



TUGAS PERTEMUAN: 10

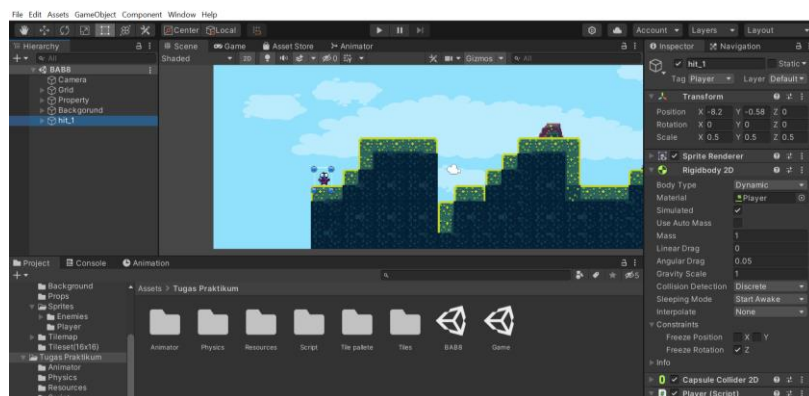
RESPAWN AND ENEMY ATTACK

NIM	:	2118059
Nama	:	Arika Risma Nabella
Kelas	:	B
Asisten Lab	:	Bagas Anardi Surya W (2118004)

1.1 Tugas 10 : Menerapkan Respawn dan Enemy Attack

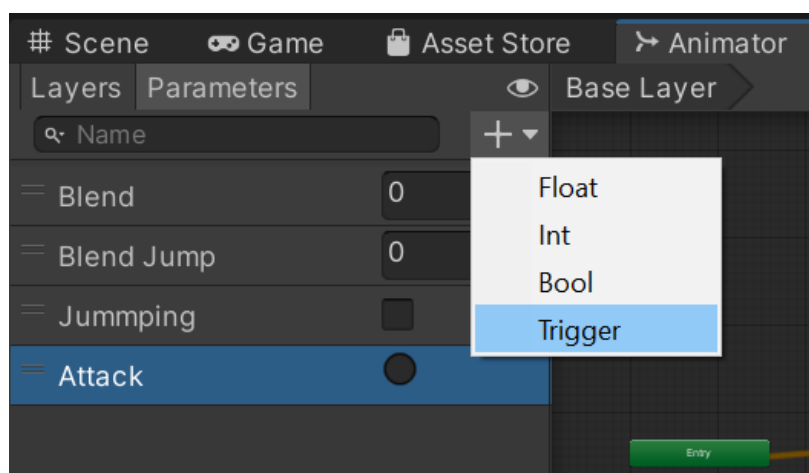
A. Mekanisme Attack

1. Buka *file* sebelumnya.



Gambar 10.1 File Project Sebelumnya

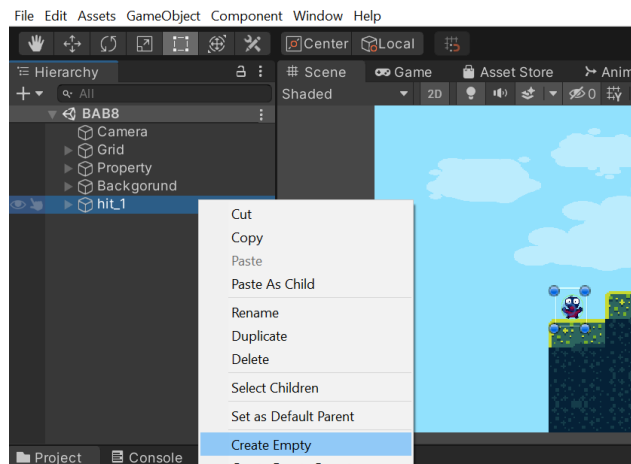
2. Kemudian buat *parameters* baru pada *tab Animator* dengan nama *Attack* dan tipe data *Trigger*.



