유니티

1주차(16.03.29)

유니티 실행-씬-오브젝트-콤포넌트
Component(콤포넌트) / ○ Edit메뉴 - Project Setting - Input -
보고있는 씬 장면으로 카메라 바로 이동 : Ctrl + Shift + F

유니티 API.

○ **C#스크립트**
- Start를 Awake로 바꿈 : 시작할 때 단 한번만 사용(Start보단 Awake)
- **Publice**으로 선언한 것은 유니티 오른쪽 메뉴에 뜸 // 공공으로 사용
- **Pribate**로 선언한 것은 유니티 오른쪽에 안 뜸 // 그 스크립트만 사용

실습 비행기/총알 발사

```
//플레이어 조작//
using UnityEngine;
using System.Collections;

public class MNewBehaviourScript : MonoBehaviour {
    private Transform myTrans; // 이름 지정, 객체를 불러오는 것, 사용자정의 자료형.
    private float v;
    private float h;

    public float moveSpeed;
    public float turnSpeed;

        // Use this for initialization
        void Awake () {
        myTrans = GetComponent<Transform>(); //GetComponent : 컴퍼넌트를 들고옴
        }                                    // Transform에 대한 컴퍼넌트만 들고옴

        // Update is called once per frame
        void Update () {    //게임하는 내내 호출이 됨.
        v = Input.GetAxis("Vertical"); //유니티의 Edite-Project Setting - Input의 내용을 불러옴
        h = Input.GetAxis("Horizontal"); //("내용")은 유니티의 Input이름이므로 오타 나도 빨간줄 안띄움

        Vector3 moveDir = new Vector3(0f, v, 0f); //Vector3는 3차원 벡터(운동 값들) 불러옴=> 새롭게 값을 줌
        myTrans.Translate(moveDir * moveSpeed * Time.deltaTime);
          //여기서부터는 유니티 에이피아이, dltaTime은 누르고있는 시간동안 값을 변경
        myTrans.Rotate(0f, 0f, -h * turnSpeed * Time.deltaTime);
        }}
//빈 오브젝트(CreatEmty) CameraRig를 생성 후 메인카메라를 넣고 메인카메라 위치 설정 후
Ctrl+Shift+F로 카메라 설정, 그리고 CameraRig 톱니바퀴에서 위치 리셋을 시켜줌
```

```
//카메라 조작(플레이어 따라다님)//
using UnityEngine;
using System.Collections;

public class CameraMove : MonoBehaviour {
    Transform myTrans;
    public Transform player;

    public float smoothTime = 1f; //얼마나 부드러운지
    private Vector3 currentVelocity = Vector3.zero; //벡터3값을 모두 0으로 줌

    public float cameraTurn = 2f;
        // Use this for initialization
        void Awake () {
        myTrans = GetComponent<Transform>();
        player = GameObject.FindGameObjectWithTag("Player").transform;} //플레이어에 태그해줌

        // Update is called once per frame
        void LateUpdate () {
        myTrans.position = Vector3.SmoothDamp(myTrans.position, player.position, ref currentVelocity,
smoothTime); //ref을 사용하면 참조 값을 가진다.
        myTrans.rotation = Quaternion.Slerp(myTrans.rotation, player.rotation, cameraTurn *
Time.deltaTime);
        }}
```

