카메라가 플레이어를 따라가는 스크립트

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

public class CameraMove : MonoBehaviour
{
    public Transform target;//따라갈 목표(객체)
    public float speed;//따라가는 속도

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void LateUpdate()
    {
        transform.position = Vector3.Lerp(transform.position, target.position, Time.deltaTime * speed);
        transform.position = new Vector3(transform.position.x, transform.position.y, -10f);
    }
}
```

화면 비율 및 사운드 스크립트

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

public class WindowSound : MonoBehaviour
{
    public AudioSource musicSource;
    int setWidth;
    int setHeight;

    // Start is called before the first frame update
    void Start()
    {
        FullScreen();
    }

    // Update is called once per frame
    void Awake()
    {

    }

    public void SetMusicVolume(float volume)
    {
        musicSource.volume = volume;
    }
    public void FullScreen()
    {
        int setWidth = 1280;
        int setHeight = 720;

        Screen.SetResolution(setWidth, setHeight, true);
    }

    public void WindowScreen()
    {
        int setWidth = 800;
        int setHeight = 600;

        Screen.SetResolution(setWidth, setHeight, false);
    }
}
```

화면 비율 및 사운드 스크립트

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

```
public class PauseScript : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenu;
    public GameObject pauseOption;
    public GameObject GoToMain;
    public GameObject ExitTheGame;


    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }

    void Resume()
    {
        pauseMenu.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
    }

    void Pause()
    {
        pauseMenu.SetActive(true);
        Time.timeScale = 0f;
        GameIsPaused = true;
    }

    public void ToSettingMenu()
    {
        pauseMenu.SetActive(false);
        pauseOption.SetActive(true);
    }
    public void ToMain()
    {

        pauseMenu.SetActive(false);
        GoToMain.SetActive(true);

    }

    public void YesMain()
    {
        SceneManager.LoadScene("mainScreen");
        Time.timeScale = 1f;
        GameIsPaused = false;
    }

    public void NoMain()
    {
        pauseMenu.SetActive(true);
        GoToMain.SetActive(false);
    }


    public void QuitGame()
    {
        pauseMenu.SetActive(false);
        ExitTheGame.SetActive(true);

    }

    public void YesQuit()
    {
```

```
        Application.Quit();
    }

    public void NoQuit()
    {
        pauseMenu.SetActive(true);
        ExitTheGame.SetActive(false);
    }

    public void BackSettingMenu()
    {
        pauseMenu.SetActive(true);
        pauseOption.SetActive(false);
    }


}
```

다음 스테이지로 변경하기위한 스크립트

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; //Scene 매니저 라이브러리 추가

public class NextStage : MonoBehaviour
{

    public string transferMapName; // 이동할 맵이름

    // Start is called before the first frame update
    void Start()
    {

    }

    // 박스 콜라이더에 닿는 순간 이벤트 발생
    private void OnTriggerEnter2D(Collider2D collision)
    {

