}
```

화살표 스크립트 설명

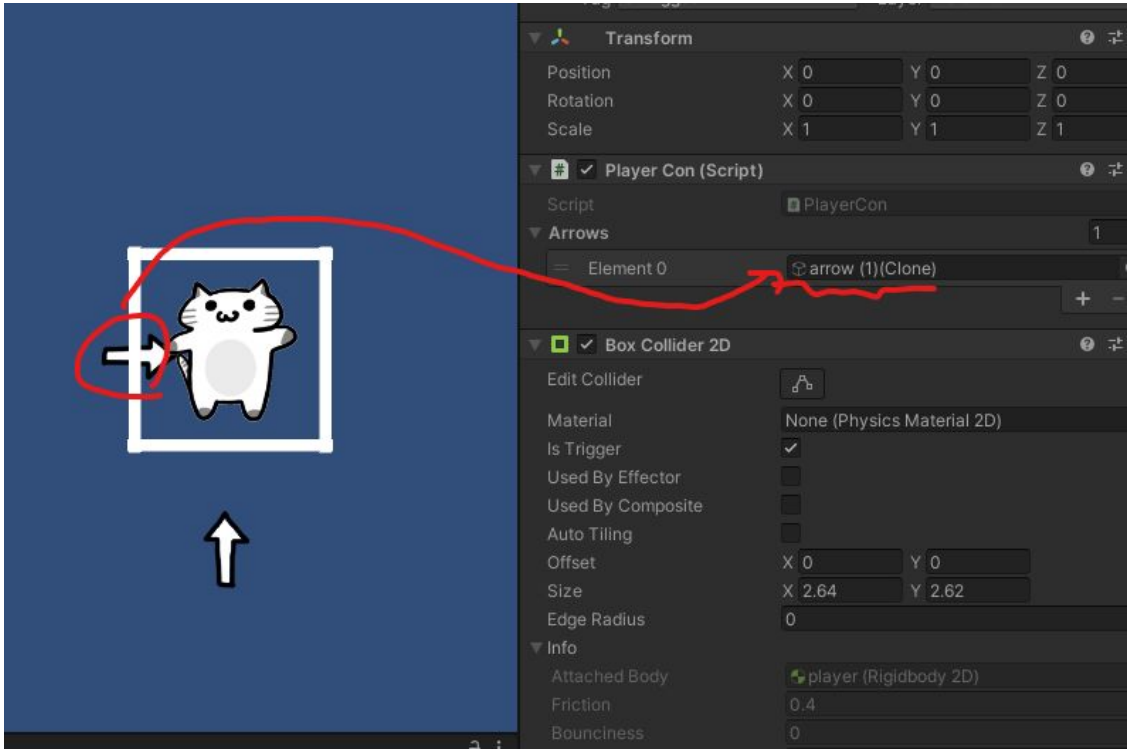
사용된 **GetComponentInChildren** 은
자식오브젝트들 중에서 사용하고 있는 컴포넌트를 검색,
최초로 검색에 나온 녀석을 가져온다.

충돌된 Player의 자식오브젝트로는
-KeyInpuRange
  -Square
  -Square1
  -Square2
…
가 있고, 이중 PlayerCon을 가진 녀석인 KeyInputRange를 가져온다.

사용된 **RemoveAt** 은
리스트에서 지울 “번째”녀석을 받는다. 예를들어, RemoveAt(1)이면 1번째 녀석을 지운다.

사용된 **FindIndex는**
리스트에서 조건에 해당하는 녀석을 숫자로 찾는다.
스크립트 상에서는 람다식으로 A라는 녀석을 찾을 거고, 이 A라는 녀석의 조건으론 해당 gameObject와 같은 녀석 찾는 다는 뜻.
즉, “나를 찾는다” 라는 뜻.
찾았으면 해당 찾은 녀석의 숫자를 반환해준다.(번째를 리턴한다)
즉, 나는 이 List에 몇번째 있는 녀석인지 검색하는 내용.

