TUGAS PERTEMUAN: 10

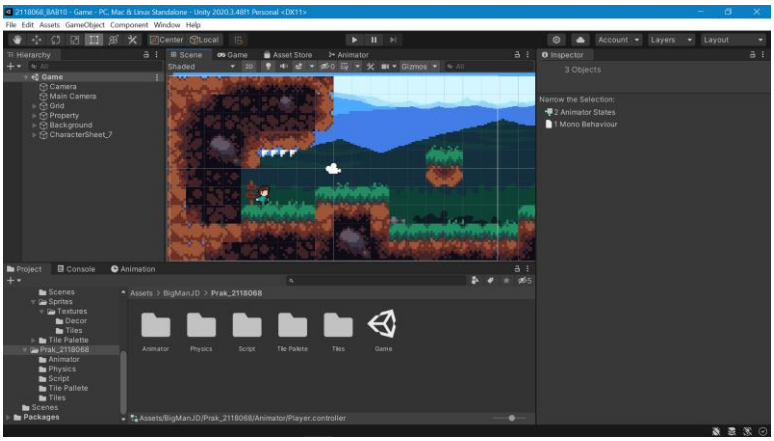
RESPAWN AND ENEMY ATTACK

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

1.1 Tugas 10 : Menerapkan Respawn dan Enemy Attack

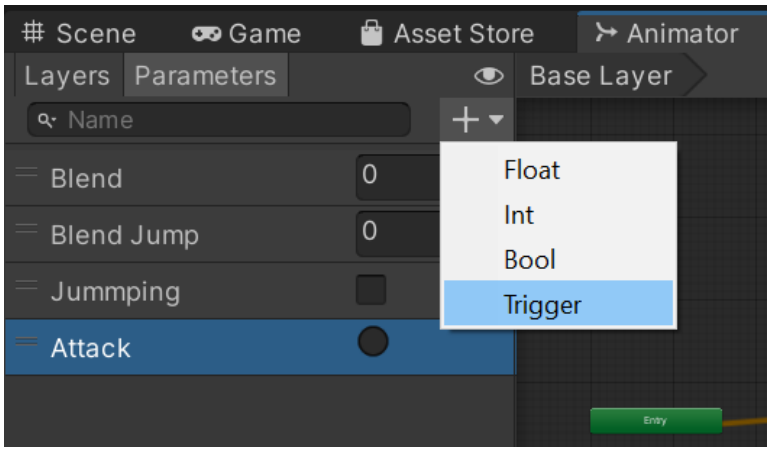
A. Mekanisme Attack

1. Buka *file* sebelumnya yang sudah memiliki animasi pada karakternya.



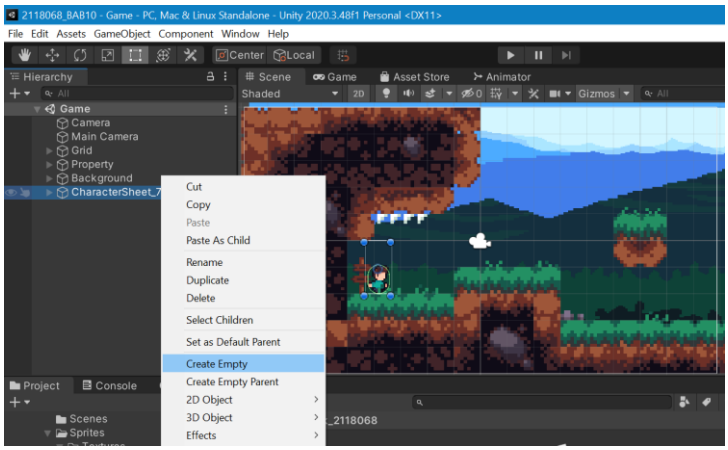
Gambar 10.1 Buka *project unity*

2. Kemudian buat *parameters* baru pada *tab Animator* beri nama *Attack* dengan tipe data *Trigger*.



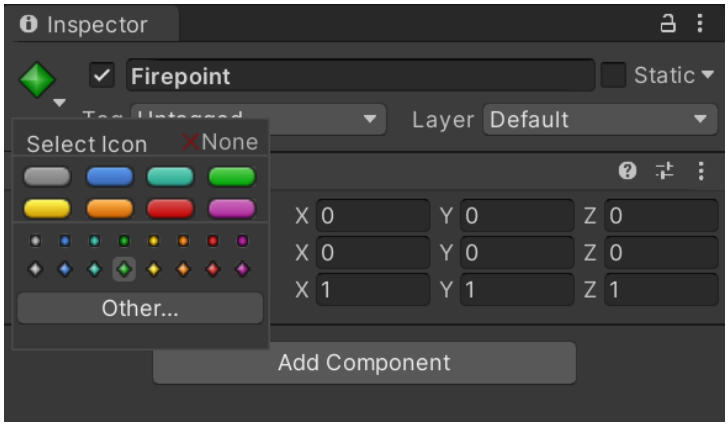
Gambar 10.2 Parameter *attack*

3. Kemudian klik kanan pada karakter dan pilih *Create Empty* beri nama *Firepoint*.



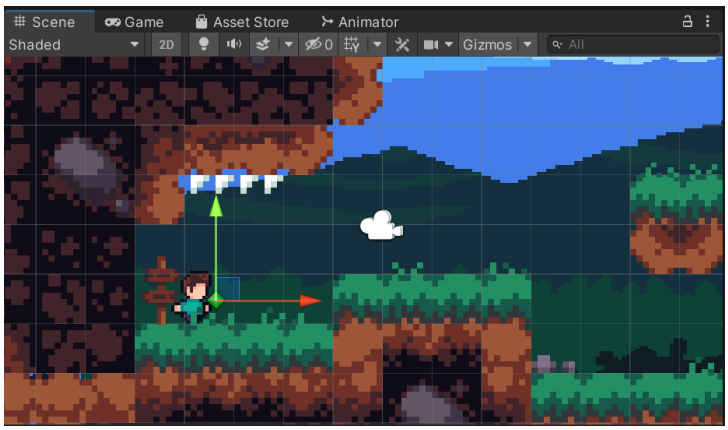
Gambar 10.3 Membuat *firepoint*

4. Jika sudah beri *icon firepoint* pada *inspector* seperti berikut.



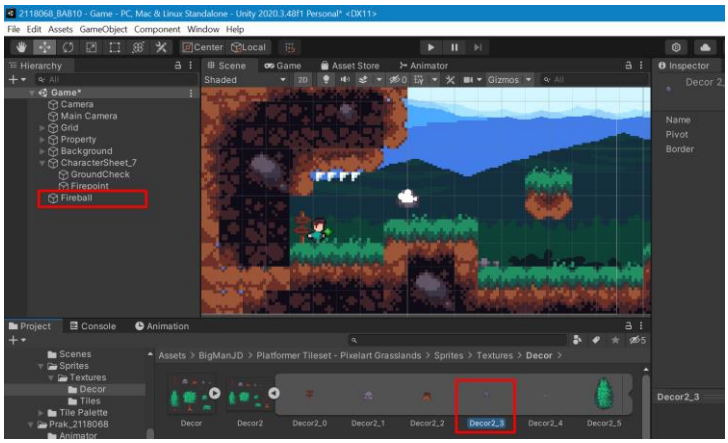
Gambar 10.4 *Icon firepoint*

5. Poisikan *firepoint* seperti berikut menggunakan *move tools*.



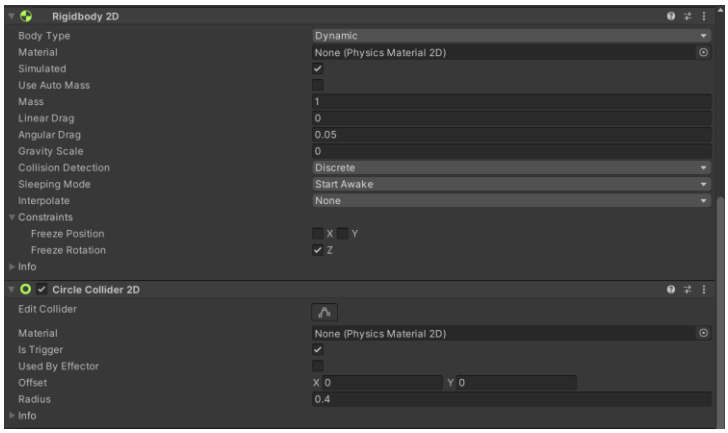
Gambar 10.5 Posisi *firepoint*

6. Dan tambahkan *asset* pada *hierarchy* kemudian ubah nama menjadi *Fireball*.



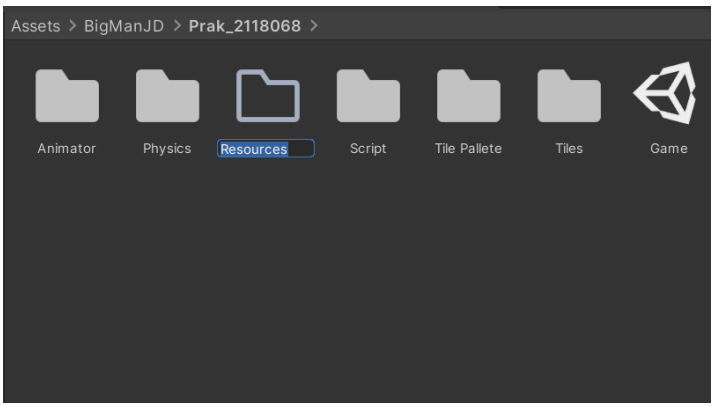
Gambar 10.6 Menambahkan *fireball*

7. Lalu beri komponen *Rigidbody 2D* dan *Circle Collider* 2D pada *Fireball* atur menjadi seperti berikut.