Gambar 10.2 Membuat Parameter

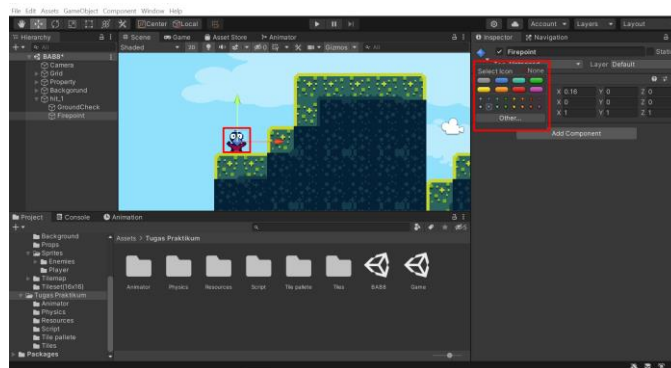


3. Klik kanan pada karakter kemudian pilih *Create Empty* dan beri nama *Firepoint*.



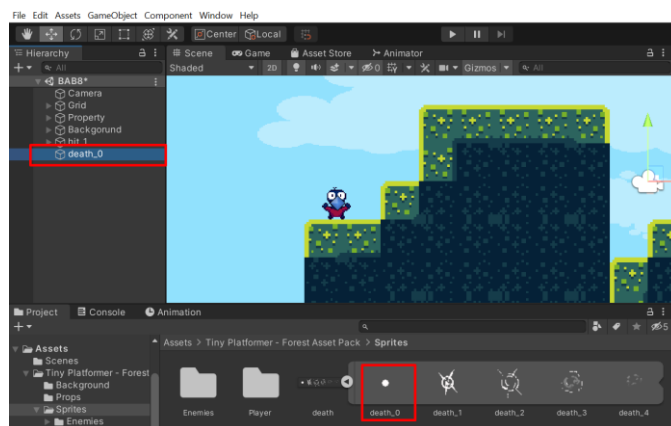
Gambar 10.3 Create Empty

4. Klik *firepoint* dan ubah icon menjadi seperti berikut.



Gambar 10.4 Ubah Icon

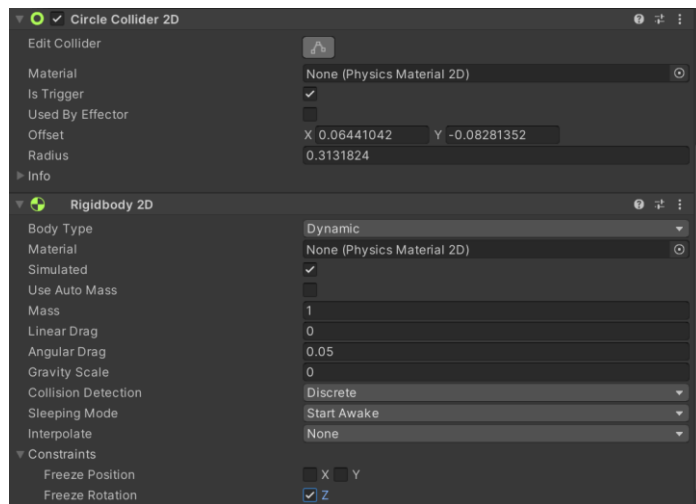
5. Dan tambahkan *asset* berikut pada *hirarky* lalu ubah nama menjadi *fireball*.



