TUGAS PERTEMUAN: 8

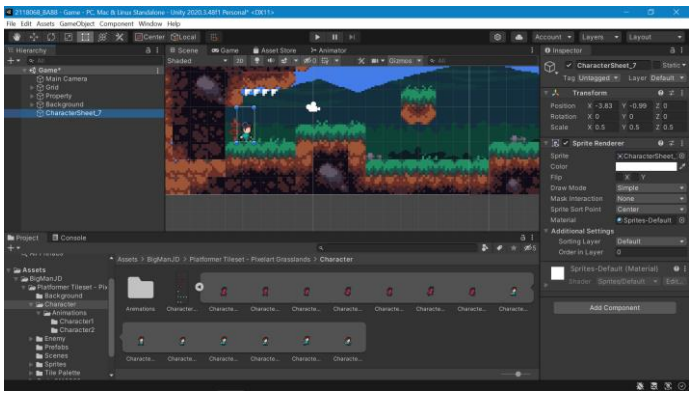
CAMERA & CHARACTER MOVEMENT

NIM	:	2118068
Nama	:	Ahmad Bahrul Ilmi
Kelas	:	B
Asisten Lab	:	Devina Dorkas Manuela (2218108)

1.1 Tugas 8 : Menerapkan Camera & Character Movement

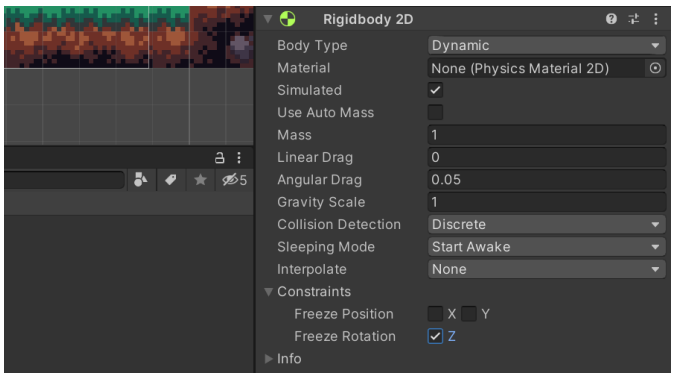
A. Membuat Pergerakan Player

1. Buka *file unity* sebelumnya yang sudah memiliki *tilemap* dan tambahkan *character idle* pada *hierarchy*.



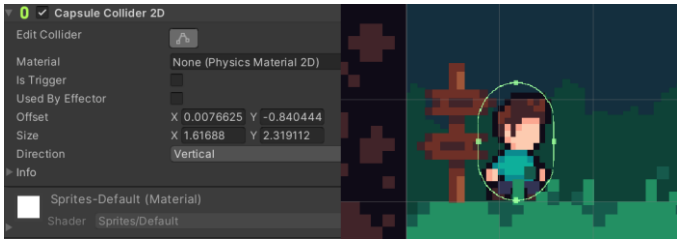
Gambar 8.1 Menambahkan *character*

2. Kemudian pada *character* tersebut tambahkan *component* yang bernama *Rigidboby 2D* dan atur *Freze Rotation Z* menjadi aktif.



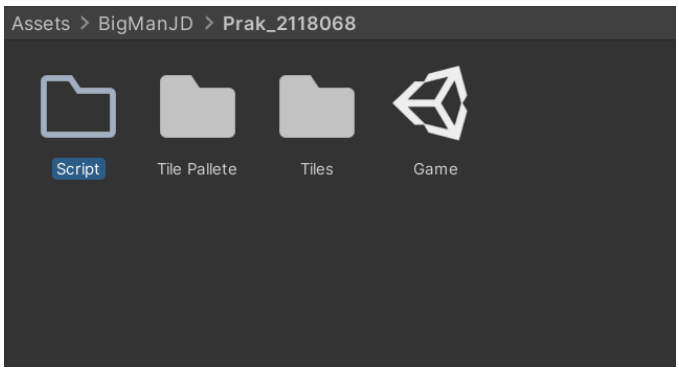
Gambar 8.2 Komponen *rigidbody 2d*

3. Lalu tambahkan juga *component* baru pada *character* tersebut yang bernama *Capsule Collider 2D* dan atur *collider* nya supaya sama dengan ukuran *character* menggunakan *Edit Collider*.