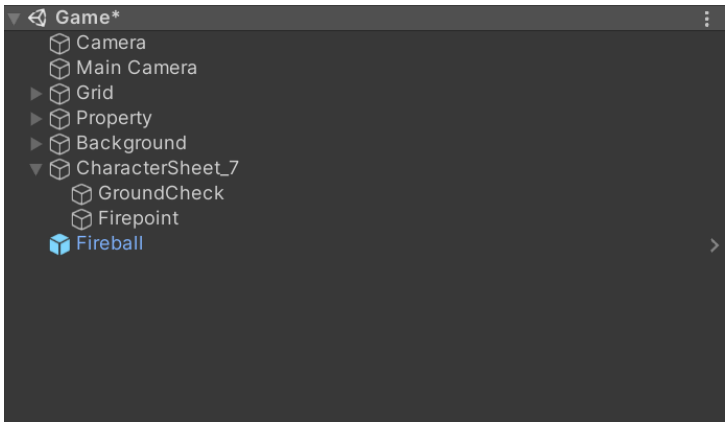
Gambar 10.7 Komponen pada *fireball*

8. Kemudian buat *folder* baru dengan nama *Resources*.



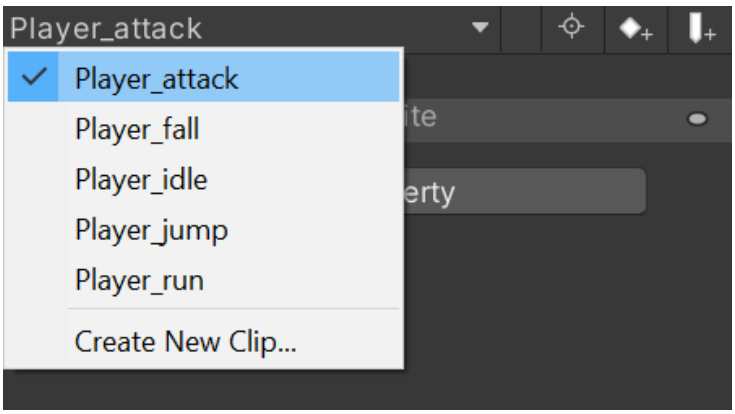
Gambar 10.8 *Folder resources*

9. Dan letakkan *fireball* pada *folder resources* maka *fireball* akan menjadi seperti berikut.



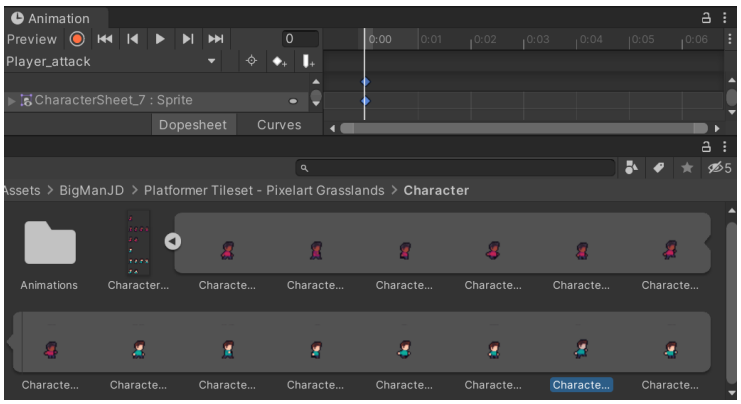
Gambar 10.9 Tampilan *fireball*

10. Kemudian klik karakter dan pada *menu animation* buat *clip* baru dengan nama *Player_attack*.



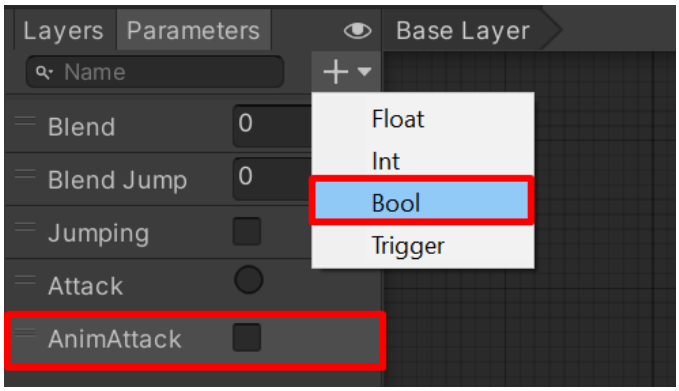
Gambar 10.10 *Animation player attack*

11. Dan tambahkan *asset* berikut pada *timeline player_attack*.



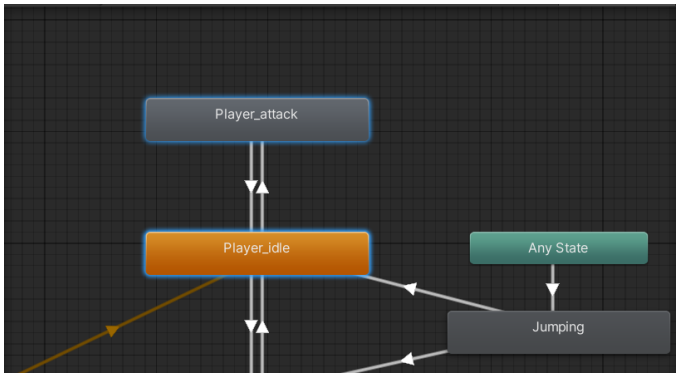
Gambar 10.11 *Assets palyer attack*

12. Pada *tab Animator* buat *parameters* baru dengan nama *AnimAttack* dan tipe data *Boolean*.



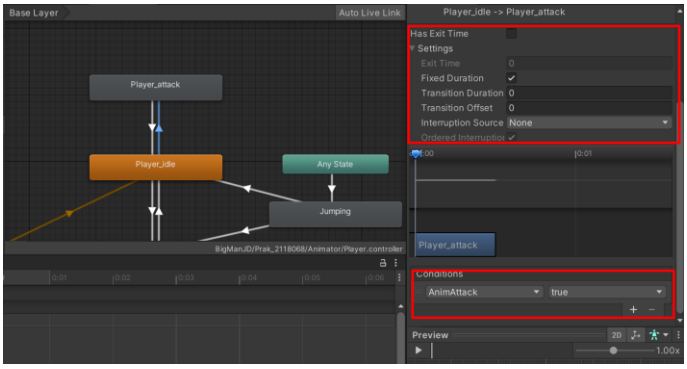
Gambar 10.12 *Parameters animattack*

13. Jika sudah buat dua *transition* pada *player_idle* ke *player_attack* seperti berikut.



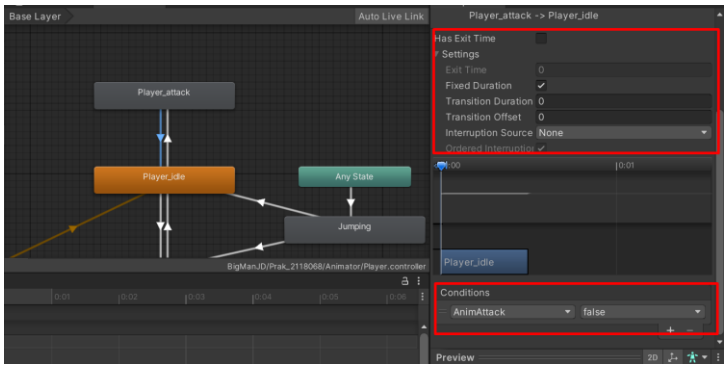
Gambar 10.13 *Transition* pada *animator*

14. Klik transisi *palyer_idle* ke *player_attack* pada *inspector* ubah *condition* menjadi *AnimAttack* dengan kondisi *true* dan juga *settings* menjadi seperti berikut.



