L A M P I R A N

AIBotScript.cs

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

public class AIBotScript : MonoBehaviour
{
    public float speed = 5.0f;

    private Rigidbody2D rb;

    public ParticleController particleController;

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

        Physics2D.IgnoreLayerCollision(9, 10);
    }

    void FixedUpdate()
    {
        // Menggerakkan bot sepeda ke depan
        Vector2 moveDirection = new Vector2(1, 0); // Bot akan bergerak ke arah kanan (1,0)
        rb.velocity = moveDirection * speed;
    }

    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            // Hentikan bot sementara
            rb.velocity = Vector2.zero;
            // Atau lakukan sesuatu yang sesuai dengan permainan Anda.

            Debug.Log("tabrakan..awww");
            //Destroy(gameObject);

            //Variable ini di ambil dari ParticleController.cs

            particleController.PlayTouchParticle();
        }
    }
}
```

AudioManager.cs

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



public class AudioManager : MonoBehaviour
{
    public static AudioManager Instance;


    public Sound[] musicSounds, sfxSounds;
    public AudioSource musicSource, sfxSource;


    private AudioSource activeSfxSource; // AudioSource untuk sfx aktif


    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;

DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }


    private void Start()
    {
        PlayMusic("Theme");
    }


    public void PlayMusic(string name)
    {
        Sound s = Array.Find(musicSounds, x => x.name == name);


        if (s == null)
        {
            Debug.Log("Sound tidak ditemukan");
        }
        else
        {
            musicSource.clip = s.clip;
            musicSource.Play();
        }
    }
```

```
    public void PlaySFX(string name)
    {
        Sound s =
Array.Find(sfxSounds, x => x.name
== name);


        if (s == null)
        {
            Debug.Log("Sound tidak
ditemukan");
        }
        else
        {

sfxSource.PlayOneShot(s.clip);
            sfxSource.Play();
        }
    }


    public void StopSFX(string name)
    {
        if (activeSfxSource != null &&
activeSfxSource.clip != null &&
activeSfxSource.clip.name == name)
        {
            activeSfxSource.Stop();
        }
    }


}
```

CollectFuel.cs

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


public class CollectFuel :
MonoBehaviour
{


    private void
OnTriggerEnter2D(Collider2D
collision)
    {

if(collision.gameObject.CompareTag
("Player"))
        {

FuelController.instance.FillFuel();
            Destroy(gameObject);

AudioManager.Instance.PlaySFX("p
oint");
            ScoreManager.scoreCount +=
120;
        }
    }
}
```

DisplayDistanceText.cs

```
using System.Collections;
```

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


public class DisplayDistanceText : MonoBehaviour
{
    [SerializeField] private Slider _positionSlider;
    [SerializeField] private TextMeshProUGUI _distanceText;
   [SerializeField] private TextMeshProUGUI _JarakGOText;
   [SerializeField] private Transform _playerTrans;



    private Vector2 _startPosition;
    private Vector2 _PlayerPosition;


    private void Start()
    {
        _startPosition = _playerTrans.position;
        _PlayerPosition = _startPosition;
        _positionSlider.value = _startPosition.x;
    }

    private void Update()
    {


        Vector2 distance = (Vector2)_playerTrans.position - _startPosition;
        distance.y = 0f;


        if (distance.x < 0)
        {
            distance.x = 0;
        }


        _positionSlider.value = distance.x;



        _distanceText.text = distance.x.ToString("F0") + "m";
        _JarakGOText.text = "Jarak :" + distance.x.ToString("F0") + "m";
    }


}
```

