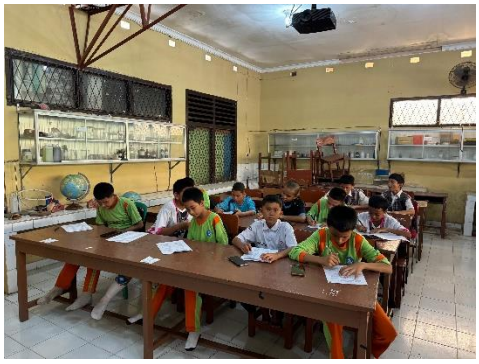
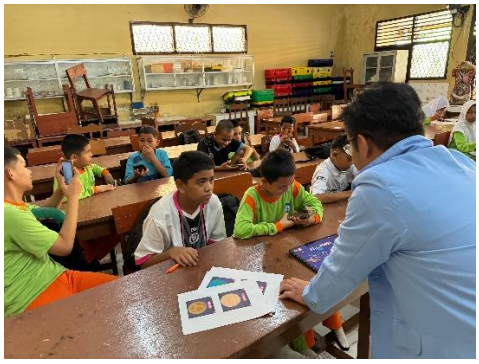
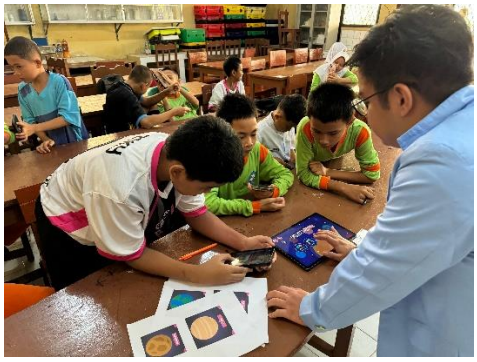
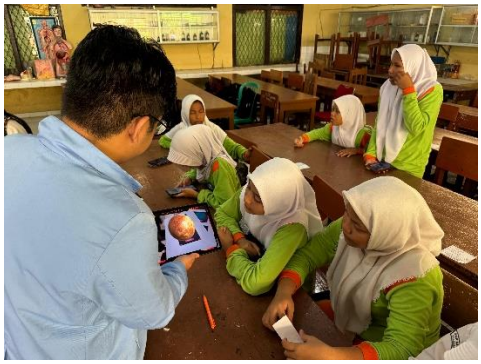
Lampiran









Validator Ahli Materi

Nama	**Maryani. S.P.d**
Instansi	**SMP Negeri 2 Palembang**
Jabatan	**Guru Ilmu Pengetahuan Alam**

Validator Ahli Media 1

Nama	**Muhammad Solihin Ansrulloh**
Institusi	**CEO Cikara Studio**
Pengalaman Kerja	**Designer and game developer**

Validator Ahli Media 2

Nama	**Firman Setiawan**
Institusi	**Cikara Studio**
Pengalaman Kerja	**Programmer Cikara Studio**

Validator Ahli Media 3

Nama	**Wahyu M Rizqi**
Institusi	**Cikara Studio**
Pengalaman Kerja	**Tester Application Cikara Studio**

Script button pindah scene

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneLoadeDestroy : MonoBehaviour
{
    // Metode untuk memuat scene baru dengan menghancurkan
AudioManager
    public void LoadSceneWithDestroy()
    {
        // Panggil metode DestroyAudioManager sebelum memuat scene
baru
        AudioManager.Instance.DestroyAudioManager();

        // Load scene baru, ganti "SceneBaru" dengan nama scene yang
ingin dimuat
        SceneManager.LoadScene("pilihmenu");
    }
}
```

Script quit game

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

public class MainMenu : MonoBehaviour
{
    public void TombolKeluar()
    {
        Application.Quit();
        Debug.Log("Game Close");
    }
```

Script download marker

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

public class PlanetMarker : MonoBehaviour
{
```

```
    public void Gdrive()
   {
   Application.OpenURL("https://drive.google.com/drive/folders/18UFN
Sd2vAZMI8LF9uijaC6-bTJgf-rgF?usp=sharing");
   }
}
```

TouchInteraction Planet

