LAMPIRAN

Script Pemrograman *PlayerController*

```
using System.Collections;
using UnityEngine;
using UnityEngine.InputSystem.Interactions;

public class PlayerController : MonoBehaviour
{
    public float moveSpeed;
    public float rotateSpeed;

    private SimpleControls m_Controls;
    private Vector2 m_Rotation;

    public void Awake()
    {
        m_Controls = new SimpleControls();
    }

    public void OnEnable()
    {
        m_Controls.Enable();
    }

    public void OnDisable()
    {
        m_Controls.Disable();
    }

    public void Update()
    {
        var look = m_Controls.gameplay.look.ReadValue<Vector2>();
        var move = m_Controls.gameplay.move.ReadValue<Vector2>();

        Look(look);
        Move(move);
    }

    private void Move(Vector2 direction)
    {
        if (direction.sqrMagnitude < 0.01)
            return;
        var scaledMoveSpeed = moveSpeed * Time.deltaTime;
        var move = Quaternion.Euler(0, transform.eulerAngles.y, 0) *
new Vector3(direction.x, 0, direction.y);
        transform.position += move * scaledMoveSpeed;
    }

    private void Look(Vector2 rotate)
    {
        if (rotate.sqrMagnitude < 0.01)
            return;
        var scaledRotateSpeed = rotateSpeed * Time.deltaTime;
        m_Rotation.y += rotate.x * scaledRotateSpeed;
```

```
        m_Rotation.x = Mathf.Clamp(m_Rotation.x - rotate.y *
scaledRotateSpeed, -89, 89);
        transform.localEulerAngles = m_Rotation;
    }
}
```

Script Pemrograman Quiz*Manager*

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

public class QuizManager : MonoBehaviour
{
    public static QuizManager Instance;

    [System.Serializable]
    public class Question
    {
        public string questionString;
        public string clueString;
        public string[] answerString;

        public int idCorrectAns;
        public Sprite questionSprite;
    }

    public Question[] questionBank;
    public Question currentQuestion;

    public Image imageCanvas;
    [SerializeField] private Sprite[] imageSprite; // The sprite to
change to
    [SerializeField] public bool shouldChangeImage;
    private const string ShouldChangeImageKey = "ShouldChangeImage";

    public ImageSpriteChecker imageSpriteChecker;

    public int currentID;
    public int randomID;
    public int QuotesRanID;
    public int maxQuestion;
    public int answerQuestion;
    public int[] prevListID;
    public int scoreGame;
    public int maxScoreGame;

    public TextMeshProUGUI gameScore;
    public TextMeshProUGUI endGameScore;

    public Image imageQuestion;
    public TextMeshProUGUI textQuestion;
    public TextMeshProUGUI textClue;
    public TextMeshProUGUI[] textAnswer;
    public Sprite[] buttonAns;
```

```
    public TextMeshProUGUI endQoutes;
    public string[] endQoutesTextLow;
    public string[] endQoutesTextMid;
    public string[] endQoutesTextHigh;

    public GameObject loadingDivider;
    public GameObject popUpPanel;

    public TextMeshProUGUI EndRankText;
    public string[] EndRankString;

    private bool theEnd = false;

    private void Awake()
    {
        Instance = this;
    }

    public void Start()
    {
        LoadQuestionBank();
        LoadQuestionText();
        gameScore.text = answerQuestion.ToString();

        shouldChangeImage = PlayerPrefs.GetInt(ShouldChangeImageKey, 0)
== 1;

        if (!shouldChangeImage)
        {
            imageCanvas.sprite = imageSprite[0]; // Set the initial
sprite
        }

    }

    public void Update()
    {
        gameScore.text = answerQuestion.ToString();
        endGameScore.text = scoreGame.ToString();
    }

    public void LoadQuestionRandom()
    {
        //di acak dulu value currentID
        randomID = Random.Range(0, questionBank.Length);

        bool flag = false;

        for (int i = 0; i < prevListID.Length; i++)
        {
            if (randomID == prevListID[i])
            {
                flag = true;
                Debug.Log("same number = " + randomID);
                break;
            }
        }

        if (flag == false)
```

```
        {
            prevListID[answerQuestion] = randomID;
            currentID = randomID;
        }
        else
        {
            LoadQuestionRandom();
        }
    }

    public void LoadQuestionBank()
    {
        LoadQuestionRandom();
        currentQuestion = questionBank[currentID];
    }

    public void LoadQuestionText()
    {
        textQuestion.GetComponent<TextMeshProUGUI>().text =
currentQuestion.questionString;
        textClue.GetComponent<TextMeshProUGUI>().text =
currentQuestion.clueString;
        imageQuestion.sprite = currentQuestion.questionSprite;


        for (int i = 0; i < textAnswer.Length; i++)
        {
            textAnswer[i].text = currentQuestion.answerString[i];
        }

        if (shouldChangeImage)
        {
            imageCanvas.sprite = imageSprite[1]; // Set the correct
sprite
        }
    }

    public void CompareAnswer(int id)
    {
        if (currentQuestion.idCorrectAns == id)
        {
            Debug.Log("benar");
            /* AudioManager.Instance.PlaySFX("Answer"); */
            loadingDivider.SetActive(true);
            StartCoroutine(QuestionChecker());
            AddScore();

            if (scoreGame >= 20)
            {
                shouldChangeImage = true;
                // Save the state of shouldChangeImage to PlayerPrefs
                PlayerPrefs.SetInt(ShouldChangeImageKey,
shouldChangeImage ? 1 : 0);
                PlayerPrefs.Save();
            }
        }
        else if (currentQuestion.idCorrectAns != id)
        {
            Debug.Log("Salah");
            /* AudioManager.Instance.PlaySFX("Answer"); */
```