2주차(16.04.03)

```
//총알 무브//
using UnityEngine;
using System.Collections;

public class BulletMove : MonoBehaviour {

    public float moveSpeed;
    Transform myTrans;

        // Use this for initialization
        void Awake () {
        myTrans = GetComponent<Transform>();
        Destroy(gameObject, 3f); //3초 뒤 오브젝트가 사라짐
        }

        // Update is called once per frame
        void Update () {
        myTrans.Translate(0f, moveSpeed * Time.deltaTime, 0f);
        }
}
//총알을 계속 씬안에 둘 수 없으니 사용하는 것이 프리펩(Prefabs)
Bullet 안에 BulletMove 스크립트를 넣음 -> Prefabs폴더안에 총알 오브젝트를 넣고
Spawn을 생성함
```

```
Move 스크립트에 총알(Bullete) 추가
public class Move : MonoBehaviour {
    private Transform myTrans;
    private float h;
    private float v;

    public float t; //총알
    public float moveSpeed;
    public float turnSpeed;
    public GameObject bullet; //게임오브젝트 총알
    public Transform bulletSpawn; //총알생성


        void Awake () {
        myTrans = GetComponent<Transform>();
        }

        // Update is called once per frame
        void Update () {
        v = Input.GetAxis("Vertical");      // 수직
        h = Input.GetAxis("Horizontal");   // 수평

        Vector3 moveDir = new Vector3(0f, v, 0f);
        myTrans.Translate(moveDir * moveSpeed * Time.deltaTime);
        myTrans.Rotate(0f, 0f, -h * turnSpeed * Time.deltaTime);

        if (Input.GetKeyDown(KeyCode.Space)) //스페이스바 누르는 순간 총알 발사
            Instantiate(bullet, bulletSpawn.position, bulletSpawn.rotation);
             //이건 유니티 자체 동적할당

        }
}
//Airplane 안에 Spawn을 넣고 오른쪽 무브 스크립트에 Bullete에 Bullete프리펩을 설정하고 Bullet Spawn에 Spawn(Transform)을 설정한다.
```

* 오른쪽옵션 Collider박스 : HitBox와 비슷한 개념
  - Collider 박스 안의 Is Trigger를 체크하면 지나가는 것은 인식을 하지만 물체자체는 통과한다.
  - ex) 어느 공간을 지나가면 이벤트 발생할 때 빈 오브젝트로 이즈 트리거를 하여 사용

3주차(16.04.12)

강체

- 중력 받고 다른 오브젝트랑 충돌하면 튕기는 것
- Material에서 Physic Material 생성 하여 Bounce 값을 설정하고 공에 넣는다.

리지드바디(Rigidbody)

- 오른쪽 탭 - Add Component - Rigidbody(실체화)
- 오브젝트에는 리지드 바디를 씀 (실체화)
- 캐릭터에게는 캐릭터 컨트롤러 (리지드 바디 대용)
- 오른쪽 탭 - Constraints에 Freeze를 설정하면 움직이지 않는다.

좌표

* 지역좌표
 - 캐릭터만의 좌표
* 전역좌표
 - 아무리 이동해도 좌표가 변하지 않음

○ **조인트(관절, 축)** : 스켈레톤 형식 ex) 관절, 문

* Component - Physics - Hinge Joint // 리지드바디 꼭 설정
* HingeJoint에서 Axis로 어느쪽으로 회전할 것인지 결정, Anchor로 회전할 위치 변경

* 마우스 클릭시 회전

```
using UnityEngine;
using System.Collections;

public class NewBehaviourScript : MonoBehaviour {

    public float forceAmount = 1000f;

        // Use this for initialization
        void OnMouseDown() {  //마우스 클릭했을 때
        GetComponent<Rigidbody>().AddForce(transform.forward*forceAmount, ForceMode.Acceleration);
        }
}
```

* Fixed Joint : 체인형식으로 조인트 // 오른쪽 탭 Connected Body에 자기 위의 오브젝트를 넣음
* Spring Joint : 스프링처럼 조인트 // 오른쪽 탭 Connected Body에 자기 위의 오브젝트를 넣음
* Charictor Joint : 캐릭터의 죽은 모습 표현할 때 적절

○ **캐릭터 관절 설정 (RagDoll)**

* **캐릭터 관절 설정** : 오브젝트 우클릭 - 3D오브젝트 - RagDoll - 각각의 오브젝트에 넣어둠

씬 이동

* Scene Moce 스크립트 생성
  - File - BuildSetting - 버튼 하나 생성

```
using UnityEngine;
using System.Collections;

public class SceneMove : MonoBehaviour {

        void OnMouseDown () {
            if(Input.GetMouseButtonDown(0))
        {
            Application.LoadLevel("Ragdoll"); //"Ragdoll"은 이동할 파일명 입력
        }
        }
}
```

카메라 이동(FPS 컨트롤러)

