LAMPIRAN

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

public class menu_utama : MonoBehaviour
{

    public GameObject menu;
    public GameObject tentang;
    public GameObject belajar;
    public GameObject main_puzzle;
    public GameObject rumah_adat;
    public GameObject rumah_bone;
    public GameObject rumah_wajo;
    public GameObject rumah_bulukumba;
    public GameObject rumah_parepare;
    public GameObject rumah_sinjai;
    public GameObject rumah_barru;
    public GameObject rumah_makassar;
    public GameObject rumah_pinrang;
    public GameObject rumah_luwu;
    public GameObject rumah_soppeng;
    public GameObject rumah_sidrap;
    public GameObject tarian;
    public GameObject tari_pajoge;
    public GameObject tari_panjaga;
    public GameObject tari_paddupa;
    public GameObject alat_musik;
    public GameObject lagu_daerah;
    public GameObject badik;
    public GameObject bodo;
    public GameObject gendang1;
    public GameObject kecapi;
    public GameObject indo_logo;
    public GameObject alosi;
    public GameObject bulu_alauna;
    public GameObject puzzle_gambar;
    public GameObject quiz_video;
    public GameObject quiz_lagu;
    public GameObject cara_bermain;
    public GameObject puzzle_video1;
    public GameObject puzzle_lagu1;


    private TimerManager timerManager; // Referensi ke
TimerManager
    public GameObject puzzleGambarPanel; // Assign
panel puzzle_gambar


    // Start is called before the first frame update
    void Start()
    {
        menu.SetActive (true);
    }

    public void tentangClicked() {
        menu.SetActive (false);
        tentang.SetActive (true);
    }

    public void keluarClicked() {
        Application.Quit();
    }

    public void kembaliTentang() {
        menu.SetActive (true);
        tentang.SetActive (false);
    }

    //cara bermain
    public void caraClicked() {
            menu.SetActive (false);
            cara_bermain.SetActive (true);
```

```
}

public void kembaliCara() {
    menu.SetActive (true);
    cara_bermain.SetActive (false);
}

// belajar
public void belajarClicked() {
    menu.SetActive (false);
    belajar.SetActive (true);
}

public void kembaliBelajar() {
    menu.SetActive (true);
    belajar.SetActive (false);
}

// puzzle
public void mainPuzzleClicked() {
    menu.SetActive (false);
    main_puzzle.SetActive (true);
}

public void kembaliMainPuzzle() {
    menu.SetActive (true);
    main_puzzle.SetActive (false);
}

// puzzle gambar
public void mainPuzzleGmbClicked() {
    main_puzzle.SetActive (false);
    puzzle_gambar.SetActive (true);
}

// puzzle lagu
public void mainPuzzleLaguClicked()
{
    main_puzzle.SetActive(false);
    puzzle_lagu1.SetActive(true);
}

// public void kembaliMainPuzzleGmb() {
//     main_puzzle.SetActive (true);
//     puzzle_gambar.SetActive (false);
// }

// quiz video
public void mainQuizVideoClicked()
{
    main_puzzle.SetActive(false);
    quiz_video.SetActive(true);
}

public void mainPuzzleVideo1Clicked()
{
    main_puzzle.SetActive(false);
    puzzle_video1.SetActive(true);
}

// quiz lagu
public void mainQuizLaguClicked()
{
    main_puzzle.SetActive(false);
    quiz_lagu.SetActive(true);
}

// alat musik
public void alatMusikClicked() {
    belajar.SetActive (false);
    alat_musik.SetActive (true);
}
```

```
public void kembaliAlatMusik() {
    belajar.SetActive (true);
    alat_musik.SetActive (false);
}

// alat musik gendang
public void gendang1Clicked() {
    alat_musik.SetActive (false);
    gendang1.SetActive (true);
   Debug.Log("[Debug] gendang diaktifkan di sini.");
}

public void kembaliGendang() {
    alat_musik.SetActive (true);
    gendang1.SetActive (false);
}

// alat musik kecapi
public void kecapiClicked() {
    alat_musik.SetActive (false);
    kecapi.SetActive (true);
}

public void kembaliKecapi() {
    alat_musik.SetActive (true);
    kecapi.SetActive (false);
}

//lagu_daerah
public void laguDaerahClicked() {
    belajar.SetActive (false);
    lagu_daerah.SetActive (true);
}

public void kembaliLaguDaerah() {
    belajar.SetActive (true);
    lagu_daerah.SetActive (false);
}


//lagu_daerah indo_logo
public void indoLogoClicked() {
    lagu_daerah.SetActive (false);
    indo_logo.SetActive (true);
}

public void kembaliIndoLogo() {
    lagu_daerah.SetActive (true);
    indo_logo.SetActive (false);
}

//lagu_daerah alosi
public void alosiClicked() {
    lagu_daerah.SetActive (false);
    alosi.SetActive (true);
}

public void kembaliAlosi() {
    lagu_daerah.SetActive (true);
    alosi.SetActive (false);
}

//lagu_daerah bulu_alauna
public void buluAlaunaClicked() {
    lagu_daerah.SetActive (false);
    bulu_alauna.SetActive (true);
}

public void kembaliBuluAlauna() {
    lagu_daerah.SetActive (true);
    bulu_alauna.SetActive (false);
```

```
}

// tarian
public void tarianClicked() {
    belajar.SetActive (false);
    tarian.SetActive (true);
}

public void kembaliTarian() {
    belajar.SetActive (true);
    tarian.SetActive (false);
}

// tarian pajoge
public void tarianPajogeClicked()
{
    tarian.SetActive(false);
    tari_pajoge.SetActive(true);
}

public void kembaliTariPajoge()
{
    tarian.SetActive(true);
    tari_pajoge.SetActive(false);
}

// tarian panjaga
public void tarianPanjagaClicked()
{
    tarian.SetActive(false);
    tari_panjaga.SetActive(true);
}

public void kembaliTariPanjaga()
{
    tarian.SetActive(true);
    tari_panjaga.SetActive(false);
}

// tarian paddupa
public void tarianPaddupaClicked()
{
    tarian.SetActive(false);
    tari_paddupa.SetActive(true);
}

public void kembaliTariPaddupa()
{
    tarian.SetActive(true);
    tari_paddupa.SetActive(false);
}

//baju_adat
public void bajuAdatClicked() {
    belajar.SetActive (false);
    bodo.SetActive (true);
}

//bodo
public void kembaliBodo() {
    belajar.SetActive (true);
    bodo.SetActive (false);
}

//senjata
public void senjataClicked() {
    belajar.SetActive (false);
    badik.SetActive (true);
}

//badik
public void kembaliBadik() {
```

```
    belajar.SetActive (true);
    badik.SetActive (false);
}

// rumah adat
public void rumahAdatClicked() {
    belajar.SetActive (false);
    rumah_adat.SetActive (true);
}

public void kembaliRumahAdat() {
    belajar.SetActive (true);
    rumah_adat.SetActive (false);
}

public void rumah_bone_clicked() {
    belajar.SetActive (false);
    rumah_bone.SetActive (true);
}

public void kembaliRumahBone() {
    belajar.SetActive (true);
    rumah_bone.SetActive (false);
}

public void rumah_wajo_clicked() {
    belajar.SetActive (false);
    rumah_wajo.SetActive (true);
}

public void kembaliRumahWajo() {
    belajar.SetActive (true);
    rumah_wajo.SetActive (false);
}

public void rumah_bulukumba_clicked() {
    belajar.SetActive (false);
    rumah_bulukumba.SetActive (true);
}

public void kembaliRumahBulukumba() {
    belajar.SetActive (true);
    rumah_bulukumba.SetActive (false);
}

public void rumah_parepare_clicked() {
    belajar.SetActive (false);
    rumah_parepare.SetActive (true);
}

public void kembaliRumahParepare() {
    belajar.SetActive (true);
    rumah_parepare.SetActive (false);
}

public void rumah_sinjai_clicked() {
    belajar.SetActive (false);
    rumah_sinjai.SetActive (true);
}

public void kembaliRumahSinjai() {
    belajar.SetActive (true);
    rumah_sinjai.SetActive (false);
}

public void rumah_barru_clicked() {
    belajar.SetActive (false);
    rumah_barru.SetActive (true);
}

public void kembaliRumahBarru() {
    belajar.SetActive (true);
```

```
    rumah_barru.SetActive (false);
}

public void rumah_makassar_clicked() {
    belajar.SetActive (false);
    rumah_makassar.SetActive (true);
}

public void kembaliRumahMakassar() {
    belajar.SetActive (true);
    rumah_makassar.SetActive (false);
}

public void rumah_pinrang_clicked() {
    belajar.SetActive (false);
    rumah_pinrang.SetActive (true);
}

public void kembaliRumahPinrang() {
    belajar.SetActive (true);
    rumah_pinrang.SetActive (false);
}

public void rumah_luwu_clicked() {
    belajar.SetActive (false);
    rumah_luwu.SetActive (true);
}

public void kembaliRumahLuwu() {
    belajar.SetActive (true);
    rumah_luwu.SetActive (false);
}

public void rumah_soppeng_clicked() {
    belajar.SetActive (false);
    rumah_soppeng.SetActive (true);
    }

    public void kembaliRumahSoppeng() {
        belajar.SetActive (true);
        rumah_soppeng.SetActive (false);
    }

    public void rumah_sidrap_clicked() {
        belajar.SetActive (false);
        rumah_sidrap.SetActive (true);
    }

    public void kembaliRumahSidrap() {
        belajar.SetActive (true);
        rumah_sidrap.SetActive (false);
    }