Gambar 10.5 Tambahkan Asset Tembakan

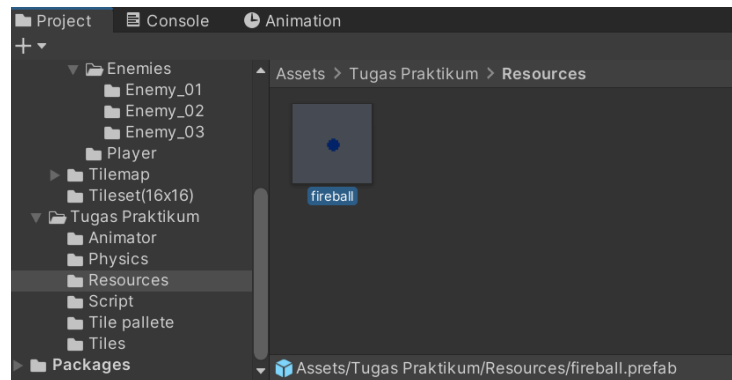


6. Dan tambahkan komponen dan atur komponen seperti berikut pada *fireball*.



Gambar 10.6 Setting Fireball

7. Buat *folder* baru dengan nama *Resources* dan masukan *fireball* pada *hirarky* ke dalam *folder*.



Gambar 10.7 Buat Folder Baru

8. Pada *script player* tambahkan variabel berikut.

```
public GameObject bullet;  
public Transform firePoint;
```

9. Dan tambahkan perintah berikut dibawah fungsi *fixedUpdate*.

```
IEnumerator Attack()  
{  
  
    animator.SetTrigger("Attack");  
    yield return new WaitForSeconds(0.25f);  
  
    float direction = 1f;  
  
    GameObject fireball = Instantiate(bullet,  
    firePoint.position, Quaternion.identity);
```

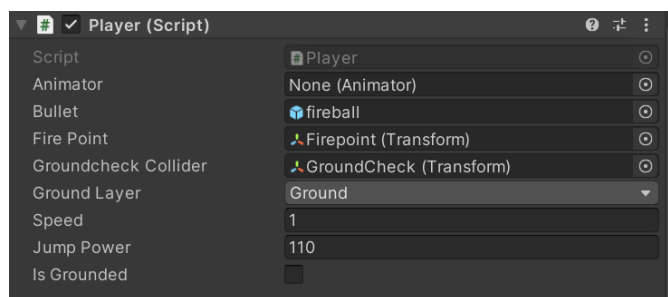


```
fireball.GetComponent<Rigidbody2D>().velocity =  
new Vector2(direction * 10f, 0);  
  
Destroy(fireball, 2f);  
}
```

10. Tambahkan kondisi berikut pada fungsi *Update*.

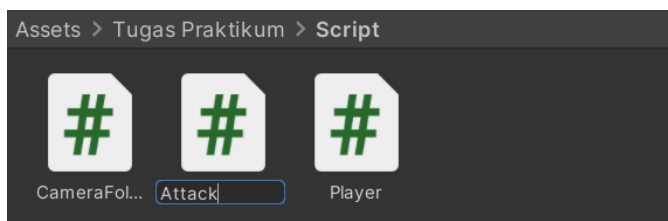
```
if (Input.GetKeyDown(KeyCode.C))  
{  
    StartCoroutine(Attack());  
}
```

11. Kemudian klik karakter dan atur *script Player* pada *inspector* menjadi seperti berikut.



Gambar 10.8 Setting Player Script

12. Buat *script* dengan nama *Attack* pada *folder Script*.



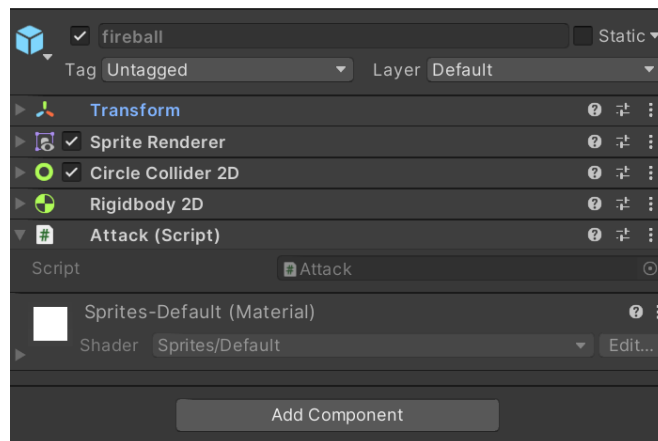
Gambar 10.9 Buat Script

13. Dan beri *source code* berikut.

```
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);  
        }  
    }  
}
```