Gambar 10.14 Transisi *player_idle* ke *player_attack*

15. Klik juga pada transisi player_attack ke player_idle ubah kondisi serta settings nya menjadi seperti berikut.



Gambar 10.15 Transisi *player_attack* ke *player_idle*

16. Pada *script Player* tambahkan variabel berikut.

```
public GameObject bullet;
public Transform firePoint;
bool animAttack = false;
```

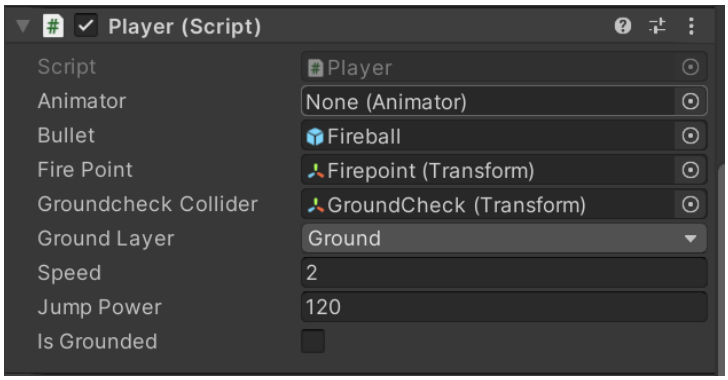
17. Dan tambahkan fungsi *Attack* dibawah fungsi *FixedUpdate*.

```
IEnumerator Attack()
  {
    //menyerang
    animator.SetTrigger("Attack");
    //animasi menyerang
    animAttack = true;
    animator.SetBool("AnimAttack", true);

    yield return new WaitForSeconds(0.25f);

    float direction = facingRight ? 1f : -1f;

    GameObject Fireball = Instantiate(bullet,
firePoint.position, Quaternion.identity);
    Fireball.GetComponent<Rigidbody2D>().velocity = new
Vector2(direction * 10f, 0);

    animator.SetBool("AnimAttack", false);
    animAttack = false;

    Destroy(Fireball, 2f);
  }
```

18. Kemudian buat kondisi baru pada fungsi *Update*.

```
if (Input.GetKeyDown(KeyCode.Return) && !animAttack)
  {
    StartCoroutine(Attack());
  }
```

19. Setelah itu klik karakter dan pada *inspector* ubah *Script Player* menjadi seperti berikut.



Gambar 10.16 Tampilan *script player*

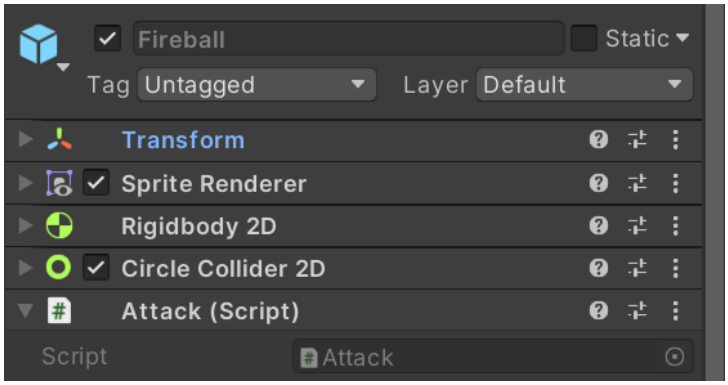
20. Lalu buat *file script* baru letakan pada *folder Script* dan beri nama *Attack*.

