TUGAS PERTEMUAN: 8

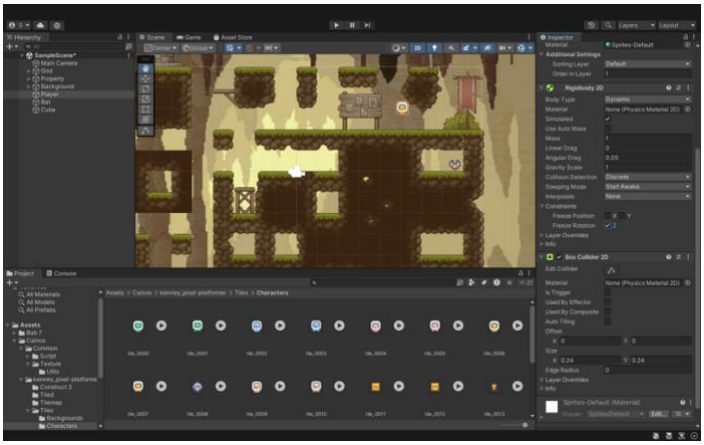
CAMERA & CHARACTER MOVEMENT

NIM	:	2118044
Nama	:	Yudistira Samuel Sura
Kelas	:	C
Asisten Lab	:	Difa Fisabilillah (2118052)

8.1 Tugas 1 : Character Movement

A. Membuat Pergerakan Karakter

1. Buka projek unity sebelumnya yang telah berisi *tilemap* dan karakter.

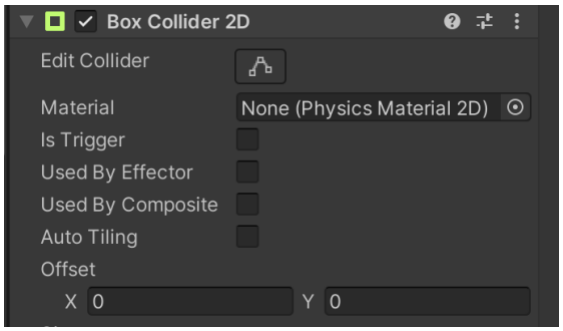


Gambar 8.1 Membuka Projek Unity

2. Klik player dan tambahkan *component* Rigidbody 2D, sesuaikan pengaturannya seperti gambar berikut, centang pada *Freeze Rotation* Z.

Gambar 8.2 Mengatur RigidBody 2D *Player*

3. Lalu tambahkan komponen *Capsule Collider* 2D di player, lalu klik *icon* sebelah kanan *edit collider*.

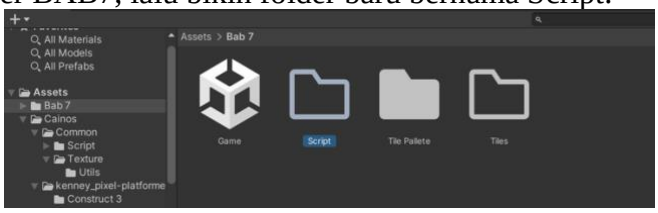


Gambar 8.3 Komponen *Capsule Collider* 2D

4. Lalu cocokkan garis oval dengan karakternya atau bisa diinputkan *offset* X, Y dan juga *size* X, Y nya.

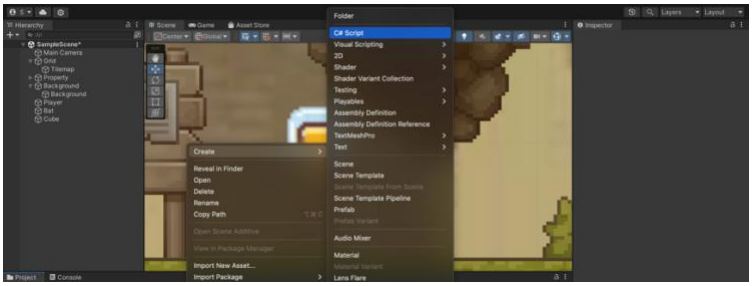
Gambar 8.4 Menyesuaikan Garis Oval dengan Karakter

5. Buka folder BAB7, lalu bikin folder baru bernama Script.



Gambar 8.5 Membuat Folder *Script*

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

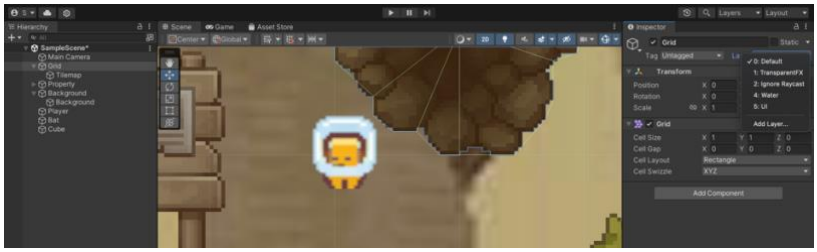


Gambar 8.6 Membuat C# Script

7. *Drag & drop script player* ke dalam Hirarki player, lalu klik 2x pada *script player*, maka akan masuk kedalam *text editor* dan masukkan *source code* berikut.