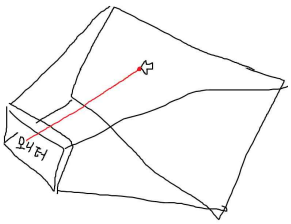
* FPS컨트롤러 : 밑 탭 - 제일 윗 폴더 - 우클릭 - import Package - Characters 클릭 - import클릭 - 밑 탭 - Standard Assets - Characters - FirstPersonCharacter

보이지 않는 벽

* 빈 오브젝트 생성 - 오른쪽 탭 - Box Collider 넣어줌

6주차(16.05.03)

충돌체크(Raycast)



○ 충돌체크 : 마우스로 거리재기, 적 지정 등(레이저 쏘듯이)

```
//Player 안에 넣어주는 Raycast 스크립트//
using UnityEngine;
using System.Collections;

public class Raycast : MonoBehaviour {

        public float moveSpeed;
        public float turnSpeed;

        private CharacterConctroller _cc;

        void Awake ()
        {
            _cc = GetComponent<CharaterContriller>();
        }

        void Updata ()
        {
            if(Input.GetMouseButton(0))
            {
                Vector3 mousePos = Input.mousePosition;  // 마우스 눌렀을 때의 좌표

                Camera mainCamera = Camera.main;
                // 메인카메라 불러오는 이유가 화면이랑 마우스누른애랑의 거리를 재어주기 때문

                Ray ray = mainCamera.ScreenPointToRay(mousePos); // ray는 메인카메라의 렌즈위치
                PaycastHit hitInfo;

                if(Physics.Raycast(ray, out hitInfo, 100f))  //out은 참조하는 것, 100f은 레이저의 길이
                {                       // ref와 out의 차이는 ref는 초기화해야하지만 out은 안해도 됨
                    Debug.Log(hitInfo.point);  // 마우스 눌러서 맞은 곳

                    Vector3 moveDir = hitInfo.point - transform.position; // transform은 캐릭터가 있는 위치
                    Vector3 dirXZ = new Vector3(moveDir.x, 0f, moveDir.z); // x,y,z축의 위치 설정
                    _cc.Move(dirXZ.normalized * moveSpeed * Time.dltaTime);
                }
            }
        }
}
```

○ 캐릭터 이동 지점에 마우스 포인터 표시

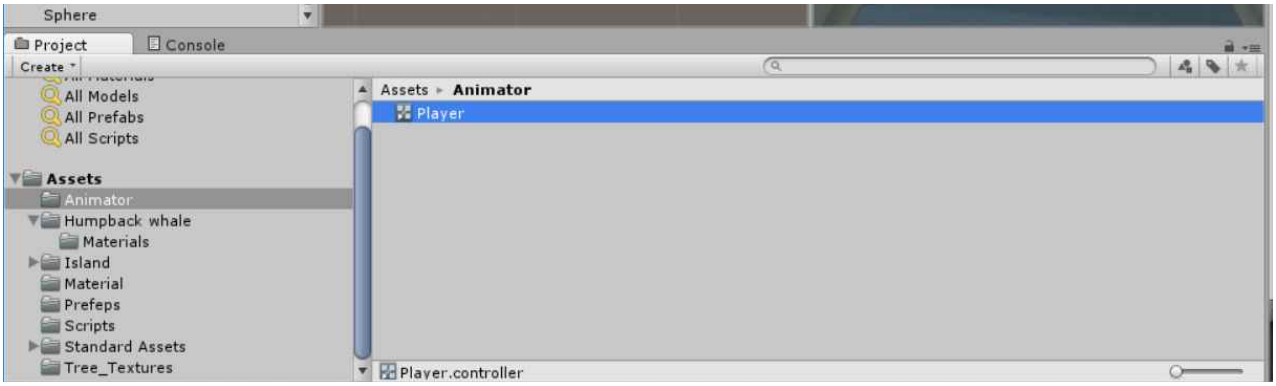
```
//이동할 곳에 마우스 포인터를 표시해주는 것// +참고 레이케스트
using UnityEngine;
using System.Collections;

public class RayTest : MonoBehaviour {

    public bool isMoveState = false;
    public float moveSpeed;
    public float turnSpeed;

    private CharacterController _cc;
    private GameObject movePoint;

        void Awake ()
    {
        _cc = GetComponent<CharacterController>();
        movePoint = GameObject.FindGameObjectWithTag("MovePoint");
        movePoint.SetActive(false); // 생성하자마자 안보이게하기 위함
        }

        // Update is called once per frame
        void Update ()
    {
        if(Input.GetMouseButton(0))
        {
            Vector3 mousePos = Input.mousePosition;
            Camera mainCamera = Camera.main;
            Ray ray = mainCamera.ScreenPointToRay(mousePos);
            RaycastHit hitInfo;

            if(Physics.Raycast(ray, out hitInfo, 100f))
            { //레이저시작지점, 레이저 맞은 지점 , 레이저 길이
                movePoint.transform.position = hintInfo.point;
                movePoint.transform.position += new Vector3(0f, 0.01f, 0f);
                // 0.01인 이유는 맵이랑 겹쳐진 포인트를 선명하게 보이기 하기위해 살짝 띄워준것
                movePoint.SetActive(true); // 찍자마자 켜야하기때문에 true
                isMoveState = true; // 무브 포인트를 보이게 하기 위함.
            }
        }
       if (isMoveState) // 표시가 나고 이동하기 위한 이동을 밑에 다 작성.
        {
            Vector3 moveDir = movePoint.transform.position - _cc.transform.position;
            Vector3 dirXZ = new Vector3(moveDir.x, 0f, moveDir.z);

            if (dirXZ != Vector3.zero)
            { //dirXZ가 Vector 제로가 아닌 경우 찍는 방향으로 캐릭터가 바라보게 돌려줌
                Quaternion tageRot = Quaternion.LookRotation(dirXZ); //Quaternion은 4원소의 기능, 회전 값
                 _cc.transform.rotation =
                    Quaternion.RotateTowards(_cc.transform.rotation, tagetRot, turnSpeed * Time.deltaTime);
            } //RotateTowards은 프레임, 회전을 어디까지 해야할지 값을 넘겨주는 것

            Vector3 framePos =
              Vector3.MoveTowards(_cc.transform.position, _cc.transform.position + dirXZ, moveSpeed * Time.deltaTime);

            Vertor3 frameMoveDir = framePos - _cc.transform.position;

            _cc.Move(frameMoveDir + Physics.gravity * Time.deltaTime);
            if (_cc.transform.position == _cc.transform.position + dirXZ) // 캐릭터가 도착했을 때 기능을 끔
            {
                movePoint.SetActive(false);
                isMoveState = false;
            }
        }
        else
        {
            //쉬는 로직.
        }
    }
}
```

