LAMPIRAN

I. Script C#

a. gameController

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

namespace VRproject
{
    public class gameController : MonoBehaviour
    {
        public static gameController instance;

        public GameObject mainMenuScreen;

        public GameObject startMenu;
        public GameObject levelMenu;
        public GameObject titleText;
        public GameObject optionMenu;
        public GameObject creditsMenu;
        public GameObject scoreMenu;
        public GameObject tutorialMenu;

        public Text[] nameText;
        public Text[] scoreText;
        public Text[] dateText;

        public int score = 0;

        public AudioSource bgmSource;
        public AudioSource sfxSource;
        public AudioClip[] bgm;
        public AudioClip[] sfx;
        public Slider bgmVolume;
        public Slider sfxVolume;

        public Toggle timeToggle;

        [SerializeField] private Text scoreNameText = null;
        [SerializeField] private Text scoreNumberText = null;

        public bool isEasy;
        public bool isMedium;
        public bool isHard;
        public Text lvlText;

        public Dropdown scoreDropdown;

        void Awake()
        {
            if (instance == null)
            {
                instance = this;
            }
        }
        void Start()
        {
            showMainMenu();
            bgmSource = GetComponent<AudioSource>();
            sfxSource = GetComponent<AudioSource>();

            if (!bgmSource.playOnAwake)
            {
                StartCoroutine(playMainBGM());
            }
        }

        public bool IsEasy()
        {
            return isEasy = true;
        }

        public bool IsMedium()
        {
            return isMedium = true;
        }

        public bool IsHard()
        {
            return isHard = true;
        }

        public void scoreInitialization(ScoreEntryData entryData)
        {
            scoreNameText.text = entryData.name;
            scoreNumberText.text = entryData.score.ToString();
        }

        private void playBGM(int sound)
        {
            bgmSource.clip = bgm[sound];
            bgmSource.Play();
        }

        private void playSFX(int sound)
        {
            sfxSource.clip = sfx[sound];
            sfxSource.Play();
        }

        public void regSliderBGM()
        {
            bgmVolume.onValueChanged.AddListener(delegate {
changeBGMVolume(bgmVolume.value); });
        }

        public void regSliderSFX()
        {
            sfxVolume.onValueChanged.AddListener(delegate {
changeSFXVolume(sfxVolume.value); });
        }

        public void changeBGMVolume(float sliderValue)
        {
            bgmSource.volume = sliderValue;
        }

        public void changeSFXVolume(float sliderValue)
        {
            sfxSource.volume = sliderValue;
        }
        private IEnumerator playMainBGM()
        {
            yield return new WaitForSeconds(0f);
            playBGM(0);
        }

        public void showMainMenu()
        {
            mainMenuScreen.SetActive(true);
            titleText.SetActive(true);
            startMenu.SetActive(true);
            levelMenu.SetActive(false);
            optionMenu.SetActive(false);
            creditsMenu.SetActive(false);
            scoreMenu.SetActive(false);
            tutorialMenu.SetActive(false);

            isEasy = false;
            isMedium = false;
            isHard = false;
        }

        public void showLevelMenu()
        {
            mainMenuScreen.SetActive(true);
            titleText.SetActive(true);
            levelMenu.SetActive(true);
            startMenu.SetActive(false);
            optionMenu.SetActive(false);
            creditsMenu.SetActive(false);
            scoreMenu.SetActive(false);
            tutorialMenu.SetActive(false);
        }

        public void showOptionMenu()
        {
            titleText.SetActive(false);
            startMenu.SetActive(false);
            levelMenu.SetActive(false);
            optionMenu.SetActive(true);
            creditsMenu.SetActive(false);
            scoreMenu.SetActive(false);
            tutorialMenu.SetActive(false);
        }

        public void showCreditMenu()
        {
            titleText.SetActive(false);
            startMenu.SetActive(false);
            levelMenu.SetActive(false);
            optionMenu.SetActive(false);
            creditsMenu.SetActive(true);
            scoreMenu.SetActive(false);
            tutorialMenu.SetActive(false);
        }

        public void showTutorialMenu()
        {
            titleText.SetActive(false);
            startMenu.SetActive(false);
            levelMenu.SetActive(false);
            optionMenu.SetActive(false);
            creditsMenu.SetActive(false);
            scoreMenu.SetActive(false);
            tutorialMenu.SetActive(true);
        }
        public void showScoreMenu()
        {
            titleText.SetActive(false);
            startMenu.SetActive(false);
            levelMenu.SetActive(false);
            optionMenu.SetActive(false);
            creditsMenu.SetActive(false);
            scoreMenu.SetActive(true);
            tutorialMenu.SetActive(false);

            scoreEntry(scoreDropdown);
        }

        public void clickEasyBtn()
        {
            IsEasy();
            gameStart();
        }

        public void clickMediumBtn()
        {
            IsMedium();
            gameStart();
        }

        public void clickHardBtn()
        {
            IsHard();
            gameStart();
        }

        public void gameStart()
        {

            SceneManager.LoadScene("SampleScene");
        }

        public void dorpdownChange()
        {
            scoreDropdown.onValueChanged.AddListener(delegate
            {
                scoreEntry(scoreDropdown);
            });
        }

        public void scoreEntry(Dropdown dropdown)
        {
            if (dropdown.value == 0)
            {
                for (int i = 0; i < ScoreBoard.EntryCount; i++)
                    ScoreBoard.GetEntry(i);


