LAMPIRAN

AppManager

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

public class AppManager :
MonoBehaviour
{
   public GameObject MainPanel;
   public GameObject CircleMenu;

   void Start()
   {
      CircleMenu.SetActive(false);
      MainPanel.SetActive(true);
   }

   public void ExploreMode()
   {
      MainPanel.SetActive(false);
      CircleMenu.SetActive(true);
   }

   public void ShowMainPanel()
   {
      MainPanel.SetActive(true);
      CircleMenu.SetActive(false);
   }

   /*
   public void TentangPanelshow()
   {
      MainPanel.SetActive(false);
      TentangPanel.SetActive(true);
   }
   */

   public void QuitApp()
   {
      // save any game data here
#if UNITY_EDITOR
      // Application.Quit() does not
work in the editor so
      //
UnityEditor.EditorApplication.isPlay
ing need to be set to false to end the
game

UnityEditor.EditorApplication.isPlay
ing = false;
#else
       Application.Quit();
#endif
   }
}
```

BukaTutupPintu

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

public class BukaTutupPintu : MonoBehaviour
{
   public GameObject PintuObj;

   public void bukatutupPintu()
   {
      if (PintuObj != null)
      {
         Animator animator = PintuObj.GetComponent<Animator>();
         if (animator != null)
         {
            bool isOpen = animator.GetBool("show");
            animator.SetBool("show", !isOpen);
         }
      }
   }
}
```

Door Rotation

```
using UnityEngine;
using System.Collections;

public class DoorRotation : MonoBehaviour
{
   [Header("Door Settings")]
   public float openAngle = 90f;
   public float speed = 2f;

   [Header("Audio")]
   public AudioSource audioSource;
   public AudioClip openSound;
   public AudioClip closeSound;

   private bool isOpen = false;
   private bool isMoving = false;

   private Quaternion closedRotation;
   private Quaternion openRotation;

   void Start()
   {
      // Simpan rotasi awal pintu
      closedRotation = transform.localRotation;

      // Buat rotasi saat pintu terbuka
      openRotation = closedRotation * Quaternion.Euler(0, openAngle, 0);
   }

   public void ToggleDoor()
   {
      if (isMoving)
         return;
```

```
        if (isOpen)
        {

StartCoroutine(MoveDoor(closedRot
ation, false));
        }
        else
        {

StartCoroutine(MoveDoor(openRota
tion, true));
        }
    }

    IEnumerator
MoveDoor(Quaternion
targetRotation, bool opening)
    {
        isMoving = true;

        // Mainkan suara
        if (audioSource != null)
        {
            if (opening && openSound
!= null)

audioSource.PlayOneShot(openSoun
d);

            if (!opening && closeSound
!= null)

audioSource.PlayOneShot(closeSoun
d);
        }

        Quaternion startRotation =
transform.localRotation;
        float time = 0f;

        while (time < 1f)
        {
            time += Time.deltaTime *
speed;

            transform.localRotation =
Quaternion.Slerp(
                startRotation,
                targetRotation,
                time
            );

            yield return null;
        }

        transform.localRotation =
targetRotation;

        isOpen = opening;
        isMoving = false;
    }
}
```

FadeEffectScene

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

public class FadeEffectScene : MonoBehaviour {

   public Texture2D fadeOutTexture;
   public float fadeSpeed = 0.8f;

   private int drawDepth = -1000;
   private float alpha = 1.0f;
   private int fadeDir = -1;

   void OnGUI()
   {
      alpha += fadeDir * fadeSpeed * Time.deltaTime;
      alpha = Mathf.Clamp01(alpha);