```
            loadingDivider.SetActive(true);
            StartCoroutine(QuestionChecker());

        }

        if (answerQuestion != maxQuestion - 1)
        {
            answerQuestion++;
            LoadQuestionBank();
            LoadQuestionText();

            // Set the sprite to index 1 if shouldChangeImage is true
            if (shouldChangeImage)
            {
                imageCanvas.sprite = imageSprite[1];
            }
        }
        else
        {
            theEnd = true;
            StartCoroutine(QuestionChecker());

            if (scoreGame <= 40 && scoreGame >= 0)
            {
                RandomQoutes();
                //random 3 kalimat low
                EndRankText.text = EndRankString[0];

            }
            else if (scoreGame <= 80 && scoreGame > 40)
            {
                RandomQoutes2();
                //popup panel aktif dan random 3 kalimat mid
                EndRankText.text = EndRankString[1];
            }
            else if (scoreGame <= 100 && scoreGame > 80)
            {
                RandomQoutes3();
                //popup panel aktif adn random 3 kalimat high
                EndRankText.text = EndRankString[2];
            }
        }
    }


    private void RandomQoutes()
    {
        QuotesRanID = Random.Range(0, 2);

        for (int i = 0; i < 3; i++)
        {
            endQoutes.GetComponent<Text>().text =
endQoutesTextLow[QuotesRanID];
        }

    }

    private void RandomQoutes2()
    {
        QuotesRanID = Random.Range(0, 2);
```

```
        for (int i = 0; i < 3; i++)
        {
            endQoutes.GetComponent<Text>().text =
endQoutesTextMid[QuotesRanID];
        }

    }

    private void RandomQoutes3()
    {
        QuotesRanID = Random.Range(0, 2);

        for (int i = 0; i < 3; i++)
        {
            endQoutes.GetComponent<Text>().text =
endQoutesTextHigh[QuotesRanID];
        }

    }
    public void AddScore()
    {
        if (scoreGame >= maxScoreGame)
        {
            scoreGame = maxScoreGame;
        }
        else
        {
            scoreGame += 10;
        }

    }

    IEnumerator QuestionChecker()
    {
        yield return new WaitForSeconds(0.5f);
        loadingDivider.SetActive(false);

        if (theEnd == true)
        {
            popUpPanel.SetActive(true);
            PanelPopUpClose();
        }
    }


    public void PanelPopUpClose()
    {
        StartCoroutine(ClosePopUpAfterDelay());
    }

    private IEnumerator ClosePopUpAfterDelay()
    {
        // Reset the QuizManager after a delay
        StartCoroutine(ResetQuizManagerAfterDelay());
        yield return new WaitForSeconds(4f); // Wait for 4 seconds
        popUpPanel.SetActive(false); // Deactivate the popup panel

        // Close quiz for all TriggerQuiz instances in the scene
```

```
        TriggerQuiz[] triggerQuizzes =
FindObjectsOfType<TriggerQuiz>();
        foreach (TriggerQuiz triggerQuiz in triggerQuizzes)
        {
            triggerQuiz.CloseQuiz();
        }

    }

    private IEnumerator ResetQuizManagerAfterDelay()
    {
        yield return new WaitForSeconds(2f); // Wait for 4 seconds

        currentID = -1;
        answerQuestion = 0;
        randomID = 0;
        QuotesRanID = 0;
        scoreGame = 0;

        for (int i = 0; i < prevListID.Length; i++)
        {
            prevListID[i] = -1;
        }

        // Load new questions for the next quiz round
        LoadQuestionBank();
        LoadQuestionText();

        theEnd = false;
    }

}
```

Script* Pemrograman *TriggerMateri

```
using UnityEngine;
using UnityEngine.InputSystem;

public class TriggerMateri : MonoBehaviour
{
    public GameObject materi;
    public GameObject materiPanel;
    public GameObject player;
    private bool isInRange = false;

    [SerializeField] private Transform targetPosition;
    [SerializeField] private Vector3 targetRotation;

    [SerializeField] private GameObject materiManager;

    [SerializeField] private PlayerController playerController;

    private InputAction aButtonAction;
    private InputAction bButtonAction;
    private InputAction yButtonAction;

    private void Start()
    {
```

```
        // Get the InputAction for the A button
        aButtonAction = new InputAction(binding:
"<Gamepad>/buttonSouth");
        aButtonAction.Enable();
        aButtonAction.performed += OnAButtonPressed;