                for (int i = 0; i < nameText.Length; i++)
                {
                    ScoreBoard.ScoreEntry entry = ScoreBoard.GetEntry(i);
                    nameText[i].text = entry.name;
                    dateText[i].text = entry.date;
                    scoreText[i].text = entry.score.ToString();
                }
            } else if (dropdown.value == 1)
            {
                for (int i = 0; i < ScoreBoardMedium.EntryCount; i++)
                    ScoreBoardMedium.GetEntry(i);
                for (int i = 0; i < nameText.Length; i++)
                {
                    ScoreBoardMedium.ScoreEntry entryMedium =
ScoreBoardMedium.GetEntry(i);
                    nameText[i].text = entryMedium.name;
                    dateText[i].text = entryMedium.date;
                    scoreText[i].text = entryMedium.score.ToString();
                }
            } else if (dropdown.value == 2)
            {
                for (int i = 0; i < ScoreBoardHard.EntryCount; i++)
                    ScoreBoardHard.GetEntry(i);


                for (int i = 0; i < nameText.Length; i++)
                {
                    ScoreBoardHard.ScoreEntry entryHard =
ScoreBoardHard.GetEntry(i);
                    nameText[i].text = entryHard.name;
                    dateText[i].text = entryHard.date;
                    scoreText[i].text = entryHard.score.ToString();
                }
            }
        }

        public void hideMainMenu()
        {
            mainMenuScreen.SetActive(false);
        }



        public void quitGame()
        {
            Application.Quit();
        }
    }
}
```

b. inGameMune

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

namespace VRproject
{
    public class inGameMune : MonoBehaviour
    {

        public GameObject endGameScreen;

        public static inGameMune main;

        public Text historyText1;
        public Text historyText2;
        public Text historyText3;
        public Text historyText3Sub;
        public Text historyText3Sub2;
        public GameObject inputScreen;
        public InputField input;
        public static string inputText;
        public GameObject gameOver;
        public Text scoreText;
        public GameObject UIHelper;
        private TouchScreenKeyboard keyboard;

        void Start()
        {
            showGameOver();
            historyText1.CrossFadeAlpha(1.0f, 0.0f, false);
            historyText2.CrossFadeAlpha(0.0f, 0.0f, false);
            historyText3.CrossFadeAlpha(0.0f, 0.0f, false);
            historyText3Sub.CrossFadeAlpha(0.0f, 0.0f, false);
            historyText3Sub2.CrossFadeAlpha(0.0f, 0.0f, false);
        }

       void Update()
        {
            scoreText.text = gameController.instance.score.ToString();
        }

        public void showGameOver()
        {
            endGameScreen.SetActive(true);
            inputScreen.SetActive(false);
            gameOver.SetActive(false);
            UIHelper.SetActive(false);

            StartCoroutine(fadeHistory());
        }
        IEnumerator fadeHistory()
        {
            historyText1.CrossFadeAlpha(1.0f, 0.5f, false);
            Debug.Log("HT1 running");

            yield return new WaitForSeconds(10);

            historyText1.CrossFadeAlpha(0.0f, 0.5f, false);
            Debug.Log("HT1 withdrawing");

            yield return new WaitForSeconds(1);

            historyText2.CrossFadeAlpha(1.0f, 0.5f, false);

            yield return new WaitForSeconds(10);

            historyText2.CrossFadeAlpha(0.0f, 0.5f, false);

            yield return new WaitForSeconds(1);

            historyText3.CrossFadeAlpha(1.0f, 0.5f, false);

            yield return new WaitForSeconds(5);

            historyText3Sub.CrossFadeAlpha(1.0f, 0.5f, false);

            yield return new WaitForSeconds(2);

            historyText3Sub2.CrossFadeAlpha(1.0f, 0.5f, false);

            yield return new WaitForSeconds(10);

            historyText3.CrossFadeAlpha(0.0f, 0.5f, false);
            historyText3Sub.CrossFadeAlpha(0.0f, 0.5f, false);
            historyText3Sub2.CrossFadeAlpha(0.0f, 0.5f, false);

            yield return new WaitForSeconds(1);

            UIHelper.SetActive(true);
            inputScreen.SetActive(true);
            keyboard = TouchScreenKeyboard.Open("",
TouchScreenKeyboardType.Default);
            gameOver.SetActive(false);
        }

        public void GOScreen()
        {
            inputText = input.text;

            inputScreen.SetActive(false);
            keyboard.active = false;
            gameOver.SetActive(true);
        }

        public void Restart()
        {
            SceneManager.LoadScene("SampleScene");
        }

        public void toMainMenu()
        {
            SceneManager.LoadScene("MainMenu");
        }
    }
}
```

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI TEKNIK INFORMATIKA
JL. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122

No. Skripsi : 561

FORM VERIFIKASI

ABSTRAK BAHASA INGGRIS DAN TATA TULIS BUKU LAPORAN SKRIPSI

Nama Mahasiswa : Ika Puspa Fairuz Wiwanata **NIM** : 1741720018
Tanggal Ujian : 10 Juni 2021
Judul : Rancang Bangun Game Obscure History Menggunakan Media Virtual Reality

NO	BAGIAN YANG DIVERIFIKASI	NAMA VERIFIKATOR	TANGGAL VERIFIKASI	TTD
1	Abstrak Berbahasa Inggris	Atiqah Nurul Asri, S.Pd., M.Pd.	26 Agustus 2021	[illegible]
2	Tata Tulis Buku Laporan Skripsi	Hendra Pradibta, SE., M.Sc	1 September 2021	[illegible]