    //rumah pare
    public void keluarRumahPareClicked() {
        belajar.SetActive (true);
        rumah_parepare.SetActive (false);
    }

    public void kembaliRumahPare() {
        rumah_adat.SetActive (true);
        rumah_parepare.SetActive (false);
    }

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

    }
}
using UnityEngine;
```

```
public class PuzzleDropZone : MonoBehaviour
{
    public float snapThreshold = 50f; // Jarak maksimal untuk snap

    public RectTransform FindNearestEmptySlot(RectTransform droppedPiece)
    {
        RectTransform[] slots = GetComponentsInChildren<RectTransform>();
        RectTransform nearestSlot = null;
        float minDistance = float.MaxValue;

        foreach (RectTransform slot in slots)
        {
            // Abaikan slot grid itu sendiri atau slot yang sudah terisi
            if (slot == (RectTransform)this.transform || slot.childCount > 0)
            {
                continue;
            }

            // Validasi apakah slot berada di dalam panel
            Vector2 slotPosition = slot.anchoredPosition;
            Debug.Log($"[FindNearestSlot] Checking Slot {slot.name}, Anchored Position: {slotPosition}");

            // Hitung jarak ke slot
            float distance = Vector2.Distance(droppedPiece.position, slot.position);

            if (distance < minDistance && distance < snapThreshold)
            {
                minDistance = distance;
                nearestSlot = slot;
            }
        }
        if (nearestSlot != null)
        {
            Debug.Log($"[FindNearestSlot] Final nearest slot: {nearestSlot.name}, Anchored Position: {nearestSlot.anchoredPosition}");
        }
        else
        {
            Debug.Log("[FindNearestSlot] No valid slot found.");
        }

        return nearestSlot;
    }
}
using UnityEngine;
using UnityEngine.UI;
using System.Linq;
using System.Collections; // Tambahkan ini
using UnityEngine.Video;

public class PuzzleManager : MonoBehaviour
{
    public RectTransform panelPuzzleArea; // Panel utama untuk menyusun puzzle
    public RectTransform[] puzzlePieces; // Semua kepingan puzzle
    public RectTransform[] slots; // Semua slot di area puzzle
    // public int rows = 3; // Jumlah baris grid
    // public int columns = 3; // Jumlah kolom grid
    public int rows; // Jumlah baris grid (diatur dinamis)
    public int columns; // Jumlah kolom grid (diatur dinamis)
    private Vector2[] targetPositions; // Posisi target untuk setiap kepingan puzzle
    public Vector3[] slotCenters; // Titik tengah semua slot
```

```
    public TimerManager timerManager; // Referensi ke TimerManager

    public Button startButton; // Tombol Start

    public Button restartButton; // Tombol Restart

    public Button nextLevelButton; // Tombol Next Level

    public Button succesButton; // Tombol success

    public Text tx_waktuHabis;

   // private int correctCounter = 0; // Counter untuk jumlah puzzle yang terpasang benar

    public Text tx_deskpuzzle; // Referensi ke teks

    public RectTransform panelKepingan; // Panel untuk kepingan puzzle

    public GameObject panelKonfirmasi; // Assign panel_konfirmasi di Inspector

    // public Button kembali; // Tombol success

    public GameObject main_puzzle; // Referensi ke panel menu utama

    public GameObject puzzle_gambar; // Referensi ke panel puzzle gambar

    public GameObject puzzle_gambar2; // Panel puzzle gambar kedua

    public GameObject puzzle_gambar3; // Panel puzzle gambar ketiga

    // public GameObject gendang; // Panel puzzle gambar ketiga

    public bool isGameStarted = false; // Status apakah permainan sudah dimulai



    //untuk puzzle video/lagu

    public Button closePanelVideo, btVidPlay1, btVidPlay2, btVidPlay3, btVidPlay4;

    public GameObject puzzleVideo1, panelVideo1, panelVideo2, panelVideo3, panelVideo4, panelVideoFull; // Assign panel_konfirmasi di Inspector

    public Text tx_puzvid; // Referensi ke teks

    public VideoPlayer videoPlayer1, videoPlayer2, videoPlayer3, videoPlayer4, videoFull;

    //public GameObject slot1, slot2, slot3, slot4, puzzle1, puzzle2, puzzle3, puzzle4;

    public GameObject puzzleLagu;

    //untuk puzzle lagu

    public AudioSource lagu1, lagu2, lagu3, lagu4, laguFull; // AudioSource untuk memutar lagu

    public GameObject[] allPanels;



    void Start()

    {

        InitializeSlotsAndPieces();

        GenerateGrid();

        RandomizePuzzlePieces();



       // Pastikan tombol restart dan next level tidak aktif di awal

        restartButton.gameObject.SetActive(false);

        nextLevelButton.gameObject.SetActive(false);

        succesButton.gameObject.SetActive(false);

        tx_deskpuzzle.gameObject.SetActive(false);

        tx_waktuHabis.gameObject.SetActive(false);



        // Pasang event listener untuk timer

        //  timerManager.OnTimerComplete += HandleTimerComplete;



        // Tombol start memulai permainan

        startButton.onClick.AddListener(StartGame);

        // Tombol restart memulai ulang permainan

       restartButton.onClick.AddListener(RestartGame);

        // Tombol next level (dapat diimplementasikan nanti)


nextLevelButton.onClick.AddListener(NextLevel);



        // Pastikan panel konfirmasi tidak terlihat saat puzzle dimulai

        if (panelKonfirmasi != null)

        {

            panelKonfirmasi.SetActive(false);

        }
```

```
        //// Cari objek di scene berdasarkan namanya
        //puzzleVideo1 = GameObject.Find("PanelVideo1");

        //if (puzzleVideo1 == null)
        //{
        //    Debug.LogError("PanelVideo1 tidak ditemukan! Pastikan namanya benar.");
        //}

        //untuk puzzle video
        if (puzzleVideo1 != null && puzzleVideo1.activeSelf)
        {
          closePanelVideo.gameObject.SetActive(false);
          btVidPlay1.gameObject.SetActive(true);
          btVidPlay2.gameObject.SetActive(true);
          btVidPlay3.gameObject.SetActive(true);
          btVidPlay4.gameObject.SetActive(true);
          tx_puzvid.gameObject.SetActive(true);
          panelVideo1.SetActive(false);
          panelVideo2.SetActive(false);
          panelVideo3.SetActive(false);
          panelVideo4.SetActive(false);
          panelVideoFull.SetActive(false);

         // Tambahkan event untuk menutup panel saat video selesai
          if (videoPlayer1 != null) videoPlayer1.loopPointReached += (vp) => ClosePanel(panelVideo1);
          if (videoPlayer2 != null) videoPlayer2.loopPointReached += (vp) => ClosePanel(panelVideo2);
          if (videoPlayer3 != null) videoPlayer3.loopPointReached += (vp) => ClosePanel(panelVideo3);
          if (videoPlayer4 != null) videoPlayer4.loopPointReached += (vp) => ClosePanel(panelVideo4);
        } else
        {
          Debug.LogError("puzzleVideo1 tidak ditemukan! Pastikan namanya benar.");
        }

        //untuk puzzle video/lagu
       if (puzzleLagu != null && puzzleLagu.activeSelf)
        {
          closePanelVideo.gameObject.SetActive(false);
          btVidPlay1.gameObject.SetActive(true);
          btVidPlay2.gameObject.SetActive(true);
          btVidPlay3.gameObject.SetActive(true);
          btVidPlay4.gameObject.SetActive(true);
          tx_puzvid.gameObject.SetActive(true);
          panelVideo1.SetActive(false);
          panelVideo2.SetActive(false);
          panelVideo3.SetActive(false);
          panelVideo4.SetActive(false);
          panelVideoFull.SetActive(false);
          lagu1.gameObject.SetActive(false);
          lagu2.gameObject.SetActive(false);
          lagu3.gameObject.SetActive(false);
          lagu4.gameObject.SetActive(false);
          laguFull.gameObject.SetActive(false);
        }
        else
        {
          Debug.LogError("puzzleLagu tidak ditemukan! Pastikan namanya benar.");
        }

    }

    void InitializeSlotsAndPieces()
    {
```

```
        // Cari semua slot dari panelPuzzleArea
        RectTransform[] children = panelPuzzleArea.GetComponentsInChildren<RectTransform>();
        slots = children.Where(slot => slot != panelPuzzleArea).ToArray();

        Debug.Log($"[InitializeSlotsAndPieces] Jumlah slot ditemukan: {slots.Length}");

        // Cari semua puzzle pieces dari panel tertentu
        GameObject panelKepinganObj = GameObject.Find("panel_kepingan");
        if (panelKepinganObj != null)
        {
            panelKepingan = panelKepinganObj.GetComponent<RectTransform>();
            RectTransform[] pieces = panelKepingan.GetComponentsInChildren<RectTransform>();
            puzzlePieces = pieces.Where(piece => piece != panelKepingan.GetComponent<RectTransform>()).ToArray();

