TUGAS PERTEMUAN: 8

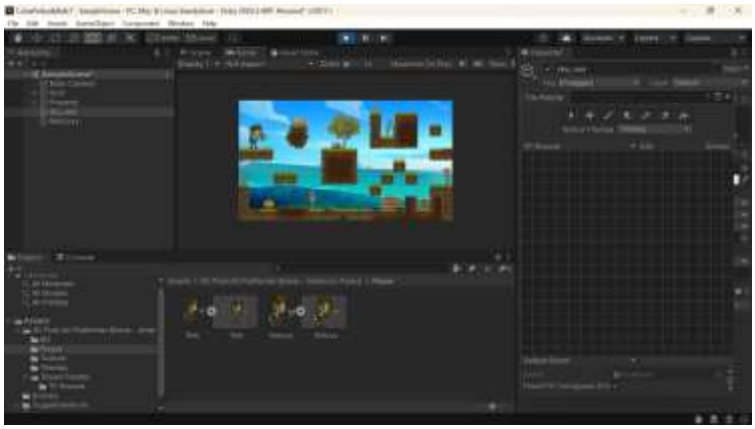
CAMERA & CHARACTER MOVEMENT

NIM	:	2118132
Nama	:	Madelbertha Fridolin Puka
Kelas	:	D
Asisten Lab	:	Abidzar Al Giffari (2218013)

8.1 Tugas 8 : Membuat Pergerakan Player, Camera Movement

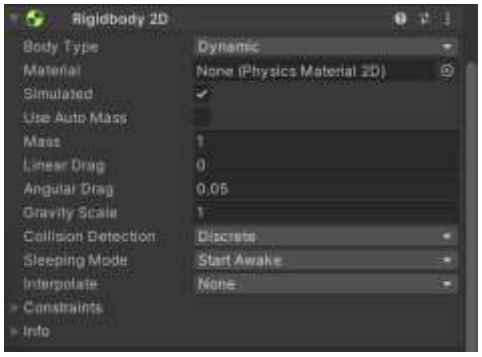
A. Membuat Pergerakan Player

1. Buka *project Unity* sebelumnya yang sudah berisikan *asset* yang akan dipakai. Tambahkan *player* bernama Reilyy, *Import* kedalam Hirarki.



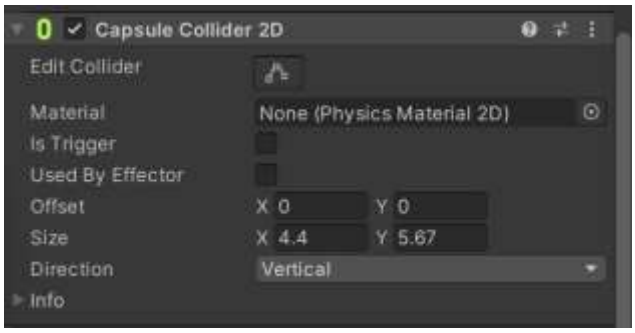
Gambar 8.1 Tampilan Penambahan *Player*

2. Klik *player* tambahkan *Component Rigidbody* 2D, sesuaikan *setting*-nya seperti gambar berikut, Centang pada *Freeze Rotation* Z



Gambar 8.2 Tampilan *Setting Rigidbody* 2D

3. Lalu tambahkan komponen *Capsule Collider* di Reilyyy, lalu klik *icon* sebelah kanan *edit colider*



Gambar 8.4 Tampilan Folder Tiles

4. Lalu cocokan garis oval degan karakternya atau bisa di inputkan *Offset* X, Y dan juga *Size* X, Y nya

Gambar 8.5 Tampilan Karakter

5. Buka Folder praktikum, lalu bikin folder baru bernama *Script*

Gambar 8.6 Tampilan Folder *Script*

6. Masuk kedalam folder *Script*, lalu buat C# *Script*, beri nama *Player*

Gambar 8.7 Tampilan Folder *Script*

7. *Drag* & *drop script player* kedalam Hirarki Reilyyy, lalu klik 2x pada *script player* maka akan masuk kedalam *text editor* seperti ini

Gambar 8.8 Tampilan Hirarki Reilyy

8. Masukan *source code* dibawah ini, pastikan nama *public class* harus sama dengan nama *file* yang dibuat.

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

public class Player : MonoBehaviour
{
    Rigidbody2D rb;
    [SerializeField] float speed = 1;
    float horizontalValue;
    bool facingRight;
    Vector3 scaleValue = new Vector3(0.2818362f,
0.2818362f, 0.2818362f); // Inisialisasi nilai awal