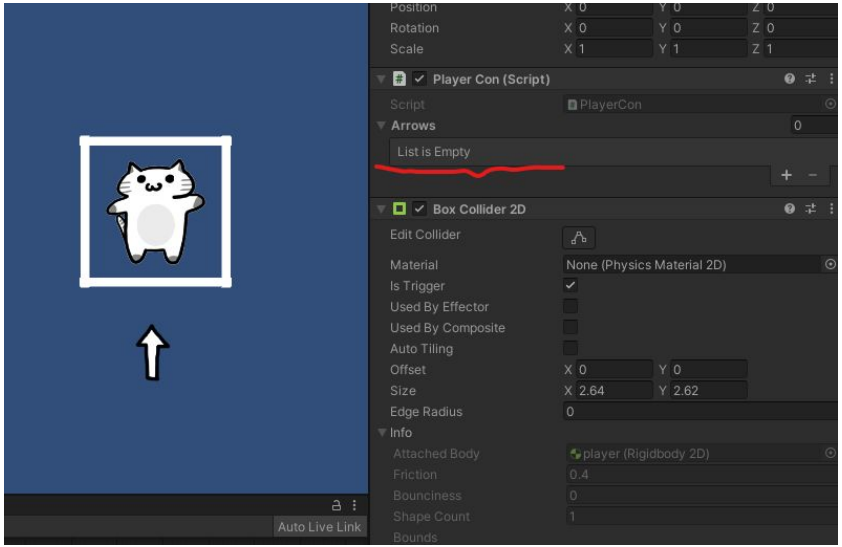
PlayerCon 구성



상자(범위 콜라이더) 안으로 들어오면
List에 포함되는 것을 확인할 수 있음.

PlayerCon 스크립트의
OnTriggerEnter2D 의 내용

PlayerCon 구성

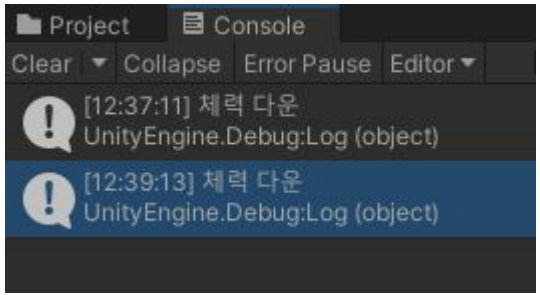


중앙에 Player(BoxCollider2D)에 닿으면 삭제됨과 동시에

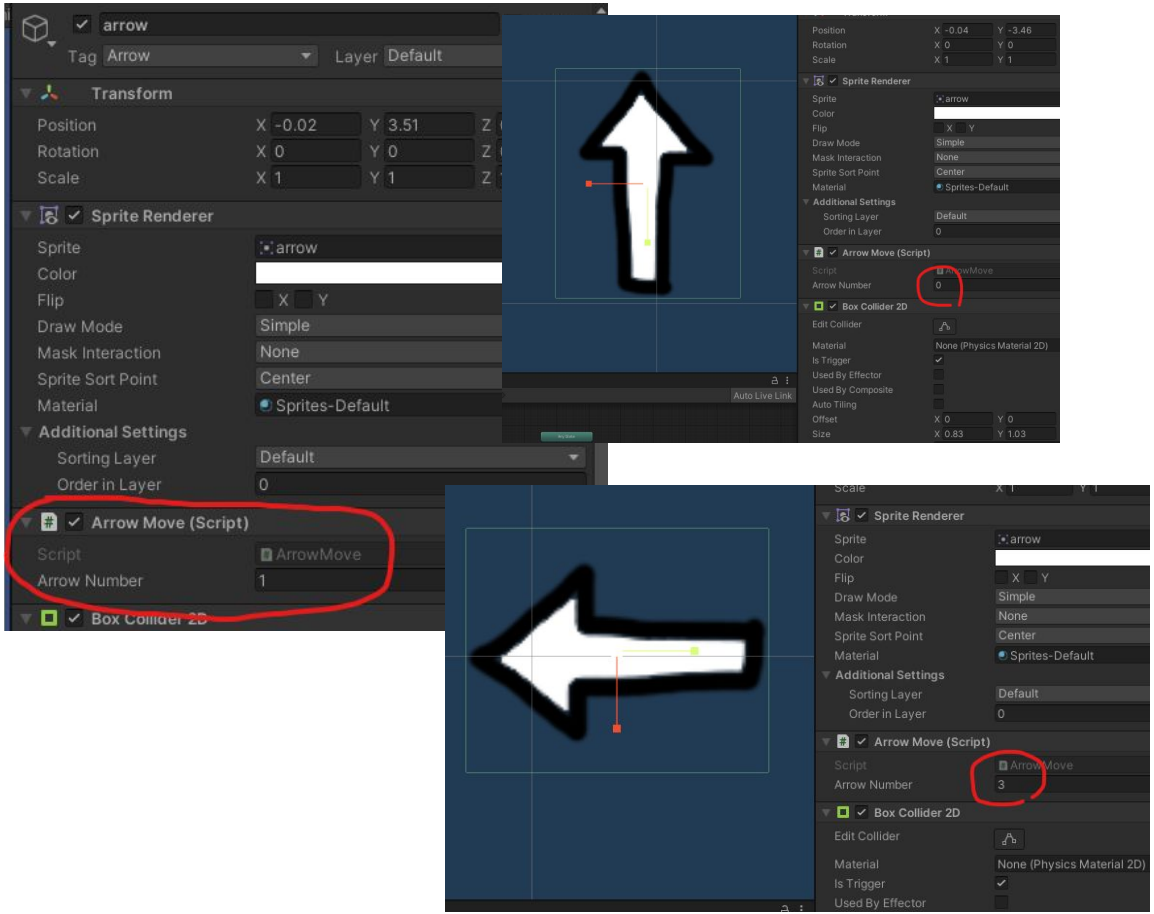
List에서 없어짐.