DisplaySkill.cs

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

```
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class DisplaySkill : MonoBehaviour
{
    public static DisplaySkill instance;

    public GameObject FloatingTextSkillPrefab;
    public TextMeshProUGUI skillText;
    public TextMeshProUGUI hiSkillText;
    public static int skillCount;
    public static int hiSkillCount;

    private readonly Animator animator;

private void Awake()
{
    if (instance == null)
    {
        instance = this;
    }

    Time.timeScale = 1f;
}

// Start is called before the first frame update
void Start()
    {
        if (PlayerPrefs.HasKey("HighSkill"))
        {
            hiSkillCount = PlayerPrefs.GetInt("HighSkill");
        }

    }

    public void ShowFloatingTextSkill()
    {
        // Instansiasi prefab di posisi sepeda, dengan rotasi default
        var go = Instantiate(FloatingTextSkillPrefab, transform.position, Quaternion.identity, transform);

        // Ambil komponen TextMeshProUGUI dari objek yang diinstansiasi
```

```
        if (go.TryGetComponent<TextMeshProUGUI>(out var textMesh))
        {
            // Set nilai teks dengan nilai skill
            textMesh.text = "Skill: " + MathF.Round(skillCount);
            Debug.Log("nilai DisplaySkillnya Berfungsi.. ");
            // Memainkan animasi dengan Trigger atau nama animasi
            if (animator != null)
            {

animator.SetTrigger("SkillSound");
                // Atau jika Anda ingin langsung memainkan animasi tertentu:
                // animator.Play("NamaAnimasi");
            }
        }
        else
        {
            Debug.LogError("Prefab tidak memiliki komponen TextMeshProUGUI!");
        }


        // Mulai coroutine untuk menyembunyikan teks setelah 3 detik
        StartCoroutine(HideFloatingText(go));


        // Tidak perlu mengaktifkan prefab asli karena kita sudah membuat instansinya
        // Debug.Log("Skill Activated Float Text! Value: " + skillValue);
    }


    IEnumerator HideFloatingText(GameObject textObject)
    {
        // Tunggu selama 3 detik
        yield return new WaitForSeconds(1f);


        // Hapus objek teks
        Destroy(textObject);
    }



    // Method untuk menambah skill
    public void AddSkill(int amount)
    {
        skillCount += amount;


        if (skillCount > hiSkillCount)
        {
```

```
            hiSkillCount = skillCount;

PlayerPrefs.SetInt("HighSkill", hiSkillCount);
        }


    }


    // Update is called once per frame
    void Update()
    {
        if (skillCount > hiSkillCount)
{

            hiSkillCount = skillCount;

PlayerPrefs.SetInt("HighSkill", hiSkillCount);
        }
        skillText.text = "Skill: " + MathF.Round(skillCount);
        hiSkillText.text = "Hi-Skill: " + hiSkillCount;
    }
}
```

DriverDeathFromHead.cs

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

public class DriverDeathFromHead : MonoBehaviour
{


    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Ground"))
        {

GameManager.instance.GameOver();

AudioManager.Instance.PlaySFX("GameOver");

AudioManager.Instance.PlaySFX("Jatuh");
        }
    }
}
```

DriverDeathFromHeadAI.cs

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

```
public class
DriverDeathFromHeadAI :
MonoBehaviour
{


   private void
OnCollisionEnter2D(Collision2D
collision)
   {
      if
(collision.gameObject.CompareTag(
"Ground"))
      {

//GameManager.instance.GameOver(
);
         //Destroy(gameObject);



//AudioManager.Instance.PlaySFX("
Jatuh");