            Debug.Log($"[InitializeSlotsAndPieces] Jumlah puzzle pieces ditemukan: {puzzlePieces.Length}");
        }
        else
        {
            Debug.LogError("[InitializeSlotsAndPieces] panel_kepingan tidak ditemukan.");
        }
    }

    void Awake()
    {
        for (int i = 0; i < slots.Length; i++)
        {
            Debug.Log($"[Inisialisasi] Slot {slots[i].name} di urutan {i}. Diharapkan puzzle: {puzzlePieces[i].name}");
        }
    }

    public void StartGame()
    {
        startButton.gameObject.SetActive(false);      // Sembunyikan tombol start
        restartButton.gameObject.SetActive(true);     // Tampilkan tombol restart
        //transform.SetAsLastSibling(); // Pastikan tombol Start berada di lapisan paling depan
        timerManager.StartTimer(); // Jalankan timer
        tx_waktuHabis.gameObject.SetActive(false);
        isGameStarted = true; // Permainan dimulai

        //if (puzzleLagu.activeSelf)
        //{
        //    lagu1.gameObject.SetActive(true);
        //}
    }

    public void RestartGame()
    {
        // Pindahkan semua kepingan ke panel_kepingan
        foreach (var piece in puzzlePieces)
        {
            piece.SetParent(panelKepingan);
        }

        RandomizePuzzlePieces(); // Acak ulang puzzle
        timerManager.ResetTimer(); // Reset timer
        startButton.gameObject.SetActive(true);       // Tampilkan tombol start
        restartButton.gameObject.SetActive(false);    // Sembunyikan tombol restart
        nextLevelButton.gameObject.SetActive(false); // Sembunyikan tombol next level
        tx_waktuHabis.gameObject.SetActive(false);
        tx_deskpuzzle.gameObject.SetActive(false);
```

```
        isGameStarted = false;
    }

    public void NextLevel()
    {
        // Debug.Log("Pindah ke level berikutnya."); // Tambahkan logika untuk pindah level
        RestartGame();

        if (puzzle_gambar.activeSelf)
        {
            // Pindah ke level berikutnya
            puzzle_gambar.SetActive(false);
            puzzle_gambar2.SetActive(true);
        }
        else if (puzzle_gambar2.activeSelf)
        {
            puzzle_gambar2.SetActive(false);
            puzzle_gambar3.SetActive(true);
        }
        else if (puzzle_gambar3.activeSelf)
        {
            // Jika sudah di level terakhir, sembunyikan tombol Next
            // Debug.Log("Ini adalah level terakhir.");
            nextLevelButton.gameObject.SetActive(false);
        }

    }

    void GenerateGrid()
    {
        int gridSize = rows * columns;
        if (puzzlePieces.Length != gridSize || slots.Length != gridSize)
        {
            Debug.LogError("[GenerateGrid] Jumlah puzzle pieces atau slots tidak sesuai dengan grid size.");
            return;
        }
    }

    void RandomizePuzzlePieces()
    {

        if (panelKepingan == null)
        {
            Debug.LogError("[RandomizePuzzlePieces] panel_kepingan belum diinisialisasi.");
            return;
        }

        foreach (var piece in puzzlePieces)
        {
            // Pastikan setiap kepingan berada di panel_kepingan
            if (piece.parent != panelKepingan)
            {
                piece.SetParent(panelKepingan);
            }
            // Hitung batas area acak berdasarkan ukuran panel_kepingan
            float minX = -panelKepingan.rect.width / 2 + piece.rect.width / 2;
            float maxX = panelKepingan.rect.width / 2 - piece.rect.width / 2;
            float minY = -panelKepingan.rect.height / 2 + piece.rect.height / 2;
            float maxY = panelKepingan.rect.height / 2 - piece.rect.height / 2;

            // Atur posisi acak di dalam batas panel_kepingan
            piece.anchoredPosition = new Vector2(
                Random.Range(minX, maxX),
                Random.Range(minY, maxY)
```

```
            );
        }
    }

    public Vector2 GetTargetPosition(int index)
    {
        if (index < 0 || index >= targetPositions.Length)
        {
            Debug.LogError($"Index {index} di luar batas targetPositions.");
            return Vector2.zero; // Kembalikan nilai default jika index tidak valid
        }

        return targetPositions[index];
    }

    public bool CheckPuzzleState()
    {

        int correctCounter = 0; // Counter untuk jumlah puzzle yang benar

        Debug.Log("[CheckPuzzleState] Memulai validasi puzzle...");

        for (int i = 0; i < slots.Length; i++)
        {
            RectTransform slot = slots[i];

            if (slot.childCount > 0)
            {
                RectTransform child = slot.GetChild(0) as RectTransform;

                if (child == puzzlePieces[i]) // Cek apakah puzzle piece sesuai
                {
                    correctCounter++;
                    Debug.Log($"[CheckPuzzleState] Slot {slot.name} berisi puzzle {child.name} dan sudah benar.");
                }
            }
        }

        // Cek apakah semua puzzle sudah di tempat yang benar
        if (correctCounter == slots.Length)
        {
            Debug.Log("[CheckPuzzleState] Semua puzzle sudah benar! Puzzle selesai!");
            // Tambahkan logika lain, misalnya memunculkan tombol "Next Level"
            // succesButton.gameObject.SetActive(true);

            StartCoroutine(ShowSuccessThenDescription()); // Jalankan coroutine
            if (timerManager != null)
            {
                timerManager.StopTimer(); // Hentikan timer
            }
            return true; // Puzzle selesai
        }
        else
        {
            return false; // Puzzle belum selesai
            Debug.Log($"[CheckPuzzleState] {correctCounter} dari {slots.Length} puzzle sudah benar.");
        }

    }

    // Coroutine untuk menampilkan succesButton lalu mengganti dengan tx_deskripsi
    private IEnumerator ShowSuccessThenDescription()
    {
        succesButton.gameObject.SetActive(true); // Tampilkan tombol sukses
```

```
        yield return new WaitForSeconds(2f); // Tunggu beberapa detik

        succesButton.gameObject.SetActive(false); // Sembunyikan tombol sukses

        if (puzzleVideo1 != null && puzzleVideo1.activeSelf)

        {

            Debug.Log("masuk ke else1!");

            CloseAllPanels();

            panelVideoFull.SetActive(true);

            //if (videoFull != null) videoFull.loopPointReached += (vp) => ClosePanel(panelVideoFull);

           closePanelVideo.gameObject.SetActive(true);

            videoFull.Play();

        }

        else if (puzzleLagu != null && puzzleLagu.activeSelf)

        {

            Debug.Log("masuk ke else2!");

            laguFull.gameObject.SetActive(true);

           closePanelVideo.gameObject.SetActive(true);

            laguFull.Play();

        }

        else

        {

            Debug.Log("masuk ke else!");

          tx_deskpuzzle.gameObject.SetActive(true); // Tampilkan deskripsi

            nextLevelButton.gameObject.SetActive(true);

        }

    }


    public void kembaliMainPuzzleGmb()

    {


        // Tampilkan panel konfirmasi keluar

        if (panelKonfirmasi != null)

        {

            Debug.Log($"Panel Konfirmasi ditemukan: {panelKonfirmasi.name}");

            panelKonfirmasi.SetActive(true);

        }

        else

        {

            Debug.LogError("Panel Konfirmasi tidak ditemukan!");

        }


    }


    private void DebugActivePanels()

    {

        GameObject[] rootObjects = UnityEngine.SceneManagement.SceneManager.GetActiveScene().GetRootGameObjects();

        Debug.Log("Panels yang aktif di scene:");

        foreach (var obj in rootObjects)

        {

            if (obj.activeSelf)

            {

                Debug.Log($"Panel Aktif: {obj.name}");

                CheckChildObjects(obj);

            }

        }

    }


    void CheckChildObjects(GameObject parent)

    {

        foreach (Transform child in parent.transform)

        {

            if (child.gameObject.activeSelf)

            {

                Debug.Log($"-- Child Active: {child.gameObject.name}");

                CheckChildObjects(child.gameObject); // Rekursif untuk anak-anak

            }
```

```
    }
}

//untuk puzzle video
public void jalankanVideo1()
{
    if (isGameStarted)
    {
        CloseAllPanels();
        panelVideo1.SetActive(true);
        videoPlayer1.Play();
       closePanelVideo.gameObject.SetActive(true);
}
    else
    {
        Debug.Log("isGameStarted= false");
    }
}

//untuk puzzle video
public void jalankanVideo2()
{
    if (isGameStarted)
    {
        CloseAllPanels();
        panelVideo2.SetActive(true);
        videoPlayer2.Play();
      closePanelVideo.gameObject.SetActive(true);
    }
    else
    {
        Debug.Log("isGameStarted= false");
    }
}

//untuk puzzle video
public void jalankanVideo3()
{
    if (isGameStarted)
    {
        CloseAllPanels();
        panelVideo3.SetActive(true);
        videoPlayer3.Play();
      closePanelVideo.gameObject.SetActive(true);
    }
    else
    {
        Debug.Log("isGameStarted= false");
    }
}

//untuk puzzle video
public void jalankanVideo4()
{
    if (isGameStarted)
    {
        CloseAllPanels();
        panelVideo4.SetActive(true);
        videoPlayer4.Play();
      closePanelVideo.gameObject.SetActive(true);
    }
    else
    {
        Debug.Log("isGameStarted = false");
    }