Gambar 10.17 *File script attack*

21. Beri *source code* berikut pada *file attack*.

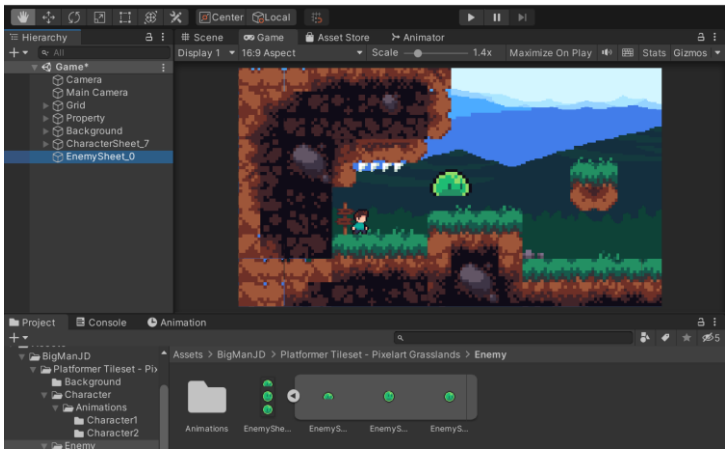
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Attack : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Enemy"))
        {
            Destroy(gameObject);
            Destroy(collision.gameObject);
        }
    }
}
```

22. Klik *fireball* pada *folder resources* dan tambahkan *script Attack*.



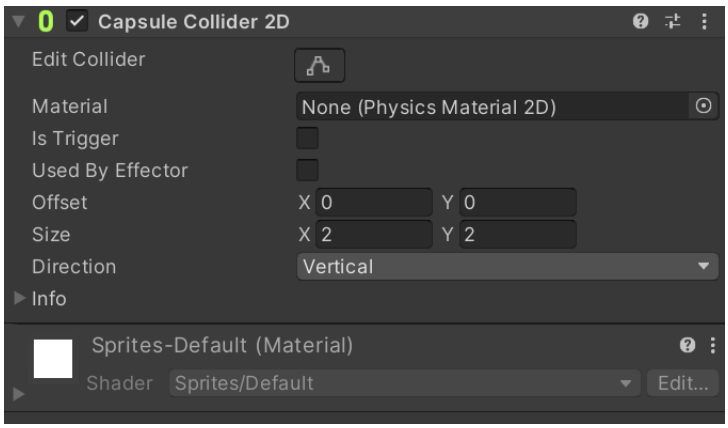
Gambar 10.18 *Script attack* pada *fireball*

23. Setelah itu tambahkan *enemy1* pada *hierarchy* untuk mencoba menyerang.



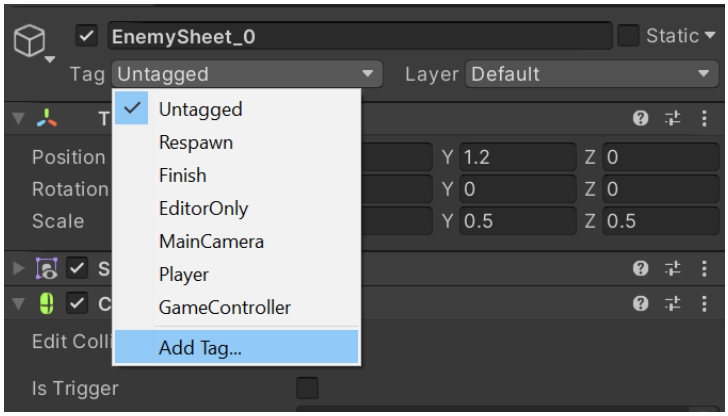
Gambar 10.19 Menambah *enemy1*

24. Beri komponen *Capsule Collider 2D* pada *enemy1*.



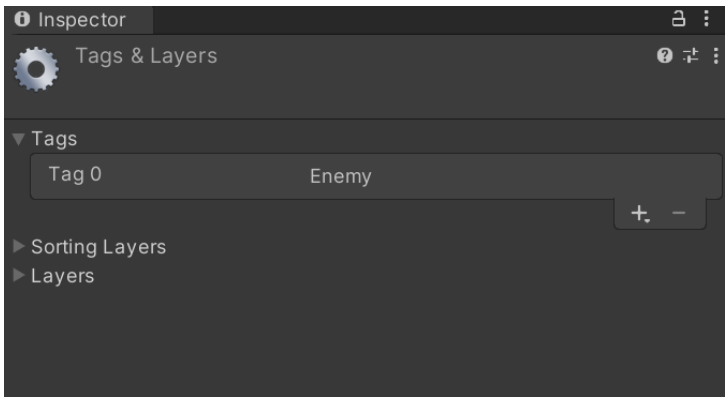
Gambar 10.20 Komponen pada *enemy1*

25. Buat *tags* baru dengan cara klik *tags* dan pilih *Add Tags*.



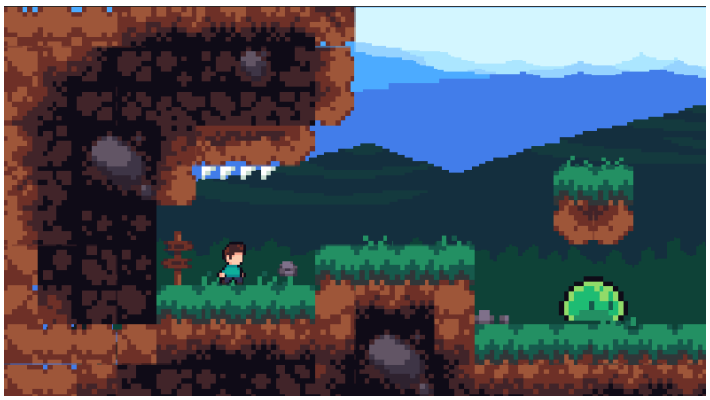
Gambar 10.21 Menambah *tags* baru

26. Klik *icon plus* dan beri nama *Enemy*, dan ubah *tags enemy1* menjadi *Enemy*.



Gambar 10.21 Tampilan *Tags enemy*

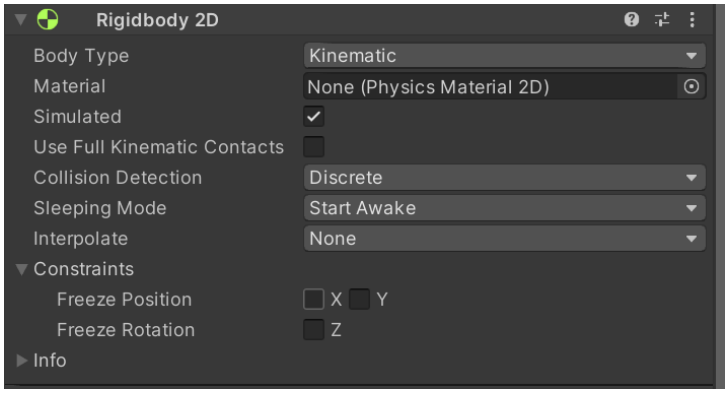
27. Jika sudah *run project* maka sekarang karakter bisa menyerang dengan klik *Enter* pada *keyboard* dan *fireball* akan mengikuti arah karakter dan juga memiliki animasi menyerang saat karakter mengeluarkan *fireball*.



Gambar 10.22 Hasil mekanisme *attack*

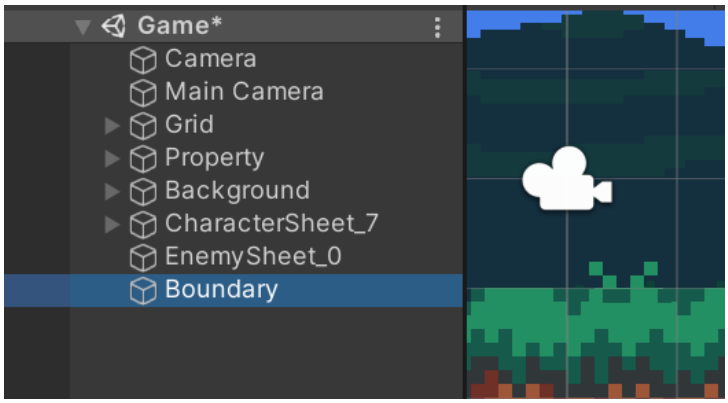
B. Enemy Behavior NPC

1. Pada *enemy1* sebelumnya tambahkan komponen *Rigidbody 2D* dan atur menjadi seperti berikut.



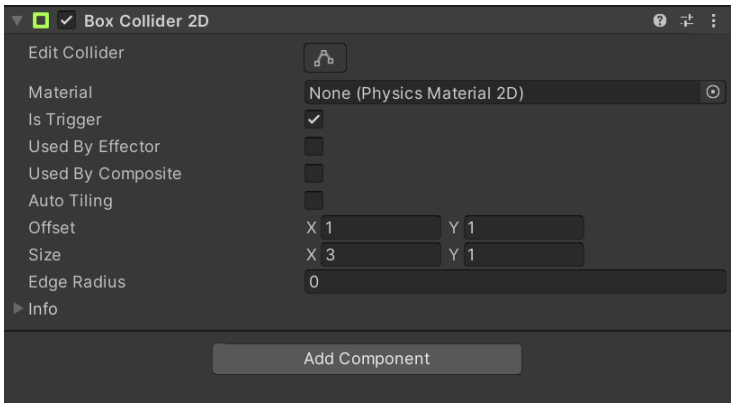
Gambar 10.23 Komponen *rigidbody 2d enemy1*

2. Klik kanan pada *hierarchy* dan pilih *create empty* beri nama *Boundary*.



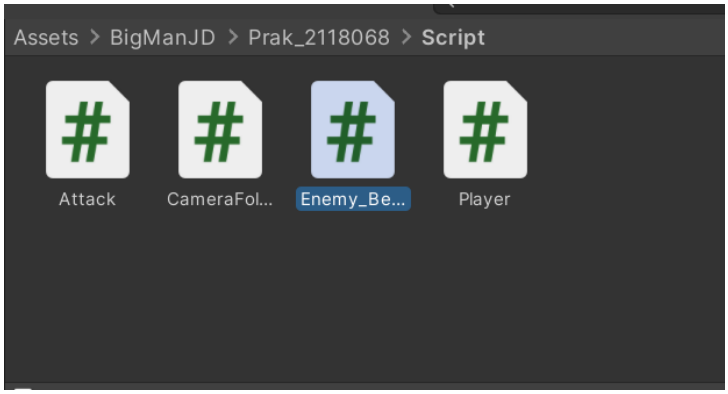
Gambar 10.24 Membuat *object boundary*

3. Tambahkan komponen *Box Collider 2D* pada *boundary* dan atur menjadi seperti berikut.



Gambar 10.25 Komponen pada *boundary*

4. Buat *file csharp* baru pada *folder script* dan beri nama *Enemy_Behavior*.

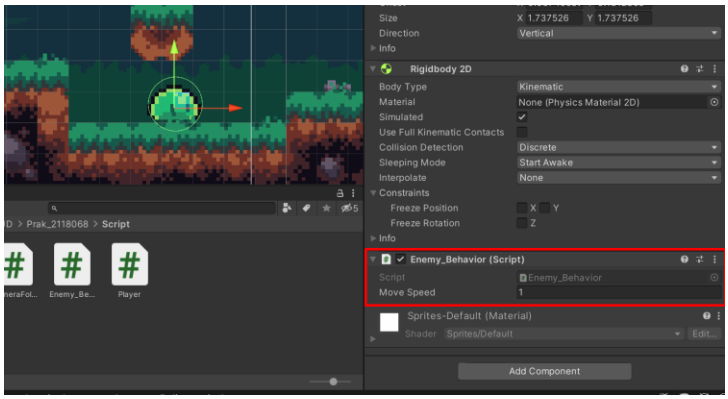


Gambar 10.26 *File enemy_behavior*

5. Dan beri *source code* berikut.

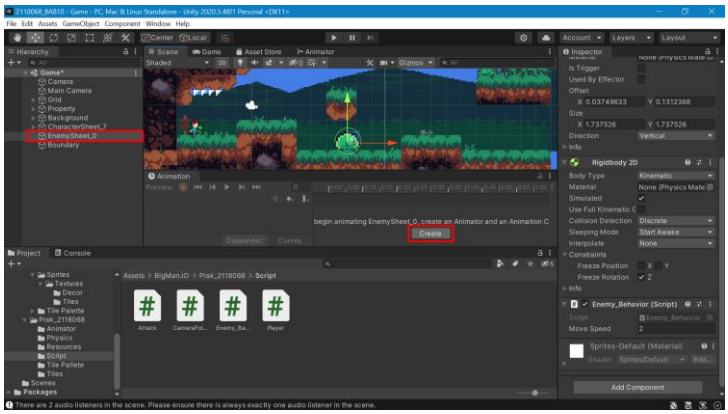
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy_Behavior : MonoBehaviour
{
    [SerializeField] float moveSpeed = 1f;
    Rigidbody2D rb;

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

    void Update()
    {
        if (isFacingRight())
        {
            rb.velocity = new Vector2(moveSpeed, 0f);
        }
        else
        {
            rb.velocity = new Vector2(-moveSpeed, 0f);
        }
    }

    private bool isFacingRight()
    {
        return transform.localScale.x > Mathf.Epsilon;
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        transform.localScale = new Vector2(-
transform.localScale.x, transform.localScale.y);
    }
}
```