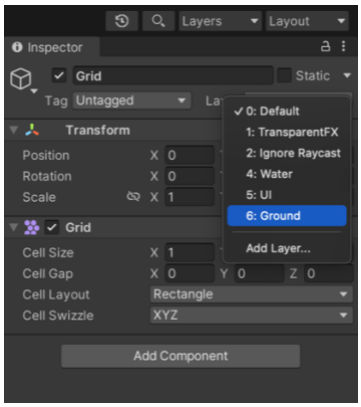
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour
{
  Rigidbody2D rb;

  [SerializeField] float speed = 1;
  float horizontalValue;
  bool facingRight;

  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)
    {
      // ukuran player
      transform.localScale = new Vector3(-6, 6, 6);
      facingRight = false;
    }

    else if (!facingRight && dir > 0)
    {
      // ukuran player
      transform.localScale = new Vector3(6, 6, 6);
      facingRight = true;
    }

    #endregion
  }
}
```

8. Untuk membuat *player* loncat menggunakan spasi, kita perlu membuat GroundCheck dengan cara klik Grid pada Hierarchy, pergi ke *inspector*, pilih *layer*, lalu klik *Add Layer* dan buat *Ground* pada *User Layer* 6.



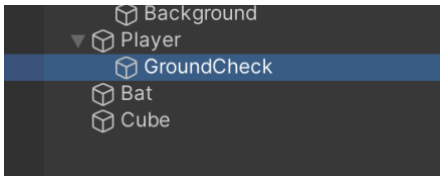
Gambar 8.7 Membuat *Ground* pada *User Layer* 6

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



Gambar 8.8 Mengubah Layer

10. Klik kanan pada player, lalu *Create empty*, beri nama *GroundCheck.*



Gambar 8.9 Membuat *Create Empty*

11. Klik pada Hirarki *GroundCheck*, lalu gunakan "*Move Tools*" untuk memindahkan ke bagian bawah *player* seperti gambar berikut.

Gambar 8.10 Memindahkan *GroundCheck*

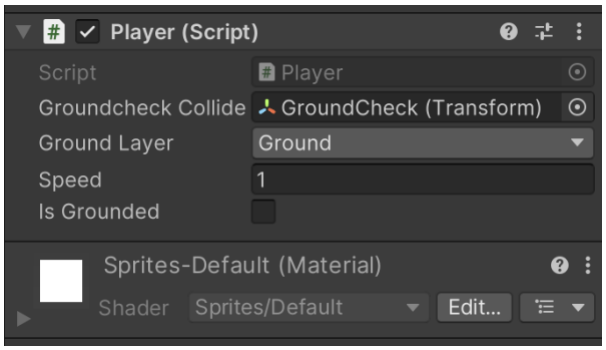
12. Kembali ke *script player* dan tambahkan *source code* berikut dibawah *source code* RigidBody2D rb; .

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

const float groundCheckRadius = 0.2f; // +
SerializeField] float speed = 1;
float horizontalValue;

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

13. Buat void *ground check* dibawah void fixedUpdate & ubah void fixedUpdate, seperti berikut.

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

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

14. Klik player, lalu ke *inspector* ke *component Player script*, selanjutnya di bagian "*Groundcheck collider*" tekan *icon*, lalu pilih yang *GroundCheck Transform* dan pada *Ground Layer* pilih *Ground*.



Gambar 8.11 Mengubah *GroundCheck Collider* dan *Ground Layer*

15. Lalu untuk membuat *player* melompat, tambahkan *script* jumPower di bawah float speed = 1 dan untuk bool jump letakkan dibawah bool facingRight.