      }
   }
}
```

DriveSepeda.cs

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

public class DriveSepeda :
MonoBehaviour
{
   [SerializeField] private
Rigidbody2D _frontTireRB;
   [SerializeField] private
Rigidbody2D _backTireRB;
   [SerializeField] private
Rigidbody2D _gearTireRB;
   [SerializeField] private
Rigidbody2D _sepedaRB;
   [SerializeField] private float
_speed = 150f;
   [SerializeField] private float
_rotationSpeed = 300f;
   public MovementJoystick
movementJoystick;


   public AudioSource moveSound;
// Referensi ke komponen
AudioSource untuk suara
pergerakan.
   public AudioSource stopSound;
   private float _moveInput;



   private void Update()
   {
      _moveInput =
Input.GetAxisRaw("Horizontal");
```

```
        // Cek jika pemain sedang bergerak (input non-nol).
        if (Mathf.Abs(_moveInput) > 0.01f)
        {
            // Mainkan suara pergerakan jika belum sedang diputar.
            if (!moveSound.isPlaying && !stopSound.isPlaying)
            {
                moveSound.Play();
                stopSound.Stop();
                //Debug.Log("move sound bunyi..");
            }
        }
        else if (movementJoystick.joystickVec.x != 0)
        {
            // Mainkan suara pergerakan jika belum sedang diputar.
            if (!moveSound.isPlaying && !stopSound.isPlaying)
            {
                moveSound.Play();
                stopSound.Stop();
                //Debug.Log("move sound bunyi..");
            }
        }
        else
        {
            // Berhenti memutar suara pergerakan jika tidak ada input.
            moveSound.Stop();
            stopSound.Play();
            //Debug.Log("move sound tidak bunyi..");
        }

Physics2D.IgnoreLayerCollision(8, 9);

Physics2D.IgnoreLayerCollision(8, 10);
    }

    private void FixedUpdate()
    {
        if (movementJoystick.joystickVec.x != 0)
        {
            // _sepedaRB.velocity = new Vector2(movementJoystick.joystickVec.y * _speed);
            _frontTireRB.AddTorque(-movementJoystick.joystickVec.x * _speed * Time.fixedDeltaTime);
            _backTireRB.AddTorque(-movementJoystick.joystickVec.x * _speed * Time.fixedDeltaTime);
```

```
            _gearTireRB.AddTorque(-
movementJoystick.joystickVec.x *
_speed * Time.fixedDeltaTime);


_sepedaRB.AddTorque(movementJo
ystick.joystickVec.x *
_rotationSpeed *
Time.fixedDeltaTime);


        }

        else

        {


_sepedaRB.AddTorque(movementJo
ystick.joystickVec.x * 0);

        }

        _frontTireRB.AddTorque(-
_moveInput * _speed *
Time.fixedDeltaTime);

        _backTireRB.AddTorque(-
_moveInput * _speed *
Time.fixedDeltaTime);

        _gearTireRB.AddTorque(-
_moveInput * _speed *
Time.fixedDeltaTime);


_sepedaRB.AddTorque(_moveInput
* _rotationSpeed *
Time.fixedDeltaTime);


    }

}
```

EnvironmentGenerator.cs

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.U2D;


[ExecuteInEditMode]

public class EnvironmentGenerator :
MonoBehaviour

{

    [SerializeField] private
SpriteShapeController
_spriteShapeController;


    [SerializeField, Range(3f, 108f)]
private int _levelLength = 50;

    [SerializeField, Range(1f, 50f)]
private float _xMultiplier = 2f;

    [SerializeField, Range(1f, 50f)]
private float _yMultiplier = 2f;

    [SerializeField, Range(0f, 1f)]
private float _curveSmoothness =
0.5f;

    [SerializeField] private float
_moiseStep = 0.5f;

    [SerializeField] private float
_bottom = 10f;


    private Vector3 _lastPos;


    private void OnValidate()

    {
```

```
_spriteShapeController.spline.Clear()
;


      for (int i = 0; i < _levelLength;
i++)
      {
          _lastPos = transform.position
+ new Vector3(i * _xMultiplier,
Mathf.PerlinNoise(0, i * _moiseStep)
* _yMultiplier);

_spriteShapeController.spline.InsertP
ointAt(i, _lastPos);


          if(i != 0 && i !=
_levelLength - 1)
          {

_spriteShapeController.spline.SetTan
gentMode(i,
ShapeTangentMode.Continuous);

_spriteShapeController.spline.SetLef
tTangent(i, Vector3.left *
_xMultiplier * _curveSmoothness);

_spriteShapeController.spline.SetRig
htTangent(i, Vector3.right *
_xMultiplier * _curveSmoothness);
          }
      }

_spriteShapeController.spline.InsertP
ointAt(_levelLength, new
Vector3(_lastPos.x,
transform.position.y - _bottom));