6. Setelah itu klik *enemy1* pada *hierarchy* dan tambahkan komponen *script Enemy_Behavior*.



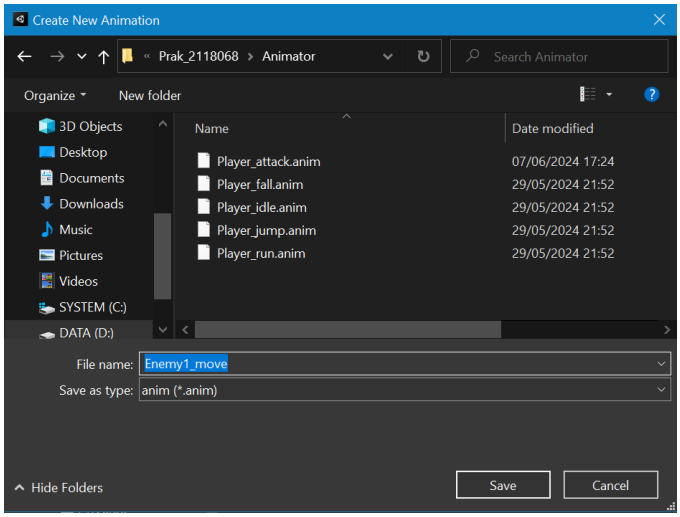
Gambar 10.27 Komponen *script enemy_behavior*

7. Kemudian klik *enemy1* dan pada *menu animation* pilih *create*.



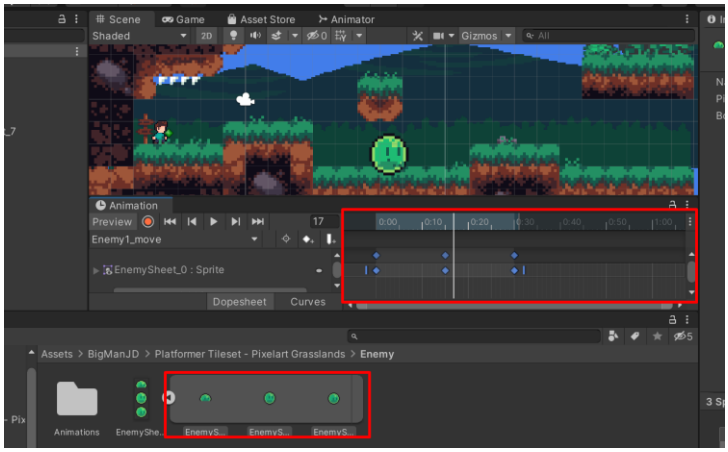
Gambar 10.28 *Create animation enemy1*

8. Simpan *animation* dengan nama *Enemy1_move* pada *folder Animator*.



Gambar 10.29 Menyimpan *animation enemy1*

9. Dan tembahkan *asset* berikut pada *timeline* dan atur juga durasinya.



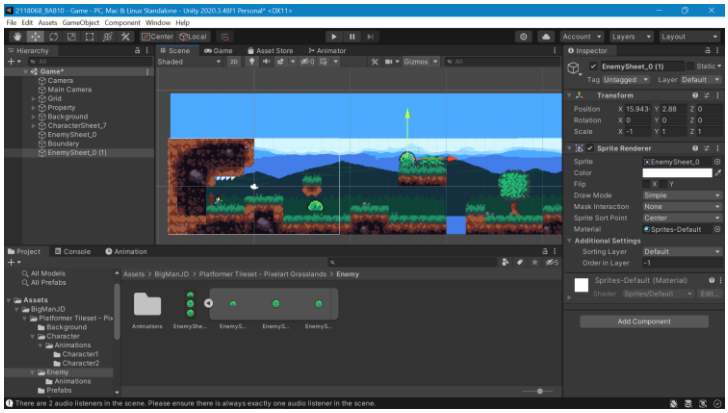
Gambar 10.30 *Animation enemy1* pada *timeline*

10. Jika di *run* maka *Enemy1* akan berjalan ke kiri dan ke kanan serta memiliki animasi berjalan.

Gambar 10.31 Hasil *enemy behavior*

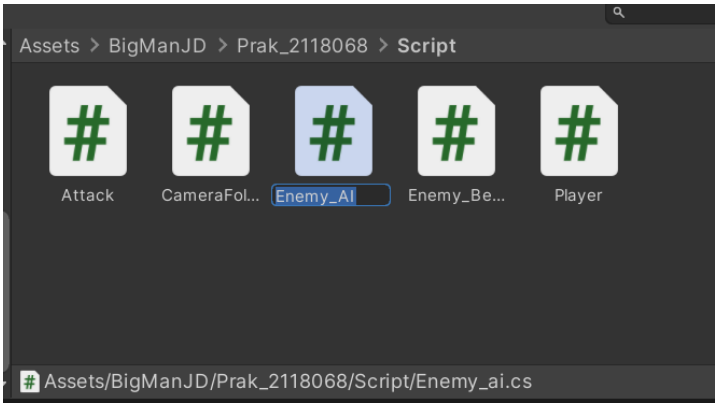
C. Enemy AI

1. Tambahkan *enemy* lagi pada *hierarchy* yang akan menjadi *enemy2*.



Gambar 10.32 Menambahkan *enemy2*

2. Dan buat *script* baru dengan nama *Enemy_AI*.



Gambar 10.33 *File script enemy_ai*

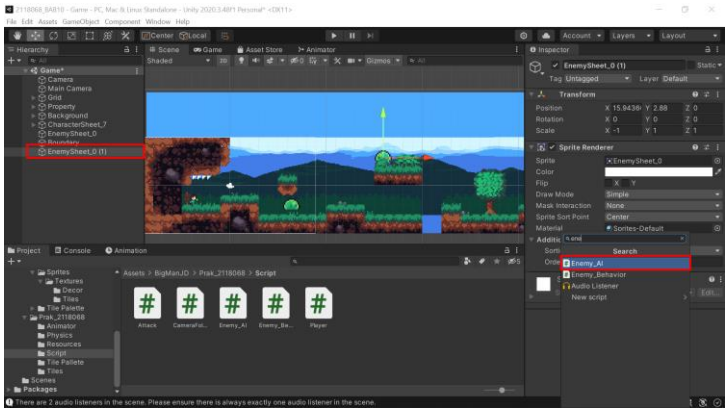
3. Dan tambahkan *source code* berikut.

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

public class Enemy_AI : MonoBehaviour
{
    public float speed; // Kecepatan gerakan musuh
    public float lineOfSite; // Jarak penglihatan musuh
    private Transform player; // Transform dari pemain
    private Vector2 initialPosition; // Posisi awal
musuh
    private bool facingRight = true; // Menyimpan arah
awal musuh (menghadap kanan)

    // Use this for initialization
    void Start()
    {
        // Mencari pemain berdasarkan tag
        player                                          =
GameObject.FindGameObjectWithTag("Player").transform;
        // Menyimpan posisi awal musuh
        initialPosition                                 =
GetComponent<Transform>().position;
    }

    // Update is called once per frame
    void Update()
    {
        // Menghitung jarak antara musuh dan pemain
        float            distanceToPlayer            =
Vector2.Distance(player.position, transform.position);

        // Jika pemain berada dalam jarak penglihatan
musuh
        if (distanceToPlayer < lineOfSite)
        {
            // Musuh bergerak menuju pemain
```

```
            transform.position                              =
Vector2.MoveTowards(this.transform.position,
player.position, speed * Time.deltaTime);
            // Menghadapkan musuh ke arah pemain
            FlipTowardsPlayer();
        }
        else
        {
            // Musuh kembali ke posisi awal
            transform.position                              =
Vector2.MoveTowards(transform.position,
initialPosition, speed * Time.deltaTime);
            //  Menghadapkan  musuh  ke  arah  awal  jika
tidak mengejar pemain
            FlipTowardsInitialPosition();
        }
    }