```
using UnityEngine;

public class TouchInteraction : MonoBehaviour
{
    private bool isRotating = false;
    private bool isScaling = false;
    private bool isDragging = false;
    private Vector2 touchStartPos;
    private Vector3 initialScale;
    private float initialDistance;
    private Vector3 initialPosition;

    void Update()
    {
        if (Input.touchCount == 1)
        {
            Touch touch = Input.GetTouch(0);

            // Deteksi mulai sentuh
            if (touch.phase == TouchPhase.Began)
            {
                isRotating = true;
                touchStartPos = touch.position;
            }
            // Deteksi pergerakan sentuh
            else if (touch.phase == TouchPhase.Moved && isRotating)
            {
                // Hitung perubahan posisi
                Vector2 deltaPos = touch.position - touchStartPos;

                // Terapkan rotasi pada objek 3D
                transform.Rotate(Vector3.up, deltaPos.x *
Time.deltaTime * 5f);

                // Simpan posisi sentuh untuk perbandingan
selanjutnya
                touchStartPos = touch.position;
```

```
            }
            // Deteksi akhir sentuh
            else if (touch.phase == TouchPhase.Ended)
            {
                isRotating = false;
            }
        }
        else if (Input.touchCount == 2)
        {
            Touch touch1 = Input.GetTouch(0);
            Touch touch2 = Input.GetTouch(1);

            // Deteksi mulai sentuh
            if (touch1.phase == TouchPhase.Began || touch2.phase ==
TouchPhase.Began)
            {
                initialPosition = transform.position;
                initialScale = transform.localScale;
                initialDistance = Vector2.Distance(touch1.position,
touch2.position);
                isScaling = true;
                isDragging = true;
            }
            // Deteksi pergerakan sentuh
            else if (touch1.phase == TouchPhase.Moved ||
touch2.phase == TouchPhase.Moved)
            {
                // Hitung jarak sentuhan baru
                float currentDistance =
Vector2.Distance(touch1.position, touch2.position);

                if (isScaling)
                {
                    // Hitung perbandingan skala
                    float scaleFactor = currentDistance /
initialDistance;

                    // Terapkan skala pada objek 3D
                    transform.localScale = initialScale *
scaleFactor;
                }

                if (isDragging)
                {
                    // Hitung posisi rata-rata sentuhan
```

```
                    Vector2 currentMidpoint = (touch1.position +
touch2.position) / 2;
                    Vector2 startMidpoint = (touch1.position -
touch1.deltaPosition + touch2.position - touch2.deltaPosition) / 2;

                    Vector2 deltaPos = currentMidpoint -
startMidpoint;

                    // Terapkan translasi pada objek 3D
                    transform.Translate(new Vector3(deltaPos.x *
Time.deltaTime * 0.1f, 0, deltaPos.y * Time.deltaTime * 0.1f),
Space.World);
                }
            }
            // Deteksi akhir sentuh
            else if (touch1.phase == TouchPhase.Ended ||
touch2.phase == TouchPhase.Ended)
            {
                isScaling = false;
                isDragging = false;
            }
        }
    }
}
```

Looping Rotation Planet

```
using UnityEngine;

public class LoopinRotation : MonoBehaviour
{
    public float rotationSpeed = 30f; // Kecepatan rotasi dalam
derajat per detik
    public Vector3 rotationAxis = Vector3.up; // Sumbu rotasi

    void Update()
    {
        // Rotasi objek di sekitar sumbu tertentu dengan kecepatan
yang ditentukan
        transform.Rotate(rotationAxis, rotationSpeed *
Time.deltaTime);
    }
}
```

Quiz

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

public class QuizManager : MonoBehaviour
{
    [Header("Question")]
    public GameObject questionPanel;
    public TextMeshProUGUI questionText;
    public TextMeshProUGUI questionNumberText; // UI untuk
menampilkan nomor soal
    public Button[] answerButtons;
    public TextMeshProUGUI[] answerTexts;
    public List<QuestionData> questionData;
    private List<QuestionData> shuffledQuestionData; // Menyimpan
urutan acak pertanyaan
    private int questionIndex;

    [Header("Result")]
    public GameObject resultPanel;
    public TextMeshProUGUI resultText;
    public float correctAnswerCount;
    public float wrongAnswerCount;
    public float score;
    public float totalQuestions;
    public bool isEnd;

    [Header("Stars")]
    public GameObject[] stars; // UI bintang

    private void Awake()
    {
        isEnd = false;
    }

    private void Start()
    {
        questionIndex = -1;

        // Menyalin daftar pertanyaan ke daftar acak
        shuffledQuestionData = new List<QuestionData>(questionData);
        ShuffleQuestions(shuffledQuestionData);
```

```
        questionPanel.SetActive(true);
        correctAnswerCount = 0;
        wrongAnswerCount = 0;
        totalQuestions = GetTotalQuestions();

        StartQuestion(); // Memulai soal secara otomatis saat
permainan dimulai
    }

    private int GetTotalQuestions()
    {
        return shuffledQuestionData.Count;
    }

    private void StartQuestion()
    {
        questionPanel.SetActive(true);
        questionIndex++;
        ApplyQuestionUI();
    }

    private void ApplyQuestionUI()
    {
        questionText.text =
shuffledQuestionData[questionIndex].question;

        // Menampilkan nomor soal di UI
        questionNumberText.text = "Soal No. " + (questionIndex +
1).ToString();

        int correctAnswerIndex = GenerateCorrectAnswer(4);
        Debug.Log($"Correct answer on index :
{correctAnswerIndex}");

        for (int i = 0; i < answerButtons.Length; i++)
        {
            answerButtons[i].onClick.RemoveAllListeners();
        }

        List<Button> wrongAnswerButtons = new List<Button>();
        List<TextMeshProUGUI> wrongAnswerTexts = new
List<TextMeshProUGUI>();
        for (int i = 0; i < answerButtons.Length; i++)
        {
```

```
            if (i != correctAnswerIndex)
            {
                wrongAnswerButtons.Add(answerButtons[i]);
                wrongAnswerTexts.Add(answerTexts[i]);
            }
            else
            {
                answerButtons[i].onClick.AddListener(delegate {
OnClickCorrectAnswer(); });
                answerTexts[i].text =
shuffledQuestionData[questionIndex].correctAnswer;
            }
        }

        for (int i = 0; i < wrongAnswerButtons.Count; i++)
        {
            wrongAnswerButtons[i].onClick.AddListener(delegate {
OnClickWrongAnswer(); });
        }

        wrongAnswerTexts[0].text =
shuffledQuestionData[questionIndex].wrongAnswer1;
        wrongAnswerTexts[1].text =
shuffledQuestionData[questionIndex].wrongAnswer2;
        wrongAnswerTexts[2].text =
shuffledQuestionData[questionIndex].wrongAnswer3;
    }

    private int GenerateCorrectAnswer(int maxValue)
    {
        return Random.Range(0, maxValue);
    }

    private void OnClickCorrectAnswer()
    {
        Debug.Log("Correct!");
        correctAnswerCount++;

        questionPanel.SetActive(false);

        UpdateScore();
        CheckWinLose();
    }

    private void OnClickWrongAnswer()
```

```
    {
        Debug.Log("Wrong!");
        wrongAnswerCount++;

        questionPanel.SetActive(false);

        UpdateScore();
        CheckWinLose();
    }

    private void UpdateScore()
    {
        score = (correctAnswerCount / totalQuestions) * 100f;
    }

    private void CheckWinLose()
    {
        if (questionIndex >= shuffledQuestionData.Count - 1)
        {
            DisplayResult();
        }
        else
        {
            StartQuestion(); // Memulai soal berikutnya jika masih
ada pertanyaan tersisa
        }
    }

    private void DisplayResult()
    {
        resultPanel.SetActive(true);
        resultText.text = $"Score: {score}\n\nCorrect Answers:
{correctAnswerCount}\nWrong Answers: {wrongAnswerCount}";

        // Menampilkan bintang berdasarkan skor
        if (score == 100)
        {
            ShowStars(3);
        }
        else if (score >= 80)
        {
            ShowStars(2);
        }
        else if (score >= 60)
        {
```

```
            ShowStars(1);
        }
        else if (score >= 50)
        {
            ShowStars(1);
        }
        else if (score >= 40)
        {
            ShowStars(1);
        }
        else if (score >= 30)
        {
            ShowStars(1);
        }
        else if (score >= 20)
        {
            ShowStars(1);
        }
        else if (score >= 10)
        {
            ShowStars(1);
        }
        else
        {
            ShowStars(0);
        }
    }

    private void ShowStars(int starCount)
    {
        for (int i = 0; i < stars.Length; i++)
        {
            stars[i].SetActive(i < starCount);
        }
    }

    public void BackToMenu(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }

    // Mengacak urutan pertanyaan
    private void ShuffleQuestions(List<QuestionData> list)
    {
        for (int i = 0; i < list.Count; i++)
```

```
        {
            QuestionData temp = list[i];
            int randomIndex = Random.Range(i, list.Count);
            list[i] = list[randomIndex];
            list[randomIndex] = temp;
        }
    }
}

[System.Serializable]
public class QuestionData
{
    public string question;
    public string correctAnswer;
    public string wrongAnswer1;
    public string wrongAnswer2;
    public string wrongAnswer3;
}
```