        // Get the InputAction for the B button
        bButtonAction = new InputAction(binding:
"<Gamepad>/buttonEast");
        bButtonAction.Enable();
        bButtonAction.performed += OnBButtonPressed;

        // Get the InputAction for the Y button
        yButtonAction = new InputAction(binding:
"<Gamepad>/buttonNorth");
        yButtonAction.Enable();
        yButtonAction.performed += OnYButtonPressed;
    }

    private void OnDestroy()
    {
        aButtonAction.performed -= OnAButtonPressed;
        aButtonAction.Disable();

        bButtonAction.performed -= OnBButtonPressed;
        bButtonAction.Disable();

        yButtonAction.performed -= OnYButtonPressed;
        yButtonAction.Disable();
    }

    private void OnTriggerEnter(Collider other)
    {
        isInRange = true;
        materi.SetActive(true);
    }

    private void OnTriggerExit(Collider other)
    {
        isInRange = false;
        materi.SetActive(false);
    }

    public void OpenMateri()
    {
        // Set quiz panel active and disable player movement
        if (isInRange)
        {
            // Store the current player position and rotation
            Vector3 playerPosition = player.transform.position;
            Quaternion playerRotation = player.transform.rotation;

            // Set the target position and rotation where you want the
player to move
            Vector3 targetPos = targetPosition.position;
            Quaternion targetRot = Quaternion.Euler(targetRotation);

            // Move the player to the target position and rotation
            player.transform.position = targetPos;
            player.transform.rotation = targetRot;
```

```
            materiManager.SetActive(true);
            materi.SetActive(true);
            materiPanel.SetActive(true);
            if (playerController != null)
            {
                playerController.enabled = false;
            }
        }
    }

    /// <summary>
    /// Method to close the Materi
    /// </summary>
    public void CloseMateri()
    {
        if (isInRange && materiManager != null)
        {
            // Close the quiz panel and enable player movement
            materi.SetActive(false);
            materiManager.SetActive(false);
            materiPanel.SetActive(false);

            if (playerController != null)
            {
                playerController.enabled = true;
            }
        }
    }

    private void OnAButtonPressed(InputAction.CallbackContext context)
    {
        // Add your logic for handling A button press here (e.g.,
selecting something)
    }

    private void OnBButtonPressed(InputAction.CallbackContext context)
    {
        // Call the OpenMateri() method when B button is pressed
        OpenMateri();
    }

    private void OnYButtonPressed(InputAction.CallbackContext context)
    {
        // Call the CloseMateri() method when Y button is pressed
        CloseMateri();
    }
}
```

Script* Pemrograman *QuizNavigation

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

public class QuizNavigation : MonoBehaviour
{
    [SerializeField] private Button[] answerButtons;
```

```
    private int selectedButtonIndex = 0;
    private int prevButtonIndex = 0; // Keep track of the previously
selected button index
    private bool isButtonSelected = false;

    private void Start()
    {
        selectedButtonIndex = 0;
        answerButtons[selectedButtonIndex].Select();
        UpdateButtonColors(); // Set initial button colors
    }

    private void Update()
    {
        // Arrow key control
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        if (!isButtonSelected)
        {
            if (horizontalInput > 0 ||
Input.GetKeyDown(KeyCode.RightArrow))
            {
                // Move selection to the right
                SelectNextButton();
            }
            else if (horizontalInput < 0 ||
Input.GetKeyDown(KeyCode.LeftArrow))
            {
                // Move selection to the left
                SelectPreviousButton();
            }
            else if (verticalInput > 0 ||
Input.GetKeyDown(KeyCode.UpArrow))
            {
                // Move selection up
                SelectPreviousButton();
            }
            else if (verticalInput < 0 ||
Input.GetKeyDown(KeyCode.DownArrow))
            {
                // Move selection down
                SelectNextButton();
            }
        }

        // Check for button click
        if (Input.GetKeyDown(KeyCode.Joystick1Button0) ||
Input.GetKeyDown(KeyCode.Return))
        {
            // Get the button letter based on the selectedButtonIndex
            char buttonLetter = (char)(selectedButtonIndex + 65); // 65
is the ASCII value of 'A'

            // Log the button letter
            Debug.Log("Clicked Button: " + buttonLetter);
        }
    }

    private void SelectNextButton()
```

```
    {
        // Restore the color of the previous button

answerButtons[prevButtonIndex].GetComponentInChildren<Image>().color =
Color.white;

        // Move to the next button
        selectedButtonIndex++;
        if (selectedButtonIndex >= answerButtons.Length)
            selectedButtonIndex = 0;

        // Select the new button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.black;

        // Set a flag to prevent continuous selection
        isButtonSelected = true;
        StartCoroutine(ResetButtonSelection());

        // Update the previous button index
        prevButtonIndex = selectedButtonIndex;

        answerButtons[selectedButtonIndex].Select(); // Set focus to
the newly selected button
    }

    private void SelectPreviousButton()
    {
        // Restore the color of the previous button

answerButtons[prevButtonIndex].GetComponentInChildren<Image>().color =
Color.white;

        // Move to the previous button
        selectedButtonIndex--;
        if (selectedButtonIndex < 0)
            selectedButtonIndex = answerButtons.Length - 1;

        // Select the new button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.black;

        // Set a flag to prevent continuous selection
        isButtonSelected = true;
        StartCoroutine(ResetButtonSelection());

        // Update the previous button index
        prevButtonIndex = selectedButtonIndex;

        answerButtons[selectedButtonIndex].Select(); // Set focus to
the newly selected button
    }

    private void UpdateButtonColors()
    {
        for (int i = 0; i < answerButtons.Length; i++)
        {
            if (i == selectedButtonIndex)
```

```
            {
                answerButtons[i].GetComponentInChildren<Image>().color
= Color.black; // Set the selected button color
            }
            else
            {
                answerButtons[i].GetComponentInChildren<Image>().color
= Color.white; // Set the color for non-selected buttons
            }
        }
    }



    private IEnumerator ResetButtonSelection()
    {
        // Delay for button selection reset
        yield return new WaitForSeconds(0.3f);

        // Reset the flag to enable button selection again
        isButtonSelected = false;
    }


}
```

Script Pemrograman *ButtonListManager*

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

public class ButtonListManager : MonoBehaviour
{
    public List<Button> buttons;
    public EventSystem eventSystem;

    private int currentIndex = 0;
    private bool isAxisInUse = false;

    private void Start()
    {
        SelectButton(currentIndex);
    }

    public void OnNavigate(InputAction.CallbackContext context)
    {
        Vector2 navigateInput = context.ReadValue<Vector2>();

        if (!isAxisInUse && navigateInput.magnitude > 0.5f)
        {
            if (navigateInput.y > 0.5f)
            {
                SelectPreviousButton();
            }
            else if (navigateInput.y < -0.5f)
```

```
            {
                SelectNextButton();
            }

            isAxisInUse = true;
        }

        if (navigateInput.magnitude < 0.5f)
        {
            isAxisInUse = false;
        }
    }

    private void SelectNextButton()
    {
        currentIndex = (currentIndex + 1) % buttons.Count;
        SelectButton(currentIndex);
    }

    private void SelectPreviousButton()
    {
        currentIndex--;
        if (currentIndex < 0)
        {
            currentIndex = buttons.Count - 1;
        }
        SelectButton(currentIndex);
    }

    private void SelectButton(int index)
    {
        eventSystem.SetSelectedGameObject(buttons[index].gameObject);
    }
}
```

Script Pemrograman *GameManager*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine;
using UnityEngine.InputSystem; // Import the new input system namespace

public class GameManager : MonoBehaviour
{
    private void Update()
    {
        // Check for the X button press using new input system
        if (Keyboard.current.xKey.wasPressedThisFrame ||
Gamepad.current?.xButton.wasPressedThisFrame == true)
        {
            RestartScene();
        }
    }

    private void RestartScene()
    {
```

```
        SceneManager.LoadScene(0);
    }
}
```

Script Pemrograman *DisplayGaze*

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

public class DisplayGaze : MonoBehaviour
{
    [SerializeField] GameObject videoObject;
    [SerializeField] VideoPlayer videoPlayer; // Reference to the
VideoPlayer component
    [SerializeField] RenderTexture renderTexture; // Reference to the
RenderTexture

    private void Start()
    {
        SetDisplay(false);
    }

    public void OnPointerEnter()
    {
        SetDisplay(true);
    }

    public void OnPointerExit()
    {
        SetDisplay(false);
    }

    private void SetDisplay(bool glazedAt)
    {
        videoObject.SetActive(glazedAt);

        if (glazedAt)
        {
            // Play the video when gazing at the object
            videoPlayer.targetTexture = renderTexture;
            videoPlayer.Play();
        }
        else
        {
            // Stop the video and clear the target texture when not
gazing
            videoPlayer.Stop();
            videoPlayer.targetTexture = null;
            ClearRenderTexture(renderTexture);
        }
    }

    private void ClearRenderTexture(RenderTexture rt)
    {
        RenderTexture.active = rt;
        GL.Clear(true, true, Color.clear);
        RenderTexture.active = null;
```

```
        Debug.Log("canvas cleared");
    }
}
```

Script* Pemrograman *NPCController

```
using System.Collections;
using UnityEngine;

public class NPCController : MonoBehaviour
{
    [System.Serializable]
    public enum Move
    {
        Idle,
        Walk,
        DoingOne,
        DoingTwo,
        DoingThree
    }

    [System.Serializable]
    public class MovementPoint
    {
        public Transform point;
        public float stopDuration;
    }

    public Move move;

    [SerializeField] private Animator anim;
    [SerializeField] private float movementSpeed;
    public MovementPoint[] movementPoints;
    private int currentPointIndex = 0;
    private bool isMoving = false;

    private void Start()
    {
        anim = GetComponent<Animator>();
        MoveToNextPoint();
    }

    private void Update()
    {
        switch (move)
        {
            case Move.Idle:
                anim.SetBool("idle", true);
                break;
            case Move.Walk:
                anim.SetBool("idle", false);
                Walking();
                break;
        }
    }

    private void ChangeState(Move move)
    {
        this.move = move;
```

```
    }

    private void Walking()
    {
        if (isMoving)
        {
            ChangeState(Move.Walk);
            // Check if the NPC has reached the current movement point
            if (Vector3.Distance(transform.position,
movementPoints[currentPointIndex].point.position) < 0.1f)
            {
                anim.SetBool("idle", true);
                ChangeState(Move.Idle);

                // Stop the NPC at the current movement point for a
duration

StartCoroutine(StopAtCurrentPoint(movementPoints[currentPointIndex].sto
pDuration));
            }
        }
    }

    private IEnumerator StopAtCurrentPoint(float duration)
    {
        isMoving = false;
        yield return new WaitForSeconds(duration);
        MoveToNextPoint();
    }

    private void MoveToNextPoint()
    {
        if (movementPoints.Length == 0)
            return;

        currentPointIndex++;
        if (currentPointIndex >= movementPoints.Length)
            currentPointIndex = 0;

        ChangeState(Move.Walk);

        // Move the NPC to the next movement point

StartCoroutine(MoveToPoint(movementPoints[currentPointIndex].point.posi
tion));
    }

    private IEnumerator MoveToPoint(Vector3 targetPosition)
    {
        isMoving = true;
        while (Vector3.Distance(transform.position, targetPosition) >
0.1f)
        {
            // Calculate the direction to the target position
            Vector3 direction = targetPosition - transform.position;
            direction.y = 0f; // Ignore vertical distance for rotation

            // Rotate towards the target position
            if (direction != Vector3.zero)
            {
```

```
                Quaternion toRotation =
Quaternion.LookRotation(direction);
                transform.rotation =
Quaternion.Lerp(transform.rotation, toRotation, Time.deltaTime *
movementSpeed);
            }

            // Move towards the target position
            transform.position =
Vector3.MoveTowards(transform.position, targetPosition, movementSpeed *
Time.deltaTime);
            yield return null;
        }
    }

}
```

Script Pemrograman *MateriNavigation*

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

public class MateriNavigation : MonoBehaviour
{
    [SerializeField] private Button[] answerButtons;
    private int selectedButtonIndex = 0;
    private bool isButtonSelected = false;

    private void Start()
    {
        selectedButtonIndex = 0;
        answerButtons[selectedButtonIndex].Select();
    }

    private void Update()
    {
        // Arrow key control
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        if (!isButtonSelected)
        {
            if (horizontalInput > 0 ||
Input.GetKeyDown(KeyCode.RightArrow))
            {
                // Move selection to the right
                SelectNextButton();
            }
            else if (horizontalInput < 0 ||
Input.GetKeyDown(KeyCode.LeftArrow))
            {
                // Move selection to the left
                SelectPreviousButton();
            }
            else if (verticalInput > 0 ||
Input.GetKeyDown(KeyCode.UpArrow))
            {
```

```
                // Move selection up
                SelectPreviousButton();
            }
            else if (verticalInput < 0 ||
Input.GetKeyDown(KeyCode.DownArrow))
            {
                // Move selection down
                SelectNextButton();
            }
        }

        // Check for button click
        if (Input.GetKeyDown(KeyCode.Joystick1Button0) ||
Input.GetKeyDown(KeyCode.Return))
        {
            // Get the button letter based on the selectedButtonIndex
            char buttonLetter = (char)(selectedButtonIndex + 65); // 65
is the ASCII value of 'A'

            // Log the button letter
            Debug.Log("Clicked Button: " + buttonLetter);
        }
    }

    private void SelectNextButton()
    {
        // Deselect the current button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.white;

        // Move to the next button
        selectedButtonIndex++;
        if (selectedButtonIndex >= answerButtons.Length)
            selectedButtonIndex = 0;

        // Select the new button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.yellow;

        // Set a flag to prevent continuous selection
        isButtonSelected = true;
        StartCoroutine(ResetButtonSelection());
        answerButtons[selectedButtonIndex].Select(); // Set focus to
the newly selected button
    }

    private void SelectPreviousButton()
    {
        // Deselect the current button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.white;

        // Move to the previous button
        selectedButtonIndex--;
        if (selectedButtonIndex < 0)
            selectedButtonIndex = answerButtons.Length - 1;
```

```
        // Select the new button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.yellow;

        // Set a flag to prevent continuous selection
        isButtonSelected = true;
        StartCoroutine(ResetButtonSelection());
        answerButtons[selectedButtonIndex].Select(); // Set focus to
the newly selected button
    }


    private IEnumerator ResetButtonSelection()
    {
        // Delay for button selection reset
        yield return new WaitForSeconds(0.3f);

        // Reset the flag to enable button selection again
        isButtonSelected = false;
    }
}
```