```
[SerializeField] float jumpPower = 100;

bool jump;
```

16. Tambahkan juga *script* berikut di bagian void Update di bawah horizontalValue.

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

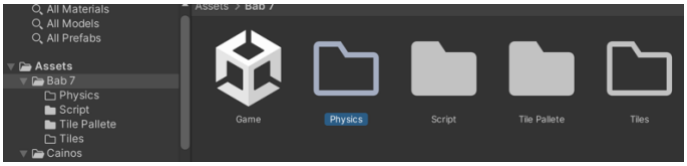
17. Tambahkan juga jump pada parameter Move di void FixedUpdate, seperti berikut.

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

18. Ubah *script* pada void Move dengan *script* berikut.

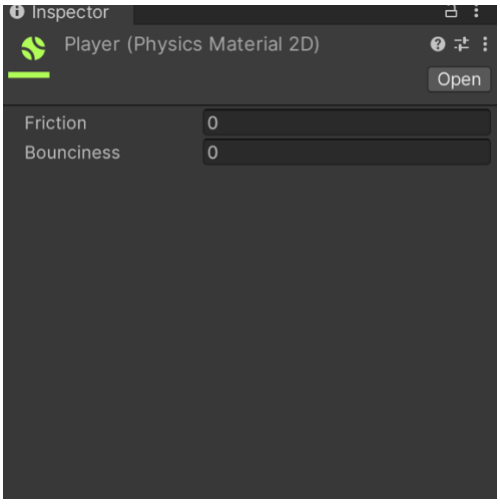
```
void Move(float dir, bool jumpflag)
 {
   if(isGrounded && jumpflag)
   {
       isGrounded = false;
       jumpflag = false;
       rb.AddForce(new Vector2(0f, jumpPower));
   }

   #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)
   {
     // ukuran player
     transform.localScale = new Vector3(-6, 6, 6);
     facingRight = false;
   }

   else if (!facingRight && dir > 0)
   {
     // ukuran player
     transform.localScale = new Vector3(6, 6, 6);
     facingRight = true;
   }

   #endregion
 }
```

19. Buat folder baru di BAB7 bernama “Physics” dan di dalam folder tersebut klik kanan, lalu pilih *create*, kemudian 2D, lalu pilih *Physics Material* 2D dan beri nama “Player”.



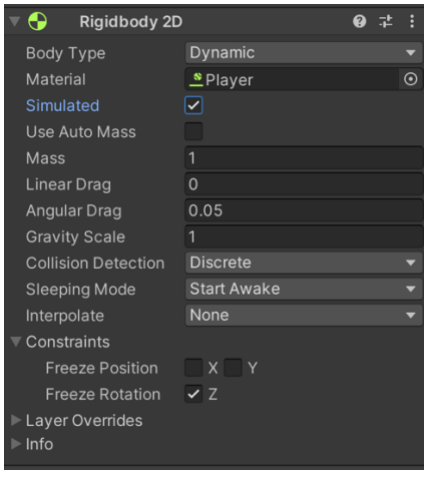
Gambar 8.12 Membuat Physics 2D

20. Klik Player (*Physics Material* 2D), di bagian menu *inspector*, *friction* & *bounces* ubah menjadi 0.



Gambar 8.13 Mengatur Inspector Player Physics Material

21. Klik Hierarchy pilih *layer* player idle 1, pada *inspector* cari Rigidbody 2D, lalu klik *icon* untuk membuka *box select physhics material* 2D, kemudian pilih *asset* Player yang sudah dibuat sebelumnya.



Gambar 8.14 Mengatur Material RigidBody 2D

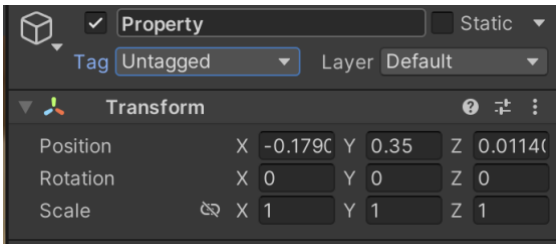
22. Tekan *play*, lalu *player* akan melompat dengan menekan spasi.

Gambar 8.15 Hasil Tampilan

8.2 Tugas 2: Camera Movement

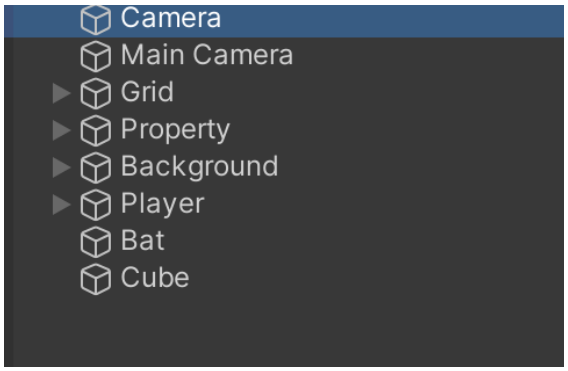
A. Membuat Camera Movement

1. Pada hirarki *Property*, ubah *inspector* pada tag *main camera* menjadi *untagged.*



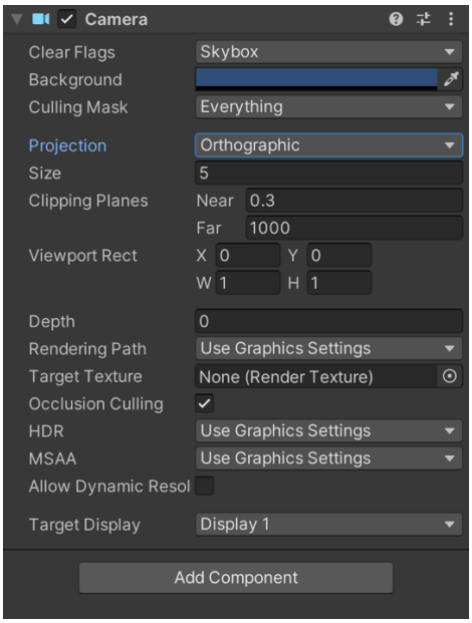
Gambar 8. 16 Mengubah Tag *Property*

2. *Create empty* pada hiraki dan *rename* menjadi *camera* dan letakkan di paling atas.



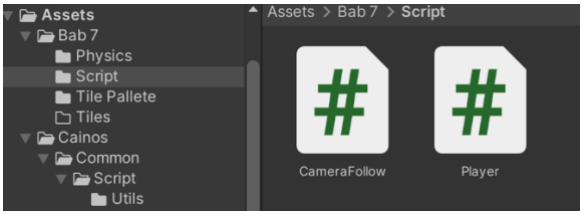
Gambar 8.17 Membuat *Create Empty Camera*

3. Kemudian tambahkan komponen *camera* dan sesuaikan pengaturan layer *camera*, seperti gambar berikut.



Gambar 8. 18 Mengatur *Layer Camera*

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



Gambar 8.19 Membuat Script CameraFollow

5. Lalu klik 2 kali dan ketikkan *source code* berikut.

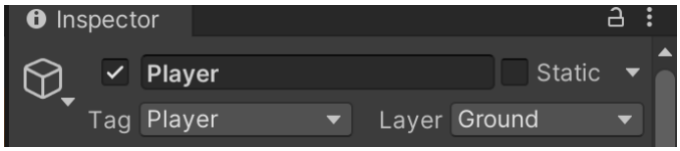
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

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);
    }
}
```

6. *Drag & drop script* “CameraFollow” ke dalam layer *camera*, lalu klik pada *camera* dan buka *inspector*, kemudian pada bagian CameraFollow (*Script*), ubah bagian max X dan max Y nya, seperti berikut.

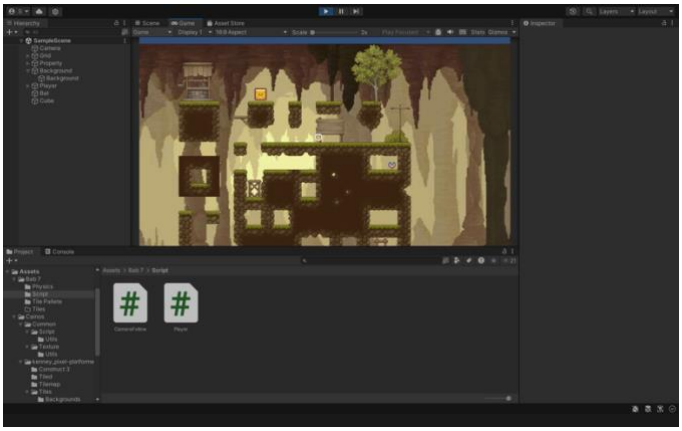
Gambar 8.20 Mengubah Max X dan Y

7. Ubah *tag* di player yang awalnya *Untagged* menjadi *Player* dan layer nya menjadi *Ground*.



Gambar 8.21 Mengubah *Tag Player*

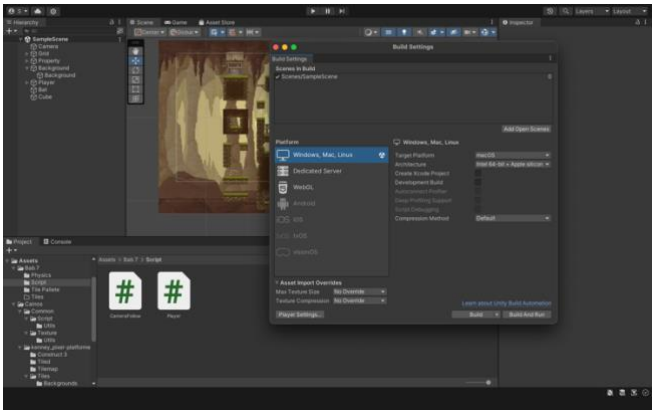
8. Tekan *play* untuk menjalankan, maka kamera akan mengikuti pergerakan karakter.



Gambar 8.22 Hasil Akhir Tampilan *Camera Movement*

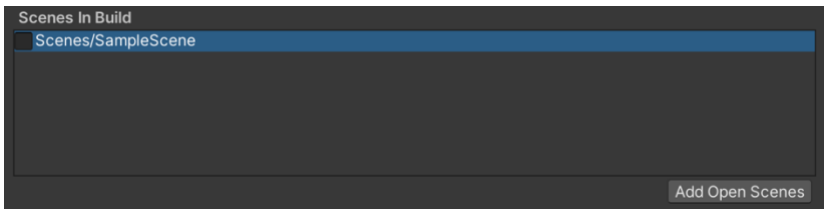
B. Render

1. Pergi ke menu file, kemudian pilih *build setting* atau Ctrl + Shift + B.



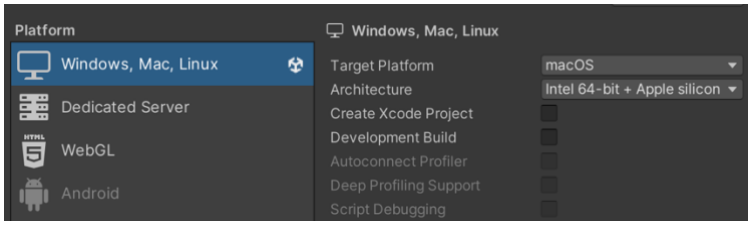
Gambar 8.24 *Build Setting*

2. Klik *add open scenes*, kemudian *uncheck* Scenes/SampleScene.



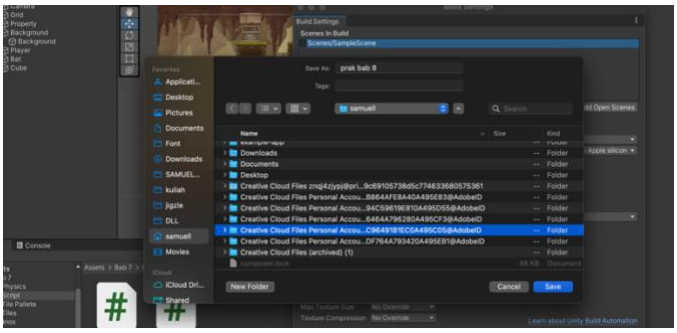
Gambar 8.25 *Add Open Scenes*

3. Pada *build setting* ini pilih PC, Mac & Linux, lalu tekan *build and run* dan pastikan pada menu *Scene in Build* berada pada *project*.



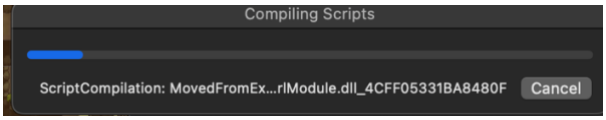
Gambar 8.26 *Build Setting*

4. Pilih lokasi project disimpan.



Gambar 8.27 Menyimpan *Project*

5. Tunggu hasil render.



Gambar 8.28 Menunggu Hasil Render

6. Hasil akhir atau hasil setelah dilakukan render.

Gambar 8.28 Hasil Akhir Tampilan Game

C. Kuis

1. Kuis CameraFollow

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

public class CameraFollow : MonoBehaviour
{
    [SerializeField] private Transform player;

    void Update(){
        transform.position = new Vector3
(player.position.x, transform.position.y,
transform.position.z)
    }
}
```

Penjelasan :

Source code diatas digunakan dalam *unity* untuk membuat kamera mengikuti gerakan *player*. *Source code* `using` digunakan untuk memanggil elemen dan *collection. Source code* `public class CameraFollow : MonoBehaviour` merupakan deklarasi *class* CameraFollow. `[SerializeField] private Transform player` merupakan *source code* yang digunakan untuk mengatur posisi atau mengacu pada objek *player*. *Source code* yang berada di dalam void update digunakan untuk mengatur posisi kamera dengan mengikuti posisi x, y atau z dari *game object* atau *player*.

D. Link Github

https://github.com/YudisSamuel/2118044_PRAK_ANIGAME