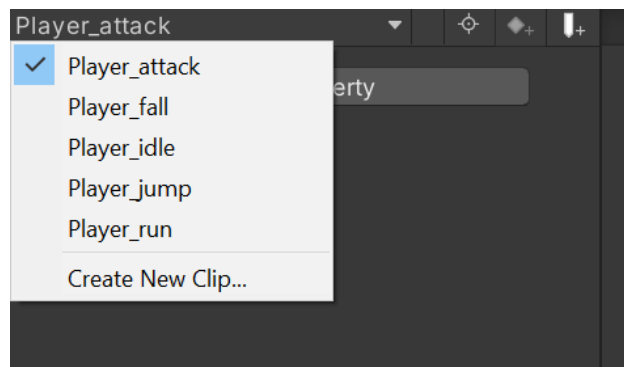


14. Setelah itu klik *fireball* pada *folder resources* dan tambahkan *script attack* pada *inspector*.



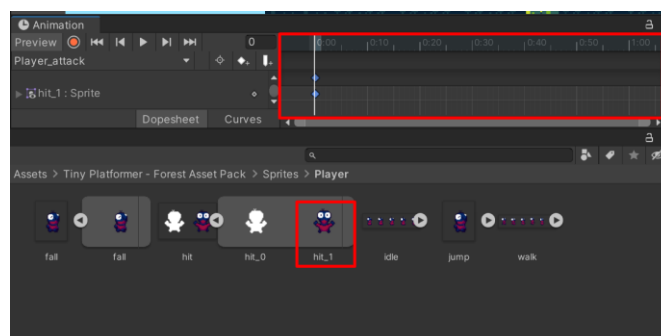
Gambar 10.10 Add Component

15. Kemudian buat *animation* baru dengan cara klik karakter dan pilih *Create New Clip* dan buat *Player_attack* untuk memberi animasi saat menyerang.



Gambar 10.11 Create Player Attack

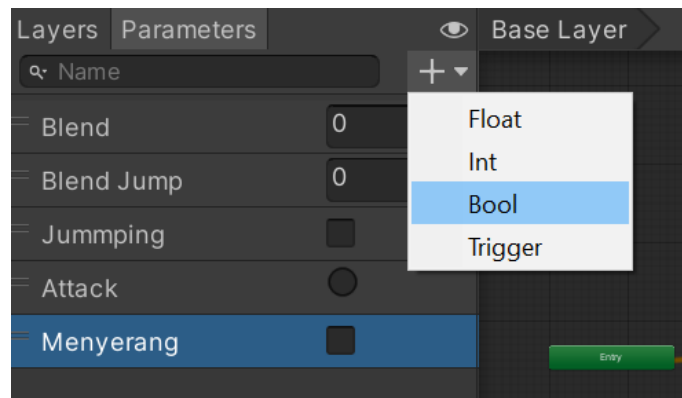
16. Dan tambahkan *asset* berikut pada *timeline* untuk memberi animasi menyerang.