      GUI.color = new Color(GUI.color.r, GUI.color.g, GUI.color.b, alpha);
      GUI.depth = drawDepth;
      GUI.DrawTexture(new Rect(0, 0, Screen.width, Screen.height), fadeOutTexture);
   }

   public float BeginFade(int direction)
   {
      fadeDir = direction;
      return (fadeSpeed);
   }

   void OnLevelWasLoaded()
   {
      BeginFade(-1);
   }
}
```

FanSpinning

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

public class FanSpinning : MonoBehaviour
{
    public GameObject CeilingFan;

    public float rotateSpeed = 50f;
    bool rotateStatus = false;

  void Start()
  {
        rotateStatus = true;
    }

  public void OnOffSpinning()
    {

        if (rotateStatus == false)
        {
            rotateStatus = false;
        }
        else
        {
            rotateStatus = true;
        }
    }

    void Update()
    {
        if (rotateStatus == true)
        {

    CeilingFan.transform.Rotate(Vector3.up, rotateSpeed * Time.deltaTime);
        }
    }
}
```

GazeInteraction

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

public class GazeInteraction :
MonoBehaviour {

   private float gazeTime = 1.2f;
   private float timer;
   private bool gazedAt;

   public Image ImgGaze;
   VRcontrol controls;

   // Update is called once per frame
   void Update()
   {
      if (gazedAt)
      {
         timer += Time.deltaTime;
         ImgGaze.fillAmount = timer
/ gazeTime;

         if (timer >= gazeTime)
         {
            // execute pointerdown
handler

ExecuteEvents.Execute(gameObject,
new
PointerEventData(EventSystem.curre
nt),
ExecuteEvents.pointerDownHandler)
;
            timer = 0f;
            enabled = false;

StartCoroutine(enableUpdate());
         }
      }

   }

   void Awake()
   {
      controls = new VRcontrol();
      controls.Player.Click.performed
+= ctx => Click();
   }

   void Click()
   {
      // execute pointerdown handler

ExecuteEvents.Execute(gameObject,
new
PointerEventData(EventSystem.curre
nt),
ExecuteEvents.pointerDownHandler)
;

   }

   IEnumerator enableUpdate()
   {
      yield return new
WaitForSeconds(1.5f);
      enabled = true;
   }

   public void PointerEnter()
   {
      gazedAt = true;
      enabled = true;
      controls.Player.Enable();
      Debug.Log("PointerEnter");
   }

   public void PointerExit()
   {
      gazedAt = false;
      enabled = false;
      controls.Player.Disable();
```

```
        timer = 0;
        ImgGaze.fillAmount = 0;
        Debug.Log("PointerExit");
    }

    public void PointerDown()
    {

StartCoroutine(PointerExitting());
    }

    IEnumerator PointerExitting()
    {
        yield return new WaitForSeconds(1f);
    }
}
```

LampControl

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

public class LampControl : MonoBehaviour
{
   public Light Lamp;
   public float LightIntensity;
   private bool active = false;

   public void TurnOnOffLight()
   {
      if (!active)
      {
         Lamp.intensity = 0f;
         active = true;
      }
      else
      {
         Lamp.intensity = LightIntensity;
         active = false;
      }
   }
}
```

Lompat

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

public class Lompat : MonoBehaviour
{
  [SerializeField]private Color warnaTakAktif;
  [SerializeField]private Color lihatPadaWarna;
  [SerializeField] private GameObject PlayerGC;

  private MeshRenderer myRenderer;
  private bool tukarwarna = false;
  private float masa = 0f;
  // Start is called before the first frame update
  void Start()
  {
    myRenderer = GetComponent<MeshRenderer>();
    myRenderer.material.color = warnaTakAktif;
  }

  // Update is called once per frame
  void Update()
  {
    if(tukarwarna)
    {
      myRenderer.material.color = Color.Lerp(myRenderer.material.color, lihatPadaWarna, Time.deltaTime / 2f);
      masa += Time.deltaTime;
      if(masa >= 2f)
      {
        //lompat
        Vector3 PoseLompat = new Vector3(transform.position.x, PlayerGC.transform.position.y, transform.position.z);

        PlayerGC.transform.position = PoseLompat;
      }
    }

  }
  public void OnPointerEnter()
  {
    GazeAt(true);
  }
  public void OnPointerExit()
  {
    GazeAt(false);
  }
  public void GazeAt(bool gazing)
  {
    if (gazing)
    {
      tukarwarna = true;

    }
    else
    {
      masa = 0f;
      tukarwarna = false;
      myRenderer.material.color = warnaTakAktif;
    }
  }
}
```

LookAtTransform

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

public class LookAtTransform : MonoBehaviour
{

	public Transform CameraObj;
	public GameObject Target;

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

		Target.transform.LookAt(new	Vector3(CameraObj.position.x,
Target.transform.position.y, CameraObj.position.z));
		Target.transform.forward *= -1;
	}
}
```

MainMenu

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

public class MainMenu :
MonoBehaviour
{
   public GameObject MainPanel,
TutorialPanel, TentangPanel;
   public string SceneReload;

   void Start()
   {
      MainPanel.SetActive(true);
      TutorialPanel.SetActive(false);
   }

   public void MainMenuScene()
   {

SceneManager.LoadScene("MainMe
nu");
   }

   public void MulaiVrScene()
   {

SceneManager.LoadScene("VRscene
s");
   }

   public void ShowMainPanel()
   {
      MainPanel.SetActive(true);
      TentangPanel.SetActive(false);
      TutorialPanel.SetActive(false);
   }

   public void ShowPanduanPanel()
   {
      TentangPanel.SetActive(false);
      MainPanel.SetActive(true);
      TutorialPanel.SetActive(true);
   }

   public void ShowTentangPanel()
   {
      TentangPanel.SetActive(true);
      MainPanel.SetActive(true);
      TutorialPanel.SetActive(false);
   }

   public void ReloadScene()
   {

SceneManager.LoadScene(SceneRel
oad);
   }

   public void QuitApp()
   {
      // save any game data here
#if UNITY_EDITOR
      // Application.Quit() does not
work in the editor so
      //
UnityEditor.EditorApplication.isPlay
ing need to be set to false to end the
game

UnityEditor.EditorApplication.isPlay
ing = false;
#else
       Application.Quit();
#endif
   }
}
```

NormalToVR

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

public class NormalToVR :
MonoBehaviour
{
   public string MainMenuScene;

   public string VRSimulasiScene;

   private bool VRswitch = false;

   public void VRSwitchMode()
   {
      if (!VRswitch)
      {

StartCoroutine(FullViewActive());
         VRswitch = true;
      }
      else
      {

StartCoroutine(VrModeActive());
         VRswitch = false;
      }
   }

   public void HomeSceneActive()
   {

StartCoroutine(NormalActive());
   }

   public void
VrSimulasiSceneActive()
   {

StartCoroutine(VRSimulasiActive())
;
   }

   public void VrMode()
   {

StartCoroutine(VrModeActive());
   }

   public void StereoMode()
   {
      StartCoroutine(StereoActive());
   }

   public void FullViewMode()
   {

StartCoroutine(FullViewActive());
   }

   private IEnumerator
VRSimulasiActive()
   {
      string desiredDevice =
"cardboard";

XRSettings.LoadDeviceByName(des
iredDevice);
      yield return null;
      XRSettings.enabled = true;

SceneManager.LoadScene(VRSimul
asiScene);
   }

   private IEnumerator
NormalActive()
   {

XRSettings.LoadDeviceByName(stri
ng.Empty);
      yield return null;
      ResetCameras();

SceneManager.LoadScene(MainMen
uScene);
```

```
    }

    private IEnumerator VrModeActive()
    {
        string desiredDevice = "cardboard";

        XRSettings.LoadDeviceByName(desiredDevice);
        yield return null;
        XRSettings.enabled = true;
        ResetVrCameras();
    }

    private IEnumerator FullViewActive()
    {

        XRSettings.LoadDeviceByName(string.Empty);
        yield return null;
        ResetCameras();
    }

    private IEnumerator StereoActive()
    {

        XRSettings.LoadDeviceByName(string.Empty);
        yield return null;
        ResetCameras();

        SceneManager.LoadScene(VRSimulasiScene);
    }

    private void ResetVrCameras()
    {
        for (int i = 0; i < Camera.allCameras.Length; i++)
        {
            Camera camera = Camera.allCameras[i];
            if (camera.enabled && camera.stereoTargetEye != StereoTargetEyeMask.Both)
            {

                camera.transform.localPosition = Vector3.zero;

                camera.transform.localRotation = Quaternion.identity;
            }
        }
    }

    private void ResetCameras()
    {
        for (int i = 0; i < Camera.allCameras.Length; i++)
        {
            Camera camera = Camera.allCameras[i];
            if (camera.enabled && camera.stereoTargetEye != 0)
            {

                camera.transform.localPosition = Vector3.zero;

                camera.transform.localRotation = Quaternion.identity;
            }
        }
    }
}
```

ObjectFollowCamera

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

public class ObjectFollowCamera : MonoBehaviour
{
   private GameObject MainCamera;
   public GameObject Object;

   void Awake()
   {
      MainCamera = GameObject.FindGameObjectWithTag("MainCamera");
   }

   void Update()
   {
      Object.transform.rotation = Quaternion.Euler(0, MainCamera.transform.eulerAngles.y, 0);
   }
}
```

PlayAnimasi

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

public class PlayAnimasi : MonoBehaviour
{
   public GameObject ObjectAnim;

   public void PlayStopObjectAnimation()
   {
      if (ObjectAnim != null)
      {
         Animator animator = ObjectAnim.GetComponent<Animator>();
         if (animator != null)
         {
            bool isOpen = animator.GetBool("play");
            animator.SetBool("play", !isOpen);
         }
      }
   }

   /*
   public void OnPointerEnter()
   {
      GazeAt(true);
   }

   public void OnPointerExit()
   {
      GazeAt(false);
   }
   public void GazeAt(bool gazing)
   {
      if (gazing)
      {
         PlayStopObjectAnimation();

      }
      else
      {

      }
   }
   */
}
```

PlayerController

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

public class PlayerController : MonoBehaviour
{
   public float runSpeed = 10f;
   public float jumpForce = 5f;
   public Rigidbody rigidbody;
   private Vector3 movement = new Vector3();
   VRcontrol controls;

   void Awake()
   {
      controls = new VRcontrol();
      controls.Player.Jump.performed += ctx => Jump();
   }

   // Update is called once per frame
   void FixedUpdate()
   {
      movement.x = runSpeed * Input.GetAxis("Horizontal");
      movement.z = runSpeed * Input.GetAxis("Vertical");


//rigidbody.MovePosition(rigidbody.position + movement * Time.fixedDeltaTime);
      transform.Translate(movement * Time.fixedDeltaTime, Camera.main.transform);

      if (Input.GetButtonDown("Jump"))
      {
         rigidbody.velocity = Vector3.up * jumpForce;
      }

   }

   void Jump()
   {
      rigidbody.velocity = Vector3.up * jumpForce;
   }

   void OnEnable()
   {
      controls.Player.Enable();
   }

   void OnDisable()
   {
      controls.Player.Disable();
   }
}
```

ShowHideInfo

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

public class ShowHideInfo : MonoBehaviour {

    public GameObject InfoText;
    private bool ShowInfo = false;

    public void showhideInfo()
    {
        if (!ShowInfo)
        {
            InfoText.SetActive(true);
            ShowInfo = true;
        }
        else
        {
            InfoText.SetActive(false);
            ShowInfo = false;
        }
    }

    /*
    public void OnPointerEnter()
    {
        GazeAt(true);
    }

    public void OnPointerExit()
    {
        GazeAt(false);
    }
    public void GazeAt(bool gazing)
    {
        if (gazing)
        {
            showhideInfo();

        }
        else
        {

        }
    }
    */
}
```

TriggerLamp

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

public class TriggerLamp : MonoBehaviour
{
    public Light Lighting;
    public GameObject PowerOnImg, PowerOffImg;
    private bool ActivateTrigger = false;

    void Start()
    {
        Lighting.intensity = 0f;
    }

    void OnTriggerEnter(Collider TriggerCol)
    {
        if (TriggerCol.gameObject.tag == "Hands")
        {
            LampSwitch();
        }
    }

    public void LampSwitch()
    {
        if (!ActivateTrigger)
        {
            Lighting.intensity = 2f;