}

// Menutup semua panel
public void CloseAllPanels()
{
    closePanelVideo.gameObject.SetActive(false);
```

```
        Debug.Log("masuk close all panel");


        if (puzzleVideo1 != null && puzzleVideo1.activeSelf)
        {
            Debug.Log("puzzleVideo aktif close all panels");
            panelVideo1.SetActive(false);
            panelVideo2.SetActive(false);
            panelVideo3.SetActive(false);
            panelVideo4.SetActive(false);
            panelVideoFull.SetActive(false);
            videoPlayer1.Stop();
            videoPlayer2.Stop();
            videoPlayer3.Stop();
            videoPlayer4.Stop();
            videoFull.Stop();
        }


        if (puzzleLagu != null && puzzleLagu.activeSelf)
        {
            Debug.Log("puzzleLagu aktif close all panels");
            //if (panelVideo1.activeSelf)
            //{
            //    Debug.Log("PanelVideo1 aktif");
            //    panelVideo1.SetActive(false);
            //}
            //if (panelVideo2.activeSelf)
            //{
            //    Debug.Log("PanelVideo2 aktif");
            //    panelVideo2.SetActive(false);
            //}
            //if (panelVideo3.activeSelf)
            //{
            //    Debug.Log("PanelVideo3 aktif");
            //    panelVideo3.SetActive(false);
            //}
            //if (panelVideo4.activeSelf)
            //{
            //    Debug.Log("PanelVideo4 aktif");
            //    panelVideo4.SetActive(false);
            //}
            //if (panelVideoFull.activeSelf)
            //{
            //    Debug.Log("PanelVideoFull aktif");
            //    panelVideoFull.SetActive(false);
            //}
            panelVideo1.SetActive(false);
            panelVideo2.SetActive(false);
            panelVideo3.SetActive(false);
            panelVideo4.SetActive(false);
            panelVideoFull.SetActive(false);
            laguFull.gameObject.SetActive(false);
            lagu1.gameObject.SetActive(false);
            lagu2.gameObject.SetActive(false);
            lagu3.gameObject.SetActive(false);
            lagu4.gameObject.SetActive(false);
            lagu1.Stop();
            lagu2.Stop();
            lagu3.Stop();
            lagu4.Stop();
            laguFull.Stop();
        }
        closePanelVideo.gameObject.SetActive(false);

    }

    // Menutup panel tertentu
    private void ClosePanel(GameObject panel)
    {
        panel.SetActive(false);
    }
```

```
    //untuk puzzle lagu

    public void jalankanLagu1()

    {

        if (puzzleLagu!=null && puzzleLagu.activeSelf
&& isGameStarted)

        {

            //CloseAllPanels();

            closePanelVideo.gameObject.SetActive(true);

            lagu1.gameObject.SetActive(true);

            lagu1.Play();

            // Tambahkan logika untuk menutup panel
setelah lagu selesai

            Invoke(nameof(ClosePanelAfterAudio),
lagu1.clip.length);

        }

        else

        {

            Debug.Log("isGameStarted= false");

        }

    }


    public void jalankanLagu2()

    {

        if (isGameStarted)

        {

            closePanelVideo.gameObject.SetActive(true);

            //CloseAllPanels();

            lagu2.gameObject.SetActive(true);

            lagu2.Play();

            // Tambahkan logika untuk menutup panel
setelah lagu selesai

            Invoke(nameof(ClosePanelAfterAudio),
lagu2.clip.length);

        }

        else

        {

            Debug.Log("isGameStarted= false");

        }

    }


    public void jalankanLagu3()

    {

        if (isGameStarted)

        {

            closePanelVideo.gameObject.SetActive(true);

            //CloseAllPanels();

            lagu3.gameObject.SetActive(true);

            lagu3.Play();

            // Tambahkan logika untuk menutup panel
setelah lagu selesai

            Invoke(nameof(ClosePanelAfterAudio),
lagu3.clip.length);

        }

        else

        {

            Debug.Log("isGameStarted= false");

        }

    }


    public void jalankanLagu4()

    {

        if (isGameStarted)

        {

            closePanelVideo.gameObject.SetActive(true);

            //CloseAllPanels();

            lagu4.gameObject.SetActive(true);

            lagu4.Play();

            // Tambahkan logika untuk menutup panel
setelah lagu selesai

            Invoke(nameof(ClosePanelAfterAudio),
lagu4.clip.length);

        }

        else

        {

            Debug.Log("isGameStarted= false");

        }
```

```
    }

    public void jalankanLaguFull()
    {
        if (isGameStarted)
        {
            closePanelVideo.gameObject.SetActive(true);
            //CloseAllPanels();
            laguFull.gameObject.SetActive(true);
            laguFull.Play();
            // Tambahkan logika untuk menutup panel setelah lagu selesai
            Invoke(nameof(ClosePanelAfterAudio), laguFull.clip.length);
        }
        else
        {
            Debug.Log("isGameStarted= false");
        }
    }

    void ClosePanelAfterAudio()
    {
        CloseAllPanels();
       closePanelVideo.gameObject.SetActive(false); // Nonaktifkan panel setelah lagu selesai
    }

}
    using UnityEngine;
    using UnityEngine.EventSystems;

    public class PuzzlePiece : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler
    {
        public RectTransform rectTransform;
        private Canvas canvas;
        private CanvasGroup canvasGroup;
        private RectTransform originalParent;
        public RectTransform puzzleAreaPanel; // Panel puzzle area

        private RectTransform currentSlot = null; // Slot tempat puzzle piece saat ini berada
        public PuzzleManager puzzleManager; // Referensi ke PuzzleManager

        private void Awake()
        {
            rectTransform = GetComponent<RectTransform>();
            canvas = GetComponentInParent<Canvas>();
            canvasGroup = GetComponent<CanvasGroup>();
            originalParent = rectTransform.parent.GetComponent<RectTransform>();
        }

        public void OnBeginDrag(PointerEventData eventData)
        {
            if (!puzzleManager.isGameStarted)
            {
                Debug.Log("Permainan belum dimulai. Puzzle tidak bisa dipindahkan.");
                return; // Abaikan interaksi jika permainan belum dimulai
            }

            if (puzzleManager.timerManager != null && puzzleManager.timerManager.isTimeUp)
            {
                Debug.Log("Waktu habis. Puzzle tidak bisa dipindahkan.");
                return; // Abaikan interaksi jika waktu sudah habis
            }
```

```
        canvasGroup.alpha = 0.6f;
        canvasGroup.blocksRaycasts = false;
        // Lepaskan dari slot saat mulai drag
        if (currentSlot != null)
        {
            currentSlot.DetachChildren(); // Hapus puzzle dari slot
            currentSlot = null; // Kosongkan referensi slot
        }

        // Ubah parent ke panel_kepingan untuk membebaskan drag
        rectTransform.SetParent(originalParent, true); // originalParent adalah panel_kepingan
        // Debug.Log($"[OnBeginDrag] {gameObject.name} parent set to {rectTransform.parent.name}");

    }

    public void OnDrag(PointerEventData eventData)
    {
        if (!puzzleManager.isGameStarted)
        {
            return; // Abaikan interaksi jika permainan belum dimulai
        }
        rectTransform.anchoredPosition += eventData.delta / canvas.scaleFactor;
    }

    private RectTransform FindNearestSlot(RectTransform parentPanel)
    {
        float minDistance = float.MaxValue;
        RectTransform nearestSlot = null;

        foreach (RectTransform slot in parentPanel.GetComponentsInChildren<RectTransform>())
        {
            // Lewati parent panel dan slot lama
            if (slot == parentPanel || slot == rectTransform.parent) continue;

            // Hitung posisi tengah slot
            Vector2 slotCenter = slot.TransformPoint(slot.rect.center);

            // Hitung jarak antara puzzle piece dan slot
            float distance = Vector2.Distance(rectTransform.position, slotCenter);
            // Debug.Log($"[FindNearestSlot] Checking Slot {slot.name}, Distance: {distance}");

            if (distance < minDistance)
            {
                minDistance = distance;
                nearestSlot = slot;
            }
        }

        if (nearestSlot != null)
        {
            // Debug.Log($"[FindNearestSlot] Nearest Slot Found: {nearestSlot.name}, Distance: {minDistance}");
        }
        else
        {
            // Debug.LogWarning($"[FindNearestSlot] No valid slot found for {gameObject.name}");
        }

        return nearestSlot;
    }
```

```
    public void OnEndDrag(PointerEventData eventData)
    {
        if (!puzzleManager.isGameStarted)
        {
            Debug.Log("Permainan belum dimulai. Puzzle tidak bisa dipindahkan.");
            return; // Abaikan interaksi jika permainan belum dimulai
        }

        canvasGroup.alpha = 1f; // Kembalikan transparansi
        canvasGroup.blocksRaycasts = true;

        RectTransform nearestSlot = FindNearestSlot(puzzleAreaPanel);

        if (nearestSlot != null)
        {
            // Periksa apakah slot baru sudah kosong
            if (nearestSlot.childCount > 0)
            {
                // Debug.LogWarning($"[OnEndDrag] {nearestSlot.name} is already occupied.");
                // Kembalikan ke slot lama jika slot baru terisi
                if (currentSlot != null)
                {
                    rectTransform.SetParent(currentSlot, false);
                    rectTransform.anchoredPosition = Vector2.zero;
                }
                else
                {
                    rectTransform.SetParent(originalParent, true);
                    rectTransform.anchoredPosition = Vector2.zero;
                }
                return;
            }