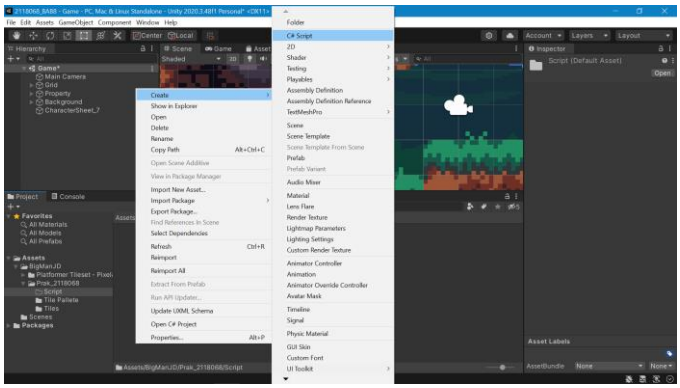
Gambar 8.3 Komponen *capsule collider 2d*

4. Dan buat *folder* pada *folder* Prak_21118068 dan beri nama *Script* yang nantinya digunakan untuk menyimpan *script* sebagai logika dari *game* atau *player* yang ada pada *game*.



Gambar 8.4 Membuat *folder script*

5. Pada *folder* tersebut tambahkan *file C Sharp* dengan cara klik kanan kemudian *Create* dan pilih *C# Script* dan beri nama *file* tersebut menjadi *Player*.

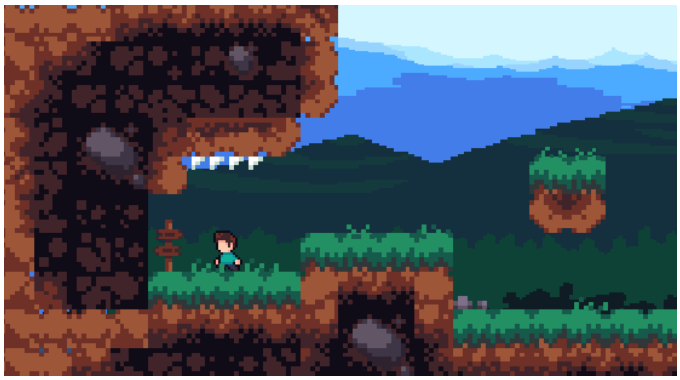


Gambar 8.5 *File c sharp*

6. *Double click* pada *file* tersebut untuk membuka *code editor* dan masukan *source* berikut untuk membuat *character* bisa bergerak.

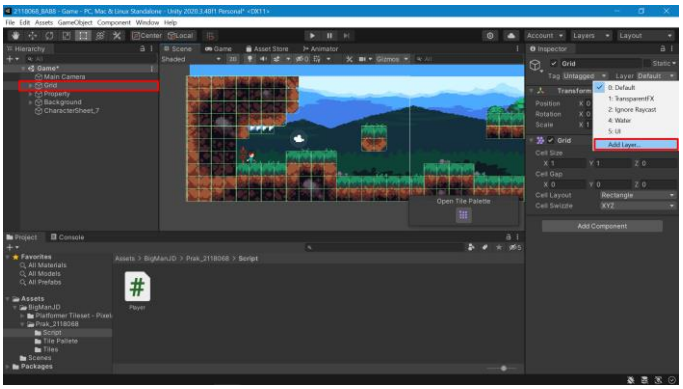
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

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(-0.5f, 0.5f,
0.5f);
      facingRight = false;
    }

    else if (!facingRight && dir > 0)
    {
      // ukuran player
      transform.localScale = new Vector3(0.5f, 0.5f,
0.5f);
      facingRight = true;
    }

