DAFTAR PUSTAKA

Christina. (n.d.). *Pengertian Animasi 3D*. Retrieved from Academia: https://www.academia.edu/RegisterToDownload/BulkDownload

Darmawan, J. (2009). Google *SketchUp* Mudah dan Cepat Menggambar 3Dimensi.

Faiztyan, I. F., Isnanto, R. R., & Widianto, D. E. (2015). Perancangan dan Pembuatan Aplikasi Visualisasi 3D Interaktif Mesjid Agung jawa Tengah Menggunakan *Unity* 3D. *Jurnal Teknologi dan Sistem Komputer*, 207-212.

Hendrawan, S. A., Isnanto, R., & Windasari, I. P. (2015). Aplikasi Visualisasi 3D Pada Struktur Sistem Rangka Manusia Berbasis Android. *Jurnal Teknologi dan Sistem Komputer* , 426-435.

Lintrami, T. (2018). *Unity 2017 Game Development Essentials Third Edition.* Birmingham: Packt Publisher.

Schreyer, A., & Hoque, S. (2009). Interactive 3D Visualization of Building Envelope Systems Using Infrared Thermography and SktechUp. *Inframation*.

Setiawan, S. I. (2011). Google *SketchUp* Perangkat Alternatif dan Permodelan 3D. *Ultimatics Vol.3*, 6-10.

Vaughan, T. (1997). Multimedia: Making it Work. *Technical Communication*, 82-84.

Yudana. (n.d.). *Pengertian Dan Konsep Restful API Programming*. Retrieved from https://www.yudana.id/pengertian-dan-konsep-restful-api-programming/

LAMPIRAN

Kode Program *ThirdPersonCharacter*

```
using UnityEngine;

namespace UnityStandardAssets.Characters.ThirdPerson
{
	[RequireComponent(typeof(Rigidbody))]
	[RequireComponent(typeof(CapsuleCollider))]
	[RequireComponent(typeof(Animator))]
	public class ThirdPersonCharacter : MonoBehaviour
	{
		[SerializeField] float m_MovingTurnSpeed = 360;
		[SerializeField] float m_StationaryTurnSpeed = 180;
		[SerializeField] float m_JumpPower = 12f;
		[Range(1f, 4f)][SerializeField] float m_GravityMultiplier = 2f;
		[SerializeField] float m_RunCycleLegOffset = 0.2f;
		[SerializeField] float m_MoveSpeedMultiplier = 1f;
		[SerializeField] float m_AnimSpeedMultiplier = 1f;
		[SerializeField] float m_GroundCheckDistance = 0.1f;

		Rigidbody m_Rigidbody;
		Animator m_Animator;
		bool m_IsGrounded;
		float m_OrigGroundCheckDistance;
		const float k_Half = 0.5f;
		float m_TurnAmount;
		float m_ForwardAmount;
		Vector3 m_GroundNormal;
		float m_CapsuleHeight;
		Vector3 m_CapsuleCenter;
		CapsuleCollider m_Capsule;
		bool m_Crouching;
```

```
        void Start()
        {
                m_Animator = GetComponent<Animator>();
                m_Rigidbody = GetComponent<Rigidbody>();
                m_Capsule = GetComponent<CapsuleCollider>();
                m_CapsuleHeight = m_Capsule.height;
                m_CapsuleCenter = m_Capsule.center;

                m_Rigidbody.constraints =
RigidbodyConstraints.FreezeRotationX | RigidbodyConstraints.FreezeRotationY |
RigidbodyConstraints.FreezeRotationZ;
                m_OrigGroundCheckDistance = m_GroundCheckDistance;
        }


        public void Move(Vector3 move, bool crouch, bool jump)
        {
                if (move.magnitude > 1f) move.Normalize();
                move = transform.InverseTransformDirection(move);
                CheckGroundStatus();
                move = Vector3.ProjectOnPlane(move, m_GroundNormal);
                m_TurnAmount = Mathf.Atan2(move.x, move.z);
                m_ForwardAmount = move.z;

                ApplyExtraTurnRotation();

                if (m_IsGrounded)
                {
                        HandleGroundedMovement(crouch, jump);
                }
                else
                {
                        HandleAirborneMovement();
```

```
                }

                ScaleCapsuleForCrouching(crouch);
                PreventStandingInLowHeadroom();
                UpdateAnimator(move);
        }


        void ScaleCapsuleForCrouching(bool crouch)
        {
                if (m_IsGrounded && crouch)
                {
                        if (m_Crouching) return;
                        m_Capsule.height = m_Capsule.height / 2f;
                        m_Capsule.center = m_Capsule.center / 2f;
                        m_Crouching = true;
                }
                else
                {
                        Ray crouchRay = new Ray(m_Rigidbody.position +
Vector3.up * m_Capsule.radius * k_Half, Vector3.up);
                        float crouchRayLength = m_CapsuleHeight -
m_Capsule.radius * k_Half;
                        if (Physics.SphereCast(crouchRay, m_Capsule.radius
* k_Half, crouchRayLength, Physics.AllLayers, QueryTriggerInteraction.Ignore))
                        {
                                m_Crouching = true;
                                return;
                        }
                        m_Capsule.height = m_CapsuleHeight;
                        m_Capsule.center = m_CapsuleCenter;
                        m_Crouching = false;
                }
        }
```

```
        void PreventStandingInLowHeadroom()
        {
            if (!m_Crouching)
            {

                Ray crouchRay = new Ray(m_Rigidbody.position +
Vector3.up * m_Capsule.radius * k_Half, Vector3.up);
                float crouchRayLength = m_CapsuleHeight -
m_Capsule.radius * k_Half;
                if (Physics.SphereCast(crouchRay, m_Capsule.radius
* k_Half, crouchRayLength, Physics.AllLayers, QueryTriggerInteraction.Ignore))
                {
                    m_Crouching = true;
                }
            }
        }


        void UpdateAnimator(Vector3 move)
        {
            m_Animator.SetFloat("Forward", m_ForwardAmount, 0.1f,
Time.deltaTime);
            m_Animator.SetFloat("Turn", m_TurnAmount, 0.1f,
Time.deltaTime);
            m_Animator.SetBool("Crouch", m_Crouching);
            m_Animator.SetBool("OnGround", m_IsGrounded);
            if (!m_IsGrounded)
            {
                m_Animator.SetFloat("Jump",
m_Rigidbody.velocity.y);
            }

            float runCycle =
```

```
                        Mathf.Repeat(
      m_Animator.GetCurrentAnimatorStateInfo(0).normalizedTime +
m_RunCycleLegOffset, 1);
                    float jumpLeg = (runCycle < k_Half ? 1 : -1) *
m_ForwardAmount;
                    if (m_IsGrounded)
                    {
                        m_Animator.SetFloat("JumpLeg", jumpLeg);
                    }
                    if (m_IsGrounded && move.magnitude > 0)
                    {
                        m_Animator.speed = m_AnimSpeedMultiplier;
                    }
                    else
                    {
                        m_Animator.speed = 1;
                    }
                }


                void HandleAirborneMovement()
                {
                    Vector3 extraGravityForce = (Physics.gravity *
m_GravityMultiplier) - Physics.gravity;
                    m_Rigidbody.AddForce(extraGravityForce);

                    m_GroundCheckDistance = m_Rigidbody.velocity.y < 0 ?
m_OrigGroundCheckDistance : 0.01f;
                }


                void HandleGroundedMovement(bool crouch, bool jump)
                {
                    if (jump && !crouch &&
m_Animator.GetCurrentAnimatorStateInfo(0).IsName("Grounded"))
```

```
                {
                        m_Rigidbody.velocity = new
Vector3(m_Rigidbody.velocity.x, m_JumpPower, m_Rigidbody.velocity.z);
                        m_IsGrounded = false;
                        m_Animator.applyRootMotion = false;
                        m_GroundCheckDistance = 0.1f;
                }
        }

        void ApplyExtraTurnRotation()
        {
                float turnSpeed = Mathf.Lerp(m_StationaryTurnSpeed,
m_MovingTurnSpeed, m_ForwardAmount);
                transform.Rotate(0, m_TurnAmount * turnSpeed *
Time.deltaTime, 0);
        }

        public void OnAnimatorMove()
        {
                if (m_IsGrounded && Time.deltaTime > 0)
                {
                        Vector3 v = (m_Animator.deltaPosition *
m_MoveSpeedMultiplier) / Time.deltaTime;

                        v.y = m_Rigidbody.velocity.y;
                        m_Rigidbody.velocity = v;
                }
        }

        void CheckGroundStatus()
        {
                RaycastHit hitInfo;
#if UNITY_EDITOR
```

```
                    Debug.DrawLine(transform.position + (Vector3.up * 0.1f),
transform.position + (Vector3.up * 0.1f) + (Vector3.down *
m_GroundCheckDistance));
#endif
                    if (Physics.Raycast(transform.position + (Vector3.up *
0.1f), Vector3.down, out hitInfo, m_GroundCheckDistance))
                    {
                           m_GroundNormal = hitInfo.normal;
                           m_IsGrounded = true;
                           m_Animator.applyRootMotion = true;
                    }
                    else
                    {
                           m_IsGrounded = false;
                           m_GroundNormal = Vector3.up;
                           m_Animator.applyRootMotion = false;
                    }
             }
      }
}
```