_spriteShapeController.spline.InsertP
ointAt(_levelLength + 1, new
Vector3(transform.position.x,
transform.position.y - _bottom));
    }
}
```

FinishFromPlayer.cs

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


public class FinishFromPlayer :
MonoBehaviour
{


   void
OnTriggerEnter2D(Collider2D
other)
   {
      if
(other.CompareTag("Finish"))
      {
         UnlockNewLevel();
```

```
GameManager.instance.Finish();

AudioManager.Instance.musicSource.Stop();

AudioManager.Instance.PlaySFX("finish");
        }
    }

    void UnlockNewLevel()
    {

if(SceneManager.GetActiveScene().buildIndex >= PlayerPrefs.GetInt("ReachedIndex"))
        {

PlayerPrefs.SetInt("ReachedIndex", SceneManager.GetActiveScene().buildIndex);

PlayerPrefs.SetInt("UnlockedLevel", PlayerPrefs.GetInt("UnlockLevel", 1) + 1);
            PlayerPrefs.Save();
        }
    }


}
```

FuelController.cs

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


public class FuelController : MonoBehaviour
{
    public static FuelController instance;


    [SerializeField] private Image _fuelImage;
    [SerializeField, Range(0.1f, 5f)] private float _fuelDrainSpeed = 1f;
    [SerializeField] private float _maxFuelAmount = 100f;
    [SerializeField] private Gradient _fuelGradient;


    private float _currentFuelAmmount;


    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
        }
```

```
    }


    private void Start()
    {
        _currentFuelAmmount = _maxFuelAmount;
        UpdateUI();
    }


    private void Update()
    {
        _currentFuelAmmount -= Time.deltaTime * _fuelDrainSpeed;
        UpdateUI();


        if (_currentFuelAmmount <= 0f)
        {

            GameManager.instance.GameOver();
        }
    }


    private void UpdateUI()
    {
        _fuelImage.fillAmount = (_currentFuelAmmount / _maxFuelAmount);
        _fuelImage.color = _fuelGradient.Evaluate(_fuelImage.fillAmount);
    }


    public void FillFuel()
    {
        _currentFuelAmmount = _maxFuelAmount;
        UpdateUI();
    }
}
```

GameManager.cs

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI; // Tambahkan ini jika menggunakan UI Text


public class GameManager : MonoBehaviour
{
    public static GameManager instance;


    [SerializeField] private GameObject _gameOverCanvas;
```

```
    [SerializeField] private
GameObject _gameFinishCanvas;

    [SerializeField] private
GameObject _gameJoystickUI;

    [SerializeField] private
GameObject _gameSettingsCanvas;


    // Variabel untuk nos

    [SerializeField] private float
cooldownSkill = 5f; // Waktu tunggu
skill dalam detik

    private bool nosTersedia = true;



    private void Awake()

    {

        if (instance == null)

        {

            instance = this;

        }


        Time.timeScale = 1f;

    }


    public void GameOver()

    {


//AudioManager.Instance.StopSFX("
NosSound");


//AudioManager.Instance.PlaySFX("
WaktuHabisSound");



_gameOverCanvas.SetActive(true);

        Time.timeScale = 0f;

        DisplaySkill.skillCount = 0;


    }


    public void Finish()

    {


//SpedaOntelSkill.instance.ResetSkill
();


AudioManager.Instance.StopSFX("N
osSound");


_gameFinishCanvas.SetActive(true);

        Time.timeScale = 0f;


_gameJoystickUI.SetActive(false);

    }


    public void RestartGame()

    {


SceneManager.LoadScene(SceneMa
nager.GetActiveScene().buildIndex);

        ScoreManager.scoreCount = 0;
```

```
        DisplaySkill.skillCount = 0;
    }

    public void Settings()
    {

_gameSettingsCanvas.SetActive(true
);
        Time.timeScale = 0f;

_gameJoystickUI.SetActive(false);
    }

    public void MainMenu()
    {

SceneManager.LoadScene("MainMe
nu");
        ScoreManager.scoreCount = 0;
        DisplaySkill.skillCount = 0;
    }

    public void Lanjut()
    {

_gameSettingsCanvas.SetActive(fals
e);
        Time.timeScale = 1f;

_gameJoystickUI.SetActive(true);
        ScoreManager.scoreCount = 0;
        DisplaySkill.skillCount = 0;
    }

    public void ExitButton()
    {
        Application.Quit();
        Debug.Log("Aplikasi Keluar");
    }

    public void Level1()
    {

SceneManager.LoadScene("Level1")
;
    }

    public void Level2()
    {

SceneManager.LoadScene("Level2")
;
    }

    public void Level3()
    {

SceneManager.LoadScene("Level3")
;
```

```
    }

    public void MuatLevel(int indexLevel)
    {
        string namaScene = $"Level{indexLevel}";
        SceneManager.LoadScene(namaScene);
    }

    // Metode untuk menggunakan skill
    public void GunakanNos()
    {
        if (nosTersedia)
        {
            Debug.Log("NOS digunakan!");
            // Tambahkan efek skill di sini, misalnya menembakkan peluru, meningkatkan kekuatan, dll.
            NosControl.instance.FillNos();
            NosControl.instance.TerapkanNOS();
            AudioManager.Instance.PlaySFX("NosSound");
            StartCoroutine(CooldownSkill());
        }
        else
        {
            Debug.Log("NOS belum tersedia, tunggu cooldown.");
            AudioManager.Instance.StopSFX("NosSound");
        }
    }

    private IEnumerator CooldownSkill()
    {
        nosTersedia = false;
        yield return new WaitForSeconds(cooldownSkill);
        nosTersedia = true;
        Debug.Log("NOS sudah tersedia kembali.");
    }
}
```