    #endregion
  }
}
```

7. Jika sudah masukan *script* tersebut pada *character* dengan cara dr*ag and drop* pada *character* yang ada di *hierarchy*, dan *run project* maka karakter bisa bergerak sesuai inputan *keyboard* A dan D atau *arrow left* dan *right*.



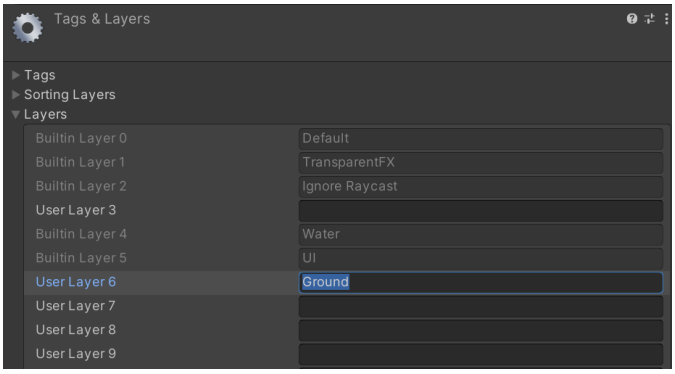
Gambar 8.6 *Character* bergerak

8. Kemudian buat *character* bisa loncat dengan cara klik pada *Grid* kemudian tambahkan *layer* baru.



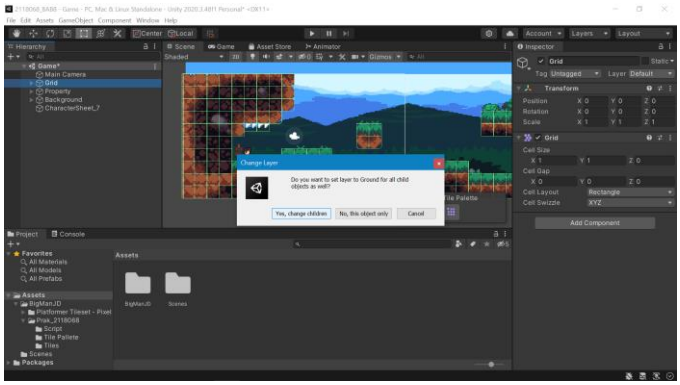
Gambar 8.7 Menambah *layer* baru pada *grid*

9. Buat pada *User Layer* 6 dan beri nama *Gorund*.



Gambar 8.8 Nama *layer* pada *grid*

10. Dan ubah *layer* pada *Grid* menjadi *Gorund* jika terdapat pemberitahuan maka pilih *Yes*, change *children*.

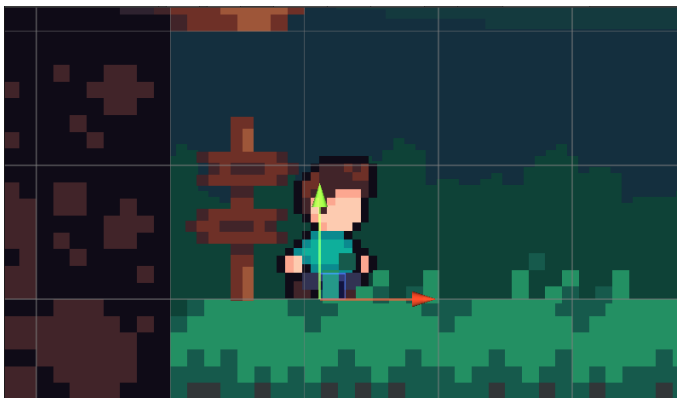


Gambar 8.9 Informasi pada *layer grid*

11. Setelah itu klik kanan pada *Hirarchy* lalu *Create Empty* dan beri nama *GroundCheck* pastikan berada didalam *Character* seperti berikut.

Gambar 8.10 Menambah *groundcheck*

12. Klik pada *GroundCheck* dan atur supaya titik *point* supaya berada dibawah *character* menggunakan *Move Tools*.



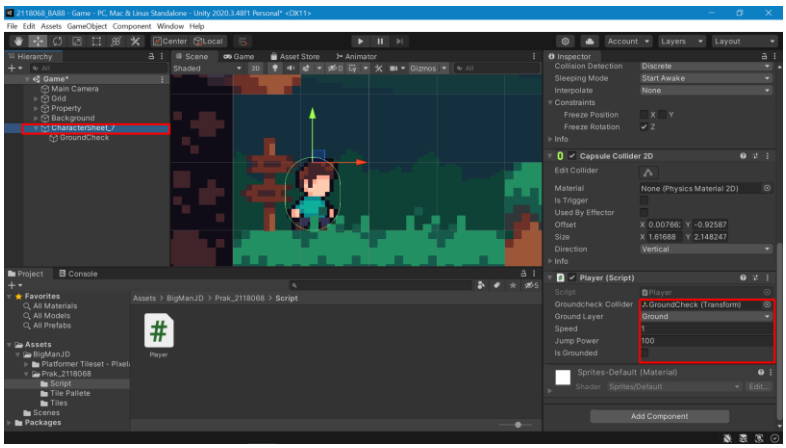
Gambar 8.11 Mengatur posisi *groundcheck*

13. Dan ubah *source* sebelumnya pada *file player* menjadi seperti berikut.