            // Kosongkan slot lama
            if (currentSlot != null && currentSlot.childCount > 0)
            {
                currentSlot.DetachChildren(); // Hapus referensi dari slot lama
            }

            // Set parent ke slot terdekat
            rectTransform.SetParent(nearestSlot, false);
            rectTransform.anchoredPosition = Vector2.zero;

            // Debug.Log($"[OnEndDrag] {gameObject.name} snapped to Slot: {nearestSlot.name}");
            currentSlot = nearestSlot; // Simpan referensi ke slot saat ini
            // Panggil validasi di PuzzleManager
            // puzzleManager.CheckPuzzleState();

        }
        else
        {
            // Jika tidak ada slot valid, kembalikan ke panel asal
            if (currentSlot != null && currentSlot.childCount > 0)
            {
                currentSlot.DetachChildren(); // Hapus referensi dari slot lama
            }

            // Jika tidak ada slot yang valid, kembalikan ke parent asal
            rectTransform.SetParent(originalParent, true);
            rectTransform.anchoredPosition = Vector2.zero;
```

```
            // Debug.Log($"[OnEndDrag] {gameObject.name} returned to original position.");
        }

        //cek selesai atau tidak puzzle nya
        puzzleManager.CheckPuzzleState();

    }
}
using UnityEngine;

public class PuzzleRandomizer : MonoBehaviour
{
    public RectTransform[] puzzlePieces; // Semua kepingan puzzle
    public RectTransform panelKepingan; // Panel tempat kepingan puzzle
    public float padding = 10f; // Jarak minimum dari tepi panel

    void Start()
    {
        // Cari semua anak dari panel_kepingan yang merupakan kepingan puzzle
        GameObject panelKepinganGO = GameObject.Find("panel_kepingan");
        if (panelKepinganGO != null)
        {
            RectTransform[] pieces = panelKepinganGO.GetComponentsInChildren<RectTransform>();
            puzzlePieces = new RectTransform[pieces.Length - 1]; // Abaikan parent (panel_kepingan)
            for (int i = 1; i < pieces.Length; i++)
            {
                puzzlePieces[i - 1] = pieces[i];
            }
            // Debug.Log($"Jumlah kepingan puzzle yang ditemukan: {puzzlePieces.Length}");
        }
        else
        {
            // Debug.LogError("panel_kepingan tidak ditemukan.");
            return;
        }

        // Acak posisi kepingan puzzle
        RandomizePositions();
    }

    void RandomizePositions()
    {
        if (panelKepingan == null)
        {
            // Debug.LogError("PanelKepingan belum diassign di Inspector!");
            return;
        }

        // Dapatkan ukuran panel
        Vector2 panelSize = panelKepingan.rect.size;

        // Debug.Log($"Panel Size: {panelSize}");

        foreach (RectTransform piece in puzzlePieces)
        {
            if (panelSize == Vector2.zero)
            {
                // Debug.LogError("Ukuran panel_kepingan tidak valid!");
                return;
            }
```

```
            // Hitung posisi acak berdasarkan pusat panel (0,0)
            float randomX = Random.Range(
                -panelSize.x / 2 + piece.rect.width / 2 + padding, // Batas kiri
                panelSize.x / 2 - piece.rect.width / 2 - padding // Batas kanan
            );

            float randomY = Random.Range(
                -panelSize.y / 2 + piece.rect.height / 2 + padding, // Batas bawah
                panelSize.y / 2 - piece.rect.height / 2 - padding  // Batas atas
            );

            // Tetapkan posisi acak ke puzzle piece
            piece.anchoredPosition = new Vector2(randomX, randomY);

            // Debug posisi acak
            // Debug.Log($"Puzzle Piece {piece.name} - Anchored Position: {piece.anchoredPosition}");
        }
    }
}
using UnityEngine;
using UnityEngine.UI;
using System.Linq;
using System.Collections; // Tambahkan ini
using UnityEngine.Video;

public class PuzzleVideoManager : MonoBehaviour
{
    public RectTransform panelPuzzleArea; // Panel utama untuk menyusun puzzle
    public RectTransform[] puzzlePieces; // Semua kepingan puzzle
    public RectTransform[] slots; // Semua slot di area puzzle
    // public int rows = 3; // Jumlah baris grid
    // public int columns = 3; // Jumlah kolom grid
    public int rows; // Jumlah baris grid (diatur dinamis)
    public int columns; // Jumlah kolom grid (diatur dinamis)
    private Vector2[] targetPositions; // Posisi target untuk setiap kepingan puzzle
    public Vector3[] slotCenters; // Titik tengah semua slot

    public TimerManager timerManager; // Referensi ke TimerManager
    public Button startButton; // Tombol Start
    public Button restartButton; // Tombol Restart
    public Button nextLevelButton; // Tombol Next Level
    public Button succesButton; // Tombol success
    public Text tx_waktuHabis;
    // private int correctCounter = 0; // Counter untuk jumlah puzzle yang terpasang benar
    public Text tx_deskpuzzle; // Referensi ke teks
    public RectTransform panelKepingan; // Panel untuk kepingan puzzle
    public GameObject panelKonfirmasi; // Assign panel_konfirmasi di Inspector
    // public Button kembali; // Tombol success
    public GameObject main_puzzle; // Referensi ke panel menu utama
    public GameObject puzzle_gambar; // Referensi ke panel puzzle gambar
    public GameObject puzzle_gambar2; // Panel puzzle gambar kedua
    public GameObject puzzle_gambar3; // Panel puzzle gambar ketiga
    // public GameObject gendang; // Panel puzzle gambar ketiga
    public bool isGameStarted = false; // Status apakah permainan sudah dimulai

    public GameObject panelVideo1; // Panel untuk video 1
    public VideoPlayer videoPlayer1; // VideoPlayer untuk video 1
```

```
    public GameObject panelVideo2; // Panel untuk video 2
    public VideoPlayer videoPlayer2; // VideoPlayer untuk video 2

    public GameObject panelVideo3; // Panel untuk video 3
    public VideoPlayer videoPlayer3; // VideoPlayer untuk video 3

    public GameObject panelVideo4; // Panel untuk video 4
    public VideoPlayer videoPlayer4; // VideoPlayer untuk video 4


    void Start()
    {
        InitializeSlotsAndPieces();
        GenerateGrid();
        RandomizePuzzlePieces();

       // Pastikan tombol restart dan next level tidak aktif di awal
        restartButton.gameObject.SetActive(false);
        nextLevelButton.gameObject.SetActive(false);
        succesButton.gameObject.SetActive(false);
        tx_deskpuzzle.gameObject.SetActive(false);
        tx_waktuHabis.gameObject.SetActive(false);

        // Pasang event listener untuk timer
        // timerManager.OnTimerComplete += HandleTimerComplete;

        // Tombol start memulai permainan
        startButton.onClick.AddListener(StartGame);
        // Tombol restart memulai ulang permainan
       restartButton.onClick.AddListener(RestartGame);
        // Tombol next level (dapat diimplementasikan nanti)
nextLevelButton.onClick.AddListener(NextLevel);

        // Pastikan panel konfirmasi tidak terlihat saat puzzle dimulai
        if (panelKonfirmasi != null)
        {
            panelKonfirmasi.SetActive(false);
        }

        // Tambahkan event untuk menutup panel saat video selesai
        if (videoPlayer1 != null) videoPlayer1.loopPointReached += (vp) => ClosePanel(panelVideo1);
        if (videoPlayer2 != null) videoPlayer2.loopPointReached += (vp) => ClosePanel(panelVideo2);
        if (videoPlayer3 != null) videoPlayer3.loopPointReached += (vp) => ClosePanel(panelVideo3);
        if (videoPlayer4 != null) videoPlayer4.loopPointReached += (vp) => ClosePanel(panelVideo4);

    }

    void InitializeSlotsAndPieces()
    {
        // Cari semua slot dari panelPuzzleArea
        RectTransform[] children = panelPuzzleArea.GetComponentsInChildren<RectTransform>();
        slots = children.Where(slot => slot != panelPuzzleArea).ToArray();

        Debug.Log($"[InitializeSlotsAndPieces] Jumlah slot ditemukan: {slots.Length}");

        // Cari semua puzzle pieces dari panel tertentu
```

```
        GameObject panelKepinganObj = GameObject.Find("panel_kepingan");

        if (panelKepinganObj != null)

        {

            panelKepingan = panelKepinganObj.GetComponent<RectTransform>();

            RectTransform[] pieces = panelKepingan.GetComponentsInChildren<RectTransform>();

            puzzlePieces = pieces.Where(piece => piece != panelKepingan.GetComponent<RectTransform>()).ToArray();



            Debug.Log($"[InitializeSlotsAndPieces] Jumlah puzzle pieces ditemukan: {puzzlePieces.Length}");

        }

        else

        {

            Debug.LogError("[InitializeSlotsAndPieces] panel_kepingan tidak ditemukan.");

        }

    }



    void Awake()

    {

        for (int i = 0; i < slots.Length; i++)