Script Pemrograman *Loader*

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

public class Loader : MonoBehaviour
{
    public void LoaderScene(int scene)
    {
        SceneManager.LoadScene(scene);
    }

}
```

Script Pemrograman *ImageSpriteChecker*

```
using UnityEngine;
using UnityEngine.UI;

public class ImageSpriteChecker : MonoBehaviour
{
    public Image imageCanvas;
    public Sprite[] imageSprite;
    public GameObject quizManagerObject; // Reference to the
QuizManager GameObject

    private void Start()
    {
        // Get the QuizManager component from the specified GameObject
        QuizManager quizManager =
quizManagerObject.GetComponent<QuizManager>();

        if (quizManager.shouldChangeImage)
```

```
        {
            imageCanvas.sprite = imageSprite[1]; // Set the correct
sprite
        }
        else
        {
            imageCanvas.sprite = imageSprite[0]; // Set the initial
sprite
        }
    }
}
```

Script Pemrograman _ButtonNavigation_

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

public class ButtonNavigation : MonoBehaviour
{
    [SerializeField] private Button[] answerButtons;
    private int selectedButtonIndex = 0;
    private bool isButtonSelected = false;

    private void Start()
    {
        selectedButtonIndex = 0;
        answerButtons[selectedButtonIndex].Select();
    }

    private void Update()
    {
        // Arrow key control
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        if (!isButtonSelected)
        {
            if (horizontalInput > 0 ||
Input.GetKeyDown(KeyCode.RightArrow))
            {
                // Move selection to the right
                SelectNextButton();
            }
            else if (horizontalInput < 0 ||
Input.GetKeyDown(KeyCode.LeftArrow))
            {
                // Move selection to the left
                SelectPreviousButton();
            }
            else if (verticalInput > 0 ||
Input.GetKeyDown(KeyCode.UpArrow))
            {
                // Move selection up
                SelectPreviousButton();
            }
            else if (verticalInput < 0 ||
Input.GetKeyDown(KeyCode.DownArrow))
```

```
            {
                // Move selection down
                SelectNextButton();
            }
        }

        // Check for button click
        if (Input.GetKeyDown(KeyCode.Joystick1Button0) ||
Input.GetKeyDown(KeyCode.Return))
        {
            // Get the button letter based on the selectedButtonIndex
            char buttonLetter = (char)(selectedButtonIndex + 65); // 65
is the ASCII value of 'A'

            // Log the button letter
            Debug.Log("Clicked Button: " + buttonLetter);
        }
    }

    private void SelectNextButton()
    {
        // Deselect the current button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.white;

        // Move to the next button
        selectedButtonIndex++;
        if (selectedButtonIndex >= answerButtons.Length)
            selectedButtonIndex = 0;

        // Select the new button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.yellow;

        // Set a flag to prevent continuous selection
        isButtonSelected = true;
        StartCoroutine(ResetButtonSelection());
        answerButtons[selectedButtonIndex].Select(); // Set focus to
the newly selected button
    }

    private void SelectPreviousButton()
    {
        // Deselect the current button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.white;

        // Move to the previous button
        selectedButtonIndex--;
        if (selectedButtonIndex < 0)
            selectedButtonIndex = answerButtons.Length - 1;

        // Select the new button

answerButtons[selectedButtonIndex].GetComponentInChildren<Image>().colo
r = Color.yellow;
```

```
        // Set a flag to prevent continuous selection
        isButtonSelected = true;
        StartCoroutine(ResetButtonSelection());
        answerButtons[selectedButtonIndex].Select(); // Set focus to
the newly selected button
    }


    private IEnumerator ResetButtonSelection()
    {
        // Delay for button selection reset
        yield return new WaitForSeconds(0.3f);

        // Reset the flag to enable button selection again
        isButtonSelected = false;
    }
}
```

Script* Pemrograman *MateriManager

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

public class MateriManager : MonoBehaviour
{
    public static MateriManager Instance;

    [System.Serializable]
    public class Materi
    {
        public string title;
        public string link;
        public string description;
        public string location;
        public Sprite image;
    }

    public Materi[] materiList;
    private int currentMateriIndex = 0;

    public TextMeshProUGUI titleText;
    public TextMeshProUGUI linkText;
    public TextMeshProUGUI descriptionText;
    public TextMeshProUGUI locationText;
    public Image image;

    private void Awake()
    {
        Instance = this;
    }

    private void Start()
    {
        ShowMateri(currentMateriIndex);
    }

    public void NextMateri()
    {
```

```
        currentMateriIndex++;
        if (currentMateriIndex >= materiList.Length)
        {
            currentMateriIndex = 0;
        }

        ShowMateri(currentMateriIndex);
    }

    public void PreviousMateri()
    {
        currentMateriIndex--;
        if (currentMateriIndex < 0)
        {
            currentMateriIndex = materiList.Length - 1;
        }

        ShowMateri(currentMateriIndex);
    }

    private void ShowMateri(int index)
    {
        titleText.text = materiList[index].title;
        linkText.text = materiList[index].link;
        descriptionText.text = materiList[index].description;
        locationText.text = materiList[index].location;
        image.sprite = materiList[index].image;
    }
}
```

Script Pemrograman _TriggerQuiz_