```
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 = 2;
  [SerializeField] float jumpPower = 120;
  float horizontalValue;

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

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

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

    if (facingRight && dir < 0)
    {
        // ukuran player
        transform.localScale = new Vector3(-0.5f, 0.5f,
0.5f);
        facingRight = false;
    }

    else if (!facingRight && dir > 0)
    {
        // ukuran player
        transform.localScale = new Vector3(0.5f, 0.5f,
0.5f);
        facingRight = true;
    }

    #endregion
  }
}
```

14. Kemudian klik pada *character* dan pastikan *GroundCheck Collider* menjadi *GroundCheck (Transform)* dan *Ground Layer* menjadi *Gorund*.

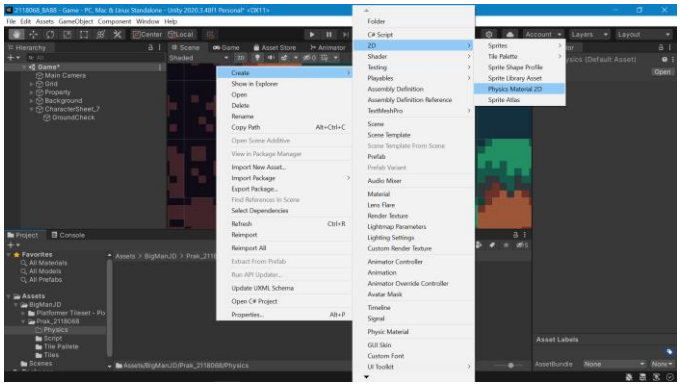


Gambar 8.12 *Gorundcheck* pada *character*

15. Buat *folder* baru lagi pada *folder* Prak_2118068 beri nama *Physics*.

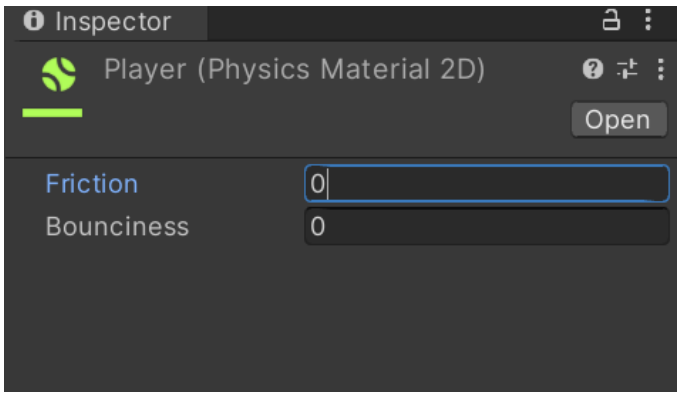
Gambar 8.13 Menambah *folder physics*

16. Pada *folder* tersebut klik kanan pilih *Create* dan 2D lalu *Physics Material 2D* dan beri nama *Player*.



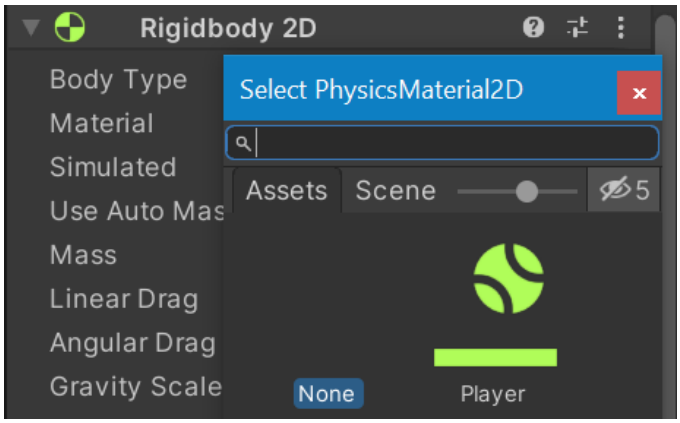