        {

            Debug.Log($"[Inisialisasi] Slot {slots[i].name} di urutan {i}. Diharapkan puzzle: {puzzlePieces[i].name}");

        }



    }



    public void StartGame()

    {

        startButton.gameObject.SetActive(false);      // Sembunyikan tombol start

        restartButton.gameObject.SetActive(true);     // Tampilkan tombol restart

       //transform.SetAsLastSibling(); // Pastikan tombol Start berada di lapisan paling depan

        timerManager.StartTimer(); // Jalankan timer

        tx_waktuHabis.gameObject.SetActive(false);

        isGameStarted = true; // Permainan dimulai

    }



    public void RestartGame()

    {

      // Pindahkan semua kepingan ke panel_kepingan

        foreach (var piece in puzzlePieces)

        {

            piece.SetParent(panelKepingan);

        }



        RandomizePuzzlePieces(); // Acak ulang puzzle

        timerManager.ResetTimer(); // Reset timer

        startButton.gameObject.SetActive(true);       // Tampilkan tombol start

        restartButton.gameObject.SetActive(false);    // Sembunyikan tombol restart

       nextLevelButton.gameObject.SetActive(false); // Sembunyikan tombol next level

        tx_waktuHabis.gameObject.SetActive(false);

        tx_deskpuzzle.gameObject.SetActive(false);

    }



    public void NextLevel()

    {

        // Debug.Log("Pindah ke level berikutnya."); // Tambahkan logika untuk pindah level

        RestartGame();



        if (puzzle_gambar.activeSelf)

        {

            // Pindah ke level berikutnya

            puzzle_gambar.SetActive(false);

            puzzle_gambar2.SetActive(true);

        }

        else if (puzzle_gambar2.activeSelf)
```

```
        {
            puzzle_gambar2.SetActive(false);
            puzzle_gambar3.SetActive(true);
        }
        else if (puzzle_gambar3.activeSelf)
        {
            // Jika sudah di level terakhir, sembunyikan tombol Next
            // Debug.Log("Ini adalah level terakhir.");
            nextLevelButton.gameObject.SetActive(false);
        }

    }

    void GenerateGrid()
    {
        int gridSize = rows * columns;
        if (puzzlePieces.Length != gridSize || slots.Length != gridSize)
        {
            Debug.LogError("[GenerateGrid] Jumlah puzzle pieces atau slots tidak sesuai dengan grid size.");
            return;
        }
    }

    void RandomizePuzzlePieces()
    {

        if (panelKepingan == null)
        {
            Debug.LogError("[RandomizePuzzlePieces] panel_kepingan belum diinisialisasi.");
            return;
        }

        foreach (var piece in puzzlePieces)
        {
            // Pastikan setiap kepingan berada di panel_kepingan
            if (piece.parent != panelKepingan)
            {
                piece.SetParent(panelKepingan);
            }
            // Hitung batas area acak berdasarkan ukuran panel_kepingan
            float minX = -panelKepingan.rect.width / 2 + piece.rect.width / 2;
            float maxX = panelKepingan.rect.width / 2 - piece.rect.width / 2;
            float minY = -panelKepingan.rect.height / 2 + piece.rect.height / 2;
            float maxY = panelKepingan.rect.height / 2 - piece.rect.height / 2;

            // Atur posisi acak di dalam batas panel_kepingan
            piece.anchoredPosition = new Vector2(
                Random.Range(minX, maxX),
                Random.Range(minY, maxY)
            );
        }
    }

    public Vector2 GetTargetPosition(int index)
    {
        if (index < 0 || index >= targetPositions.Length)
        {
            Debug.LogError($"Index {index} di luar batas targetPositions.");
            return Vector2.zero; // Kembalikan nilai default jika index tidak valid
        }

        return targetPositions[index];
    }
```

```
    public bool CheckPuzzleState()
    {

        int correctCounter = 0; // Counter untuk jumlah puzzle yang benar

        // Debug.Log("[CheckPuzzleState] Memulai validasi puzzle...");

        for (int i = 0; i < slots.Length; i++)
        {
            RectTransform slot = slots[i];

            if (slot.childCount > 0)
            {
                RectTransform child = slot.GetChild(0) as RectTransform;

                if (child == puzzlePieces[i]) // Cek apakah puzzle piece sesuai
                {
                    correctCounter++;
                    // Debug.Log($"[CheckPuzzleState] Slot {slot.name} berisi puzzle {child.name} dan sudah benar.");
                }
            }
        }

        // Cek apakah semua puzzle sudah di tempat yang benar
        if (correctCounter == slots.Length)
        {
            Debug.Log("[CheckPuzzleState] Semua puzzle sudah benar! Puzzle selesai!");
            // Tambahkan logika lain, misalnya memunculkan tombol "Next Level"
            // succesButton.gameObject.SetActive(true);

            StartCoroutine(ShowSuccessThenDescription()); // Jalankan coroutine
            if (timerManager != null)
            {
                timerManager.StopTimer(); // Hentikan timer
            }
            return true; // Puzzle selesai
        }
        else
        {
            return false; // Puzzle belum selesai
            // Debug.Log($"[CheckPuzzleState] {correctCounter} dari {slots.Length} puzzle sudah benar.");
        }

    }

    // Coroutine untuk menampilkan succesButton lalu mengganti dengan tx_deskripsi
    private IEnumerator ShowSuccessThenDescription()
    {
        succesButton.gameObject.SetActive(true);    // Tampilkan tombol sukses
        yield return new WaitForSeconds(2f); // Tunggu beberapa detik
        succesButton.gameObject.SetActive(false);   // Sembunyikan tombol sukses
        tx_deskpuzzle.gameObject.SetActive(true);   // Tampilkan deskripsi
        nextLevelButton.gameObject.SetActive(true);
    }

    public void kembaliMainPuzzleGmb()
    {

        // Tampilkan panel konfirmasi keluar
        if (panelKonfirmasi != null)
        {
            Debug.Log($"Panel Konfirmasi ditemukan: {panelKonfirmasi.name}");
```

```
            panelKonfirmasi.SetActive(true);
        }
        else
        {
            Debug.LogError("Panel Konfirmasi tidak ditemukan!");
        }

    }

    private void DebugActivePanels()
    {
        GameObject[] rootObjects = UnityEngine.SceneManagement.SceneManager.GetActiveScene().GetRootGameObjects();
        Debug.Log("Panels yang aktif di scene:");
        foreach (var obj in rootObjects)
        {
            if (obj.activeSelf)
            {
                Debug.Log($"Panel Aktif: {obj.name}");
                CheckChildObjects(obj);
            }
        }
    }

    void CheckChildObjects(GameObject parent)
    {
        foreach (Transform child in parent.transform)
        {
            if (child.gameObject.activeSelf)
            {
                Debug.Log($"-- Child Active: {child.gameObject.name}");
                CheckChildObjects(child.gameObject); // Rekursif untuk anak-anak
            }
        }
    }

    //// Membuka panel video tertentu
    //public void OpenVideoPanel(int panelIndex)
    //{
    //    CloseAllVideoPanels(); // Pastikan semua panel lain tertutup

    //    if (panelIndex >= 0 && panelIndex < videoPanels.Length)
    //    {
    //        videoPanels[panelIndex].SetActive(true); // Aktifkan panel yang dipilih
    //    }
    //    else
    //    {
    //        Debug.LogError($"Panel index {panelIndex} tidak valid!");
    //    }
    //}

    //// Menutup semua panel video
    //public void CloseAllVideoPanels()
    //{
    //    foreach (GameObject panel in videoPanels)
    //    {
    //        if (panel != null)
    //        {
    //            panel.SetActive(false);
    //        }
    //    }
    //}

    // Metode untuk menampilkan panel video 1
    public void PlayVideo1()
    {
        CloseAllPanels();
        panelVideo1.SetActive(true);
```

```
    videoPlayer1.Play();
}

// Metode untuk menampilkan panel video 2
public void PlayVideo2()
{
    CloseAllPanels();
    panelVideo2.SetActive(true);
    videoPlayer2.Play();
}

// Metode untuk menampilkan panel video 3
public void PlayVideo3()
{
    CloseAllPanels();
    panelVideo3.SetActive(true);
    videoPlayer3.Play();
}

// Metode untuk menampilkan panel video 4
public void PlayVideo4()
{
    CloseAllPanels();
    panelVideo4.SetActive(true);
    videoPlayer4.Play();
}

// Menutup semua panel
private void CloseAllPanels()
{
    panelVideo1.SetActive(false);
    panelVideo2.SetActive(false);
    panelVideo3.SetActive(false);
    panelVideo4.SetActive(false);
}
    // Menutup panel tertentu
    private void ClosePanel(GameObject panel)
    {
        panel.SetActive(false);
    }

}
    using UnityEngine;
    using UnityEngine.EventSystems;

    public class PuzzleVideoPiece : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler
    {
        public RectTransform rectTransform;
        private Canvas canvas;
        private CanvasGroup canvasGroup;
        private RectTransform originalParent;
       public RectTransform puzzleAreaPanel; // Panel puzzle area

       private RectTransform currentSlot = null; // Slot tempat puzzle piece saat ini berada
        public PuzzleManager puzzleManager; // Referensi ke PuzzleManager

        private void Start()
        {
            if (puzzleManager == null)
            {
                Debug.LogError("PuzzleVideoPiece : PuzzleManager tidak terhubung! Pastikan script terhubung di Inspector.");
            }
        }

        private void Awake()
        {
```

```
        rectTransform = GetComponent<RectTransform>();

        canvas = GetComponentInParent<Canvas>();

        canvasGroup = GetComponent<CanvasGroup>();

        originalParent = rectTransform.parent.GetComponent<RectTransform>();