```
using UnityEngine;
using UnityEngine.InputSystem;

public class TriggerQuiz : MonoBehaviour
{
    public static TriggerQuiz Instance;

    public GameObject quiz;
    public GameObject quizPanel;
    public GameObject player;
    [SerializeField] private Transform targetPosition;
    [SerializeField] private Vector3 targetRotation;
    private bool isInRange = false;
    [SerializeField] private GameObject quizManager;
    [SerializeField] private PlayerController playerController;

    private InputAction aButtonAction;
    private InputAction bButtonAction;


    private void Start()
    {
        quiz.SetActive(false);
        quizPanel.SetActive(false);

        Instance = this;
        // Get the InputActions for the A and B buttons
```

```
        aButtonAction = new InputAction(binding:
"<Gamepad>/buttonSouth");
        aButtonAction.Enable();
        aButtonAction.performed += OnAButtonPressed;

        bButtonAction = new InputAction(binding:
"<Gamepad>/buttonEast");
        bButtonAction.Enable();
        bButtonAction.performed += OnBButtonPressed;
    }

    private void OnDestroy()
    {
        aButtonAction.performed -= OnAButtonPressed;
        aButtonAction.Disable();

        bButtonAction.performed -= OnBButtonPressed;
        bButtonAction.Disable();
    }

    private void OnTriggerEnter(Collider other)
    {
        isInRange = true;
        quiz.SetActive(true);
    }

    private void OnTriggerExit(Collider other)
    {
        isInRange = false;
        quiz.SetActive(false);
    }

    public void OpenQuiz()
    {
        // Set quiz panel active and disable player movement
        if (isInRange)
        {
            // Store the current player position and rotation
            Vector3 playerPosition = player.transform.position;
            Quaternion playerRotation = player.transform.rotation;

            // Set the target position and rotation where you want the
player to move
            Vector3 targetPos = targetPosition.position;
            Quaternion targetRot = Quaternion.Euler(targetRotation);

            // Move the player to the target position and rotation
            player.transform.position = targetPos;
            player.transform.rotation = targetRot;

            // Activate quiz panel and other necessary objects
            quizManager.SetActive(true);
            quiz.SetActive(true);
            quizPanel.SetActive(true);

            // Disable player controller to prevent movement during the
quiz
            if (playerController != null)
            {
                playerController.enabled = false;
```

```
            }


        }
    }

    public void CloseQuiz()
    {
        if (quizPanel.activeSelf) // Check if the quiz panel is active
        {
            // Close the quiz panel and enable player movement
            quizPanel.SetActive(false);
            quizManager.SetActive(false);
            if (playerController != null)
            {
                playerController.enabled = true;
            }
        }
    }

    private void OnAButtonPressed(InputAction.CallbackContext context)
    {
        // Add your logic for handling A button press here (e.g.,
selecting something)
    }

    private void OnBButtonPressed(InputAction.CallbackContext context)
    {
        // Call the OpenQuiz() method when B button is pressed
        OpenQuiz();
    }
}
```

Script Pemrograman _NPCGaze_

```
using UnityEngine;

public class NPCGaze : MonoBehaviour
{
    public GameObject dialogue;
    public Animator anim;

    private void Start()
    {
        SetDialogue(false);
        anim = GetComponent<Animator>();
    }

    public void OnPointerEnter()
    {
        SetDialogue(true);
        anim.SetBool("Talk", true);
        Debug.Log("OnPointerEntertrue");
    }

    public void OnPointerExit()
    {
```

```
            SetDialogue(false);
            anim.SetBool("Talk", false);
            Debug.Log("OnPointerExitfalse");
        }