Gambar 8.14 *Physics material 2d*

17. Kemudian atur *Friction* pada *Physics Material 2D* yang sudah dibuat menjadi 0.



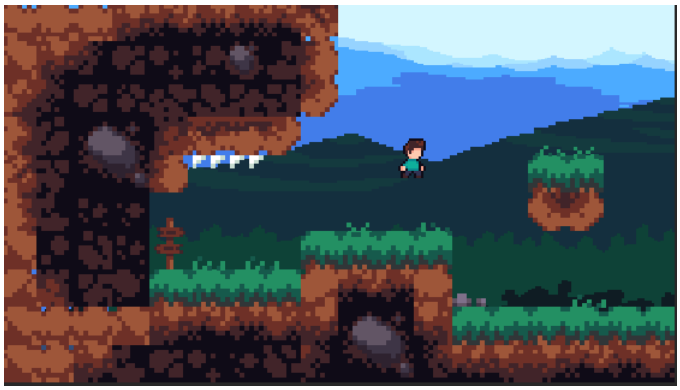
Gambar 8.15 Atur *friction physics material*

18. Klik *character* pada *Hierarchy* dan cari komponen *Ridigbody* kemudian ubah *Material* pada *Rigidbody* menjadi *Player*.



Gambar 8.16 Mengatur *material character*

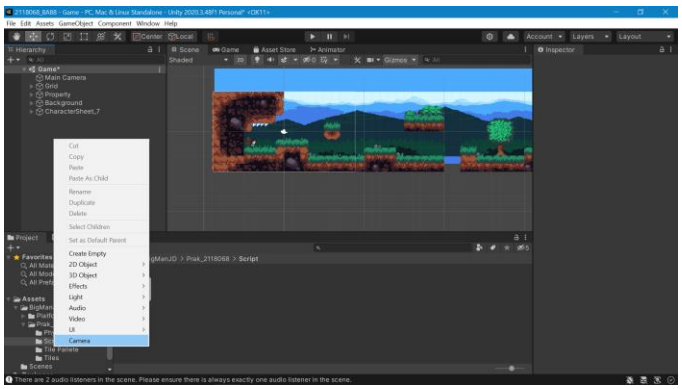
19. Jalankan *project* maka sekarang *character* bisa melompat menggunakan *Space* pada *keyboard*.



Gambar 8.17 *Character* melompat

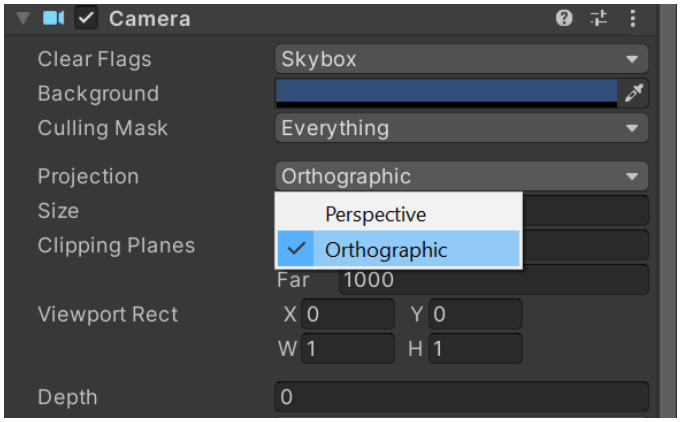
B. Membuat Pergerakan Kamera

1. Klik kanan pada *hierarchy* dan pilih *Camera*.



Gambar 8.18 Menambah *camera hierarchy*

2. Kemudian atur komponen *Camera* menjadi seperi berikut.



Gambar 8.19 Mengatur komponen *camera*

3. Buat *file C Sharp* baru pada *folder Script* dan beri nama *CameraFollow*.

Gambar 8.20 Menambah *file c sharp camera*

4. *Double click* pada *file* tersebut dan masukan *source code* berikut.

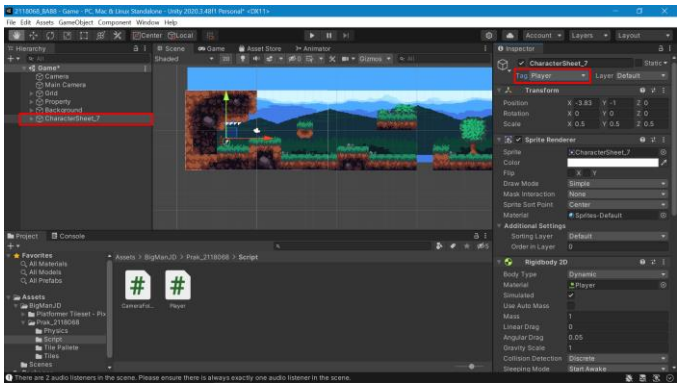
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

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);
    }
}
```