Kode Program *ThirdPersonUserControl*

```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;
namespace UnityStandardAssets.Characters.ThirdPerson
{
    [RequireComponent(typeof (ThirdPersonCharacter))]
    public class ThirdPersonUserControl : MonoBehaviour
    {
        private ThirdPersonCharacter m_Character;
        private Transform m_Cam;
        private Vector3 m_CamForward;
        private Vector3 m_Move;
        private bool m_Jump;
        private void Start()
        {
            if (Camera.main != null)
            {
                m_Cam = Camera.main.transform;
            }
            else
            {
                Debug.LogWarning(
                    "Warning: no main camera found. Third person character
needs a Camera tagged \"MainCamera\", for camera-relative controls.",
gameObject);
            }
            m_Character = GetComponent<ThirdPersonCharacter>();
        }
        private void Update()
        {
            if (!m_Jump)
            {
                m_Jump = CrossPlatformInputManager.GetButtonDown("Jump");
            }
```

```
        }
        private void FixedUpdate()
        {
            float h = CrossPlatformInputManager.GetAxis("Horizontal");
            float v = CrossPlatformInputManager.GetAxis("Vertical");
            bool crouch = Input.GetKey(KeyCode.C);
            if (m_Cam != null)
            {
                m_CamForward = Vector3.Scale(m_Cam.forward, new Vector3(1, 0,
1)).normalized;
                m_Move = v*m_CamForward + h*m_Cam.right;
            }
            else
            {
                m_Move = v*Vector3.forward + h*Vector3.right;
            }
#if !MOBILE_INPUT
        if (Input.GetKey(KeyCode.LeftShift)) m_Move *= 0.5f;
#endif

            m_Character.Move(m_Move, crouch, m_Jump);
            m_Jump = false;
        }
    }
}
```

Kode Program *DoorScript*

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

public class DoorScript : MonoBehaviour
{
    private Animator _animator;
    public GameObject OpenPanel = null;
    private bool _isInsideTrigger = false;
    void Start()
    {
        _animator = transform.Find("Door").GetComponent<Animator>();
    }
    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
        {
            _isInsideTrigger = true;
            OpenPanel.SetActive(true);
        }
    }
    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Player")
        {
            _isInsideTrigger = false;
            _animator.SetBool("open", false);
            OpenPanel.SetActive(false);
        }
    }
    private bool IsOpenPanelActive
    {
        get
        {
```

```
            return OpenPanel.activeInHierarchy;
    }
     }
     void Update()
     {