LeaderboardManager.cs

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

```
public class LeaderboardManager : MonoBehaviour
{
    public static LeaderboardManager instance;

    public TextMeshProUGUI[] leaderboardTexts; // Array untuk menampilkan peringkat di UI
    private readonly List<PlayerScore> playerScores = new();

    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
        }
    }

    public void AddScore(string playerName, int score)
    {
        playerScores.Add(new PlayerScore(playerName, score));
        UpdateLeaderboard();
    }

    void UpdateLeaderboard()
    {
        playerScores.Sort((x, y) => y.score.CompareTo(x.score)); // Urutkan berdasarkan skor tertinggi

        for (int i = 0; i < leaderboardTexts.Length; i++)
        {
            if (i < playerScores.Count)
            {
                leaderboardTexts[i].text = $"{i + 1}. {playerScores[i].playerName} - {playerScores[i].score}";
            }
            else
            {
                leaderboardTexts[i].text = $"{i + 1}. -";
            }
        }
    }

    private class PlayerScore
    {
        public string playerName;
        public int score;

        public PlayerScore(string name, int score)
        {
            this.playerName = name;
```

```
            this.score = score;
        }
    }
}
```

LevelMenu.cs

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

public class LevelMenu : MonoBehaviour
{
    public Button[] buttons;

    private void Awake()
    {
        int unlockedLevel = PlayerPrefs.GetInt("UnlockLevel", 1);

        // Disable all buttons initially
        foreach (Button button in buttons)
        {
            button.interactable = false;
        }

        // Enable buttons up to the unlocked level
        for (int i = 0; i < buttons.Length; i++)
        {
            if (i < unlockedLevel)
            {
                buttons[i].interactable = true;
            }
        }
    }

    public void OpenLevel(int levelid)
    {
        string levelName = "Level" + levelid;
        // Optionally, check if the scene exists before loading
        if (Application.CanStreamedLevelBeLoaded(levelName))
        {
            SceneManager.LoadScene(levelName);
        }
        else
        {
```