Debug로 체력다운 띄움

ArrowMove의 OnTriggerEnter2D의 내용



ArrowMove 구성 (화살표)

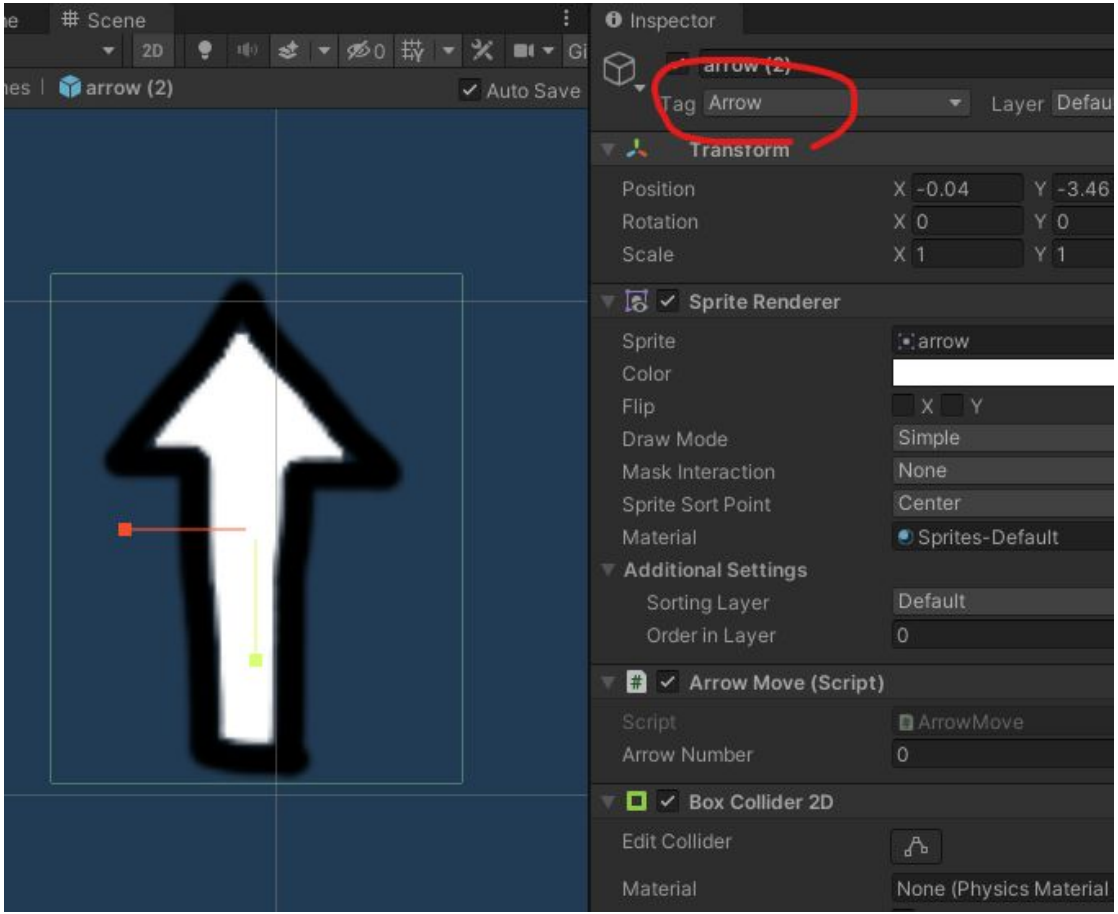


화살표 프리팹에 모두 ArrowMove 스크립트를 가지고 있음.

가지고 있는 상태에서 Inspector 창에서 넘버를 수정한다.

해당 넘버는 플레이어가 입력을 할때 마다 비교할 대상이다(위, 아래, 등등을 알기 위함)

ArrowMove 구성 (화살표)



또한 모든 화살표 들은 Tag를 가지고 있음.
(Arrow)