        if (collision.gameObject.name == "Character")
        {
            SceneManager.LoadScene(transferMapName);
        }
    }
}
```

플레이어(조작캐릭터)에 대한 전반적인 스크립트

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

public class Move : MonoBehaviour
```

```
{
    public float moveSpeed = 5f;     //이동 속도

    public float jumpSpeed = 5f;     //점프 속도

    public bool isGrounded = false;

    public int jumpCount = 2; //점프횟수     2를 3으로 바꾸면 3단 점프

    public int atk_dmg;

    Rigidbody2D rb;

    public Animator animator;

    public float curTimeFir;

    public float curTiemSec;

    public float atk_CoolTime = 0.5f;

    public Transform pos;

    public Vector2 boxSize;

    [SerializeField]
    private Slider hpbar;

    public float maxHp;
    public float curHp;
    //float imsi;

    bool isPlayerDead = false;
    public GameObject GameOverCanvas;

    void Start()

    {
        animator = GetComponent<Animator>();

        rb = GetComponent<Rigidbody2D>(); //컴포넌트를 불러옴

        jumpCount = 0;

        StartCoroutine(CheckPlayerDeath());
        //imsi = (float)curHp / (float)maxHp;
    }
```

```
IEnumerator CheckPlayerDeath()
{
    while (true)
    {
        if (curHp <= 10 && isPlayerDead == false)
        {
            isPlayerDead = true;
            yield return new WaitForSeconds(2);
            animator.SetTrigger("die");
            GameOverCanvas.SetActive(true);
        }
        yield return new WaitForEndOfFrame();
    }
}

private void OnCollisionEnter2D(Collision2D col)

{

    if (col.gameObject.tag == "Ground")

    {

        isGrounded = true;     //Ground에 닿으면 isGround는 true

        jumpCount = 2;            //Ground에 닿으면 점프횟수가 2로 초기화됨

        animator.SetTrigger("isGround");
    }

}

private void OnDrawGizmos()
{
    Gizmos.color = Color.blue;
    Gizmos.DrawWireCube(pos.position, boxSize);
}


void Update()

{
    if (isPlayerDead)
    {
        return;
    }
```

```
if (curTimeFir <= 0)
{
    if (Input.GetKeyDown(KeyCode.Z))
    {
        //공격
        Collider2D[] collider2Ds = Physics2D.OverlapBoxAll(pos.position, boxSize, 0);

        foreach (Collider2D collider in collider2Ds)
        {
            if (collider.tag == "Enemy")
            {
                collider.GetComponent<Slime_Move>().TakeDamage(atk_dmg);
            }
        }
        animator.SetTrigger("atk");
        curTimeFir = atk_CoolTime;
    }


}
else if (curTiemSec <= 0)
{


    if (Input.GetKeyDown(KeyCode.Z))
    {
        //공격
        Collider2D[] collider2Ds = Physics2D.OverlapBoxAll(pos.position, boxSize, 0);

        foreach (Collider2D collider in collider2Ds)
        {
            collider.GetComponent<Slime_Move>().TakeDamage(atk_dmg);
        }
        animator.SetTrigger("atk2");
        curTiemSec = atk_CoolTime;
    }

}


curTiemSec -= Time.deltaTime;
curTimeFir -= Time.deltaTime;



if (isGrounded)//이동 스크립트
```

```
        {

            if (jumpCount > 0)

            {

                if (Input.GetKeyDown(KeyCode.Space))    //입력키가 위화살표면 실행함

                {


                    animator.SetTrigger("Jump");

                    rb.AddForce(new Vector3(0, 1, 0) * jumpSpeed, ForceMode2D.Impulse); //위방향으로 올
라가게함

                    jumpCount--;     //점프할때 마다 점프횟수 감소

                }
            }

        }

        if (Input.GetKey(KeyCode.LeftArrow))     //왼쪽화살표 입력시 실행함

        {


            Vector3 scale = transform.localScale;

            scale.x = -Mathf.Abs(scale.x);//좌우반전

            transform.localScale = scale;

            transform.Translate(Vector3.left * moveSpeed * Time.deltaTime);

            animator.SetBool("Run", true);
            animator.SetBool("runNidle", true);
        }

        else if (Input.GetKey(KeyCode.RightArrow))     //오른쪽화살표 입력시 실행함

        {
```

```
            Vector3 scale = transform.localScale;

            scale.x = Mathf.Abs(scale.x);

            transform.localScale = scale;

            transform.Translate(Vector3.right * moveSpeed * Time.deltaTime);


            animator.SetBool("Run", true);
            animator.SetBool("runNidle", true);
        }
        else
        {
            animator.SetBool("Run", false);
            animator.SetBool("runNidle", false);
        }
    }


    public void TakeDamage(float damage)
    {
        if (curHp > 0)
        {
            curHp = curHp - damage;
        }

        else
        {
            curHp = 0;
        }

        hpbar.value = hpbar.value - damage;
       // HandleHp();
    }

    private void HandleHp()
    {
        //hpbar.value = Mathf.Lerp(hpbar.value, imsi, Time.deltaTime * 10);
    }

}
```