```
            Debug.LogError("Scene " +
levelName + " does not exist.");
        }
    }
}
```

MovementJoystick.cs

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

public class MovementJoystick :
MonoBehaviour
{
    public GameObject joystick;
    public GameObject joystickBG;
    public Vector2 joystickVec;
    private Vector2 joystickTouchPos;
    private Vector2
joystickOriginalPos;
    private float joystickRadius;
    void Start()
    {
        joystickOriginalPos =
joystickBG.transform.position;
        joystickRadius =
joystickBG.GetComponent<RectTra
nsform>().sizeDelta.x / 4;
    }

    public void PointerDown()
    {
        joystick.transform.position =
Input.mousePosition;
        joystickBG.transform.position =
Input.mousePosition;
        joystickTouchPos =
Input.mousePosition;

    }

    public void Drag(BaseEventData
baseEventData)
    {
        PointerEventData
pointerEventData = baseEventData
as PointerEventData;
        Vector2 dragPos =
pointerEventData.position;
        joystickVec = (dragPos -
joystickTouchPos).normalized;

        float joystickDist =
Vector2.Distance(dragPos,
joystickTouchPos);

        if(joystickDist <
joystickRadius)
        {
```

```
            joystick.transform.position = joystickTouchPos + joystickVec * joystickDist;


        }

        else

        {

            joystick.transform.position = joystickTouchPos + joystickVec * joystickRadius;

        }

    }


    public void PointerUp()

    {

        joystickVec = Vector2.zero;

        joystick.transform.position = joystickOriginalPos;

        joystickBG.transform.position= joystickOriginalPos;

    }

}
```

NosControl.cs

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;


public class NosControl : MonoBehaviour

{

    public static NosControl instance;


    public float speedNos = 500f;


    [SerializeField] private Rigidbody2D PlayerEffect;


    [SerializeField] ParticleSystem nosParticle;


    [SerializeField] private Image _nosImage;

    [SerializeField, Range(0.1f, 5f)] private float _nosDrainSpeed = 1f;

    [SerializeField] private float _maxNosAmount = 100f;

    [SerializeField] private Gradient _nosGradient;

    [SerializeField] private GameObject _tombol;

    // Start is called before the first frame update


    private float _currentNosAmmount;


    private void Awake()

    {
```

```
        if (instance == null)
        {
            instance = this;
        }
    }
    void Start()
    {
        //_currentNosAmmount = 0f;
        //SembunykanTombol();
        //UpdateUI();
    }

    // Update is called once per frame
    private void Update()
    {
        _currentNosAmmount -=
Time.deltaTime * _nosDrainSpeed;
        UpdateUI();

        if (_currentNosAmmount <= 0f)
        {
            // _tombol.SetActive(true);
            SembunykanTombol();

AudioManager.Instance.StopSFX("N
osSound");
            StopNosParticle();
        }
    }

    private void UpdateUI()
    {
        _nosImage.fillAmount =
(_currentNosAmmount /
_maxNosAmount);
        _nosImage.color =
_nosGradient.Evaluate(_nosImage.fil
lAmount);
    }

    public void FillNos()
    {
        _currentNosAmmount =
_maxNosAmount;
        UpdateUI();
    }

    public void TampilkanTombol()
    {
        _tombol.SetActive(true);
    }

    public void SembunykanTombol()
    {
        _tombol.SetActive(false);
    }

    public void TerapkanNOS()
    {
```

```
        // Menggerakkan bot sepeda ke depan
        Vector2 moveDirection = new Vector2(1, 0); // Bot akan bergerak ke arah kanan (1,0)
        PlayerEffect.velocity = moveDirection * speedNos;
        Debug.Log("NOS diterapkan.");
        PlayNosParticle();
    }