PowerOnImg.SetActive(true);

PowerOffImg.SetActive(false);
            ActivateTrigger = true;
        }
        else
        {
            Lighting.intensity = 0f;

PowerOnImg.SetActive(false);

PowerOffImg.SetActive(true);
            ActivateTrigger = false;
        }
    }
}
```

TriggerTV

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

public class TriggerTV : MonoBehaviour
{
   public GameObject TvOffPlane;
   public VideoPlayer VideoPlay;
   public Material TurnOnMat, TurnOffMat;
   Renderer renderer;
   Material CurrMat;
   private bool ActivateTv = false;

   public GameObject PowerOnImg, PowerOffImg;

   // Use this for initialization
   void Start()
   {
      renderer = this.GetComponent<Renderer>();
   }

   public void TurnOnOffTv()
   {
      if (!ActivateTv)
      {
         VideoPlay.Play();
         TvOffPlane.SetActive(false);
         renderer.material = TurnOnMat;

PowerOnImg.SetActive(true);

PowerOffImg.SetActive(false);
         ActivateTv = true;
      }
      else
      {
         VideoPlay.Stop();
         TvOffPlane.SetActive(true);
         renderer.material = TurnOffMat;

PowerOnImg.SetActive(false);

PowerOffImg.SetActive(true);
         ActivateTv = false;
      }
   }
}
```

TukarWarna

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

public class TukarWarna : MonoBehaviour
{
   [SerializeField]private Color warnaTakAktif;
   [SerializeField]private Color lihatPadaWarna;

   private MeshRenderer myRenderer;
   // Start is called before the first frame update
   void Start()
   {
      myRenderer = GetComponent<MeshRenderer>();
      myRenderer.material.color = warnaTakAktif;
   }

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

   }
   public void OnPointerEnter()
   {
      GazeAt(true);
   }
   public void OnPointerExit()
   {
      GazeAt(false);
   }
   public void GazeAt(bool gazing)
   {
      if (gazing)
      {
         myRenderer.material.color = lihatPadaWarna;
      }
      else
      {
         myRenderer.material.color = warnaTakAktif;
      }
   }
}
```

TukarWarnaDua

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

public class TukarWarnaDua : MonoBehaviour
{
    [SerializeField]private Color warnaTakAktif;
    [SerializeField]private Color lihatPadaWarna;

    private MeshRenderer myRenderer;
    private bool tukarwarna = false;
    // Start is called before the first frame update
    void Start()
    {
        myRenderer = GetComponent<MeshRenderer>();
        myRenderer.material.color = warnaTakAktif;
    }

    // Update is called once per frame
    void Update()
    {
        if(tukarwarna)
        {
            myRenderer.material.color = Color.Lerp(myRenderer.material.color, lihatPadaWarna, Time.deltaTime / 2f);
        }

    }
    public void OnPointerEnter()
    {
        GazeAt(true);
    }
    public void OnPointerExit()
    {
        GazeAt(false);
    }
    public void GazeAt(bool gazing)
    {
        if (gazing)
        {
            tukarwarna = true;

        }
        else
        {
            tukarwarna = false;
            myRenderer.material.color = warnaTakAktif;
        }
    }
}
```