    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        horizontalValue =
Input.GetAxisRaw("Horizontal");
    }

    void FixedUpdate()
    {
        Move(horizontalValue);
    }

    void Move(float dir)
    {
        #region gerak kanan kiri
        float xVal = dir * speed * 100 *
Time.fixedDeltaTime;
        Vector2 targetVelocity = new Vector2(xVal,
rb.velocity.y);
```

```
        rb.velocity = targetVelocity;

        if (facingRight && dir < 0)
        {
            // Ubah nilai scaleValue sesuai kebutuhan
            scaleValue   =   new   Vector3(-0.2818362f,
0.2818362f, 0.2818362f);
            facingRight = false;
        }
        else if (!facingRight && dir > 0)
        {
            // Ubah nilai scaleValue sesuai kebutuhan
            scaleValue   =   new   Vector3(0.2818362f,
0.2818362f, 0.2818362f);
            facingRight = true;
        }

        //     Terapkan     nilai     scaleValue     ke
transform.localScale
        transform.localScale = scaleValue;
        #endregion
    }
}
```

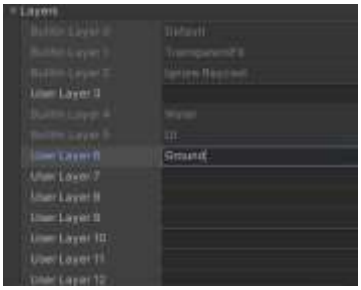
9. Untuk mencoba *Source code* diatas berhasil, Tekan *dikeyboard* "a" atau "*left arrow*" untuk ke arah kiri, tekan "d" atau "*right arrow*" untuk ke arah kanan

Gambar 8.10 Tampilan Hasil Pergerakan Karakter

10. Untuk membuat *player* loncat menggunakan spasi, kita perlu membuat *GorundCheck* dengan cara, klik *Grid* pada *Hierarchy*, pergi ke *inspector*, pilih *Layer*, Klik *Add Layer*

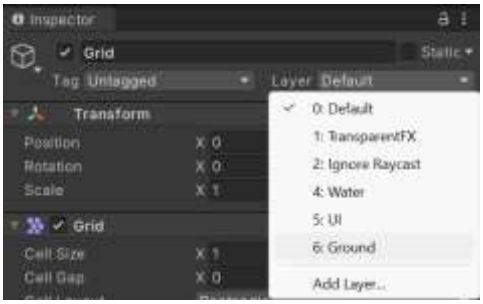
Gambar 8.11 Tampilan *Asset Texture*

11. Lalu isi "*Ground*" pada *User Layer* 6



Gambar 8.12 Tampilan *User Layer* 6

12. *Ubah Layer* menjadi *Ground*, jika muncul *pop up Change Layer*, klik yes saja



Gambar 8.13 Tampilan Pengubahan *Layer Ground*

13. Klik kanan pada Reilyyy, lalu *Create empty*, beri nama *GroundCheck*

Gambar 8.15 Tampilan *Create empty*

14. Klik pada Hirarki *GorundCheck*, lalu gunakan "*Move Tools*" untuk memindahkan ke bagian bawah *Player* seperti gambar berikut.

Gambar 8.16 Tampilan *Move Tools*

15. Kembali ke *script Player* tambahkan *source code* seperti ini

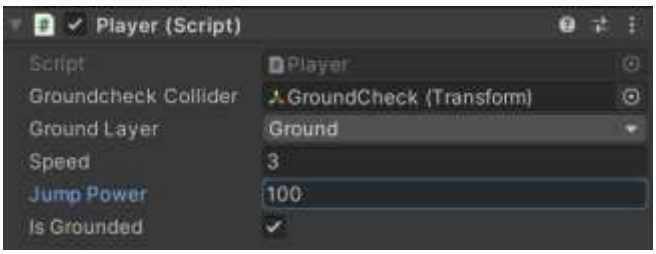
```
[SerializeField] Transform groundcheckCollider;
[SerializeField] LayerMask groundLayer;

const float groundCheckRadius = 0.2f; // +
[SerializeField] float speed = 1;
[SerializeField] float jumpPower = 100;
```