    }


    public void OnBeginDrag(PointerEventData eventData)

    {

        if (!puzzleManager.isGameStarted)

        {

            Debug.Log("Permainan belum dimulai. Puzzle tidak bisa dipindahkan.");

            return; // Abaikan interaksi jika permainan belum dimulai

        }


        if (puzzleManager.timerManager != null && puzzleManager.timerManager.isTimeUp)

        {

            Debug.Log("Waktu habis. Puzzle tidak bisa dipindahkan.");

            return; // Abaikan interaksi jika waktu sudah habis

        }


        canvasGroup.alpha = 0.6f;

        canvasGroup.blocksRaycasts = false;

        // Lepaskan dari slot saat mulai drag

        if (currentSlot != null)

        {

            currentSlot.DetachChildren(); // Hapus puzzle dari slot

            currentSlot = null; // Kosongkan referensi slot

        }

        // Ubah parent ke panel_kepingan untuk membebaskan drag

        rectTransform.SetParent(originalParent, true); // originalParent adalah panel_kepingan

        // Debug.Log($"[OnBeginDrag] {gameObject.name} parent set to {rectTransform.parent.name}");


    }


    public void OnDrag(PointerEventData eventData)

    {

        if (!puzzleManager.isGameStarted)

        {

            return; // Abaikan interaksi jika permainan belum dimulai

        }

        rectTransform.anchoredPosition += eventData.delta / canvas.scaleFactor;

    }


    private RectTransform FindNearestSlot(RectTransform parentPanel)

    {

        float minDistance = float.MaxValue;

        RectTransform nearestSlot = null;


        foreach (RectTransform slot in parentPanel.GetComponentsInChildren<RectTransform>())

        {

            // Lewati parent panel dan slot lama

            if (slot == parentPanel || slot == rectTransform.parent) continue;


            // Hitung posisi tengah slot

            Vector2 slotCenter = slot.TransformPoint(slot.rect.center);


            // Hitung jarak antara puzzle piece dan slot
```

```
            float distance = Vector2.Distance(rectTransform.position, slotCenter);

            // Debug.Log($"[FindNearestSlot] Checking Slot {slot.name}, Distance: {distance}");

            if (distance < minDistance)
            {
                minDistance = distance;
                nearestSlot = slot;
            }
        }

        if (nearestSlot != null)
        {
            // Debug.Log($"[FindNearestSlot] Nearest Slot Found: {nearestSlot.name}, Distance: {minDistance}");
        }
        else
        {
            // Debug.LogWarning($"[FindNearestSlot] No valid slot found for {gameObject.name}");
        }

        return nearestSlot;
    }

    public void OnEndDrag(PointerEventData eventData)
    {
        if (!puzzleManager.isGameStarted)
        {
            Debug.Log("Permainan belum dimulai. Puzzle tidak bisa dipindahkan.");
            return; // Abaikan interaksi jika permainan belum dimulai
        }

        canvasGroup.alpha = 1f; // Kembalikan transparansi
        canvasGroup.blocksRaycasts = true;

        RectTransform nearestSlot = FindNearestSlot(puzzleAreaPanel);

        if (nearestSlot != null)
        {
            // Periksa apakah slot baru sudah kosong
            if (nearestSlot.childCount > 0)
            {
                // Debug.LogWarning($"[OnEndDrag] {nearestSlot.name} is already occupied.");
                // Kembalikan ke slot lama jika slot baru terisi
                if (currentSlot != null)
                {
                    rectTransform.SetParent(currentSlot, false);
                    rectTransform.anchoredPosition = Vector2.zero;
                }
                else
                {
                    rectTransform.SetParent(originalParent, true);
                    rectTransform.anchoredPosition = Vector2.zero;
                }
                return;
            }

            // Kosongkan slot lama
            if (currentSlot != null && currentSlot.childCount > 0)
            {
                currentSlot.DetachChildren(); // Hapus referensi dari slot lama
            }

            // Set parent ke slot terdekat
```

```
            rectTransform.SetParent(nearestSlot, false);

            rectTransform.anchoredPosition = Vector2.zero;


            // Debug.Log($"[OnEndDrag] {gameObject.name} snapped to Slot: {nearestSlot.name}");

            currentSlot = nearestSlot; // Simpan referensi ke slot saat ini

            // Panggil validasi di PuzzleManager

            // puzzleManager.CheckPuzzleState();



        }

        else

        {

            // Jika tidak ada slot valid, kembalikan ke panel asal

            if (currentSlot != null && currentSlot.childCount > 0)

            {

                currentSlot.DetachChildren(); // Hapus referensi dari slot lama

            }



            // Jika tidak ada slot yang valid, kembalikan ke parent asal

            rectTransform.SetParent(originalParent, true);

            rectTransform.anchoredPosition = Vector2.zero;


            // Debug.Log($"[OnEndDrag] {gameObject.name} returned to original position.");

        }



        //cek selesai atau tidak puzzle nya

        puzzleManager.CheckPuzzleState();



    }

}

using System.Collections;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.Video;


public class QuizManager : MonoBehaviour

{

    [Header("Panel and Buttons")]

    public GameObject panelVideo; // Panel untuk memutar video

    public GameObject panelChoices; // Panel untuk pilihan

    public Button btA, btB, btC, btStart; // Tombol pilihan

    public GameObject btBenar, btSalah; // Indikator jawaban benar/salah



    [Header("Video Settings")]

    public VideoPlayer videoPlayer; // VideoPlayer untuk memutar video

    public float videoDuration = 10f; // Durasi video (jika VideoPlayer tidak mendukung durasi)



    [Header("Next Question")]

    public GameObject nextPanel; // Panel pertanyaan berikutnya

    public GameObject currentPanel; // Panel pertanyaan saat ini



    private bool isAnswerCorrect; // Status jawaban



    public GameObject mainPuzzlePanel; // Panel utama (main_puzzle)

    public GameObject[] activePanels; // Semua panel lain yang mungkin aktif



    void Start()

    {

        // Pastikan panel video aktif dan panel lainnya tidak aktif di awal

        panelVideo.SetActive(false);
```

```
        panelChoices.SetActive(false);
        btBenar.SetActive(false);
        btSalah.SetActive(false);

        // Sambungkan tombol pilihan ke metode penanganan jawaban
        btA.onClick.AddListener(() => CheckAnswer("A"));
        btB.onClick.AddListener(() => CheckAnswer("B"));
        btC.onClick.AddListener(() => CheckAnswer("C"));

       // Periksa apakah VideoPlayer mendukung durasi
        if (videoPlayer != null)
        {
            videoPlayer.loopPointReached += OnVideoFinished; // Event saat video selesai
        }
    }

    public void StartQuiz()
    {
        panelVideo.SetActive(true); // Tampilkan panel video
        panelChoices.SetActive(false); // Sembunyikan panel pilihan

        // Sembunyikan tombol Start
        btStart.gameObject.SetActive(false);

        if (videoPlayer != null)
        {
            videoPlayer.Play(); // Mulai video
        }
        else
        {
          // Jika VideoPlayer tidak tersedia, gunakan timer sebagai alternatif
            StartCoroutine(VideoTimer(videoDuration));
        }
    }

    void OnVideoFinished(VideoPlayer vp)
    {
        // Ketika video selesai, tampilkan pilihan
        panelVideo.SetActive(false);
        //panelChoices.SetActive(true);
        btA.gameObject.SetActive(true);
        btB.gameObject.SetActive(true);
        btC.gameObject.SetActive(true);
    }

    IEnumerator VideoTimer(float duration)
    {
        yield return new WaitForSeconds(duration);
       OnVideoFinished(null); // Panggil logika setelah video selesai
    }

    public void CheckAnswer(string selectedAnswer)
    {
        // Tentukan apakah jawaban benar atau salah
       //isAnswerCorrect = (selectedAnswer == "A"); // Misalnya jawaban benar adalah "A"

        // Variabel untuk menentukan apakah jawaban benar
        bool isAnswerCorrect = false;

       // Tentukan jawaban benar berdasarkan nama panel aktif
        if (currentPanel.name == "quiz_pajoge")
        {
            isAnswerCorrect = (selectedAnswer == "A");
        }
        else if (currentPanel.name == "quiz_panjaga")
```

```
        {
            isAnswerCorrect = (selectedAnswer == "B");
        }
        else if (currentPanel.name == "quiz_paddupa")
        {
            isAnswerCorrect = (selectedAnswer == "C");
        }
        else
        {
            Debug.LogError($"Panel tidak dikenal: {currentPanel.name}");
        }

        if (isAnswerCorrect)
        {
            Debug.Log("Jawaban benar!");
            btBenar.SetActive(true);
            btSalah.SetActive(false);
            StartCoroutine(ProceedToNextQuestion());
        }
        else
        {
            Debug.Log("Jawaban salah!");
            btBenar.SetActive(false);
            btSalah.SetActive(true);
           StartCoroutine(HideWrongAnswerFeedback());
        }
    }