    // Menghadapkan musuh ke arah pemain
    void FlipTowardsPlayer()
    {
        if (player.position.x > transform.position.x &&
!facingRight)
        {
            Flip();
        }
        else       if       (player.position.x       <
transform.position.x && facingRight)
        {
            Flip();
        }
    }

    // Menghadapkan musuh ke arah awal
    void FlipTowardsInitialPosition()
    {
        if (initialPosition.x > transform.position.x &&
!facingRight)
        {
            Flip();
        }
        else       if       (initialPosition.x       <
transform.position.x && facingRight)
        {
            Flip();
        }
    }

    // Membalik arah musuh
    void Flip()
    {
        facingRight = !facingRight;
        Vector3 localScale = transform.localScale;
        localScale.x *= -1;
        transform.localScale = localScale;
    }

    //  Untuk  menggambar  jarak  penglihatan  musuh  di
editor
```

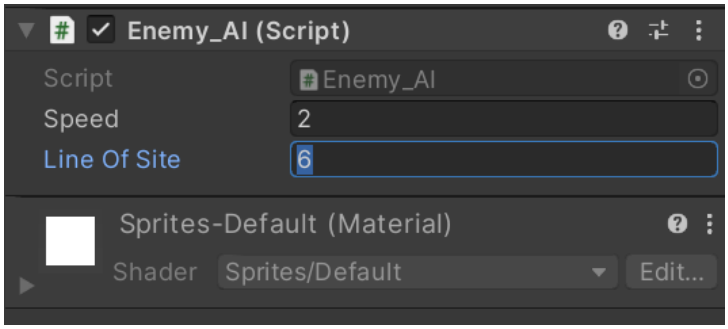
```
    private void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(transform.position,
lineOfSite);
    }
}
```

4. Kemudian pada *Enemy2* tambahkan komponen *script enemy_ai*.



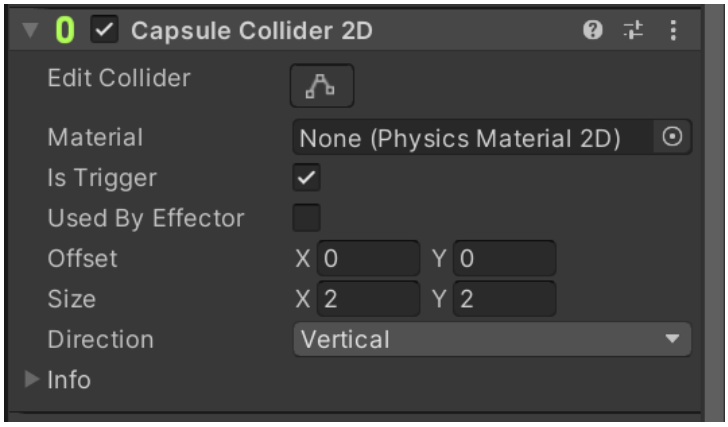
Gambar 10.34 Komponen *script enemy_ai*

5. Dan atur *speed* serta *line of site* menjadi seperti berikut.



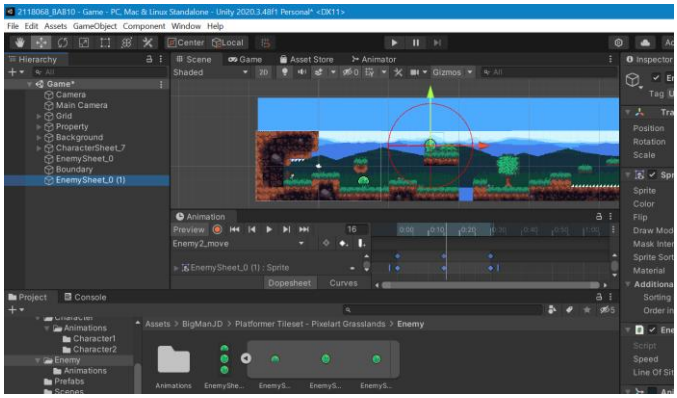
Gambar 10.35 Tampilan *script enemy_ai*

6. Tambahkan juga komponen *Capsule Collider 2D* pada *Enemy2*.



Gambar 10.36 *Capsule collider 2d enemy2*

7. Tambahkan juga animasi dengan cara yang sama klik *create* dan simpan animasi dengan nama *enemy2_move* dan tambahkan *asset* pada *timeline*.



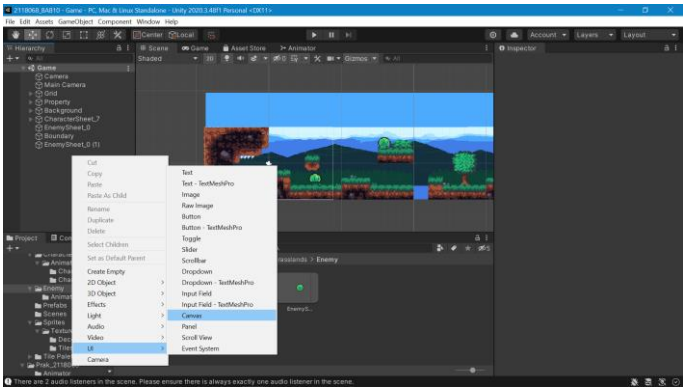
Gambar 10.37 Membuat animasi *enemy2*

8. Jika di *run* maka *enemy2* akan mengikuti pergerakan karakter atau *player* dan juga *enemy2* memiliki animasi.

Gambar 10.38 Hasil penerapan *enemy ai*

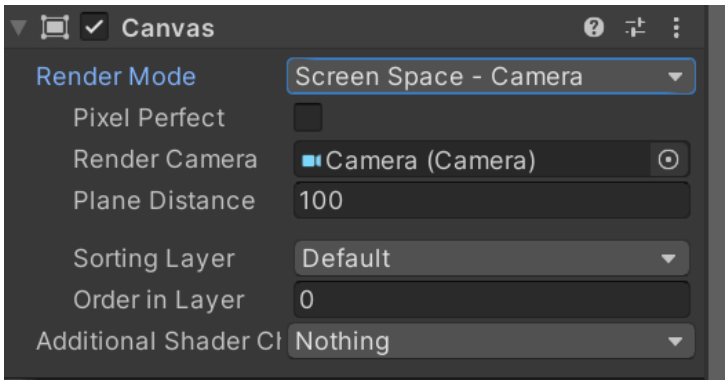
D. Respawn

1. Klik kanan pada *hierarchy* dan pilih UI kemudian *Canvas* untuk membuat *health bar*.



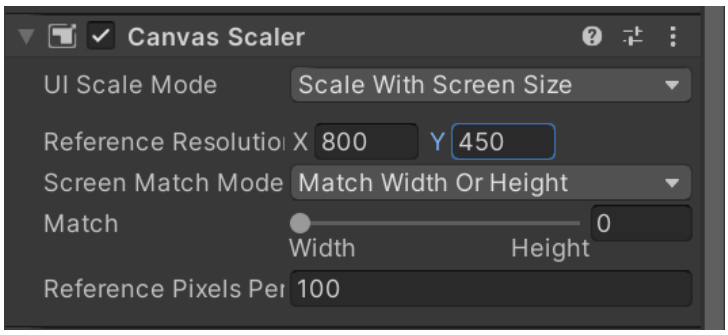
Gambar 10.39 Menambah objek UI

2. Klik *canvas* dan atur komponen *canvas* pada *inspector* menjadi seperti berikut.



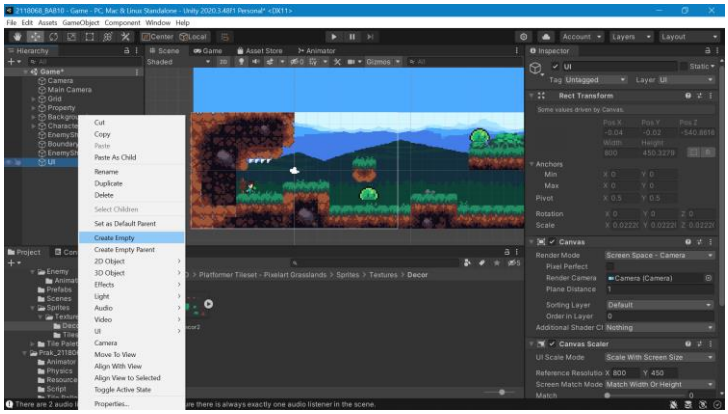
Gambar 10.40 Komponen *canvas*

3. Dan atur juga komponen canvas scaler menjadi seperti berikut.



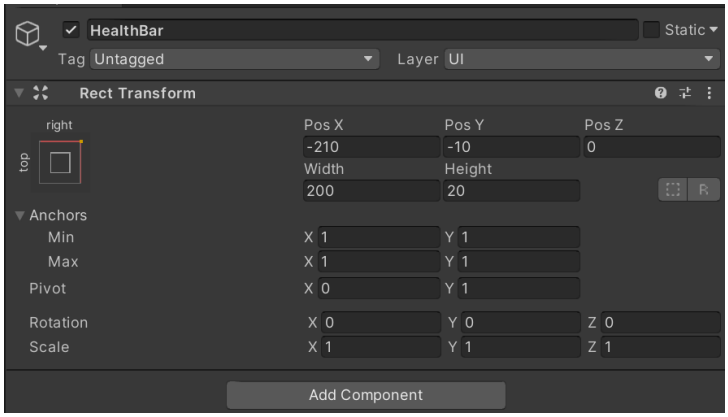
Gambar 10.41 Komponen *canvas scaler*

4. Klik kanan pada *UI* dan pilih *Create Empty* beri nama *HealthBar*.



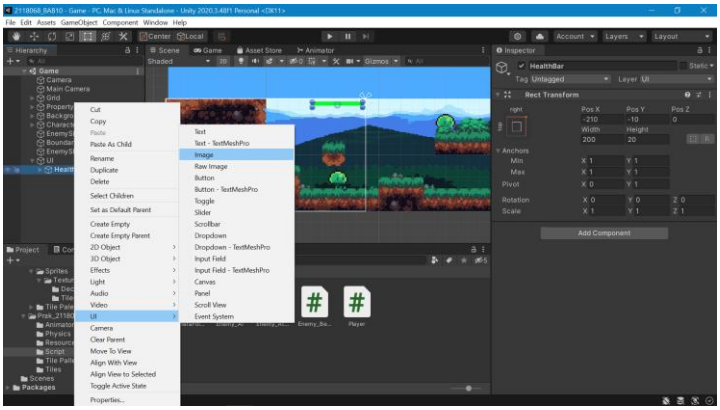
Gambar 10.42 Membuat objek *health bar*

5. Ubah *react transform* menjadi seperti berikut pada objek *health bar*.



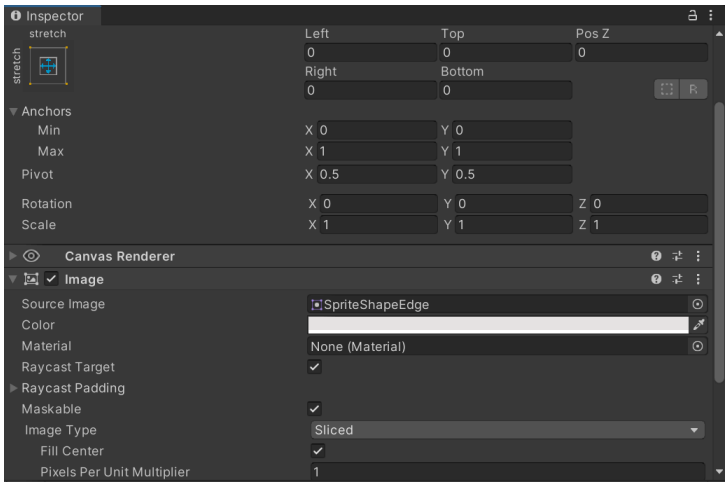
Gambar 10.43 Komponen *react transform*

6. Klik kanan pada *health bar* dan pilih UI kemudian pilih *Image* beri nama BG.



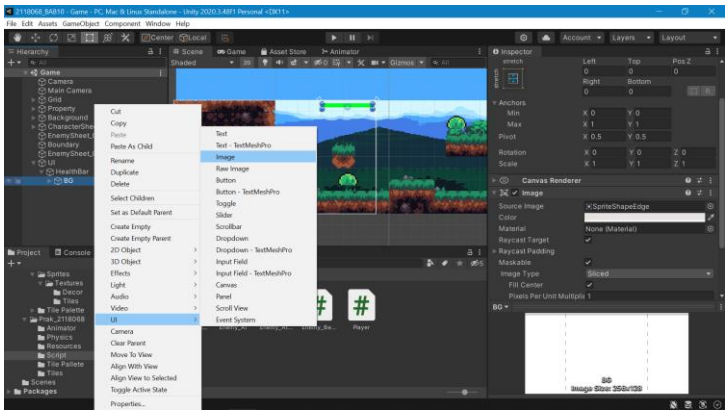
Gambar 10.44 Membuat objek *background*

7. Dan ubah komponen BG pada *inspector* menjadi seperti berikut.



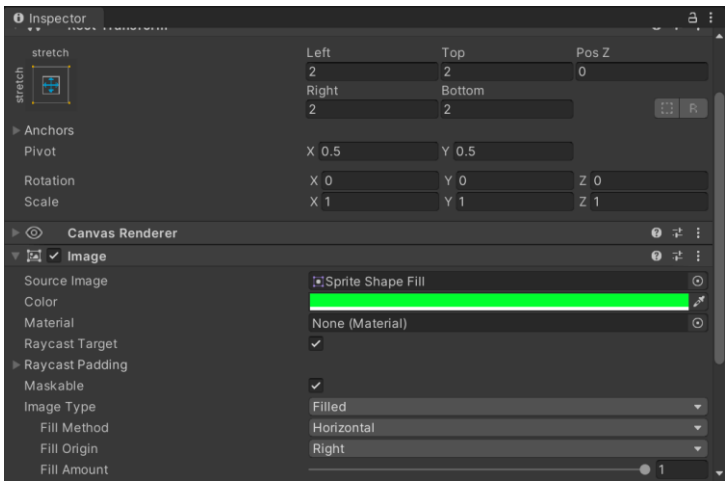
Gambar 10.45 Tampilan komponen BG

8. Klik kanan lagi pada BG dan pilih UI kemudian *Image* dan beri nama *Fill*.



Gambar 10.46 Membuat objek *fill*

9. Dan atur juga komponen *fill* pada *inspector* menjadi seperti berikut.



Gambar 10.47 Tampilan komponen *fill*

10. Pada *script* player tambahkan variabel berikut.