    public void PlayNosParticle()
    {
        nosParticle.Play();
    }

    public void StopNosParticle()
    {
        nosParticle.Stop();
    }
}
```

ParticleController.cs

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

public class ParticleController : MonoBehaviour
{
    [Header("===Interaksi Particle Debu===")]
    [SerializeField] ParticleSystem movementParticle;

    [Range(0, 10)]
    [SerializeField] int occurAfterVelocity;

    [Range(0, 0.2f)]
    [SerializeField] float dustFormationPeroid;
    [SerializeField] Rigidbody2D playerRb;

    float counter;
    bool isOnGround;

    [Header("===Interaksi Partikel rumput dan sentuhan ===")]
    [SerializeField] ParticleSystem fallParticle;
    [SerializeField] ParticleSystem touchParticle;

    private void Update()
    {
        counter += Time.deltaTime;
```

```
        if (isOnGround &&
Mathf.Abs(playerRb.velocityX) >
occurAfterVelocity)
        {
            if(counter >
dustFormationPeroid)
            {
                movementParticle.Play();
                counter = 0;
            }
        }
    }

    //Fungsi ini dikaitkan ke
AIBotScript.CS
    public void PlayTouchParticle()
    {
        touchParticle.Play();
    }

    private void
OnTriggerEnter2D(Collider2D
collision)
    {
        if
(collision.CompareTag("Ground"))
        {
            fallParticle.Play();
            isOnGround = true;
AudioManager.Instance.PlaySFX("b
an");
        }
    }

    private void
OnTriggerExit2D(Collider2D
collision)
    {
        if
(collision.CompareTag("Ground"))
        {
            isOnGround = false;
        }
    }
}
```

RotateObject.cs

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

public class RotateObject :
MonoBehaviour
{
    private Rigidbody2D rb;
    // Start is called before the first
frame update
    void Start()
```

```
    {
        rb = GetComponent<Rigidbody2D>();
    }

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

        rb.rotation += Time.deltaTime * -100;
    }
}
```

ScoreManager.cs

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

public class ScoreManager : MonoBehaviour
{

    public TextMeshProUGUI scoreText;
    public TextMeshProUGUI hiScoreText;

    public static int scoreCount;
    public static int hiScoreCount;

    // Start is called before the first frame update
    void Start()
    {
        if (PlayerPrefs.HasKey("HighScore"))
        {
            hiScoreCount = PlayerPrefs.GetInt("HighScore");
        }

    }

    // Method untuk menambah skor
    public void AddScore(int amount)
    {
        scoreCount += amount;

        if (scoreCount > hiScoreCount)
        {
            hiScoreCount = scoreCount;

            PlayerPrefs.SetInt("HighScore", hiScoreCount);
            // Menambahkan skor ke leaderboard
```

```
LeaderboardManager.instance.AddScore("Player1", scoreCount);
        }

    }

    // Update is called once per frame
    void Update()
    {
        if (scoreCount > hiScoreCount)
        {
            hiScoreCount = scoreCount;

PlayerPrefs.SetInt("HighScore", hiScoreCount);
        }
        scoreText.text = "Score: " + MathF.Round(scoreCount);
        hiScoreText.text = "Hi-Score: " + hiScoreCount;
    }

}
```

Skill.cs

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

//Ketika menyentuh point skill

public class Skill : MonoBehaviour
{

    public float DestroyTime = 1f;
    public Vector3 offset = new Vector2(0, 2);
    public float TambahSkill = 5f;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {

NosControl.instance.TampilkanTombol();

NosControl.instance.FillNos();

            // Tambah Skill
            DisplaySkill.skillCount += (int)TambahSkill;

DisplaySkill.instance.ShowFloatingTextSkill();
```

```
        // Pindahkan objek sebelum dihancurkan
        transform.localPosition += offset;

        // Mainkan suara efek skill
AudioManager.Instance.PlaySFX("SkillSound");

        // Hancurkan objek setelah jeda waktu
        Destroy(gameObject, DestroyTime);
      }
   }

}
```

Sound.cs

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

[System.Serializable]
public class Sound
{
   public string name;
   public AudioClip clip;
}
```

SpedaOntelSkill.cs

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

public class SpedaOntelSkill : MonoBehaviour
{
   public static SpedaOntelSkill instance;

   public GameObject FloatingTextPrefab;

   public Rigidbody2D _sepedaRotation;
   private float initialRotation;
   public bool skillActivated = false;
   public float skillValue = 100f; // Nilai skill yang dikeluarkan
   public float RotateValue = 90f; // Rotasi skill yang dikeluarkan
   private readonly Animator animator;

   private void Awake()
   {
```

```
        if (instance == null)
        {
            instance = this;
        }
    }

    void Start()
    {
        if (_sepedaRotation != null)
        {
            initialRotation =
_sepedaRotation.rotation;
        }
        else
        {
            initialRotation =
transform.eulerAngles.z;
        }
    }



    void Update()
    {
        DetectRotation();
    }


    void DetectRotation()
    {
        float currentRotation =
(_sepedaRotation != null) ?
_sepedaRotation.rotation :
transform.eulerAngles.z;
        float rotationDifference =
Mathf.Abs(currentRotation -
initialRotation);