Gambar 10.12 Asset Timeline

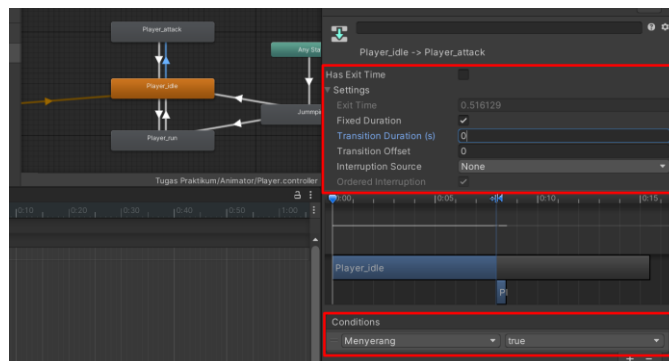


17. Buat parameter baru pada *tab animator* beri nama Menyerang dengan tipe data *Bool*.



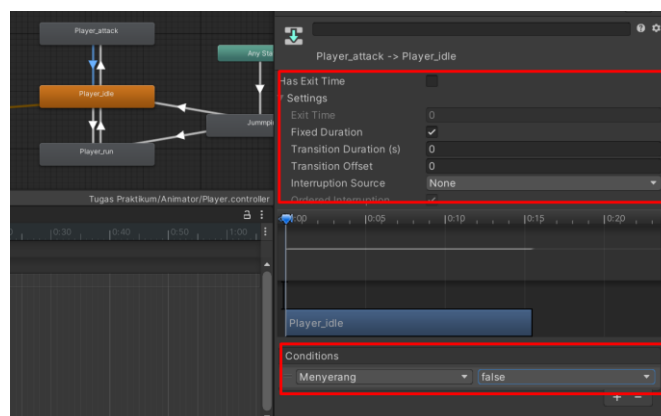
Gambar 10.13 New Parameter

18. Buat transisi pada *player_idle* ke *player_attack* dan klik arah panah beri kondisi Menyerang dengan kondisi *true* dan atur juga *setting* nya.



Gambar 10.14 Transisi Player Idle

19. Buat transisi lagi pada *player_attack* ke *player_idle* dan beri kondisi Menyerang dengan kondisi *false*.



Gambar 10.15 Transisi Player Attack



20. Pada *script Player* tambahkan variabel berikut.

```
bool menyerang;
```

21. Kemudian ubah kondisi berikut pada *void update*.

```
if (Input.GetKeyDown(KeyCode.C) && !menyerang)
{
    StartCoroutine(Attack());
}
```

22. Ubah fungsi *Attack* seperti berikut.

```
IEnumerator Attack()
{
    animator.SetTrigger("Attack");
    menyerang = true;
    animator.SetBool("Menyerang", 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("Menyerang", false);
    menyerang = false;

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

23. Dan ubah juga fungsi *Move*.

```
void Move(float dir, bool jumpflag, bool serangFlag)
{
    float xVal = dir * speed * 100 * Time.fixedDeltaTime;
    Vector2 targetVelocity = new Vector2(xVal,
    rb.velocity.y);
    rb.velocity = targetVelocity;
    menyerang = serangFlag;

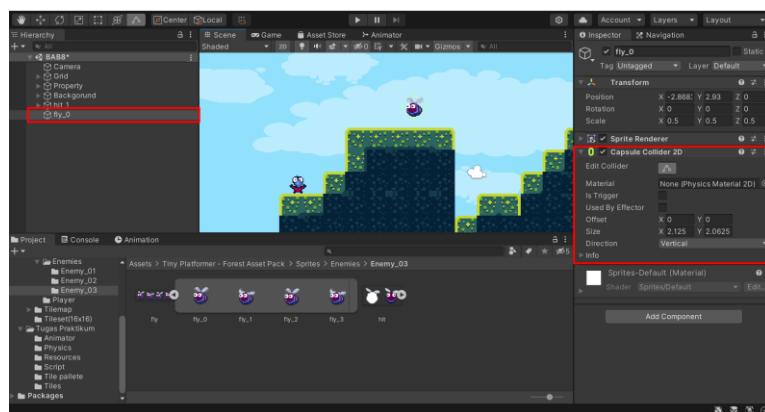
    if(isGrounded && serangFlag)
    {
        serangFlag = true;
    }

    if(isGrounded && jumpflag)
    {
        isGrounded = false;
        jumpflag = false;
        rb.AddForce(new Vector2(0f, jumpPower));
    }
    if (facingRight && dir < 0)
    {
        // ukuran player
    }
}
```



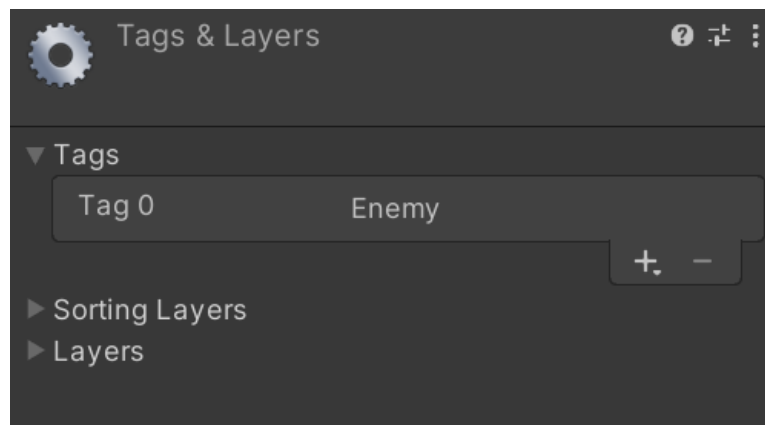
```
transform.localScale = new Vector3(-0.5f, 0.5f, 0.5f);  
facingRight = false;  
}  
  