```
float horizontalValue;

[SerializeField] bool isGrounded; // +
bool facingRight;
bool jump;
```

16. Buat *void ground check* dibawah *void fixedUpdate* & tambahkan *GorunCheck*(); pada *void fixedUpdate*

```
void FixedUpdate()
{
    GroundCheck();
    Move(horizontalValue);
}

void GroundCheck()
{
     isGrounded = false;
     Collider2D[] colliders =
Physics2D.OverlapCircleAll(groundcheckCollider.positio
n, groundCheckRadius, groundLayer);
     if (colliders.Length > 0)
     isGrounded = true;
}
```

17. Klik Reilyyy, lalu ke *inspector* ke *effect Player script* di bagian "*Grouncheck collider*" tekan *icon* lalu pilih yang *GroundCheck Transform*, dan pada *Ground Layer* pilih *Ground*



Gambar 8.19 Tampilan *Asset* didalam *Property*

18. Lalu untuk membuat *player* melompat tambahkan *script* berikut

```
[SerializeField] float jumpPower = 100;

bool jump;
```

19. Tambahkan juga *script* berikut di bagian *void update*

```
if (Input.GetButtonDown("Jump"))
jump = true;
else if (Input.GetButtonUp("Jump"))
jump = false;
```

20. Tambahkan juga *jump* pada parameter *Move*

```
void FixedUpdate()
{
```

```
        GroundCheck();
        Move(horizontalValue, jump);
}
```

21. Tambahkan *script* berikut pada *void Move*

```
bool jumpflag

if(isGrounded && jumpflag)
{
    isGrounded = false;
    jumpflag = false;
    rb.AddForce(new Vector2(0f, jumpPower));
```

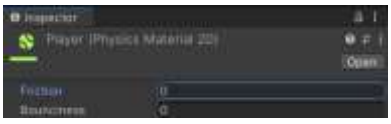
22. Buat folder baru di praktikum bernama "*Physics*"

Gambar 8.24 Tampilan Penambahan Folder *Physics*

23. Didalam folder *Pyshics create* > 2D > *physical material* 2d , beri nama "*Player*"

Gambar 8.25 Penambahan *Physical Material* 2D

24. Klik *Player* (*Physics Material* 2D), dibagian menu *inspector*, *friction* & *bounciness* ubah menjadi 0



Gambar 8.26 Tampilan *Player* (*Physics Material* 2D)

25. Klik *Hierarchy* pilih layer *player*, pada *Inspector* cari *Rigidbody* 2D lalu klik icon untuk membuka *box select physhics material* 2D, lalu pilih *asset Player* yang sudah kita buat tadi.

Gambar 8.27 Tampilan *Rigidbody* 2D

26. Jika di *play* karakter akan melompat dengan menekan spasi

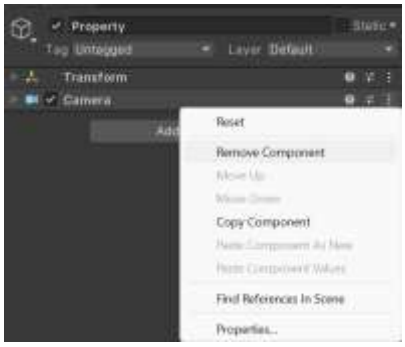
Gambar 8.28 Tampilan Hasil

B. Camera Movement

1. Pada Hirarki *Property* Ubah *Inspector* pada tag Main camera Menjadi *untagged*

Gambar 8.29 Tampilan *Property*

2. Pada *Effect Camera* pilih *Remove Component*



Gambar 8.30 Tampilan *Remove Compenent*

3. *Create Empty* pada Hirarki, dan *Rename* Menjadi Camera

Gambar 8.31 Tampilan Camera

4. Sesuaikan *Setting Layer* Camera seperti gambar dibawah ini

Gambar 8.32 Tampilan *Setting Layer* Camera

5. Buat file *script* baru di folder *Script* dengan nama "*CameraFollow*"

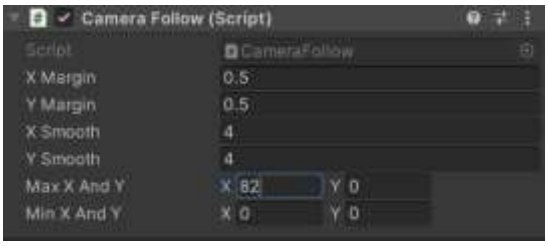
Gambar 8.33 Tampilan *CameraFollow*

6. Masukan *script* berikut pada *CameraFollow*

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

public class CameraFollow : MonoBehaviour
{

    public float xMargin = 0.5f;
    public float yMargin = 0.5f;
    public float xSmooth = 4f;
    public float ySmooth = 4f;
    public Vector2 maxXAndY;
    public Vector2 minXAndY;
    private Transform player;

    void Awake()
    {
        player                                              =
GameObject.FindGameObjectWithTag("Player").transform;
    }

    bool CheckXMargin()
    {
        return     Mathf.Abs(transform.position.x     -
player.position.x) > xMargin;
    }
```