        // Deteksi rotasi 180 derajat
        if (rotationDifference >=
RotateValue && !skillActivated)
        {
            ActivateSkill();
            skillActivated = true;


        }
    }


    void ActivateSkill()
    {
        // Logika untuk mengeluarkan
nilai skill
        Debug.Log("Skill Activated!
Value: " + skillValue);


        DisplaySkill.skillCount +=
(int)skillValue;


AudioManager.Instance.PlaySFX("S
killSound");


        if (FloatingTextPrefab)
```

```
        {
            ShowFloatingText();
        }

        // Anda bisa menambahkan efek visual, suara, atau mekanisme lainnya di sini
    }

    void ShowFloatingText()
    {
        ResetSkill();
        // Instansiasi prefab di posisi sepeda, dengan rotasi default
        var go = Instantiate(FloatingTextPrefab, transform.position, Quaternion.identity, transform);

        // Ambil komponen TextMeshProUGUI dari objek yang diinstansiasi
        if (go.TryGetComponent<TextMeshProUGUI>(out var textMesh))
        {
            // Set nilai teks dengan nilai skill
            textMesh.text = "Hebat!!! Anda Mendapatkan Skill: " + skillValue.ToString();
            Debug.Log("nilai SepedaOntelSkill Berfungsi.. ");
            // Memainkan animasi dengan Trigger atau nama animasi
            if (animator != null)
            {
                animator.SetTrigger("SkillSound");
                // Atau jika Anda ingin langsung memainkan animasi tertentu:
                // animator.Play("NamaAnimasi");
            }
        }
        else
        {
            Debug.LogError("Prefab tidak memiliki komponen TextMeshProUGUI!");
        }

        // Mulai coroutine untuk menyembunyikan teks setelah 3 detik
        StartCoroutine(HideFloatingText(go));

        // Tidak perlu mengaktifkan prefab asli karena kita sudah membuat instansinya
        // Debug.Log("Skill Activated Float Text! Value: " + skillValue);
    }
```

```
    IEnumerator
HideFloatingText(GameObject
textObject)
    {
        // Tunggu selama 1 detik
        yield return new
WaitForSeconds(1f);


        // Hapus objek teks
        Destroy(textObject);


    }


    public void ResetSkill()
    {
        //skillValue = 0f;
        skillActivated = false;
        initialRotation =
transform.eulerAngles.z; // Set initial
rotation kembali ke posisi sekarang
    }
```

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : *Muh Rifli Aris*	Pembimbing I : *Ade Hastuty, S.T., S.Kom. MT.*
NIM : *218280009*	Pembimbing II : *Mugaffir Yunus, S.T., MT.*
Judul Skripsi : *APLIKASI GAME BALAP SEPEDA ONTEL BERBASIS ANDROID*	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Perbaikan Pada Abstrak		Konsultasi 1 Perbaikan Pada Prosedure Standar Penelitian	
Konsultasi 2 Menambahkan Abstrak dan kesimpulan		Konsultasi 2 menambahkan fitur waktu	
Konsultasi 3 Menambahkan class Diagram		Konsultasi 3 Acc	
Konsultasi 4 20/09/2024 Acc ujian		Konsultasi 4	
Konsultasi 5		Konsultasi 5 Acc Ujian tutup	

Lanjut ke halaman sebelah

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare,

Mengetahui
Ketua Program Studi

Marlina, S.Kom.,M.Kom.
NBM. 1162 680

Mahasiswa

Muh Rifli Aris
NIM. 218280009

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik