상당히 많은 스크립트가 사용되었는데, 그 중 핵심이 되는 코드만 기술했습니다.

1. 카메라 무빙

using System.Collections;

using System.Collections.Generic;

using System.Collections.Specialized;

using System.Security.Cryptography;

using UnityEngine;

public class MainCamera : MonoBehaviour

{

 public Camera[] subCameras;

 public void MoveCamera(int index)

 {

 Vector3 new\_position = subCameras[index].transform.position;

 Vector3 new\_rotation = subCameras[index].transform.eulerAngles;

 iTween.MoveTo(this.gameObject, iTween.Hash("position", new\_position, "easetype", iTween.EaseType.easeOutBack, "time", 8.0f));

 iTween.RotateTo(this.gameObject, iTween.Hash("rotation", new\_rotation, "easetype", iTween.EaseType.easeOutBack, "time", 8.0f));

 }

 private void Start()

 {

 MoveCamera(0);

 }

}

게임상에서 처음 카메라 위치와 바꿀 카메라 위치를 각각 설정해 놓은 후, new\_position 변수에 좌표값을 저장한다. 그 후, 유니티 에셋 스토어에서 다운받은 iTween함수를 활용하여 카메라 이동 시간, 모션 등을 설정한다. 여기에서는 easetype라는 모션과 8.0f라는 이동시간을 사용하였다. 해당 스크립트는 게임 시작화면, 튜토리얼에서 본게임 화면으로 넘어갈 때 사용하였다.

2. 페이드 아웃

using System;

using System.Collections;

using System.Collections.Generic;

using System.Reflection;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class FadeOUT : MonoBehaviour

{

 public Image Panel;

 public Image LOGO;

 public Slider progressbar;

 float time = 0f;

 float F\_time = 1f;

 public void Fade()

 {

 StartCoroutine(FadeFlow());

 }

IEnumerator FadeFlow()

 {

 if (progressbar.value == 1f)

 {

 yield return new WaitForSeconds(3f);

 Panel.gameObject.SetActive(true);

 time = 0f;

 Color alpha = Panel.color;

 while (alpha.a < 1f)

 {

 time += Time.deltaTime / F\_time;

 alpha.a = Mathf.Lerp(0, 1, time);

 Panel.color = alpha;

 yield return null;

 Destroy(LOGO);

 }

 SceneManager.LoadScene(1);

 }

 }

}

게임상에서 canvas를 추가해 페이드 아웃시 변경될 색을 미리 지정해놓고 투명도를 0으로 설정해 눈에 보이지 않게 설정한다. 이벤트 실행 시, 투명도가 조금씩 증가하며 지정된 색으로 화면이 전부 가려진다. 여기에서는 로딩이 100% 완료되면 3초 기다렸다가 검정 화면으로 점차 변하며, 모두 실행되면 다음 씬으로 넘어가도록 설정해놓았다. 해당 스크립트는 시작화면에서 튜토리얼 화면으로 넘어갈 때 사용하였다.

3. 페이드 인

using System;

using System.Collections;

using System.Collections.Generic;

using System.Reflection;

using UnityEngine;

using UnityEngine.UI;

public class FadeIN : MonoBehaviour

{

 public GameObject leapmotion;

 public GameObject interaction\_manager;

 public GameObject warning;

 public GameObject sun;

 public GameObject moon;

 public GameObject cloud;

 public GameObject gauge;

 public GameObject gaugeBar;

 public GameObject bud;

 public GameObject flower;

 public GameObject leaf1;

 public GameObject leaf2;

 public GameObject roseSoil;

 public Image Panel;

 float time = 0f;

 float F\_time = 1f;

void Start()

 {

 Fade();

 leapmotion.transform.position=new Vector3(-0.68f, 2.569f, -5.7799f);

 interaction\_manager.transform.position = new Vector3(-0.68f, 2.74f, -6.154f);

 warning.SetActive(false);

 sun.SetActive(false);

 moon.SetActive(false);

 cloud.SetActive(false);

 gauge.SetActive(false);

 gaugeBar.SetActive(false);

 bud.SetActive(false);

 flower.SetActive(false);

 leaf1.SetActive(false);

 leaf2.SetActive(false);

 roseSoil.SetActive(false);

 }

 public void Fade()

 {

 StartCoroutine(FadeFlow());

 }

 IEnumerator FadeFlow()

 {

 yield return new WaitForSeconds(1f);

 Panel.gameObject.SetActive(true);

 time = 0f;

 Color alpha = Panel.color;

 while (alpha.a > 0f)

 {

 time += Time.deltaTime / F\_time;

 alpha.a = Mathf.Lerp(1, 0, time);

 Panel.color = alpha;

 yield return null;

 }

 }

}

게임상에서 canvas를 추가해 페이드 아웃될 때와 동일한 색을 미리 지정해놓고 투명도를 최대로 설정해 화면 내 존재하는 다른 오브젝트들이 눈에 보이지 않게 설정한다. 이벤트 실행 시, 투명도가 조금씩 감소하며 색이 옅어지면서 화면이 나타난다. 여기에서는 씬이 전환된 후 1초 기다렸다가 실행되도록 설정해놓았다. 해당 스크립트는 시작화면에서 튜토리얼 화면으로 넘어올 때 자연스러운 씬 전환효과를 내기 위해 사용하였다. 또한 보면 public GameObject로 상당히 많은 변수들이 설정되어 있는 것을 확인할 수 있는데. 이는 모두 동일 씬에서 사용할 오브젝트들이다. 당장 사용하지 않을 것들은 false로 설정해 화면에서 보이지 않게 하였다.

4. 충돌 이벤트

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SUN : MonoBehaviour

{

 public int s = 0, sf = 1;

 public GameObject gauge;

 public GameObject sun;

 public GameObject ball;

void Update()

 {

 float speed = 0.0018f;

 if (sf == 1 && s>0 && s < 20)

 {

 gauge.transform.Translate(0, speed, 0);

 s = s + 1;

 if (s == 20)

 {

 sf = sf + 1;

 }

 }

 else if (sf == 2 && s > 20 && s < 40)

 {

 gauge.transform.Translate(0, speed, 0);

 s = s + 1;

 if (s == 40)

 {

 sf = sf + 1;

 }

 }

 else if (sf == 3 && s > 40 && s < 60)

 {

 gauge.transform.Translate(0, speed, 0);

 s = s + 1;

 if (s == 60)

 {

 sf = sf + 1;

 }

 }

 }

void OnCollisionEnter(Collision col)

 {

 if (col.gameObject.name == "Sphere")

 {

 s = s + 1;

 }

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

 {

 sun.transform.position = new Vector3(-0.1f, 0.148f, 0.35f);

 }

 }

}

void OnCollisionEnter 함수는 parameter가 지정된 물체와 충돌했을 시 실행되는 라이브러리 함수이다. col.gameObject.name == “Sphere” 같은 경우, 충돌하는 오브젝트의 이름이 “Sphere”일 때 실행이 되며, col.gameObject.tag == “barrier” 같은 경우, 충돌하는 오브젝트의 tag명이 “barrier”일 때 실행이 된다.