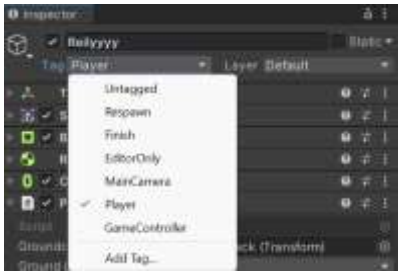
```
    bool CheckYMargin()
    {
        return     Mathf.Abs(transform.position.y     -
player.position.y) > yMargin;
    }

    void FixedUpdate()
    {
        TrackPlayer();
    }

    void TrackPlayer()
    {
        float targetX = transform.position.x;
        float targetY = transform.position.y;
        if (CheckXMargin())
            targetX  =  Mathf.Lerp(transform.position.x,
player.position.x,
            xSmooth * Time.deltaTime);
        if (CheckYMargin())
            targetY  =  Mathf.Lerp(transform.position.y,
player.position.y,
            ySmooth * Time.deltaTime);
            targetX  =  Mathf.Clamp(targetX,  minXAndY.x,
maxXAndY.x); targetY =
            Mathf.Clamp(targetY,            minXAndY.y,
maxXAndY.y); transform.position = new
            Vector3(targetX,                   targetY,
transform.position.z);
    }
}
```

7. Pada camera, buka *inspector setting* menjadi berikut



Gambar 8.34 Tampilan *Setting Camera Follow*

8. Ubah *tag* di *Player Untagged* menjadi ”*Player*”



Gambar 8.35 Tampilan *Tag Player*

9. Tekan *play* untuk menjalankan, maka sekarang kamera akan mengikuti pergerakan karakter

Gambar 8.36 Tampilan Hasil

KUIS

1. *Soucer Code Player*

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

public class Player : MonoBehaviour
{
    Rigidbody2D rb;

    [SerializeField] Transform groundcheckCollider;
    [SerializeField] LayerMask groundLayer;


    const float groundCheckRadius = 0.2f; // +
    [SerializeField] float speed = 1;
    [SerializeField] float jumpPower = 100;

    float horizontalValue;

    [SerializeField] bool isGrounded; // +
    bool facingRight;
    bool jump;


    Vector3 scaleValue = new Vector3(0.2818362f, 0.2818362f,
0.2818362f); // Inisialisasi nilai awal

    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        horizontalValue = Input.GetAxisRaw("Horizontal");
        if (Input.GetButtonDown("Jump"))
             jump = true;
        else if (Input.GetButtonUp("Jump"))
              jump = false;

    }

    void FixedUpdate()
    {
        GroundCheck();
        Move(horizontalValue, jump);
    }

    void GroundCheck()
    {
        isGrounded = false;
        Collider2D[]               colliders               =
Physics2D.OverlapCircleAll(groundcheckCollider.position,
groundCheckRadius, groundLayer);
        if (colliders.Length > 0)
        isGrounded = true;
```

```
    }


    void Move(float dir, bool jumpflag)
    {
        if(isGrounded && jumpflag)
        {
            isGrounded = false;
            jumpflag = false;
            rb.AddForce(new Vector2(0f, jumpPower));
        }


        #region gerak kanan kiri
        float xVal = dir * speed * 100 * Time.fixedDeltaTime;
        Vector2   targetVelocity   =   new   Vector2(xVal,
rb.velocity.y);
        rb.velocity = targetVelocity;

        if (facingRight && dir < 0)
        {
            // Ubah nilai scaleValue sesuai kebutuhan
            scaleValue = new Vector3(-0.2818362f, 0.2818362f,
0.2818362f);
            facingRight = false;
        }
        else if (!facingRight && dir > 0)
        {
            // Ubah nilai scaleValue sesuai kebutuhan
            scaleValue = new Vector3(0.2818362f, 0.2818362f,
0.2818362f);
            facingRight = true;
        }