    IEnumerator ProceedToNextQuestion()
    {
        yield return new WaitForSeconds(2f); // Jeda sebelum pindah ke pertanyaan berikutnya

        btBenar.SetActive(false);

        // Pindah ke panel pertanyaan berikutnya
        if (nextPanel != null)
        {
            currentPanel.SetActive(false);
            nextPanel.SetActive(true);
        }
    }

    // Coroutine untuk menyembunyikan feedback jawaban salah setelah 2 detik
    privateIEnumerator HideWrongAnswerFeedback()
    {
        yield return new WaitForSeconds(2); // Tunggu selama 2 detik
        btSalah.SetActive(false); // Sembunyikan tombol btSalah
    }

    public void BackToMainPuzzle()
    {
        // Set panel main_puzzle aktif
        mainPuzzlePanel.SetActive(true);
        currentPanel.SetActive(false);

        // Nonaktifkan semua panel yang lain
        //foreach (GameObject panel in activePanels)
        //{
        //    if (panel.activeSelf)
        //    {
        //        panel.SetActive(false);
        //        Debug.Log($"Panel {panel.name} dinonaktifkan.");
        //    }
        //}

        Debug.Log("Kembali ke panel main_puzzle.");
    }
}
using UnityEngine;
```

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

public class QuizManagerLagu : MonoBehaviour
{
    [Header("UI Elements")]
    public GameObject panelQuizLagu; // Panel utama kuis lagu
    public GameObject btA, btB, btC; // Tombol pilihan
    public GameObject btBenar; // Indikator jawaban benar
    public GameObject btSalah; // Indikator jawaban salah
    public Button btStart; // Tombol untuk memulai lagu
    public GameObject panelKonfirmasi; // Panel konfirmasi keluar

    [Header("Audio Settings")]
    public AudioSource audioSource; // AudioSource untuk memutar lagu
    public AudioClip[] laguClips; // Daftar lagu untuk kuis
    public string[] correctAnswers; // Jawaban benar untuk setiap lagu (harus sama panjang dengan laguClips)

    private int currentLaguIndex = 0; // Indeks lagu saat ini

    [Header("Next Question")]
    public GameObject nextPanel; // Panel pertanyaan berikutnya
    public GameObject currentPanel; // Panel pertanyaan saat ini
    public GameObject mainPuzzlePanel; // Panel utama (main_puzzle)

    void Start()
    {
        // Pastikan semua elemen dalam keadaan awal
        ResetUI();
        //panelQuizLagu.SetActive(false);
    }

    public void StartQuizLagu()
    {
        // Reset UI dan aktifkan panel kuis lagu
        ResetUI();
        //panelQuizLagu.SetActive(true);

        // Mulai memutar lagu pertama
        PlayCurrentLagu();
    }

    private void PlayCurrentLagu()
    {
        //if (currentLaguIndex < laguClips.Length && audioSource != null)
        // Aktifkan AudioSource jika sebelumnya disable
        if (!audioSource.gameObject.activeSelf)
        {
            audioSource.gameObject.SetActive(true);
        }

        if (audioSource != null)
        {
            //audioSource.clip = laguClips[currentLaguIndex];
            audioSource.Play();
            // Sembunyikan tombol Start
            btStart.gameObject.SetActive(false);
            StartCoroutine(ShowChoicesAfterAudio());
        }
        else
        {
            Debug.LogError("Tidak ada lagu yang tersedia atau AudioSource belum diatur!");
        }
```

```
    }

    private IEnumerator ShowChoicesAfterAudio()
    {
        // Tunggu hingga lagu selesai diputar
        yield return new WaitForSeconds(audioSource.clip.length);

        // Tampilkan pilihan jawaban
        btA.SetActive(true);
        btB.SetActive(true);
        btC.SetActive(true);
    }

    public void CheckAnswer(string selectedAnswer)
    {
        //bool isAnswerCorrect = (selectedAnswer == correctAnswers[currentLaguIndex]);
        // Variabel untuk menentukan apakah jawaban benar
        bool isAnswerCorrect = false;

       // Tentukan jawaban benar berdasarkan nama panel aktif
        if (currentPanel.name == "quiz_lagu1")
        {
            isAnswerCorrect = (selectedAnswer == "A");
        }
        else if (currentPanel.name == "quiz_lagu2")
        {
            isAnswerCorrect = (selectedAnswer == "B");
        }
        else if (currentPanel.name == "quiz_lagu3")
        {
            isAnswerCorrect = (selectedAnswer == "C");
        }
        else
        {
            Debug.LogError($"Panel tidak dikenal: {currentPanel.name}");
        }

        if (isAnswerCorrect)
        {
            Debug.Log("Jawaban benar!");
            btBenar.SetActive(true);
            btSalah.SetActive(false);

            // Lanjutkan ke lagu berikutnya setelah jeda
            StartCoroutine(ProceedToNextQuestion());
        }
        else
        {
            Debug.Log("Jawaban salah!");
            btBenar.SetActive(false);
            btSalah.SetActive(true);

            // Berikan jeda sebelum memperbolehkan mencoba lagi
            StartCoroutine(HideIncorrectMessage());
        }
    }

    private IEnumerator HideIncorrectMessage()
    {
        yield return new WaitForSeconds(2f);
        btSalah.SetActive(false);
    }

    private IEnumerator ProceedToNextQuestion()
    {
        yield return new WaitForSeconds(2f);

        btBenar.SetActive(false);
```

```
        // Pindah ke panel pertanyaan berikutnya
        if (nextPanel != null)
        {
            currentPanel.SetActive(false);
            nextPanel.SetActive(true);
        }
    }

    public void BackToMainMenu()
    {
        //panelQuizLagu.SetActive(false);
        // Tambahkan logika untuk kembali ke menu utama
        // Set panel main_puzzle aktif
        ResetUI();
        mainPuzzlePanel.SetActive(true);
        currentPanel.SetActive(false);
    }

    private void ResetUI()
    {
        btA.SetActive(false);
        btB.SetActive(false);
        btC.SetActive(false);
        btBenar.SetActive(false);
        btSalah.SetActive(false);
    }
}
using UnityEngine;
using UnityEngine.UI;
using System;

public class TimerManager : MonoBehaviour
{
    public Slider timerBar; // Progress bar untuk timer
    // public float maxTime = 60f; // Waktu maksimal dalam detik
    public float maxTime; // Waktu maksimal dalam detik

    private float currentTime; // Waktu saat ini
    private bool isTimerRunning = false; // Status timer

    public event Action OnTimerComplete; // Event ketika timer habis

    public PuzzleManager puzzleManager; // Referensi ke PuzzleManager
    public Text tx_waktuHabis; // Teks waktu habis
    public GameObject puzzle_gambar;
    public GameObject puzzle_gambar1; // Panel Puzzle Gambar 1
    public GameObject puzzle_gambar2; // Panel Puzzle Gambar 2

    public bool isTimeUp { get; private set; } = false;

    void Start()
    {
        ResetTimer(); // Inisialisasi timer
        SetMaxTimeBasedOnActivePanel();
        // Cari PuzzleManager di scene
        GameObject puzzleManagerObject = GameObject.Find("panel_puzzle_area");
        if (puzzleManagerObject != null)
        {
            puzzleManager = puzzleManagerObject.GetComponent<PuzzleManager>();
        }
        else
        {
            Debug.LogError("PuzzleManager tidak ditemukan! Pastikan objek dengan script PuzzleManager memiliki nama 'panel_puzzle_area'.");
        }
```

```
    }

    void Update()
    {
        if (isTimerRunning)
        {
            currentTime -= Time.deltaTime; // Kurangi waktu
            timerBar.value = currentTime; // Perbarui progress bar

            if (currentTime <= 0)
            {
                currentTime = 0;
                isTimerRunning = false;
                isTimeUp = true; // Tandai bahwa waktu telah habis
                OnTimerComplete?.Invoke(); // Panggil event jika waktu habis

                // Periksa status puzzle melalui CheckPuzzleState
                bool isPuzzleComplete = puzzleManager.CheckPuzzleState();

                if (!isPuzzleComplete) // Jika puzzle belum selesai
                {
                    if (tx_waktuHabis != null)
                    {
                        tx_waktuHabis.gameObject.SetActive(true); // Tampilkan pesan waktu habis
                    }
                }
            }
        }
    }

    public void StartTimer()
    {
        isTimerRunning = true;
    }

    public void StopTimer()
    {
        isTimerRunning = false;
        currentTime = 0; // Reset waktu
    }

    public void ResetTimer()
    {
        currentTime = maxTime;
        timerBar.maxValue = maxTime;
        timerBar.value = maxTime;
        // fillImage.color = normalColor; // Reset warna ke normal
        isTimerRunning = false;
    }

    void SetMaxTimeBasedOnActivePanel()
    {
        if (puzzle_gambar.activeSelf)
        {
            maxTime = 60f; // Waktu untuk panel puzzle_gambar1
        }
        else if (puzzle_gambar2.activeSelf)
        {
            maxTime = 180f; // Waktu untuk panel puzzle_gambar2
        }
        else
        {
            maxTime = 480f; // Waktu default jika tidak ada panel aktif
        }
```

```
        Debug.Log($"Max Time set to: {maxTime} for
active panel");

    }

}
```