VRcontrol

```
// GENERATED AUTOMATICALLY FROM 'Assets/VRcontrol.inputactions'

using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.InputSystem;
using UnityEngine.InputSystem.Utilities;

public class @VRcontrol : IInputActionCollection, IDisposable
{
    public InputActionAsset asset { get; }
    public @VRcontrol()
    {
        asset = InputActionAsset.FromJson(@"{
    ""name"": ""VRcontrol"",
    ""maps"": [
        {
            ""name"": ""Player"",
            ""id"": ""1f70bd5b-3792-4a93-bed4-2a4f09c27450"",
            ""actions"": [
                {
                    ""name"": ""Click"",
                    ""type"": ""Button"",
                    ""id"": ""29e9d48a-c85a-4a88-9dcc-72f8412ee81b"",
                    ""expectedControlType"": ""Button"",
                    ""processors"": """",
                    ""interactions"": ""Press""
                },
                {
                    ""name"": ""Jump"",
                    ""type"": ""Button"",
                    ""id"": ""738ec309-66fd-44d3-8f11-fc7dcf1e44e1"",
                    ""expectedControlType"": ""Button"",
                    ""processors"": """",
                    ""interactions"": ""Press""
                }
            ],
            ""bindings"": [
                {
                    ""name"": """",
                    ""id"": ""2f3cce7d-776b-4945-91c2-1fac48f45a3b"",
                    ""path"": ""<Gamepad>/buttonSouth"",
                    ""interactions"": ""Press"",
                    ""processors"": """",
                    ""groups"": ""XboxControl"",
                    ""action"": ""Click"",
                    ""isComposite"": false,
                    ""isPartOfComposite"": false
                },
                {
                    ""name"": """",
                    ""id"": ""f3f56785-6875-42d2-9713-91f8c9131088"",
                    ""path"": ""<Gamepad>/buttonWest"",
                    ""interactions"": ""Press"",
                    ""processors"": """",
                    ""groups"": ""XboxControl"",
                    ""action"": ""Jump"",
                    ""isComposite"": false,
                    ""isPartOfComposite"": false
                }
            ]
```

```
        }
    ],
    ""controlSchemes"": [
        {
            ""name"": ""XboxControl"",
            ""bindingGroup"":
""XboxControl"",
            ""devices"": [
                {
                    ""devicePath"":
""<XInputController>"",
                    ""isOptional"": false,
                    ""isOR"": false
                }
            ]
        }
    ]
}");
        // Player
        m_Player                    =
asset.FindActionMap("Player",
throwIfNotFound: true);
        m_Player_Click              =
m_Player.FindAction("Click",
throwIfNotFound: true);
        m_Player_Jump               =
m_Player.FindAction("Jump",
throwIfNotFound: true);
    }

    public void Dispose()
    {

UnityEngine.Object.Destroy(asset);
    }

    public                  InputBinding?
bindingMask
    {
        get => asset.bindingMask;
        set =>  asset.bindingMask  =
value;
    }

    public
ReadOnlyArray<InputDevice>?
devices
    {
        get => asset.devices;
        set => asset.devices = value;
    }

    public
ReadOnlyArray<InputControlSchem
e>          controlSchemes          =>
asset.controlSchemes;

    public bool Contains(InputAction
action)
    {
        return asset.Contains(action);
    }

    public IEnumerator<InputAction>
GetEnumerator()
    {
        return asset.GetEnumerator();
    }

    IEnumerator
IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }

    public void Enable()
    {
        asset.Enable();
    }

    public void Disable()
    {
        asset.Disable();
    }

    // Player
    private readonly InputActionMap
m_Player;
    private                 IPlayerActions
m_PlayerActionsCallbackInterface;
    private readonly InputAction
m_Player_Click;
```

```
    private readonly InputAction m_Player_Jump;
    public struct PlayerActions
    {
        private @VRcontrol m_Wrapper;
        public PlayerActions(@VRcontrol wrapper) { m_Wrapper = wrapper; }
        public InputAction @Click => m_Wrapper.m_Player_Click;
        public InputAction @Jump => m_Wrapper.m_Player_Jump;
        public InputActionMap Get() { return m_Wrapper.m_Player; }
        public void Enable() { Get().Enable(); }
        public void Disable() { Get().Disable(); }
        public bool enabled => Get().enabled;
        public static implicit operator InputActionMap(PlayerActions set) { return set.Get(); }
        public void SetCallbacks(IPlayerActions instance)
        {
            if (m_Wrapper.m_PlayerActionsCallbackInterface != null)
            {
                @Click.started -= m_Wrapper.m_PlayerActionsCallbackInterface.OnClick;
                @Click.performed -= m_Wrapper.m_PlayerActionsCallbackInterface.OnClick;
                @Click.canceled -= m_Wrapper.m_PlayerActionsCallbackInterface.OnClick;
                @Jump.started -= m_Wrapper.m_PlayerActionsCallbackInterface.OnJump;
                @Jump.performed -= m_Wrapper.m_PlayerActionsCallbackInterface.OnJump;
                @Jump.canceled -= m_Wrapper.m_PlayerActionsCallbackInterface.OnJump;
            }
            m_Wrapper.m_PlayerActionsCallbackInterface = instance;
            if (instance != null)
            {
                @Click.started += instance.OnClick;
                @Click.performed += instance.OnClick;
                @Click.canceled += instance.OnClick;
                @Jump.started += instance.OnJump;
                @Jump.performed += instance.OnJump;
                @Jump.canceled += instance.OnJump;
            }
        }
    }
    public PlayerActions @Player => new PlayerActions(this);
    private int m_XboxControlSchemeIndex = -1;
    public InputControlScheme XboxControlScheme
    {
        get
        {
            if (m_XboxControlSchemeIndex == -1) m_XboxControlSchemeIndex = asset.FindControlSchemeIndex("XboxControl");
            return asset.controlSchemes[m_XboxControlSchemeIndex];
        }
    }
    public interface IPlayerActions
    {
```

```
        void OnClick(InputAction.CallbackContext context);
        void OnJump(InputAction.CallbackContext context);
    }
}
```

