LISTING PROGRAM

AngleVirtualJoystick.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class AngleVirtualJoystick : MonoBehaviour, IDragHandler, IPointerUpHandler,
IPointerDownHandler
{
    [SerializeField]
    private float speed;
    ///private Image joy;
    public Vector3 InputDirection { set; get; }
    // Start is called before the first frame update

    private void Awake()
    {
        this.transform.SetParent(GameObject.Find("Control Character
Panel").GetComponent<Transform>(), false);
    }
    void Start()
    {
        //bgImg = GetComponent<Image>();
        //joy = transform.GetChild(0).GetComponent<Image>();
        InputDirection = Vector3.zero;
    }

    // Update is called once per frame

    public virtual void OnDrag(PointerEventData ped)
    {
        InputDirection = ped.delta*Time.deltaTime*speed;
    }
    public virtual void OnPointerUp(PointerEventData ped)
    {
        InputDirection = Vector3.zero;
        //joy.rectTransform.anchoredPosition = Vector3.zero;

    }
    public virtual void OnPointerDown(PointerEventData ped)
    {
        OnDrag(ped);

    }
}
```

CameraForward.cs

```
using UnityEngine;

namespace Hanafi
{
    /// <summary>
    /// Camera work. Follow a target
    /// </summary>
    public class CameraForwad : MonoBehaviour
    {
        #region Private Fields

        [Tooltip("The distance in the local x-z plane to the target")]
        [SerializeField]
        private float distance = 7.0f;
```

```
        [Tooltip("The height we want the camera to be above the target")]
        [SerializeField]
        private float height = 3.0f;

        [Tooltip("The Smooth time lag for the height of the camera.")]
        [SerializeField]
        private float heightSmoothLag = 0.3f;

        [Tooltip("Allow the camera to be offseted vertically from the target, for example giving more view of the sceneray and less ground.")]
        [SerializeField]
        private Vector3 centerOffset = Vector3.zero;

        [Tooltip("Set this as false if a component of a prefab being instanciated by Photon Network, and manually call OnStartFollowing() when and if needed.")]
        [SerializeField]
        private bool followOnStart = false;

        // cached transform of the target
        Transform cameraTransform;

        // maintain a flag internally to reconnect if target is lost or camera is switched
        bool isFollowing;

        // Represents the current velocity, this value is modified by SmoothDamp() every time you call it.
        private float heightVelocity;

        // Represents the position we are trying to reach using SmoothDamp()
        private float targetHeight = 100000.0f;

        #endregion

        #region MonoBehaviour Callbacks

        /// <summary>
        /// MonoBehaviour method called on GameObject by Unity during initialization phase
        /// </summary>
        void Start()
        {
            // Start following the target if wanted.
            if (followOnStart)
            {
                OnStartFollowing();
            }

        }

        /// <summary>
        /// MonoBehaviour method called after all Update functions have been called. This is useful to order script execution. For example a follow camera should always be implemented in LateUpdate because it tracks objects that might have moved inside Update.
        /// </summary>
        void LateUpdate()
        {
            // The transform target may not destroy on level load,
```

```
            // so we need to cover corner cases where the Main Camera is different
everytime we load a new scene, and reconnect when that happens
            if (cameraTransform == null && isFollowing)
            {
                OnStartFollowing();
            }

            // only follow is explicitly declared
            if (isFollowing)
            {
                Apply();
            }
        }

        #endregion

        #region Public Methods

        /// <summary>
        /// Raises the start following event.
        /// Use this when you don't know at the time of editing what to follow, typically
instances managed by the photon network.
        /// </summary>
        public void OnStartFollowing()
        {
            cameraTransform = Camera.main.transform;
            isFollowing = true;
            // we don't smooth anything, we go straight to the right camera shot
            Cut();
        }

        #endregion

        #region Private Methods

        /// <summary>
        /// Follow the target smoothly
        /// </summary>
        void Apply()
        {
            Vector3 targetCenter = transform.position + centerOffset;

            // Calculate the current & target rotation angles
            float originalTargetAngle = transform.eulerAngles.y;
            float currentAngle = cameraTransform.eulerAngles.y;

            // Adjust real target angle when camera is locked
            float targetAngle = originalTargetAngle;

            currentAngle = targetAngle;

            targetHeight = targetCenter.y + height;

            // Damp the height
            float currentHeight = cameraTransform.position.y;
            currentHeight = Mathf.SmoothDamp(currentHeight, targetHeight, ref
heightVelocity, heightSmoothLag);

            // Convert the angle into a rotation, by which we then reposition the camera
            Quaternion currentRotation = Quaternion.Euler(0, currentAngle, 0);
```

```
            // Set the position of the camera on the x-z plane to:
            // distance meters behind the target
            cameraTransform.position = targetCenter;
            cameraTransform.position += currentRotation * Vector3.back * distance;

            // Set the height of the camera
            cameraTransform.position = new Vector3(cameraTransform.position.x,
currentHeight, cameraTransform.position.z);

            // Always look at the target
            SetUpRotation(targetCenter);
        }


        /// <summary>
        /// Directly position the camera to a the specified Target and center.
        /// </summary>
        void Cut()
        {
            float oldHeightSmooth = heightSmoothLag;
            heightSmoothLag = 0.001f;

            Apply();

            heightSmoothLag = oldHeightSmooth;
        }

        /// <summary>
        /// Sets up the rotation of the camera to always be behind the target
        /// </summary>
        /// <param name="centerPos">Center position.</param>
        void SetUpRotation(Vector3 centerPos)
        {
            Vector3 cameraPos = cameraTransform.position;
            Vector3 offsetToCenter = centerPos - cameraPos;

            // Generate base rotation only around y-axis
            Quaternion yRotation = Quaternion.LookRotation(new
Vector3(offsetToCenter.x, 0, offsetToCenter.z));

            Vector3 relativeOffset = Vector3.forward * distance + Vector3.down *
height;
            cameraTransform.rotation = yRotation *
Quaternion.LookRotation(relativeOffset);

        }

        #endregion
    }
}
```

GameManager.cs

```
using System;
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;
using Photon.Pun;
using Photon.Realtime;
using UnityEngine.UI;

namespace Hanafi
```

```
{
    public class GameManager : MonoBehaviourPunCallbacks
    {
        #region Public Field
        [Tooltip("The prefab to use for representing the player")]
        public GameObject playerPrefab;
        #endregion

        #region Private Field
        GameObject player;
        int respownPoint;
        bool mulai;
        #endregion
        #region Photon Callbacks
        public override void OnPlayerEnteredRoom(Player other)
        {
            Debug.LogFormat("OnPlayerEnteredRoom() {0}", other.NickName); // not
seen if you're the player connecting
            if (PhotonNetwork.IsMasterClient)
            {
                Debug.LogFormat("OnPlayerEnteredRoom IsMasterClient {0}",
PhotonNetwork.IsMasterClient); // called before OnPlayerLeftRoom
                //LoadArena();
            }
        }


        public override void OnPlayerLeftRoom(Player other)
        {
            Debug.LogFormat("OnPlayerLeftRoom() {0}", other.NickName); // seen when
other disconnects
            if (PhotonNetwork.IsMasterClient)
            {
                Debug.LogFormat("OnPlayerLeftRoom IsMasterClient {0}",
PhotonNetwork.IsMasterClient); // called before OnPlayerLeftRoom
                //LoadArena();
            }
        }
        /// <summary>
        /// Called when the local player left the room. We need to load the launcher
scene.
        /// </summary>

        //Metode override dari MonoBehaviourPunCallbacks
        //meload scane 0 ketika meninggalkan room
        public override void OnLeftRoom(){
            SceneManager.LoadScene(0);
        }
        #endregion
        #region Public Methods
        //Metode yang dapat dipanggil secara bebas
        public void LeaveRoom(){
            PhotonNetwork.LeaveRoom();
        }
        #endregion