```
public int nyawa;
[SerializeField] public int maxNyawa = 3;
[SerializeField] Vector3 respawn_loc;
public bool play_again;
```

11. Kemudian tambahkan perintah berikut pada fungsi *Awake*.

```
nyawa = maxNyawa;
respawn_loc = transform.position;
```

12. Dan buat fungsi *playagain*.

```
void playagain()
  {
    if(play_again == true)
    {
      nyawa = maxNyawa;
      transform.position = respawn_loc;
```

```
      play_again = false;
    }
  }
```

13. Setelah itu tambahkan kondisi berikut pada fungsi *Update*.

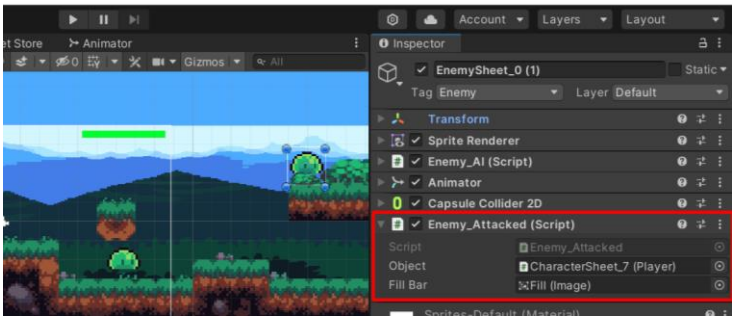
```
if(nyawa < 0)
{
  playagain();
}
if(transform.position.y < -10)
{
  play_again = true;
  playagain();
}
```

14. Dan buat *file Csharp* baru beri nama *Enemy Attacked* dan tambahkan *source code* berikut.

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

public class Enemy_Attacked : MonoBehaviour
{
    [SerializeField] public Player Object;
    public Image fillBar;

    void Start()
    {
        if (Object == null)
        {
            Object                                          =
GameObject.FindWithTag("Player").GetComponent<Player>()
;
        }
    }

    void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            Object.nyawa--;

            if (Object.nyawa < 0)
            {
                Object.play_again = true;
            }
            UpdateHealthBar();
        }
        if (collision.CompareTag("Enemy"))
        {
            LoseHealth(25);
        }
    }

    public void LoseHealth(int value)
```