         if (IsOpenPanelActive && _isInsideTrigger)
         {
             if (Input.GetKeyDown(KeyCode.E))
             {
                 OpenPanel.SetActive(false);
                 _animator.SetBool("open", true);
             }
         }
      }
}
```

Kode Program *TurnRight*

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

[ExecuteInEditMode]
public class TurnRight : MonoBehaviour
{

    string URL = "http://localhost/new.php";
    public string[] usersData;
    public static string baish;
    IEnumerator Start()
    {
        WWW users = new WWW(URL);
        yield return users;
        string usersDataString = users.text;
        usersData = usersDataString.Split(';');

        baish = "Ruang Dosen 4";

    }

    string getValueData(string data, string index)
    {
        string value = data.Substring(data.IndexOf(index) + index.Length);
        if (value.Contains("|"))
        {
            value = value.Remove(value.IndexOf("|"));
        }

        return value;
    }
```

```
    [Space(10)]
    [Header("Toggle for the gui on off")]
    public bool GuiOn;
    [Space(10)]
    [Header("The text to Display on Trigger")]
    [Tooltip("To edit the look of the text Go to Assets > Create > GUIskin. Add the new Guiskin to the Custom Skin proptery. If you select the GUIskin in your project tab you can now adjust the Label section to change this text")]

    public string Text ="Nama Ruangan : "+baish;

    [Tooltip("This is the window Box's size. It will be mid screen. Add or reduce the X and Y to move the box in Pixels. ")]
    public Rect BoxSize = new Rect(0, 0, 200, 100);
    [Space(10)]
    [Tooltip("To edit the look of the text Go to Assets > Create > GUIskin. Add the new Guiskin to the Custom Skin proptery. If you select the GUIskin in your project tab you can now adjust the font, colour, size etc of the text")]
    public GUISkin customSkin;
    void OnTriggerEnter()
    {
        GuiOn = true;
    }

    void OnTriggerExit()
    {
        GuiOn = false;
    }

    void OnGUI()
    {
        if (customSkin != null)
        {
            GUI.skin = customSkin;
        }
```

```
        if (GuiOn == true)
        {
            GUI.BeginGroup(new Rect((Screen.width - BoxSize.width) / 2,
(Screen.height - BoxSize.height) / 2, BoxSize.width, BoxSize.height));

            GUI.Label(BoxSize, baish);

            GUI.EndGroup();

        }


    }

}
```

Kode Program *Cube*

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

public class Cube1 : MonoBehaviour
{
    env myEnv = new env();
    string URL = "http://perencanaan.unhas.ac.id: ";
    public string[] usersData;
    public static string nama;
    public static string nip;
    public static string alamat;
    public static string email;
    public static string noHp;
    IEnumerator Start()
    {
        var form = new WWWForm();
        form.AddField("userAccount", "");
        form.AddField("userPassword", "");
        var headers1 = new Hashtable();

        var www1 = new WWW("http://perencanaan.unhas.ac.id: /login", form.data,
headers1);
        yield return www1;
        string myToken = "Bearer " + getValueData(www1.text, "token\": \"");
        myToken = myToken.Substring(0, myToken.Length - 3);

        Debug.Log(myToken);
        Dictionary<string, string> headers = new Dictionary<string, string>();
        headers.Add("Authorization", myToken);
```

```
        WWW users = new WWW(URL, null, headers);
        yield return users;
        string usersDataString = users.text;
        usersData = usersDataString.Split(';');

    private Cube getData()
    {
    [Space(10)]
    [Header("Toggle for the gui on off")]
    public bool GuiOn;
    [Space(10)]
    [Header("The text to Display on Trigger")]
    [Tooltip("To edit the look of the text Go to Assets > Create > GUIskin. Add the new Guiskin to the Custom Skin proptery. If you select the GUIskin in your project tab you can now adjust the Label section to change this text")]

    public string Text = "" + nip;

    [Tooltip("This is the window Box's size. It will be mid screen. Add or reduce the X and Y to move the box in Pixels. ")]
    public Rect BoxSize = new Rect(0, 0, 500, 100);



    [Space(10)]
    [Tooltip("To edit the look of the text Go to Assets > Create > GUIskin. Add the new Guiskin to the Custom Skin proptery. If you select the GUIskin in your project tab you can now adjust the font, colour, size etc of the text")]
    public GUISkin customSkin;

    void OnTriggerEnter()
    {
        GuiOn = true;
    }
```

```
    void OnTriggerExit()
    {
        GuiOn = false;
    }

    void OnGUI()
    {

        if (customSkin != null)
        {
            GUI.skin = customSkin;
        }

        if (GuiOn == true)
        {
            GUI.BeginGroup(new Rect((Screen.width - BoxSize.width) / 2,
(Screen.height - BoxSize.height) / 2, BoxSize.width, BoxSize.height));

            GUI.Label(BoxSize, nama + "\n" + nip + "\n" + alamat + "\n" + email
+ "\n" + noHp);

            GUI.EndGroup();

        }
    }
}
```