        #region Private Methods
        void Start()
        {
            if (PhotonNetwork.IsMasterClient) {Debug.Log("Master");}
```

```
            Debug.Log("Registered Time Delta Start " +
PhotonNetwork.ServerTimestamp);
            mulai = true;
            if (playerPrefab == null)
            {
                Debug.LogError("<Color=Red><a>Missing</a></Color> playerPrefab
Reference. Please set it up in GameObject 'Game Manager'", this);
            }
            else
            {
                if (PlayerManager.LocalPlayerInstance == null)
                {
                    Debug.LogFormat("We are Instantiating LocalPlayer from {0}",
SceneManagerHelper.ActiveSceneName);
                    // kita ada di kamar. menelurkan karakter untuk pemain lokal. itu
akan disinkronkan dengan menggunakan PhotonNetwork.Instantiate
                    respownPoint = PhotonNetwork.CurrentRoom.PlayerCount;
                    player =PhotonNetwork.Instantiate(
                        this.playerPrefab.name,
                        GameObject.Find("Respown"+
respownPoint).GetComponent<Transform>().position,
                        Quaternion.identity,
                        0
                        );
                }
                else
                {
                    Debug.LogFormat("Ignoring scene load for {0}",
SceneManagerHelper.ActiveSceneName);
                }
            }
        }
        private void FixedUpdate()
        {
            if (GameObject.Find("Environment").GetComponent<GamePlay>().mulaiPlay)
{
                TeleportPlayer();
            }

        }
        public void TeleportPlayer() {
            if (mulai)
            {
                string tempatPlayer = (PlayerPrefs.GetInt("TypeLabirin") == 1) ?
PlayerPrefs.GetInt("TypeLabirinDiff") + 1 + "" : "";
                player.transform.position = GameObject.Find("RespownPlayer" +
tempatPlayer +""+ respownPoint).GetComponent<Transform>().position;
                mulai = false;
            }
        }
        #endregion
    }
}
```

GamePlay.cs

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

```
namespace Hanafi
{
    public class GamePlay : MonoBehaviour
    {

        #region Field
        GameObject countDown,readykuy;
        float timeLeft = 100.0f;
        float timeLeftReady = 5.0f;
        public bool mulaiPlay;
        [SerializeField] GameObject aStart;
        [SerializeField] GameObject npc1, npc2, npc3;

        //Exekutor bool
        bool eksekusiSudah;
        #endregion
        private void Start()
        {
            countDown = GameObject.Find("CoolDown");
            readykuy= GameObject.Find("KeteranganUI");
            //demageEffect = GameObject.Find("BloodDmg");
            mulaiPlay = false;
            eksekusiSudah = false;
        }
        private void Update()
        {
            if (!eksekusiSudah)
            {
                if (PlayerPrefs.GetInt("DebugerMode") == 1)
                {
                    timeLeftReady -= Time.deltaTime;
                    countDown.GetComponent<Text>().text = "GO!!!";
                    if (timeLeftReady < 0)
                    {
                        GoPlay();
                    }
                }
                else if (PlayerPrefs.GetInt("DebugerMode") == 0)
                {
                    if (PhotonNetwork.CurrentRoom.PlayerCount == 1)
                    {
                        countDown.GetComponent<Text>().text = "....";
                        timeLeft = 100.0f;
                    }
                    else if (PhotonNetwork.CurrentRoom.PlayerCount == 2)
                    {
                        timeLeft -= Time.deltaTime;
                        countDown.GetComponent<Text>().text = "" +
Mathf.Round(timeLeft);
                        if (timeLeft < 0)
                        {
                            GoPlay();
                        }
                    }
                    else if (PhotonNetwork.CurrentRoom.PlayerCount == 3)
                    {
                        timeLeftReady -= Time.deltaTime;
                        countDown.GetComponent<Text>().text = "GO!!!";
                        if (timeLeftReady < 0)
                        {
                            GoPlay();
```

```
                    }
                }
            }
        }
    }
    void GoPlay() {
        countDown.SetActive(false);
        readykuy.SetActive(false);
        //Menutup koneksi player baru
        PhotonNetwork.CurrentRoom.IsOpen = false;
        //membuat room menjadi tidak terlihat oleh player baru
        PhotonNetwork.CurrentRoom.IsVisible = false;
        mulaiPlay = true;
        CreateNPC(PlayerPrefs.GetInt("NPCOnMap"));
        eksekusiSudah = true;
    }
    public void CreateNPC(int i){
        string tempatNPC =
            (PlayerPrefs.GetInt("TypeLabirin") == 1) ?
            PlayerPrefs.GetInt("TypeLabirinDiff") + 1 + "" : "";
        PhotonNetwork.InstantiateSceneObject(aStart.name, Vector3.zero,
Quaternion.identity);
        if (i == 1){
            PhotonNetwork.InstantiateSceneObject(
                npc1.name,
                GameObject.Find("RespownNPC"+
tempatNPC+"1").GetComponent<Transform>().position,
                Quaternion.identity);
        }else if (i == 2){
            PhotonNetwork.InstantiateSceneObject(
                npc1.name,
                GameObject.Find("RespownNPC" + tempatNPC +
"1").GetComponent<Transform>().position,
                Quaternion.identity);
            PhotonNetwork.InstantiateSceneObject(
                npc2.name,
                GameObject.Find("RespownNPC" + tempatNPC +
"2").GetComponent<Transform>().position,
                Quaternion.identity);
        }else if (i == 3){
            PhotonNetwork.InstantiateSceneObject(
                npc1.name,
                GameObject.Find("RespownNPC" + tempatNPC +
"1").GetComponent<Transform>().position,
                Quaternion.identity);
            PhotonNetwork.InstantiateSceneObject(
                npc2.name,
                GameObject.Find("RespownNPC" + tempatNPC +
"2").GetComponent<Transform>().position,
                Quaternion.identity);
            PhotonNetwork.InstantiateSceneObject(
                npc3.name,
                GameObject.Find("RespownNPC" + tempatNPC +
"3").GetComponent<Transform>().position,
                Quaternion.identity);
        }
    }


}
}
```

GenerateObject.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
public class GenerateObject : MonoBehaviourPunCallbacks
{
    #region Public Fields
    [Tooltip("Object anything")]
    public GameObject objectIni;
    #endregion

    private void Start()
    {
        if (objectIni == null)
        {
            Debug.LogError("<Color=Red><a>Missing</a></Color> Bola Reference.
Please set it up in GameObject 'Game Manager'", this);
        }
        else
        {
            PhotonNetwork.InstantiateSceneObject(this.objectIni.name, new
Vector3(0f, 30f, 0f), Quaternion.identity, 0);


        }
    }

}
```