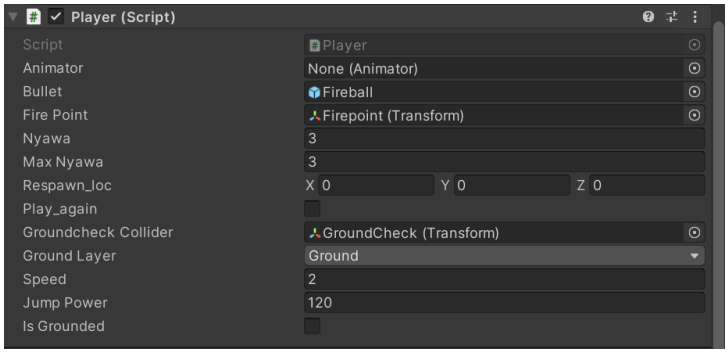
```
    {
        if (Object.nyawa <= 0)
            return;

        Object.nyawa -= value;

        UpdateHealthBar();

        if (Object.nyawa <= 0)
        {
            fillBar.fillAmount = 0f;
        }
    }

    public void UpdateHealthBar()
    {
        float fillAmount = (float)Object.nyawa / Object.maxNyawa;
        fillBar.fillAmount = fillAmount;
    }
}
```

15. Lalu pada *Enemy1* dan *Enemy2* tambahkan komponen *script Enemy_Attacked* dan atur *Object* menjadi *player* serta *Fill Bar* menjadi *Fill (image).*



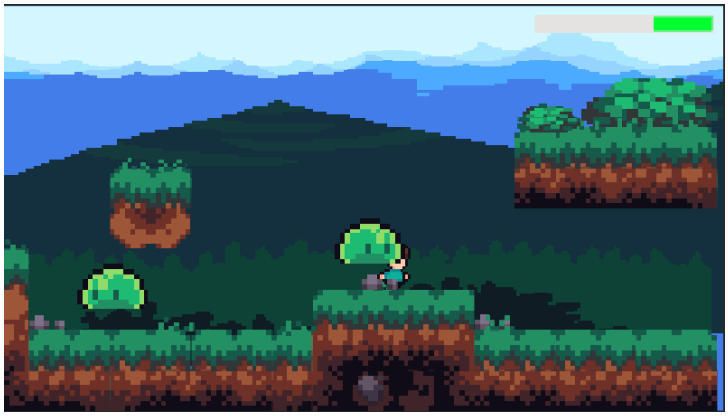
Gambar 10.48 Komponen *script enemy attacked*

16. Kemudian pada komponen *script Player* tambahkan Nyawa dan *Max* Nyawa menjadi seperti berikut.



Gambar 10.49 Komponen *script player* nyawa

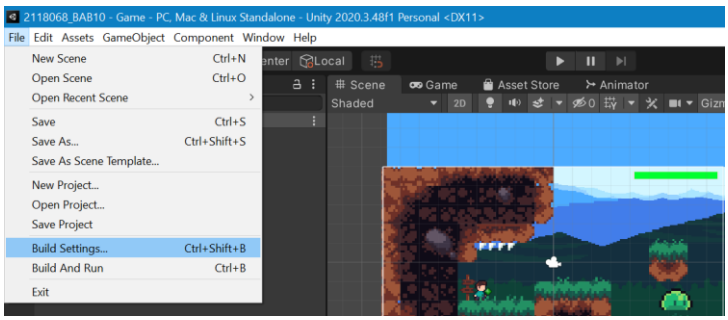
17. Jika di *run* ketika karakter menyentuh *Enemy* nyawa akan berkurang dan juga *fill health bar* akan berkurang, jika nyawa 0 maka kembali ke posisi awal.



Gambar 10.50 Hasil penerapan *respawn*

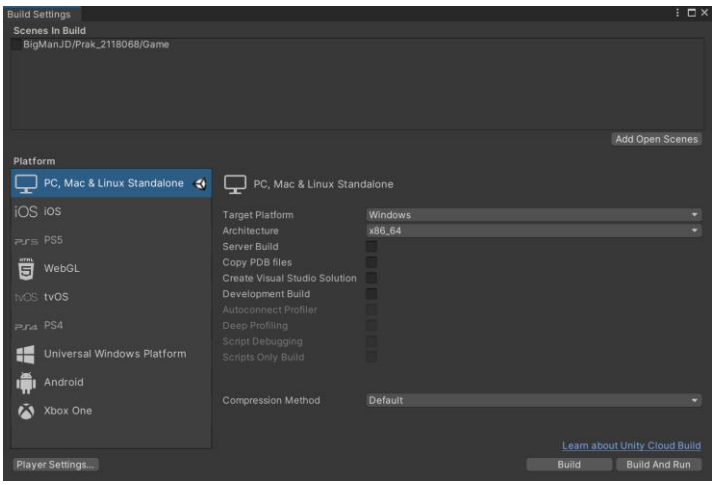
E. Render

1. Klik *menu file* dan pilih *Build Settings.*



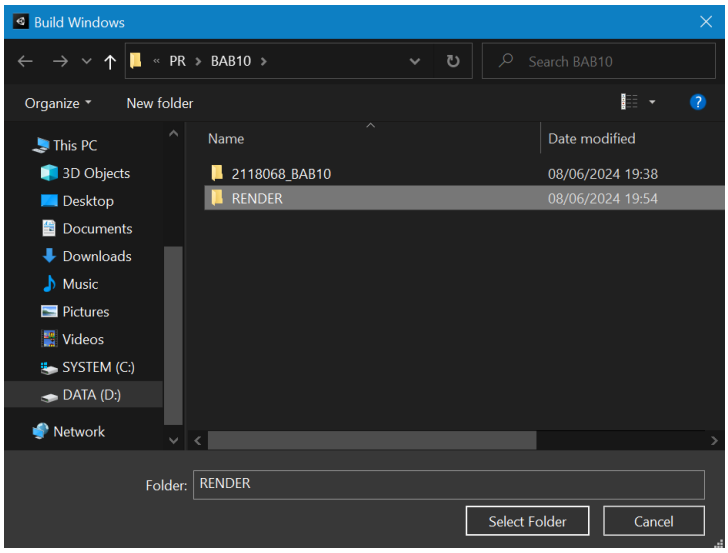
Gambar 10.51 *Render project*

2. Dan atur menjadi seperti berikut lalu klik *Build.*



Gambar 10.52 Tampilan *build settings*

3. Pilih *folder* untuk menyimpan dan klik *Select Folder* kemudian tunggu hingga proses *render selesai*.



Gambar 10.53 *Folder* penyimpanan *render*

4. Jika selesai maka akan terdapat sebuah *file exe* pada *folder* penyimpanan jika dibuka hasilnya akan seperti berikut.

Gambar 10.54 Hasil penerapan *render*

F. Link Github

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

G. Kuis

1. Tambahan

```
using UnityEngine;

public class PlayerAttack : MonoBehaviour
{
    public float attackRange = 2.0f;
    public int attackDamage = 10;

    void Update()
```

```
    {
        if (Input.GetButtonDown("Fire1"))
        {
            PerformMeleeAttack();
        }
    }

    void PerformMeleeAttack()
    {
        RaycastHit hit;
        if          (Physics.Raycast(transform.position,
transform.forward, out hit, attackRange))
        {
            if (hit.collider.CompareTag("Enemy"))
            {
                EnemyHealth        enemyHealth        =
hit.collider.GetComponent<EnemyHealth>();

                if (enemyHealth != null)
                {

enemyHealth.TakeDamage(attackDamage);
                }
            }
        }
    }
}
```

Penjelasan:

Pada *source code* PlayerAttack yang diimplementasikan memungkinkan pemain untuk melakukan serangan jarak dekat dengan menekan tombol "*Fire1*". Serangan ini dilakukan dengan menembakkan *ray* dari posisi pemain ke arah depan hingga jarak tertentu yang ditentukan oleh variabel *attackRange*. Jika ray tersebut mengenai objek dengan *tag "Enemy"*, komponen *EnemyHealth* pada objek tersebut akan dicari. Jika ditemukan, *method TakeDamage* pada komponen tersebut akan dipanggil, mengurangi kesehatan musuh sesuai dengan jumlah kerusakan yang ditentukan oleh variabel *attackDamage*. Kode ini memastikan bahwa serangan hanya berlaku pada musuh yang berada dalam jangkauan serangan, memberikan mekanisme yang efisien untuk mengelola interaksi tempur dalam permainan.