        // Terapkan nilai scaleValue ke transform.localScale
        transform.localScale = scaleValue;
        #endregion
    }
}
```

Analisa

Pada codingan diatas, digunakan untuk mengatur gerakan ke kanan dan kekiri serta lompatan dari karakter. Pada awal, terdapat beberapa *namespace* yang digunakan untuk mengakses kelas dan fungsi yang dibutuhkan dalam *Unity*. Kemudian terdapat kelas bernama *player* yang mewarisi dari *MonoBehaviour* yang merupakan kelas dasar dalam Unity.

Didalam kelas *player* terdapat beberapa variabel. Variabel rb digunakan untuk mengatur gerakan dari objek. Variabel *groundcheckCollider* digunakan untuk memeriksa apakah karakter berada diatas tanah. Variabel *speed*,

jumPower dan *IsGrounded* digunakan untuk mengatur kecepatan, daya loncatan dan status karakter yang berada di tanah atau tidak.

Terdapat juga fungsi *void Awake*() yang digunakan untuk inisialisasi *Rigidbody* 2D. Fungsi *void Update*() digunakan untuk mengambil inputan dari pengguna, seperti gerakan horizontal dan inputan lompatan. Fungsi *void FixedUpdate*() digunakan untuk memeriksa apakah karakter berada di atas tanah serta menggerakkan karakter berdasarkan input yang diterima dari *Update*().Fungsi void *GroundCheck*() digunakan untuk memeriksa apakah karakter berada di atas tanah dengan melakukan pengecekan lingkaran di sekitar *groundcheckCollider*. Fungsi *void Move*() digunakan untuk karakter berdasarkan input yang diterima. Jika karakter berada di atas tanah dan input lompatan diterima, maka gaya lompatan diterapkan pada *Rigidbody2D*. Bagian #region digunakan untuk mengatur gerakan ke kanan dan ke kiri dengan mengubah kecepatan *horizontal* karakter serta mengatur arah hadap karakter dengan mengubah skala lokal objek.

2. *Source Code Camera Follow*

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

public class CameraFollow : MonoBehaviour
{

    public float xMargin = 0.5f;
    public float yMargin = 0.5f;
    public float xSmooth = 4f;
    public float ySmooth = 4f;
    public Vector2 maxXAndY;
    public Vector2 minXAndY;
    private Transform player;

    void Awake()
    {
        player                                                    =
GameObject.FindGameObjectWithTag("Player").transform;
    }

    bool CheckXMargin()
    {
        return        Mathf.Abs(transform.position.x        -
player.position.x) > xMargin;
    }

    bool CheckYMargin()
    {
```

```
        return        Mathf.Abs(transform.position.y        -
player.position.y) > yMargin;
    }

    void FixedUpdate()
    {
        TrackPlayer();
    }

    void TrackPlayer()
    {
        float targetX = transform.position.x;
        float targetY = transform.position.y;
        if (CheckXMargin())
            targetX    =    Mathf.Lerp(transform.position.x,
player.position.x,
            xSmooth * Time.deltaTime);
        if (CheckYMargin())
            targetY    =    Mathf.Lerp(transform.position.y,
player.position.y,
            ySmooth * Time.deltaTime);
            targetX   =   Mathf.Clamp(targetX,   minXAndY.x,
maxXAndY.x); targetY =
            Mathf.Clamp(targetY,   minXAndY.y,   maxXAndY.y);
transform.position = new
            Vector3(targetX, targetY, transform.position.z);
    }
}
```

Analisa

Codingan diatas digunakan untuk mengontrol kamera agar mengikuti pergerakan dari *player*. Pada bagian awal terdapat beberapa *namespace* yang digunakan untuk mengimport namespace yang dibutuhkan. Kemudian terdapat kelas *CameraFollow* yang mewarisi dari kelas *MonoBehaviour*. Terdapat berapa variabel yang digunakan salah satunya player yang digunakan untuk menyimpan referensi *transform* dari *player*. Kemudian terdapat beberapa fungsi *void* salah satunya *TrackPlayer* yang digunakan untuk mengatur posisi kamera agar mengikuti pergerakan dari *player*.