Launcher.cs

```
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;
using ExitGames.Client.Photon;
namespace Hanafi
{
    public class Launcher : MonoBehaviourPunCallbacks
    {
        #region Private Serializable Fields

        #endregion

        #region Public Fields
        [Tooltip("The Ui Panel to let the user enter name, connect and play")]
        [SerializeField]
        private GameObject controlPanel;
        [Tooltip("The UI Label to inform the user that the connection is in progress")]
        [SerializeField]
        private GameObject progressLabel;
        [SerializeField]
        private GameObject menuRule;
        #endregion

        #region Private Fields
        /// <summary>
        /// This client's version number. Users are separated from each other by
gameVersion (which allows you to make breaking changes).
        /// </summary>
```

```
        string gameVersion = "1";
        /// <summary>
        /// The maximum number of players per room. When a room is full, it can't
be joined by new players, and so new room will be created.
        /// </summary>
        [Tooltip("The maximum number of players per room. When a room is full, it
can't be joined by new players, and so new room will be created")]
        [SerializeField]
        private byte maxPlayersPerRoom = 3;
        bool isConnecting;
        #endregion

        #region MonoBehaviour CallBacks
        /// <summary>
        /// MonoBehaviour method called on GameObject by Unity during early
initialization phase.
        /// </summary>
        void Awake()
        {
            // #Critical
            // this makes sure we can use PhotonNetwork.LoadLevel() on the master
client and all clients in the same room sync their level automatically
            PhotonNetwork.AutomaticallySyncScene = true;
        }
        private void Start()
        {
            progressLabel.SetActive(false);
            controlPanel.SetActive(true);
        }

        /// <summary>
        /// MonoBehaviour method called on GameObject by Unity during initialization
phase.
        /// </summary>
        #endregion

        #region MonoBehaviourPunCallbacks Callbacks


        public override void OnConnectedToMaster()
        {
            Debug.Log("PUN Basics Tutorial/Launcher: OnConnectedToMaster() was
called by PUN");
            // #Critical: Yang pertama kami coba lakukan adalah bergabung dengan ruang
potensial yang ada. Jika ada, bagus, lain, kita akan dipanggil kembali dengan
OnJoinRandomFailed ()
            if (isConnecting)
            {
                JoinKeRoom();
                isConnecting = false;
            }
        }

        void JoinKeRoom() {
            Hashtable prop=null;
            if (PlayerPrefs.GetInt("TypeLabirin") == 0)
            {
                prop = new Hashtable()
                {
                    { "tm", 0},
                    { "nc", PlayerPrefs.GetInt("NPCOnMap") }
```

```
            };
        }
        else if (PlayerPrefs.GetInt("TypeLabirin") == 1)
        {
            prop = new Hashtable()
            {
                { "tm", PlayerPrefs.GetInt("TypeLabirinDiff") + 1 },
                { "nc", PlayerPrefs.GetInt("NPCOnMap") }
            };
        }
        // #Critical: Yang pertama kami coba lakukan adalah bergabung dengan ruang potensial yang ada. Jika ada, bagus, lain, kita akan dipanggil kembali dengan OnJoinRandomFailed ()
        PhotonNetwork.JoinRandomRoom(prop, maxPlayersPerRoom);
    }

    public override void OnDisconnected(DisconnectCause cause)
    {
        progressLabel.SetActive(false);
        controlPanel.SetActive(true);
        Debug.LogWarningFormat("PUN Basics Tutorial/Launcher: OnDisconnected() was called by PUN with reason {0}", cause);
    }

    public override void OnJoinRandomFailed(short returnCode, string message)
    {
        Debug.Log("PUN Basics Tutorial/Launcher:OnJoinRandomFailed() was called by PUN. No random room available, so we create one.\nCalling: PhotonNetwork.CreateRoom");
        RoomOptions roomOptions = new RoomOptions();
        roomOptions.MaxPlayers = maxPlayersPerRoom;
        roomOptions.IsOpen = true;
        roomOptions.IsVisible = true;
        roomOptions.CustomRoomPropertiesForLobby = new string[] { "tm","nc" };
        if (PlayerPrefs.GetInt("TypeLabirin") == 0)
        {
            roomOptions.CustomRoomProperties = new Hashtable()
            {
                { "tm", 0},
                { "nc", PlayerPrefs.GetInt("NPCOnMap") }
            };
        }
        else if (PlayerPrefs.GetInt("TypeLabirin") == 1)
        {
            roomOptions.CustomRoomProperties = new Hashtable()
            {
                { "tm", PlayerPrefs.GetInt("TypeLabirinDiff")+1 },
                {"nc", PlayerPrefs.GetInt("NPCOnMap") }
            };
        }
        // #Critical: kami gagal bergabung dengan ruang acak, mungkin tidak ada atau semuanya penuh. Jangan khawatir, kami membuat ruangan baru.
        PhotonNetwork.CreateRoom(null, roomOptions, TypedLobby.Default);
    }

    public override void OnJoinedRoom()
    {
        Debug.Log("PUN Basics Tutorial/Launcher: OnJoinedRoom() called by PUN. Now this client is in a room.");
```

```
            // #Critical: Kami hanya memuat jika kami adalah pemain pertama, jika tidak,
kami mengandalkan `PhotonNetwork.AutomaticallySyncScene` untuk menyinkronkan
adegan instance kami.
            if (PhotonNetwork.CurrentRoom.PlayerCount == 1)
            {
                if (PlayerPrefs.GetInt("TypeLabirin") == 1)
                    PhotonNetwork.LoadLevel("Arena Dinamis");
                else
                    PhotonNetwork.LoadLevel("Arena Statis");
            }
        }

        #endregion

        #region Public Methods
        /// <summary>
        /// Start the connection process.
        /// - If already connected, we attempt joining a random room
        /// - if not yet connected, Connect this application instance to Photon Cloud
Network
        /// </summary>
        public void Connect()
        {
            if (PlayerPrefs.GetString("PlayerName").Trim().Equals(""))
            {
                menuRule.GetComponent<MainMenu>().Pengaturan();
            }
            else
            {
                isConnecting = PhotonNetwork.ConnectUsingSettings();
                progressLabel.SetActive(true);
                controlPanel.SetActive(false);
                Debug.Log("PUN Basics Tutorial/Launcher: Run connect method");
                // kami memeriksa apakah kami terhubung atau tidak, kami bergabung
jika kami terhubung, jika tidak kami memulai koneksi ke server.
                if (PhotonNetwork.IsConnected)
                {
                    Debug.Log("PUN Basics Tutorial/Launcher: Photon ready was
connected");
                    JoinKeRoom();
                }
                else
                {
                    Debug.Log("PUN Basics Tutorial/Launcher:
PhotonNetwork.IsConnected is not connected");
                    // #Critical, pertama-tama kita harus terhubung ke Photon Online
Server.
                    PhotonNetwork.ConnectUsingSettings();
                    PhotonNetwork.GameVersion = gameVersion;
                }
            }
        }
        #endregion
    }
}
```

LogSaverAndSender.cs

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

using System.IO;
```

```
using System.Net;
using System.Net.Mail;
using System.Net.Security;
using System.Security.Cryptography.X509Certificates;
using System;
using System.Text;
using UnityEngine.UI;

namespace Hanafi
{

    public class LogSaverAndSender : MonoBehaviour
    {
        public bool enableSave = true;
        public bool enableMailing = true;

        public string yourEmail = "fromemail@gmail.com";
        public string yourEmailPassword = "password";
        public string toEmail = "toemail@gmail.com";
        public string nama_folder = "data";

        private string name_file, temp_name;
        int NPCint = 0;

        [Serializable]
        public struct Logs
        {
            public string condition;
            public string stackTrace;
            public LogType type;

            public string dateTime;

            public Logs(string condition, string stackTrace, LogType type, string
dateTime)
            {
                this.condition = condition;
                this.stackTrace = stackTrace;
                this.type = type;
                this.dateTime = dateTime;
            }
        }

        [Serializable]
        public class LogInfo
        {
            public List<Logs> logInfoList = new List<Logs>();
        }

        LogInfo logs = new LogInfo();

        void OnEnable()
        {
            //Email last saved log
            if (enableMailing)
            {
                mailLog();
            }

            //Subscribe to Log Event
            Application.logMessageReceived += LogCallback;
```

```
        }

        //Called when there is an exception
        void LogCallback(string condition, string stackTrace, LogType type)
        {
            //Create new Log
            Logs logInfo = new Logs(condition, stackTrace, type,
DateTime.Now.ToString("yyyy-MM-ddTHH:mm:sszzz"));

            //Add it to the List
            logs.logInfoList.Add(logInfo);
        }

        void mailLog()
        {
            //Read old/last saved log
            LogInfo loadedData = DataSaver.loadData<LogInfo>(name_file,
nama_folder);
            string date = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:sszzz");

            //Send only if there is something to actually send
            if (loadedData != null && loadedData.logInfoList != null

                && loadedData.logInfoList.Count > 0)
            {

                Debug.Log("Found log to send!");

                //Convert to json
                string messageToSend = JsonUtility.ToJson(loadedData, true);

                string attachmentPath = Path.Combine(Application.persistentDataPath,
"data");
                attachmentPath = Path.Combine(attachmentPath, name_file);

                //Finally send email
                sendMail(yourEmail, yourEmailPassword, toEmail, "Log: " + date,
messageToSend, attachmentPath);

                //Clear old log
                DataSaver.deleteData(name_file, nama_folder);
            }
        }

        void sendMail(string fromEmail, string emaiPassword, string toEmail, string
eMailSubject, string eMailBody, string attachmentPath = null)
        {
            try
            {
                MailMessage mail = new MailMessage();

                mail.From = new MailAddress(fromEmail);
                mail.To.Add(toEmail);
                mail.Subject = eMailSubject;
                mail.Body = eMailBody;

                if (attachmentPath != null)
                {
                    System.Net.Mail.Attachment attachment = new
System.Net.Mail.Attachment(attachmentPath);
                    mail.Attachments.Add(attachment);
```

```
            }

            SmtpClient smtpClient = new SmtpClient();
            smtpClient.Host = "smtp.gmail.com";
            smtpClient.Port = 587;
            smtpClient.DeliveryMethod = SmtpDeliveryMethod.Network;
            smtpClient.Credentials = new System.Net.NetworkCredential(fromEmail,
emaiPassword) as ICredentialsByHost;
            smtpClient.EnableSsl = true;
            ServicePointManager.ServerCertificateValidationCallback =
                delegate (object s, X509Certificate certificate, X509Chain chain,
SslPolicyErrors sslPolicyErrors)
                { return true; };
            smtpClient.Send(mail);
        }
        catch (Exception e) { }
    }

    void OnDisable()
    {
        //Un-Subscribe from Log Event
        Application.logMessageReceived -= LogCallback;
    }
    /*
    //Save log  when focous is lost
    void OnApplicationFocus(bool hasFocus)
    {
        if (!hasFocus)
        {
            //Save
            if (enableSave)
                DataSaver.saveData(logs, name_file);
        }
    }
    //Save log on exit
    void OnApplicationPause(bool pauseStatus)
    {
        if (pauseStatus)
        {
            //Save
            if (enableSave)
                DataSaver.saveData(logs, name_file);
        }
    }
    */
    public void SimpanData()
    {
        if (enableSave)
            DataSaver.saveData(logs, name_file + " " + NPCint, nama_folder);
    }
    private void Start()
    {
        nama_folder = PlayerPrefs.GetString("PlayerName");
        NPCint = PlayerPrefs.GetInt("NPCOnMap");
        temp_name = "";
        if (PlayerPrefs.GetInt("TypeLabirin") == 1)
        {
            int difficult = PlayerPrefs.GetInt("TypeLabirinDiff");
            temp_name = temp_name + (difficult + 1);
        }
        else
```

```
            {
                temp_name = temp_name + "0";
            }
        }
        private void FixedUpdate()
        {
            name_file = temp_name + "\\" + DateTime.Now;
        }
        public class DataSaver
        {
            //Save Data
            public static void saveData<T>(T dataToSave, string dataFileName, string
nama_folder)
            {
                string tempPath = Path.Combine(Application.persistentDataPath,
nama_folder);
                tempPath = Path.Combine(tempPath, dataFileName + ".json");

                //Convert To Json then to bytes
                string jsonData = JsonUtility.ToJson(dataToSave, true);
                byte[] jsonByte = Encoding.ASCII.GetBytes(jsonData);

                //Create Directory if it does not exist
                if (!Directory.Exists(Path.GetDirectoryName(tempPath)))
                {
                    Directory.CreateDirectory(Path.GetDirectoryName(tempPath));
                }
                //Debug.Log(path);

                try
                {
                    File.WriteAllBytes(tempPath, jsonByte);
                    Debug.Log("Saved Data to: " + tempPath.Replace("/", "\\"));
                }
                catch (Exception e)
                {
                    Debug.LogWarning("Failed To PlayerInfo Data to: " +
tempPath.Replace("/", "\\"));
                    Debug.LogWarning("Error: " + e.Message);
                }
            }

            //Load Data
            public static T loadData<T>(string dataFileName, string nama_folder)
            {
                string tempPath = Path.Combine(Application.persistentDataPath,
nama_folder);
                tempPath = Path.Combine(tempPath, dataFileName + ".json");

                //Exit if Directory or File does not exist
                if (!Directory.Exists(Path.GetDirectoryName(tempPath)))
                {
                    Debug.LogWarning("Directory does not exist");
                    return default(T);
                }

                if (!File.Exists(tempPath))
                {
                    Debug.Log("File does not exist");
                    return default(T);
                }
```

```
            //Load saved Json
            byte[] jsonByte = null;
            try
            {
                jsonByte = File.ReadAllBytes(tempPath);
                Debug.Log("Loaded Data from: " + tempPath.Replace("/", "\\"));
            }
            catch (Exception e)
            {
                Debug.LogWarning("Failed To Load Data from: " +
tempPath.Replace("/", "\\"));
                Debug.LogWarning("Error: " + e.Message);
            }

            //Convert to json string
            string jsonData = Encoding.ASCII.GetString(jsonByte);

            //Convert to Object
            object resultValue = JsonUtility.FromJson<T>(jsonData);
            return (T)Convert.ChangeType(resultValue, typeof(T));
        }

        public static bool deleteData(string dataFileName, string nama_folder)
        {
            bool success = false;

            //Load Data
            string tempPath = Path.Combine(Application.persistentDataPath,
nama_folder);
            tempPath = Path.Combine(tempPath, dataFileName + ".json");

            //Exit if Directory or File does not exist
            if (!Directory.Exists(Path.GetDirectoryName(tempPath)))
            {
                Debug.LogWarning("Directory does not exist");
                return false;
            }

            if (!File.Exists(tempPath))
            {
                Debug.Log("File does not exist");
                return false;
            }

            try
            {
                File.Delete(tempPath);
                Debug.Log("Data deleted from: " + tempPath.Replace("/", "\\"));
                success = true;
            }
            catch (Exception e)
            {
                Debug.LogWarning("Failed To Delete Data: " + e.Message);
            }

            return success;
        }
    }
}
```

MainMenu.cs

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

public class MainMenu : MonoBehaviour
{
    public GameObject PanelMainMenu;
    public GameObject PanelSetting;
    // Start is called before the first frame update
    void Start()
    {
        PanelMainMenu.SetActive(true);
        PanelSetting.SetActive(false);
    }
    public void Pengaturan() {
        PanelMainMenu.SetActive(false);
        PanelSetting.SetActive(true);
    }
    public void Kembali() {
        PanelMainMenu.SetActive(true);
        PanelSetting.SetActive(false);
    }
    // Update is called once per frame
    void Update()
    {

    }
    public void ExitGame() {
        Application.Quit();
    }
}
```

MenuScript.cs

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


namespace Hanafi
{
    public class MenuScript : MonoBehaviourPunCallbacks
    {
        [SerializeField]
        private GameObject MenuPanel;
        //[SerializeField]
        //private GameObject MasterPanel;
        [SerializeField]
        private GameObject labirinGenerate;
        // Start is called before the first frame update
        void Start()
        {

            MenuPanel.SetActive(false);
            if (PhotonNetwork.IsMasterClient)
            {
                if (PlayerPrefs.GetInt("TypeLabirin") == 1)
                {
                    //MasterPanel.SetActive(true);
```

```
                int difficult = PlayerPrefs.GetInt("TypeLabirinDiff");
                if (difficult == 0)
                {
                    labirinGenerate.GetComponent<MazeGenerator>().Simpel();
                }
                else if (difficult == 1)
                {
                    labirinGenerate.GetComponent<MazeGenerator>().Sedang();
                }
                else if (difficult == 2)
                {
                    labirinGenerate.GetComponent<MazeGenerator>().Kompleks();
                }
            }
        }
        //else MasterPanel.SetActive(false);

    }

    // Update is called once per frame

    public void OpenMenuPanel()
    {
        MenuPanel.SetActive(true);
    }
    public void CloseMenuPanel()
    {
        MenuPanel.SetActive(false);
    }
    public void CloseMasterPanel()
    {
        //MasterPanel.SetActive(false);
    }
}
}
```

NamaPlayerNow.cs

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

public class NamePlayerNow : MonoBehaviour
{
    const string playerNamePrefKey = "PlayerName";
    Text textValue;
    string defaultName;
    // Start is called before the first frame update
    void Start()
    {
        defaultName = string.Empty;
        textValue = this.GetComponent<Text>();
        if (textValue != null)
        {
            if (PlayerPrefs.HasKey(playerNamePrefKey))
            {
                defaultName = PlayerPrefs.GetString(playerNamePrefKey);
                textValue.text = defaultName;
            }
        }
    }
```

```
    // Update is called once per frame
    void FixedUpdate()
    {
        if (textValue != null)
        {
            if (PlayerPrefs.HasKey(playerNamePrefKey))
            {
                defaultName = PlayerPrefs.GetString(playerNamePrefKey);
                textValue.text = defaultName;
            }
        }
    }
}
```

NavigationVirtualJoystick.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class NavigationVirtualJoystick : MonoBehaviour, IDragHandler,
IPointerUpHandler, IPointerDownHandler
{
    private Image bgImg;
    private Image joy;
    public Vector3 InputDirection { set; get; }
    // Start is called before the first frame update
    void Start()
    {
        bgImg = GetComponent<Image>();
        joy = transform.GetChild(0).GetComponent<Image>();
        InputDirection = Vector3.zero;
    }

    // Update is called once per frame

    private void Awake()
    {
        this.transform.SetParent(GameObject.Find("Control Character
Panel").GetComponent<Transform>(), false);
    }
    public virtual void OnDrag(PointerEventData ped)
    {
        Vector2 pos = Vector2.zero;
        if
(RectTransformUtility.ScreenPointToLocalPointInRectangle(bgImg.rectTransform,
ped.position, ped.pressEventCamera, out pos))
        {
            pos.x = (pos.x / bgImg.rectTransform.sizeDelta.x);
            pos.y = (pos.y / bgImg.rectTransform.sizeDelta.y);

            float x = (bgImg.rectTransform.pivot.x == 1) ? pos.x * 2 + 1 : pos.x *
2 - 1;
            float y = (bgImg.rectTransform.pivot.y == 1) ? pos.y * 2 + 1 : pos.y *
2 - 1;
            InputDirection = new Vector3(x, 0, y);
            InputDirection = (InputDirection.magnitude > 1) ?
InputDirection.normalized : InputDirection;
            joy.rectTransform.anchoredPosition = new Vector3(InputDirection.x *
(bgImg.rectTransform.sizeDelta.x / 3), InputDirection.z *
(bgImg.rectTransform.sizeDelta.y / 3));
```

```
        }
        //Debug.Log("OnDrag");
    }
    public virtual void OnPointerUp(PointerEventData ped)
    {
        InputDirection = Vector3.zero;
        joy.rectTransform.anchoredPosition = Vector3.zero;

    }
    public virtual void OnPointerDown(PointerEventData ped)
    {
        OnDrag(ped);

    }
}
```

PengukurWaktu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Pathfinding;
using Photon.Pun;
using System;

namespace Hanafi
{
    public class PengukurWaktu : MonoBehaviour
    {
        float time_start;
        float target_curent;
        AIDestinationSetter aIdestination;
        int debugMode;
        bool sampai;
        byte now_target_num;
        // Start is called before the first frame update
        void Start()
        {
            debugMode = PlayerPrefs.GetInt("DebugerMode");
            time_start = PhotonNetwork.ServerTimestamp;
            aIdestination = gameObject.GetComponent<AIDestinationSetter>();
            sampai = false;
            now_target_num = 0;
        }

        // Update is called once per frame
        void FixedUpdate()
        {
            if (debugMode == 1)
            {
                if (now_target_num != aIdestination.numberTarget) {
                    time_start = PhotonNetwork.ServerTimestamp;
                    now_target_num = aIdestination.numberTarget;
                }
                target_curent = aIdestination.remainingToTarget;
                if (target_curent < 1 && sampai==false)
                {
                    Debug.Log(gameObject.name+" need time "+Math.Abs((time_start -
PhotonNetwork.ServerTimestamp) / 1000)+"s ");
                    sampai = true;
                }
                if (target_curent > 1) {
```

```
                    sampai = false;
                }
            }
        }
    }
}
```

PengukurWaktuNPC.cs

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

namespace Hanafi
{
    public class PengukurWaktuNPC : MonoBehaviourPunCallbacks, IPunObservable
    {
        private uint count_delay = 0;
        Text status_ping;
        long ping = 0;

        void Start() {
            status_ping = GameObject.Find("Ping").GetComponent<Text>();
        }
        private void FixedUpdate()
        {
            ping = long.Parse(status_ping.text);
        }
        public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo
info)
        {
            if (stream.IsWriting)
            {
            }
            else
            {
                float lag = Mathf.Abs((float)(PhotonNetwork.Time - info.timestamp));
                count_delay++;
                string fill_log = "Registered NPC "+this.gameObject.name+" " +
count_delay + " " + ((int)(lag * 1000f)) + " " + ping;
                Debug.Log(fill_log);
            }
        }
    }
}
```

PlayerAnimatorManager.cs

```
using UnityEngine;
using System.Collections;
using Photon.Pun;
using Photon.Realtime;


namespace Hanafi
{
    public class PlayerAnimatorManager : MonoBehaviourPun
    {

        #region Private Fields
```

```
        [SerializeField]
        private float directionDampTime = 0.25f;

        private NavigationVirtualJoystick joy;
        private Animator animator;
        // Use this for initialization

        #endregion

        #region MonoBehaviour Callbacks


        private void Awake()
        {
            if (photonView.IsMine == false && PhotonNetwork.IsConnected == true)
            {
                return;
            }
            joy =
GameObject.FindGameObjectWithTag("NavigationVirtualJoystick").GetComponent<Navi
gationVirtualJoystick>();

        }
        // Use this for initialization
        void Start()
        {
            animator = GetComponent<Animator>();
            if (!animator)
            {
                Debug.LogError("PlayerAnimatorManager is Missing Animator Component",
this);
            }
        }


        // Update is called once per frame
        void Update()
        {
            if (photonView.IsMine == false && PhotonNetwork.IsConnected == true)
            {
                return;
            }
            if (!animator)
            {
                return;
            }
            //float h = Input.GetAxis("Horizontal");
            float v = joy.InputDirection.z;

            float h = joy.InputDirection.x;
            Debug.Log(h+"----------------"+v);
            if (v < 0)
            {
                //v = 0;
                animator.SetBool("BackSteep", true);
            }else animator.SetBool("BackSteep", false);
            animator.SetFloat("Speed", h * h + v * v);
            animator.SetFloat("Direction", h, directionDampTime, Time.deltaTime);
        }
```

```
        #endregion

    }
}
```

PlayerManager.cs

```
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
using Photon.Pun;

using Pathfinding;
using System.Collections;
using UnityEngine.SceneManagement;
using System;

namespace Hanafi
{

    public class PlayerManager : MonoBehaviourPunCallbacks, IPunObservable
    {
        #region Private Fields
        private Vector3 moveDirection = Vector3.zero;
        CharacterController characterController;
        [SerializeField]
        private GameObject joy;
        [SerializeField]
        private GameObject angle;
        [SerializeField]
        private GameObject actionButton1;
        [SerializeField]
        private GameObject hit;
        private NavigationVirtualJoystick joy_v;
        private AngleVirtualJoystick angle_v;
        private TapButton actionButton1_v;
        private Animator animator;
        private float directionDampTime = 0.25f;
        private uint count_delay;
        private float total_delay;
        private Text status_syns;
        private string string_avg;
        private bool hit_v;
        bool recoverHP,sudahSave;
        GamePlay gamePlay;
        GameObject _uiGo, _uiAng, _uiJoy, _uiAcB, _uiGoOther;
        private Text status_ping;
        #endregion

        #region Public Fields
        [Tooltip("The current Health of our player")]
        public float Health = 1f;
        [Tooltip("The local player instance. Use this to know if the local player
is represented in the Scene")]
        public static GameObject LocalPlayerInstance;
        [Tooltip("The Player's UI GameObject Prefab")]
        [SerializeField]
        public GameObject playerHp;
        [SerializeField]
        public GameObject OtherHp;
```

```
        public float speed = 6.0f;
        public float gravity = 20.0f;
        float timeLeft = 10.0f;
        private int debugMode=0;
        int ping=0;
        LogSaverAndSender logSave;

        //UI untuk menampilkan Game Over
        GameObject hpCritical;
        GameObject panelWinner;
        GameObject death;
        Text winnerCountDown, deathCountDown;

        //bagian ini untuk menampilkan jarak Player dengan NPC secara radius
        Text remNPC1, remNPC2, remNPC3;
        GameObject[] NPC;
        #endregion

        #region MonoBehaviour CallBacks
        void Awake()
        {
            // #Important
            // used in GameManager.cs: we keep track of the localPlayer instance to
prevent instantiation when levels are synchronized
            if (photonView.IsMine)
            {
                PlayerManager.LocalPlayerInstance = this.gameObject;
            }
            // #Critical
            // we flag as don't destroy on load so that instance survives level
synchronization, thus giving a seamless experience when levels load.
            DontDestroyOnLoad(this.gameObject);
        }

        void Start()
        {
            logSave = GameObject.Find("Log
Saver").GetComponent<LogSaverAndSender>();
            sudahSave = false;
            debugMode = PlayerPrefs.GetInt("DebugerMode");
            recoverHP = true;
            hit_v = false;
            total_delay = 0;
            count_delay = 0;
            CameraForwad _cameraWork =
this.gameObject.GetComponent<CameraForwad>();
            characterController = GetComponent<CharacterController>();
            PhotonNetwork.NickName = PlayerPrefs.GetString("PlayerName");
            if (photonView.IsMine && actionButton1 != null && joy != null && angle !=
null
                && playerHp != null && _cameraWork != null) {
                //membuat HP bar curret Player
                _uiGo = Instantiate(playerHp);
                //membuat controller
                _uiAng = Instantiate(angle);
                _uiJoy = Instantiate(joy);
                _uiAcB = Instantiate(actionButton1);
                _cameraWork.OnStartFollowing();
                _uiGo.SendMessage("SetTarget", this,
SendMessageOptions.RequireReceiver);
                joy_v = _uiJoy.GetComponent<NavigationVirtualJoystick>();
```

```
            angle_v = _uiAng.GetComponent<AngleVirtualJoystick>();
            actionButton1_v = _uiAcB.GetComponent<TapButton>();
        }
        else if (OtherHp != null)
        {
            //membuat tampilan HP bar di Player lain
            _uiGoOther = Instantiate(OtherHp);
            _uiGoOther.SendMessage("SetTarget", this,
SendMessageOptions.RequireReceiver);
        }
        animator = GetComponent<Animator>();
        if (!animator)
        {
            Debug.LogError("PlayerAnimatorManager is Missing Animator Component",
this);
        }
        if (photonView.IsMine) {
            gamePlay = GameObject.Find("Environment").GetComponent<GamePlay>();
            hpCritical = GameObject.Find("BloodDmg");
            death = GameObject.Find("Game Over");
            panelWinner = GameObject.Find("WinnerPanel");
            winnerCountDown =
GameObject.Find("WinnerCountDown").GetComponent<Text>();
            deathCountDown = GameObject.Find("Button Return to
Menu").GetComponent<Text>();
            hpCritical.SetActive(false);
            death.SetActive(false);
            panelWinner.SetActive(false);

            //bagian radar NPC
            remNPC1 = GameObject.Find("RemNPC1").GetComponent<Text>();
            remNPC2 = GameObject.Find("RemNPC2").GetComponent<Text>();
            remNPC3 = GameObject.Find("RemNPC3").GetComponent<Text>();
        }
        status_ping = GameObject.Find("Ping").GetComponent<Text>();
        status_syns = GameObject.Find("Time Latency" +
photonView.Owner.ActorNumber).GetComponent<Text>();
        string_avg = "0";
    }
    void Update()
    {
        if (photonView.IsMine)
        {
            if (GameObject.FindGameObjectsWithTag("Winner").Length != 0
&& !gameObject.tag.Trim().Equals("Winner"))
            {
                MeGameOver();
                return;
            }
            if (gameObject.tag.Trim().Equals("Winner"))
            {
                SimpanData();
                timeLeft -= Time.deltaTime;
                winnerCountDown.text = Mathf.Round(timeLeft) + "";
                panelWinner.SetActive(true);
                if (timeLeft < 0)
                {
                    PhotonNetwork.LeaveRoom();
                }
                return;
            }
```

```
            if (gameObject.tag.Trim().Equals("Player Death"))
            {

                timeLeft -= Time.deltaTime;
                deathCountDown.text = "To Menu (" + Mathf.Round(timeLeft) + ")";
                if (timeLeft < 0)
                {
                    PhotonNetwork.LeaveRoom();
                }
                return;
            }
        }
        ProcessInputs();
        hit.SetActive(hit_v);
    }
    void SimpanData() {
        if (!sudahSave)
        {
            logSave.SimpanData();
            sudahSave = true;
        }
    }
    private void FixedUpdate()
    {
        ping = PhotonNetwork.GetPing();
        if (photonView.IsMine)
        {
            if (ping > 300) status_ping.color = Color.red;
            else if (ping > 100) status_ping.color = Color.yellow;
            else if (ping > 50) status_ping.color = Color.white;
            else if (ping > 0) status_ping.color = Color.green;
            status_ping.text = ping + "";
            if (debugMode == 0)
            {
                RemeberSetUI();
            }
            if (gamePlay.mulaiPlay)
            {
                RecoverHp();
            }
            if (Health < 0.2f && Health > 0)
            {
                Berdarah(true);
            }
            else if (Health <= 0f)
            {
                MeGameOver();
            }
            /*
            // bagian auto cover HP ketika HP 50% digunakan untuk membuat player
tidak bisam mati
            if (debugMode == 1 && Health<0.5) {
                Health = 1;
            }
            */
        }
    }

    //Penerima demage
    void OnTriggerEnter(Collider other){
        if (other.tag.Trim().Equals("Finish")) {
```

```
            //script untuk finish
            gameObject.tag = "Winner";
            speed = 0;
        }
        if (!photonView.IsMine){return;}
        else if (photonView.IsMine && other.name.Trim().Equals("Hit")){
            if (gameObject.tag.Equals("Player")){
                Health -= 0.2f * Time.deltaTime;
            }
        }
    }
    void OnTriggerStay(Collider other){
        if (!photonView.IsMine){return;}
        else if (photonView.IsMine &&  other.tag.Trim().Equals("Robot")){
            if (gameObject.tag.Equals("Player")){
                Health -= 0.1f * Time.deltaTime;
            }
        }
    }
    #endregion

    #region Custom
    void ProcessInputs()
    {
        if (photonView.IsMine == false && PhotonNetwork.IsConnected == true)
return;
        if (!animator)return;

        float v = joy_v.InputDirection.z;
        float h = joy_v.InputDirection.x;
        float r = angle_v.InputDirection.x;

        if (characterController.isGrounded)
        {

            moveDirection = new Vector3(h, 0.0f, v);
            moveDirection *= speed;
            moveDirection.y -= gravity * Time.deltaTime;

characterController.Move(transform.TransformDirection(moveDirection *
Time.deltaTime*speed));
            characterController.transform.Rotate(0,r* speed*2f, 0);
            if (v < 0)
            {
                animator.SetBool("BackSteep", true);
            }
            else animator.SetBool("BackSteep", false);
            animator.SetFloat("Speed", h * h + v * v);
            animator.SetFloat("Direction", h, directionDampTime,
Time.deltaTime);
            hit_v = actionButton1_v.InputAction;
        }
    }
    void RemeberSetUI() {
        NPC = GameObject.FindGameObjectsWithTag("Robot");
        if (NPC.Length == 1)
        { // jika NPC hanya satu
            remNPC1.text = Math.Round(Jarak(NPC[0].GetComponent<Transform>(),
gameObject.transform))+" : Robot1";
        }
        else if (NPC.Length == 2)
```

```
          { // jika NPC berjumlah 2
              remNPC1.text = Math.Round(Jarak(NPC[0].GetComponent<Transform>(),
gameObject.transform)) + " : Robot1";
              remNPC2.text = Math.Round(Jarak(NPC[1].GetComponent<Transform>(),
gameObject.transform)) + " : Robot2";
          }
          else if (NPC.Length == 3)
          { // Jika NPC berjumlah 3
              remNPC1.text = Math.Round(Jarak(NPC[0].GetComponent<Transform>(),
gameObject.transform))+ " : Robot1";
              remNPC2.text = Math.Round(Jarak(NPC[1].GetComponent<Transform>(),
gameObject.transform)) + " : Robot2";
              remNPC3.text = Math.Round(Jarak(NPC[2].GetComponent<Transform>(),
gameObject.transform))+ " : Robot3";
          }
      }
      double Jarak(Transform position1, Transform position2)
      {
          double a1 = position1.position.x;
          double b1 = position1.position.z;
          double a2 = position2.position.x;
          double b2 = position2.position.z;
          double tmp = b1 - b2;
          double dist = (Math.Sin(a1 * Mathf.Deg2Rad) * Math.Sin(a2 * Mathf.Deg2Rad))
+ (Math.Cos(a1 * Mathf.Deg2Rad) * Math.Cos(a2 * Mathf.Deg2Rad) * Math.Cos(tmp *
Mathf.Deg2Rad));
          dist = Math.Acos(dist);
          dist = dist * Mathf.Rad2Deg;
          return (dist);
      }
      public void RecoverHp()
      {
          if (recoverHP)
          {
              this.Health = 1.0f;
              recoverHP = false;
          }
      }
      void Berdarah(bool i)
      {
          if (photonView.IsMine)
          {
              hpCritical.SetActive(i);
          }
      }

      void Mati(bool i)
      {
          if (photonView.IsMine)
          {
              death.SetActive(i);
          }
      }
      void MeGameOver() {
          SimpanData();
          speed = 0f;
          Berdarah(false);
          Mati(true);
          if (!gameObject.tag.Equals("Player Death"))
          {
              _uiAng.SetActive(false);
```

```
                _uiJoy.SetActive(false);
                _uiAcB.SetActive(false);
                gameObject.tag = "Player Death";
            }
        }
        /*
        void CalledOnLevelWasLoaded() {
            GameObject _uiGo = Instantiate(this.PlayerUiHpPrefab);
            _uiGo.SendMessage("SetTarget", this,
SendMessageOptions.RequireReceiver);
            GameObject _uiJoy = Instantiate(this.joy);
            //_uiJoy.SendMessage("SetTarget", this,
SendMessageOptions.RequireReceiver);
            GameObject _uiAng = Instantiate(this.angle);
            //_uiAng.SendMessage("SetTarget", this,
SendMessageOptions.RequireReceiver);

        }
        */
        #endregion
        #region IPunObservable implementation
        public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo
info){
            if (stream.IsWriting){
                // We own this player: send the others our data
                //mengirim data attack
                stream.SendNext(hit_v);
                //mengirim data HP
                stream.SendNext(Health);
                //mengirim data tag
                stream.SendNext(gameObject.tag);
            }else{
                //menerima data attack
                hit_v = (bool)stream.ReceiveNext();
                //menrima data HP
                this.Health = (float)stream.ReceiveNext();
                //menghitung waktu yang dibutuhkan untuk mengirim data
                float lag = Mathf.Abs((float)(PhotonNetwork.Time - info.timestamp));
                count_delay++;
                string fill_log = "Registered Name "
                    + this.photonView.Owner.NickName + " "
                    + count_delay + " " + ((int)(lag * 1000f)) + " "
                    + ping;
                Debug.Log(fill_log);
                status_syns.text = fill_log;
                //menerima data tag
                this.gameObject.tag = (string) stream.ReceiveNext();
            }
        }

        #endregion
    }
}
```

PlayerNameInputField.cs

```
using UnityEngine;
using UnityEngine.UI;
using Photon.Pun;
using Photon.Realtime;
using System.Collections;
using Pathfinding;
```