5. Jika sudah masukan *file* tersebut dalam *Camera* pada *hierarchy* dan atur juga *Max X And Y* yang ada pada komponen *Camera Follow* menjadi 80.

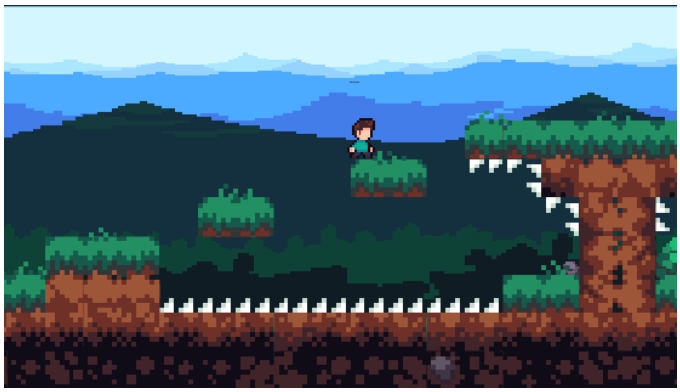
Gambar 8.21 Komponen *camera follow*

6. Setelah itu ubah *Tag* pada *character* menjadi *Player*, dengan cara klik *character* kemudian pada *inpector* ubah *tag* menjadi *Player*.