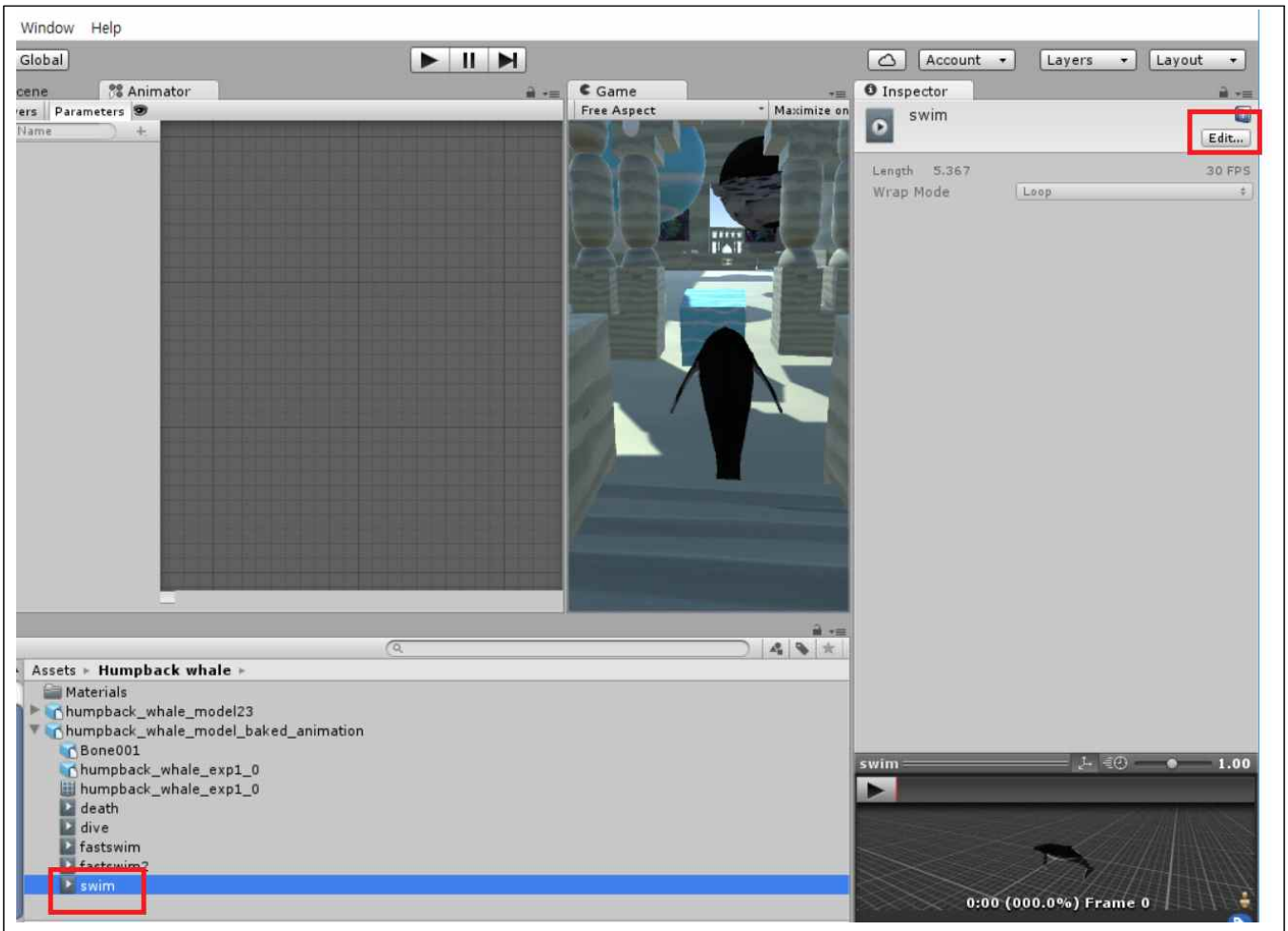
○ 메카닉 시스템

- 캐릭터 애니메이션 (스크립트 : FSMBase / CharactorState / PlayerFSM / MoveUtil)

* Coroutine(코루틴) : 어떤 루틴이 실행하고 다시 돌아간 후 다시 부르면 이어받기 // 루틴 : 함수
* IEnumerator : 반복자 // FSMMain이 불릴때마다 Idle이 불리고 그 다음 것들이 바뀌어가면서 불림