        private void SetDialogue(bool glazedOn)
        {
            if (glazedOn == true)
            {
                dialogue.SetActive(true);
            }
            else
            {
                dialogue.SetActive(false);
            }
        }
    }
```

Lembar Bimbingan Pembimbing 1

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

KONSULTASI/ BIMBINGAN TUGAS AKHIR

Nama Mahasiswa : Muhammad Benny Fathurrahman
NIM : 062030701685
Jurusan/Program Studi : D3 Teknik Komputer / Teknik Komputer
Dosen Pembimbing : Ikhthison Mekongga, S.T., M.Kom
Judul : Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1	06/06/2023	Perbaiki latar belakang	
2	04/07/2023	Bimbingan Bab II	
3	06/07/2023	Revisi Bab 2	
4	10/07/2023	Bimbingan Bab III	
5	12/07/2023	Revisi Bab 3	
6	18/07/2023	Bimbingan Bab IV	
7	19/07/2023	Acc Project	
8	25/07/2023	Bimbingan Bab V	
9	26/07/2023	Revisi Bab V	
10	27/07/2023	ACC	

Palembang, 2023

Mengetahui,
Ketua Jurusan

Azwardi, ST., M.T
NIP.197005232005011004

Lembar Bimbingan Pembimbing 2

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

KONSULTASI/ BIMBINGAN TUGAS AKHIR

Nama Mahasiswa	:	Muhammad Benny Fathurrahman
NIM	:	062030701685
Jurusan/Program Studi	:	D3 Teknik Komputer / Teknik Komputer
Dosen Pembimbing	:	Ica Admirani, S.Kom, M.Kom
Judul	:	Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	25/7-2023	Revisi Bab 1, 2 & 3	
2.	26/7-2023	Revisi Bab 1, 2 & 4	
3.	27/7-2023	Tambahkan materi dan Unity pd aplikasi	
4	31/7-2023	ACC Aplikasi perbaiki gambar tambahkan pembahasan	
5	1/8-2023	Revisi Bab 5	
6	1/8-2023	ACC LA	

Palembang, 2023

Mengetahui,
Ketua Jurusan

Azwardi, ST., M.T
NIP.197005232005011004

Rekomendasi Sidang

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REKOMENDASI UJIAN TUGAS AKHIR

Pembimbing Laporan Tugas Akhir, memberikan rekomendasi ujian laporan tugas akhir kepada,

Nama Mahasiswa	:	Muhammad Benny Fathurrahman
NIM	:	062030701685
Jurusan/Program Studi	:	D3 Teknik Komputer / Teknik Komputer
Judul Tugas Akhir	:	Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti Ujian Laporan Tugas Akhir, pada Tahun Akademik 2022/2023

Palembang, 27 Juli 2023

Disetujui oleh,

Pembimbing I

Ikhthison Mekongga, S.T., M.Kom
NIP.197705242000031002

Pembimbing II

Ica Admirani, S.Kom, M.Kom
NIP.197903282005012001

Revisi Dosen Penguji

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji	: Yulian Mirza,ST.,M.Kom
Nama Mahasiswa	: Muhammad Benny Fathurrahman
NIM	: 062030701685
Jurusan /Program Studi	: DIII Teknik Komputer
Judul LA/	: Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

No	Uraian	Paraf
	Tata tulis	

Palembang, Agustus 2023
Dosen Penguji

Yulian Mirza,ST.,M.Kom
NIP. 196607121990031003

Revisi Dosen Penguji

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji	: Hartati Deviana,ST.,M.Kom
Nama Mahasiswa	: Muhammad Benny Fathurrahman
NIM	: 062030701685
Jurusan /Program Studi	: DIII Teknik Komputer
Judul LA/	: Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

No	Uraian	Paraf
	Perbaiki tata tulis	

Palembang, Agustus 2023
Dosen Penguji

Hartati Deviana,ST.,M.Kom
NIP. 197405262008122001

Revisi Dosen Penguji

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji : Alan Novi Tompunu.ST.,MT
Nama Mahasiswa : Muhammad Benny Fathurrahman
NIM : 062030701685
Jurusan /Program Studi : DIII Teknik Komputer
Judul LA/ : Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

No	Uraian	Paraf
	Bab 3 Bab 5 Materi informasi pariwisata di Sumatera. Di buat interaktif alan.pari@gmail.com	

Palembang, 09 Agustus 2023
Dosen Penguji

Alan Novi Tompunu.ST.,MT
NIP. 197611082000031002

Revisi Dosen Penguji

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REVISI UJIAN TUGAS AKHIR

Dosen Penguji : Rian Rahmanda Putra, M.Kom
Nama Mahasiswa : Muhammad Benny Fathurrahman
NIM : 062030701685
Jurusan /Program Studi : DIII Teknik Komputer
Judul LA/ : Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

No	Uraian	Paraf
-	-	-

Palembang, Agustus 2023
Dosen Penguji

Rian Rahmanda Putra, M.Kom
NIP. 197307062005011003

Pelaksanaan Revisi

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

PELAKSANAAN REVISI UJIAN TUGAS AKHIR

Nama Mahasiswa : Muhammad Benny Fathurrahman
NIM : 062030701685
Jurusan /Program Studi : DIII Teknik Komputer
Judul LA/ Skripsi : Pengembangan Pameran Virtual Dengan Menggunakan Unity Game Engine Sebagai Sarana Pengenalan Pariwisata Yang Ada Di Sumatera Selatan

Telah melaksanakan revisi terhadap Laporan Tugas Akhir yang diujikan pada hari ....RABU..... tanggal .....9..... bulan .....AGUSTUS..... tahun .....2023..... Pelaksanaan revisi terhadap Laporan Tugas Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No	Komentar	Nama Dosen Penguji	Tanggal/ bulan	Tanda Tangan
1.	Acc	Yulian Mirza,ST.,M.Kom	14/11 2023	
2.	Acc	Alan Novi Tompunu.ST.,MT	19/12 2023	
3.	Acc	Hartati Deviana,ST.,M.Kom	28/11 2023	
4.	Acc	Rian Rahmanda Putra, M.Kom	14/11 -2023	

Palembang, November 2023
Ketua Penguji,

Yulian Mirza,ST.,M.Kom
NIP. 196607121990031003

Foto Hasil Projek