VRLookWalk

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

public class VRLookWalk : MonoBehaviour
{

    public Transform vrCamera;
    public float toggleAngle = 30.0f;
    public float speed = 3.0f;
    public bool moveForward;
    private bool Walk = false;

    private CharacterController cc;

    // Use this for initialization
    void Start()
    {
        cc = GetComponent<CharacterController>();
        enabled = false;
    }

    public void WalkMode()
    {
        if (!Walk)
        {
            enabled = true;
            Walk = true;
        }
        else
        {
            enabled = false;
            Walk = false;
        }
    }

    // Update is called once per frame
    void Update()
    {
        if (vrCamera.eulerAngles.x >= toggleAngle && vrCamera.eulerAngles.x < 90.0f)
        {
            moveForward = true;
        }
        else
        {
            moveForward = false;
        }

        if (moveForward)
        {
            Vector3 forward = vrCamera.TransformDirection(Vector3.forward);

            cc.SimpleMove(forward * speed);
        }
    }
}
```

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

PROPOSAL

Mahasiswa : FEGGY NURWINDI	**Pembimbing I** : WAHYUDDIN, S.Kom.,M.Kom
NIM : 220280028	**Pembimbing II** : Ir.Untung Suwardoyo, S.Kom.,MT.,IPP
Judul Skripsi : RANCANG BANGUN VIRTUAL REALITY GAMES HOUSE OF HORROR BERBASIS ANDROID	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Acc		Konsultasi 1 Perbaiki Cover Dokument	
Konsultasi 2		Konsultasi 2 Perbaiki Dokument Ada Versi Sistem	
Konsultasi 3		Konsultasi 3 Perbaiki Bab 3.	
Konsultasi 4		Konsultasi 4 Perbaiki Flowchart, tabel & grafik	
Konsultasi 5		Konsultasi 5 Acc Lanjut 2	

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 A4 berwarna Hijau muda dan dicetak timbal balik

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : Feggy Nurwindi	Pembimbing I : Wahyuddin, S.Kom.,M.Kom
NIM : 220280028	Pembimbing II : Ir. Untung Suwardoyo, S.Kom.,MT.,IPP
Judul Skripsi : Rancang Bangun Virtual Reality Games House Of Horror Berbasis Android	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 - perbaiki sistem - Use case diagram		Konsultasi 1	
Konsultasi 2 - Activity diagram - Sequence diagram		Konsultasi 2	
Konsultasi 3 - ACC seminar hasil		Konsultasi 3	
Konsultasi 4		Konsultasi 4	
Konsultasi 5		Konsultasi 5 Ac hal.	

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 A4 berwarna Hijau muda dan dicetak timbal balik