* 메카닉 시스템 : Create - Animator Controller





Edit - Rig 들어가서 업데이트 후 Apply, 동작 설정.
Entry가 시작이고, Make Trans 로 동작들을 연결
// 오른쪽 탭에 Has Exit Time를 체크해제(애니메이션 도중 동작 변경 가능)
// 오른쪽 탭 제일 밑, State를 Equals로 설정 후 스크립트에 설정한 동작 번호(하고자하는 동작번호) 입력
Player 오브젝트에 FSMBase 넣고,
KoKo(Player밑 오브젝트) 오른쪽 탭 Controller에 Player을 넣음

```
//MoveUtil//
using UnityEngine;
using System.Collections;

/// <summary>
/// 지정된 캐릭터를 회전 하면서 이동하게 하는 스크립트
/// </summary>
public class MoveUtil
{
    /// <summary>
    /// 캐릭터를 프레임 별로 이동 시키고 목적지까지의 거리를 넘겨줌
    /// </summary>
    /// <param name="cc"></param>
    /// <param name="target"></param>
    /// <param name="moveSpeed"></param>
    /// <param name="turnSpeed"></param>
    /// <returns></returns>
    public static float MoveFrame(CharacterController cc,
        Transform target, float moveSpeed, float turnSpeed)
    {
        Transform t = cc.transform;
        Vector3 dir = target.position - t.position;
        Vector3 dirXZ = new Vector3(dir.x, 0f, dir.z);
        Vector3 targetPos = t.position + dirXZ;
        Vector3 framePos = Vector3.MoveTowards(
            t.position,
            targetPos,
            moveSpeed * Time.deltaTime);

        //이동
        cc.Move(framePos - t.position + Physics.gravity);

        //회전도 같이 함
        RotateToDir(t, target, turnSpeed);

        return Vector3.Distance(framePos, targetPos);
    }

    /// <summary>
    /// 캐릭터를 프레임 벼롤 회전 시키는 함수
    /// </summary>
    /// <param name="self"></param>
    /// <param name="target"></param>
    /// <param name="turnSpeed"></param>
    public static void RotateToDir(Transform self, Transform target, float turnSpeed)
    {
        Vector3 dir = target.position - self.position;
        Vector3 dirXZ = new Vector3(dir.x, 0f, dir.z);
        //회전 방향이 더이상 할 필요가 없을 경우
        if (dirXZ == Vector3.zero)
            return;

        //회전.
        self.rotation = Quaternion.RotateTowards(
            self.rotation,
            Quaternion.LookRotation(dirXZ),
            turnSpeed * Time.deltaTime);
    }

    /// <summary>
    /// 긴급히 회전 함
    /// 몬스터가 맞을때 회전 함
    /// </summary>
    /// <param name="self"></param>
    /// <param name="target"></param>
    public static void RotateToDirBurst(Transform self, Transform target)
    {
        Vector3 dir = target.position - self.position;
        Vector3 dirXZ = new Vector3(dir.x, 0f, dir.z);
        //회전 방향이 더이상 할 필요가 없을 경우
        if (dirXZ == Vector3.zero)
            return;

        //회전.
        self.rotation = Quaternion.LookRotation(dirXZ);
    }
}
```

```
//CharacterState//
using UnityEngine;
using System.Collections;

public enum CharacterState
{
        Idle = 0,
    Run = 1,
    AttackRun = 2,
    Attack = 3,
    Dead = 4,
    Patrol = 5,
    Skill1 = 6,

    Max
}
```

FSMBase를 만든 이유가 플레이어하고 몬스터가 같이 공유하는 상태,

```
//FSMBase//
using UnityEngine;
using System.Collections;

public class FSMBase : MonoBehaviour {
    public CharacterState state;

    public Animator _a;
    public CharacterController player;

    public bool isNewstate;

    //Awake가 꺼진상태로도 실행
    protected virtual void Awake()
    {
        Player = GetComponent<CharacterController>();
        _a = GetComponentInChildren<Animator>();
    }

    protected virtual void OnEnable()
    {
        StartCoroutine();
    }

    public void SetState(CharacterState _newState)
    {
        isNewstate = true;
        state = _newState;
        _a.SetInterger("state", (int)state);
    }

    private IEnumerator FSMMain()
    {
        while (true)
        {
            isNewstate = false;
            yield return StartCoroutine(state.ToString());
        }
    }

    protected virtual IEnumerator Idle()
    {
        do
        {
            yield return null;
        }
        while (!isNewstate);
    }
    public bool IsDead()
    {
        return (state == CharacterState.Dead);
    }
}
```

```
무브, 턴, 돈, 점수 등 파라미터 값들을 PlayerFSM에 넣어줌
using UnityEngine;
using System.Collections;

public class FSMBase : MonoBehaviour {
    public CharacterState state;

    public Animator _a;
    public CharacterController player;

    public bool isNewstate;

    //Awake가 꺼진상태로도 실행
    protected virtual void Awake()
    {
        Player = GetComponent<CharacterController>();
        _a = GetComponentInChildren<Animator>();
    }

    protected virtual void OnEnable()
    {
        StartCoroutine();
    }

    public void SetState(CharacterState _newState)
    {
        isNewstate = true;
        state = _newState;
        _a.SetInterger("state", (int)state);
    }

    private IEnumerator FSMMain()
    {
        while (true)
        {
            isNewstate = false;
            yield return StartCoroutine(state.ToString());
        }
    }

    protected virtual IEnumerator Idle()
    {
        do
        {
            yield return null;
        }
        while (!isNewstate);
    }
    public bool IsDead()
    {
        return (state == CharacterState.Dead);
    }
}
```