Gambar 8.22 Mengubah *tag* pada *character*

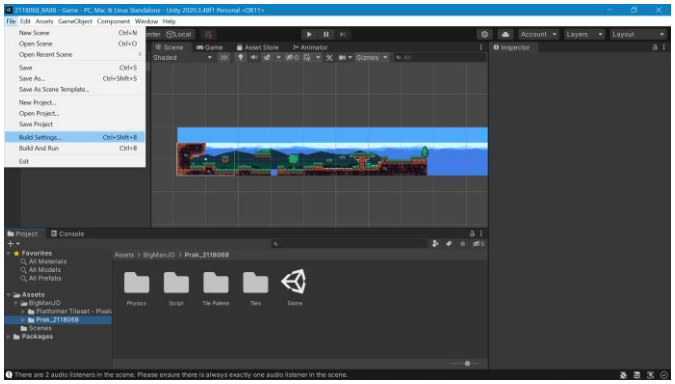
7. Maka jika *project* dijalankan *camera* akan mengikuti *player* yang bergerak.



Gambar 8.23 Kamera bergerak

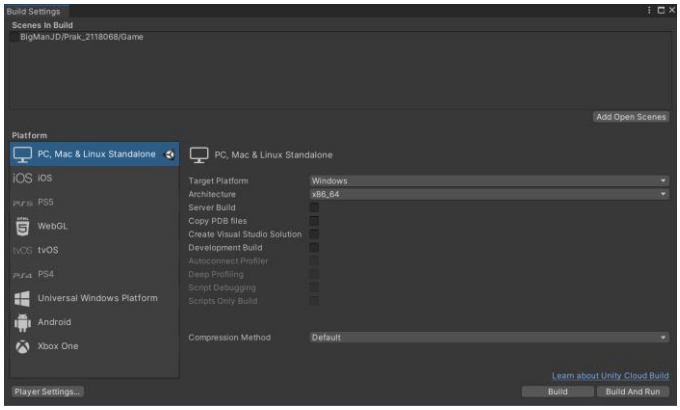
C. Render Project

1. *Render project* dengan cara klik *menu File* dan pilih *Build Setting*.



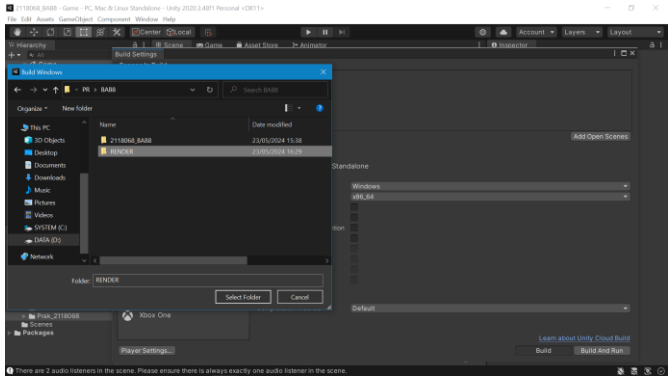
Gambar 8.24 *Build setting*

2. Dan pastikan *setting build* seperti berikut, jika sudah klik *Build*.



Gambar 8.25 *Setting build*

3. Maka akan muncul *pop up* seperti berikut untuk menyimpan *file render* jika sudah klik *Select Folder*.



Gambar 8.26 Menyimpan hasil *render*

4. Tunggu hingga proses selesai, jika sudah buka *file .exe* yang ada pada *folder* untuk mengecek hasil *render*.

Gambar 8.27 Hasil *render project*

D. Link Github

https://github.com/AhmadBahrulIlmi/2118068_PRAK_ANIGAME.git

E. Kuis

```
using System.Collections;
using System.Collections.Generic;
using UnitEngine;

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

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

Penjelasan :

Pertama terdapat pemanggilan komponen *c sharp* kemudian terdapat sebuah *class CameraFollow* yang diinisialisasi dengan *MonoBehaviour* sebagai komponen objek *unity*, yang didalam *class* tersebut terdapat deklarasi variabel player kemudian diikuti dengan *method* bertipe *void* dengan nama *Update*, yang didalam *method* tersebut terdapat sebuah perintah untuk mengatur posisi kamera yang mengikuti pergerakan player sesuai dengan koordinat x, y, dan z.