```
namespace Hanafi
{
    /// <summary>
    /// Bidang masukan nama pemain. Biarkan pengguna memasukkan namanya, akan muncul
di atas pemain dalam game.
    /// </summary>
    [RequireComponent(typeof(InputField))]
    public class PlayerNameInputField : MonoBehaviour
    {
        #region Private Constants
        // Simpan Kunci PlayerPref untuk menghindari kesalahan ketik
        const string playerNamePrefKey = "PlayerName";
        #endregion
        #region MonoBehaviour CallBacks
        // Metode MonoBehaviour memanggil GameObject oleh Unity selama fase
inisialisasi.
        void Start()
        {
            string defaultName = string.Empty;
            InputField _inputField = this.GetComponent<InputField>();
            if (_inputField != null)
            {
                if (PlayerPrefs.HasKey(playerNamePrefKey))
                {
                    defaultName = PlayerPrefs.GetString(playerNamePrefKey);
                    _inputField.text = defaultName;
                }
            }
            PhotonNetwork.NickName = defaultName;
        }
        #endregion
        #region Public Methods
        // Menetapkan nama pemain, dan menyimpannya di PlayerPrefs untuk sesi
selanjutnya.
        // <param name = "value"> Nama Pemain </param>
        public void SetPlayerName(string value)
        {
            // #Important
            if (string.IsNullOrEmpty(value)){return;}
            PhotonNetwork.NickName = value;
            PlayerPrefs.SetString(playerNamePrefKey, value);
        }
        #endregion
    }
}
```

PlayerUIHP.cs

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


namespace Hanafi
{
    public class PlayerUIHP : MonoBehaviour
    {
        #region Private Fields
```

```
        [Tooltip("UI Text to display Player's Name")]
        [SerializeField]
        private Text playerNameText;


        [Tooltip("UI Slider to display Player's Health")]
        [SerializeField]
        private Slider playerHealthSlider;
        private PlayerManager target;
        float characterControllerHeight = 0f;
        Transform targetTransform;
        Renderer targetRenderer;
        CanvasGroup _canvasGroup;
        Vector3 targetPosition;
        private GameObject HPOther1,HPOther2;
        #endregion

        #region Public Field
        [Tooltip("Pixel offset from the player target")]
        [SerializeField]
        private Vector3 screenOffset = new Vector3(0f, 30f, 0f);
        #endregion

        #region MonoBehaviour Callbacks

        void Update()
        {
            if (target == null)
            {
                Destroy(this.gameObject);
                return;
            }
            // Reflect the Player Health
            if (playerHealthSlider != null)
            {
                playerHealthSlider.value = target.Health;
            }

            //Debug.Log("-----------------> " + HPOther1.transform.childCount);
        }

        void Awake()
        {

            _canvasGroup = this.GetComponent<CanvasGroup>();
        }
        private void Start()
        {
            HPOther1 = GameObject.Find("HP Other Player 1");
            HPOther2 = GameObject.Find("HP Other Player 2");
            if (target.photonView.IsMine)
            {

this.transform.SetParent(GameObject.Find("UI").GetComponent<Transform>(),
false);
            }
            else
            {
                if (HPOther1.transform.childCount == 0)
                {
```

```
                this.transform.SetParent(GameObject.Find("HP Other Player
1").GetComponent<Transform>(), false);
            }else if (HPOther1.transform.childCount == 1)
            {
                this.transform.SetParent(GameObject.Find("HP Other Player
2").GetComponent<Transform>(), false);
            }

        }
    }
    void LateUpdate()
    {
        // Do not show the UI if we are not visible to the camera, thus avoid
potential bugs with seeing the UI, but not the player itself.
        if (targetRenderer != null)
        {
            this._canvasGroup.alpha = targetRenderer.isVisible ? 1f : 0f;
        }
    }
    #endregion


    #region Public Methods

    public void SetTarget(PlayerManager _target)
    {
        if (_target == null)
        {
            Debug.LogError("<Color=Red><a>Missing</a></Color> PlayMakerManager
target for PlayerUI.SetTarget.", this);
            return;
        }
        // Cache references for efficiency
        target = _target;
        if (playerNameText != null)
        {
            playerNameText.text = target.photonView.Owner.NickName;
        }
        targetTransform = this.target.GetComponent<Transform>();
        targetRenderer = this.target.GetComponent<Renderer>();
        CharacterController characterController =
_target.GetComponent<CharacterController>();
        // Get data from the Player that won't change during the lifetime of this
Component
        if (characterController != null)
        {
            characterControllerHeight = characterController.height;
        }
    }

    #endregion


}
}
```

SettingGamePlay.cs

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

```
using UnityEngine.UI;

public class SettingGamePlay : MonoBehaviour
{

    // Start is called before the first frame update



    public void AmbilSetting()
    {
        //Dijadikan local variable karena menghindari error saat pertama aplikasi
dijalankan akan memanggil object yang sebenarnya belum diaktifkan
        Toggle debugi = GameObject.Find("Debug").GetComponent<Toggle>();
        Toggle statis = GameObject.Find("Statis").GetComponent<Toggle>();
        Toggle dinamis = GameObject.Find("Dinamis").GetComponent<Toggle>();
        Dropdown diff = GameObject.Find("Diff").GetComponent<Dropdown>();
        Dropdown npc = GameObject.Find("NPC_Bot").GetComponent<Dropdown>();

        if (PlayerPrefs.GetInt("DebugerMode")==1)
        {
            debugi.isOn=true;
        }else if (PlayerPrefs.GetInt("DebugerMode") == 0)
        {
            debugi.isOn = false;
        }
        if (PlayerPrefs.GetInt("TypeLabirin") == 0)
        {
            statis.isOn = true;
            dinamis.isOn = false;
        }
        else if (PlayerPrefs.GetInt("TypeLabirin") == 1)
        {
           statis.isOn = false;
           dinamis.isOn = true;
           diff.value= PlayerPrefs.GetInt("TypeLabirinDiff");

        }
        npc.value = PlayerPrefs.GetInt("NPCOnMap")-1;

    }
    public void WriteString()
    {
        //Dijadikan local variable karena menghindari error saat pertama aplikasi
dijalankan akan memanggil object yang sebenarnya belum diaktifkan
        Toggle statis = GameObject.Find("Statis").GetComponent<Toggle>();
        Toggle debugi = GameObject.Find("Debug").GetComponent<Toggle>();
        Dropdown npc = GameObject.Find("NPC_Bot").GetComponent<Dropdown>();
        Dropdown diff = GameObject.Find("Diff").GetComponent<Dropdown>();

        if (debugi.isOn)
        {
            PlayerPrefs.SetInt("DebugerMode", 1);
        }
        else {
            PlayerPrefs.SetInt("DebugerMode", 0);
        }
        if (statis.isOn)
        {
            PlayerPrefs.SetInt("TypeLabirin", 0);
        }
```

```
        else {
            PlayerPrefs.SetInt("TypeLabirin", 1);
            if (diff.options[diff.value].text.Equals("Simpel"))
            {
                PlayerPrefs.SetInt("TypeLabirinDiff", 0);
            }
            else if (diff.options[diff.value].text.Equals("Medium"))
            {
                PlayerPrefs.SetInt("TypeLabirinDiff", 1);
            }
            else if (diff.options[diff.value].text.Equals("Kompleks"))
            {
                PlayerPrefs.SetInt("TypeLabirinDiff", 2);
            }
        }

        if (npc.options[npc.value].text.Equals("1 NPC"))
        {
            PlayerPrefs.SetInt("NPCOnMap", 1);
        }
        else if (npc.options[npc.value].text.Equals("2 NPC"))
        {
            PlayerPrefs.SetInt("NPCOnMap", 2);
        }
        else if (npc.options[npc.value].text.Equals("3 NPC"))
        {
            PlayerPrefs.SetInt("NPCOnMap", 3);
        }
    }


}
```

TapButton.cs

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

public class TapButton : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{

    public bool InputAction { set; get; }

    private void Awake()
    {
        this.transform.SetParent(GameObject.Find("Control Character
Panel").GetComponent<Transform>(), false);

    }
    // Start is called before the first frame update


    // Update is called once per frame

    public void OnPointerDown(PointerEventData eventData)
    {
        InputAction = true;
        Debug.Log("Input Action true");
    }
```

```
    public void OnPointerUp(PointerEventData eventData)
    {
        InputAction = false;
        Debug.Log("Input Action false");
    }
}
```

TransformFollowers.cs

```
using UnityEngine;
using System.Collections;

public class TransformFollower : MonoBehaviour
{
    [SerializeField]
    private Transform target;

    [SerializeField]
    private Vector3 offsetPosition;

    [SerializeField]
    private Space offsetPositionSpace = Space.Self;

    [SerializeField]
    private bool lookAt = true;

    private void Update()
    {
        Refresh();
    }

    public void Refresh()
    {
        if (target == null)
        {
            Debug.LogWarning("Missing target ref !", this);

            return;
        }

        // compute position
        if (offsetPositionSpace == Space.Self)
        {
            transform.position = target.TransformPoint(offsetPosition);
        }
        else
        {
            transform.position = target.position + offsetPosition;
        }

        // compute rotation
        if (lookAt)
        {
            transform.LookAt(target);
        }
        else
        {
            transform.rotation = target.rotation;
        }
    }
}
```