Joystick Controller Pesawat

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

public class JelajahController : MonoBehaviour
{
    public float moveSpeed = 5f;
    public float rotationSpeed = 5f;
    private Rigidbody2D rb;
    private Vector2 moveDirection;

    public VariableJoystick joystick;
    public bool isUseJoystick;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        ProcessInputs();
```

```
        RotatePlayer();
    }

    void FixedUpdate()
    {
        Move();
    }

    void ProcessInputs()
    {
        if (!isUseJoystick)
        {
            float moveX = Input.GetAxisRaw("Horizontal");
            float moveY = Input.GetAxisRaw("Vertical");
            moveDirection = new Vector2(moveX, moveY).normalized;
        }
        else
        {
            float moveX = joystick.Horizontal;
            float moveY = joystick.Vertical;
            moveDirection = new Vector2(moveX, moveY).normalized;
        }
    }

    void Move()
    {
        rb.velocity = moveDirection * moveSpeed;
    }

    void RotatePlayer()
    {
        if (moveDirection != Vector2.zero)
        {
            float targetAngle = Mathf.Atan2(moveDirection.y,
moveDirection.x) * Mathf.Rad2Deg;
            float angle = Mathf.LerpAngle(rb.rotation, targetAngle -
90f, rotationSpeed * Time.deltaTime); // Lerp rotation

            rb.MoveRotation(angle);
        }
    }
}
```

Controller Game Memory Puzzle

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

public class MemoryPuzzleController : MonoBehaviour
{
    public int currentCorrect;
    public int maxCorrect;
    public List<GameObject> tilePrefabs;
    public float delayCheck;
    public AudioSource SFX;
    public Transform parentObj;

    public List<Button> spawnedTiles;
    public GameObject firstSelected;
    public GameObject secondSelected;
    public GameObject resultPanel;

    public static MemoryPuzzleController instaces;

    private void Awake()
    {
        instaces = this;
    }

    private void Start()
    {
        SpawnAllPrefabsRandomly();

        currentCorrect = 0;
        maxCorrect = tilePrefabs.Count / 2;
    }

    private void SpawnAllPrefabsRandomly()
    {
        Shuffle(tilePrefabs);
        foreach (GameObject prefab in tilePrefabs)
        {
            GameObject spawnedPrefab = Instantiate(prefab,
parentObj);
            //spawnedPrefab.GetComponent<Button>().onClick.RemoveAll
Listeners();
            spawnedPrefab.GetComponent<Button>().onClick.AddListener
(delegate {
```

```
spawnedPrefab.GetComponent<TileController>().OnClickTile();
SFX.Play(); });
            spawnedTiles.Add(spawnedPrefab.GetComponent<Button>());
        }
    }

    private void Shuffle(List<GameObject> list)
    {
        int n = list.Count;
        while (n > 1)
        {
            n--;
            int k = Random.Range(0, n + 1);
            GameObject value = list[k];
            list[k] = list[n];
            list[n] = value;
        }
    }

    public IEnumerator CheckingSelectedTiles()
    {
        for (int i = 0; i < spawnedTiles.Count; i++)
        {
            spawnedTiles[i].GetComponent<Button>().interactable =
false;
        }

        if (firstSelected != null && secondSelected != null)
        {
            if (firstSelected.gameObject.name ==
secondSelected.gameObject.name)
            {
                firstSelected.GetComponent<TileController>().isCompl
ete = true;
                secondSelected.GetComponent<TileController>().isComp
lete = true;

                currentCorrect++;
            }
        }

        yield return new WaitForSeconds(delayCheck);

        if (firstSelected != null && secondSelected != null)
        {
```

```
            if (firstSelected.gameObject.name ==
secondSelected.gameObject.name)
            {
                firstSelected.GetComponent<Image>().sprite =
firstSelected.GetComponent<TileController>().completeImage;
                secondSelected.GetComponent<Image>().sprite =
secondSelected.GetComponent<TileController>().completeImage;
            }

            firstSelected = null;
            secondSelected = null;
        }

        for (int i = 0; i < spawnedTiles.Count; i++)
        {
            if
(!spawnedTiles[i].GetComponent<TileController>().isComplete)
            {
                spawnedTiles[i].GetComponent<Button>().interactable
= true;
                spawnedTiles[i].GetComponent<Image>().sprite =
spawnedTiles[i].GetComponent<TileController>().deselectedImage;
            }
        }

        if (currentCorrect >= maxCorrect)
        {
            Debug.Log("win");
            resultPanel.SetActive(true);
        }
    }
}
```

Playerhealth Game Astrounot Platform

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

public class playerhealth : MonoBehaviour
{
    public int maxHealth = 10;
    public int health;
    public GameObject gameOverUI; // Referensi ke UI Game Over
```

```
    // Start is called before the first frame update
    void Start()
    {
        health = maxHealth;
        gameOverUI.SetActive(false); // Sembunyikan UI Game Over
saat permainan dimulai
    }

    public void TakeDamage(int damage)
    {
        health -= damage;
        if (health <= 0)
        {
            health = 0; // Pastikan kesehatan tidak negatif
            GameOver(); // Panggil fungsi GameOver jika kesehatan
habis
        }
    }

    void GameOver()
    {
        // Tampilkan UI Game Over
        gameOverUI.SetActive(true);
        // Time.timeScale = 0; // Untuk menghentikan waktu permainan
(opsional)
        // Tambahan kode lain yang ingin Anda jalankan saat game
over
    }


}
```

Platformer Controller

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

public class PlatformerController : MonoBehaviour
{
    public float inputValue;

    public float moveSpeed = 5f;
    public float jumpForce = 10f;
    private Rigidbody2D rb;
```

```
    private bool isGrounded;
    public Transform groundCheck;
    public LayerMask groundLayer;
    public float groundCheckRadius = 0.2f;

    private Animator animator;
    private bool isRunAnim;
    private bool isIdleAnim;
    private bool isjumpAnim;

    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();

        isIdleAnim = false;
        isRunAnim = true;
    }

    private void Update()
    {
        isGrounded = Physics2D.OverlapCircle(groundCheck.position,
groundCheckRadius, groundLayer);

        float moveInput = inputValue;
        rb.velocity = new Vector2(moveInput * moveSpeed,
rb.velocity.y);

        CharacterFacing();
        AnimationController();
    }

    private void AnimationController()
    {
        if (inputValue == 0)
        {
            animator.Play("idle");
            isRunAnim = true;
            isIdleAnim = false;
        }
        else
        {
            if (inputValue != 0 && isRunAnim && !isIdleAnim)
            {
                animator.Play("run");
```

```
                isRunAnim = false;
                isIdleAnim = true;
            }

        }
    }

    private void CharacterFacing()
    {
        if (inputValue == -1)
        {
            GetComponent<SpriteRenderer>().flipX = true;
        }
        else if (inputValue == 1)
        {
            GetComponent<SpriteRenderer>().flipX = false;
        }
    }

    public void JumpButton()
    {
        if (isGrounded)
        {
            rb.velocity = new Vector2(rb.velocity.x, jumpForce);
            animator.Play("jump");

        }
    }

}
```

Soal Manager Game Platform

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

public class SoalManager : MonoBehaviour
{
    [Header("Soal")]
    public GameObject soalPanel;
    public TextMeshProUGUI soalText;
```

```
public Button[] answerButton;
public TextMeshProUGUI[] answerText;
public List<SoalData> soalData;
private int soalIndex;

[Header("Result")]
public GameObject resultPanel;
public TextMeshProUGUI resultText;
public float correctAnswerCount;
public float uncorrectAnswerCount;
public float score;
public float totalSoal;
public bool isEnd;

private void Awake()
{
    isEnd = false;
}

private void Start()
{
    soalIndex = -1;

    soalPanel.SetActive(false);
    correctAnswerCount = 0;
    uncorrectAnswerCount = 0;
    totalSoal = GetTotalSoal();
}

private int GetTotalSoal()
{
    int thisTotalSoal = soalData.Count;

    return thisTotalSoal;
}

public void OnStartSoal()
{
    soalPanel.SetActive(true);
    soalIndex ++;
    OnApplySoalUI();
}

private void OnApplySoalUI()
{
```

```
        soalText.text = soalData[soalIndex].soal;

        int correctAnswerIndex = OnGenerateCorrectAnswer(4);
        Debug.Log($"Correct answer on index :
{correctAnswerIndex}");

        for (int i = 0; i < answerButton.Length; i++)
        {
            answerButton[i].onClick.RemoveAllListeners();
        }

        List<Button> uncorrectAnswerButton = new List<Button>();
        List<TextMeshProUGUI> uncorrectValueText = new
List<TextMeshProUGUI>();
        for (int i = 0; i < answerButton.Length; i++)
        {
            if (i != correctAnswerIndex)
            {
                uncorrectAnswerButton.Add(answerButton[i]);
                uncorrectValueText.Add(answerText[i]);
            }
            else
            {
                answerButton[i].onClick.AddListener(delegate {
OnClickCorrectAnswer(); });
                answerText[i].text =
soalData[soalIndex].jawabanBenar;
            }
        }

        for (int i = 0; i < uncorrectAnswerButton.Count; i++)
        {
            uncorrectAnswerButton[i].onClick.AddListener(delegate {
OnClickUncorrectAnswer(); });
        }

        uncorrectValueText[0].text =
soalData[soalIndex].jawabanSalah1;
        uncorrectValueText[1].text =
soalData[soalIndex].jawabanSalah2;
        uncorrectValueText[2].text =
soalData[soalIndex].jawabanSalah3;
    }

    private int OnGenerateCorrectAnswer(int maxValue)
```

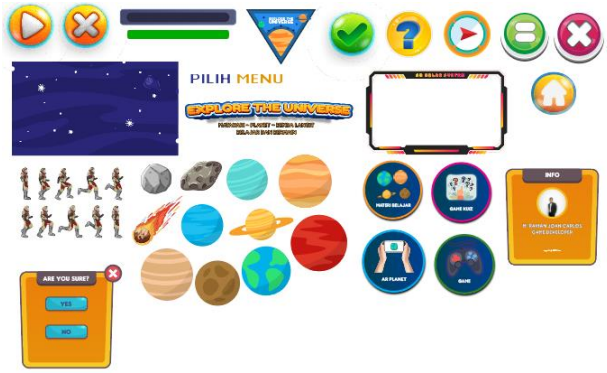
```
    {
        int value = Random.Range(0, maxValue);
        return value;
    }

    private void OnClickCorrectAnswer()
    {
        Debug.Log("BENAR");
        correctAnswerCount++;

        soalPanel.SetActive(false);

        OnUpdateScore();
        OnCheckWinLose();
    }

    private void OnClickUncorrectAnswer()
    {
        Debug.Log("SALAH");
        uncorrectAnswerCount++;

        soalPanel.SetActive(false);

        OnUpdateScore();
        OnCheckWinLose();
    }

    private void OnUpdateScore()
    {
        score = (float)(correctAnswerCount / totalSoal) * 100f;
    }

    private void OnCheckWinLose()
    {
        if (soalIndex >= soalData.Count - 1)
        {
            OnDisplayResult();
        }
    }

    private void OnDisplayResult()
    {
        resultPanel.SetActive(true);
        resultText.text = $"Score : {score}%\n\nJawaban Benar :
{correctAnswerCount}\nJawaban Salah : {uncorrectAnswerCount}";
```

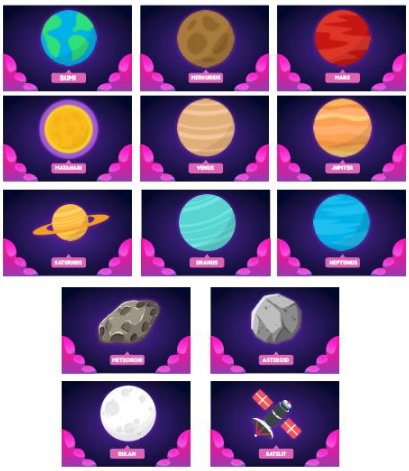
```
    }

    public void BackToMenu(string SceneName)
    {
        SceneManager.LoadScene(SceneName);
    }
}

[System.Serializable]
public class SoalData
{
    public string soal;
    public string jawabanBenar;
    public string jawabanSalah1;
    public string jawabanSalah2;
    public string jawabanSalah3;
}
```

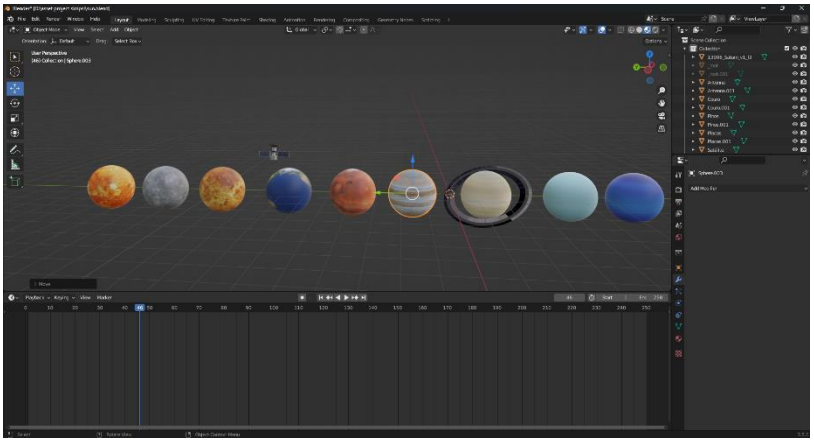
Soal Trigger Game Platform

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

public class SoalTrigger : MonoBehaviour
{
    public SoalManager SoalManager;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            SoalManager.OnStartSoal();
            GetComponent<Collider2D>().enabled = false;
            Destroy(gameObject);
        }
    }
}
```

Dokumentasi Aset







Data Siswa

No	Nama Siswa
1	M. Daffa Ramadhan
2	Yoga Adhitia Ramadhan
3	Sai Ilyas Pradika
4	M.Ibnu
5	M. Yudha Putra
6	Alief Akbar
7	M. Al Aqsha As Syifa
8	Alya Nazwa
9	Hany Tamara Lestari
10	Aqila Disya
11	Karin Juniarti
12	Mawar Yuriska
13	Miftahul Anshori
14	Nayya Zaira Assyifa
15	Silvia Destriani
16	Fitri Rahmadani
17	Ambrizal
18	M.Zaky
19	Alex Tgen
20	M. Rafaela Marco

Data Ahli Media

Nama Ahli	Hasil Kuisioner									
Firman Setiawan	4	3	4	3	4	3	4	4	3	3
Muhammad Solihin Ansrulloh	4	3	3	3	4	3	4	3	3	4
Wahyu M Rizqi	4	3	3	4	3	3	3	3	3	3