else if (!facingRight && dir > 0)  
{  
    // ukuran player  
    transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);  
    facingRight = true;  
}  
}
```

24. Kemudian tambahkan *enemy1* dan beri komponen *Capsule Collider 2D*.



Gambar 10.16 Menambahkan Enemy

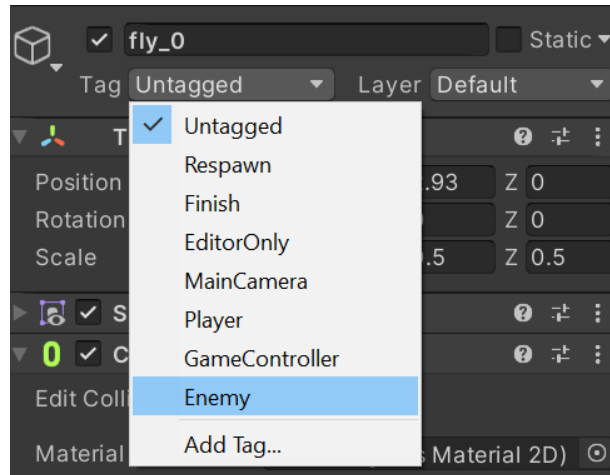
25. Tambahkan *Tags* baru dengan nama *Enemy*.



Gambar 10.17 Create Tags

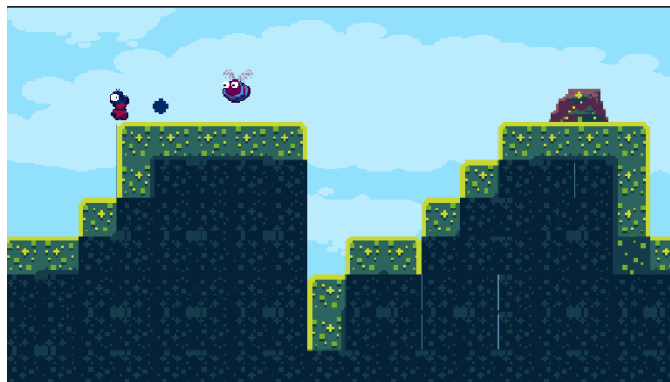


26. Dan ubah *tags* pada *Enemy1* menjadi *Enemy*.



Gambar 10.18 Ubah Tag

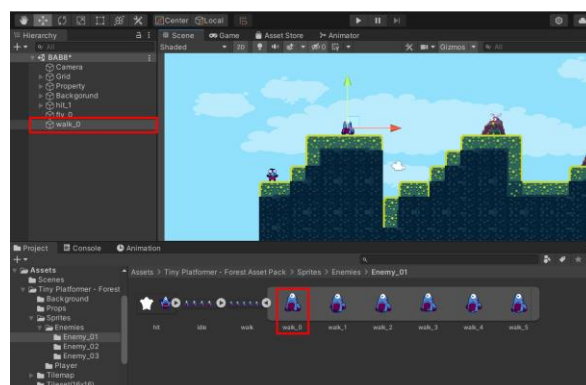
27. Jika di *run* karakter bisa menyerang dan memiliki animasi menyerang dan jika *fireball* mengenai *Enemy1* maka *Enemy1* akan hilang.



Gambar 10.19 Hasil Mekanisme Attack

B. Enemy Behavior NPC

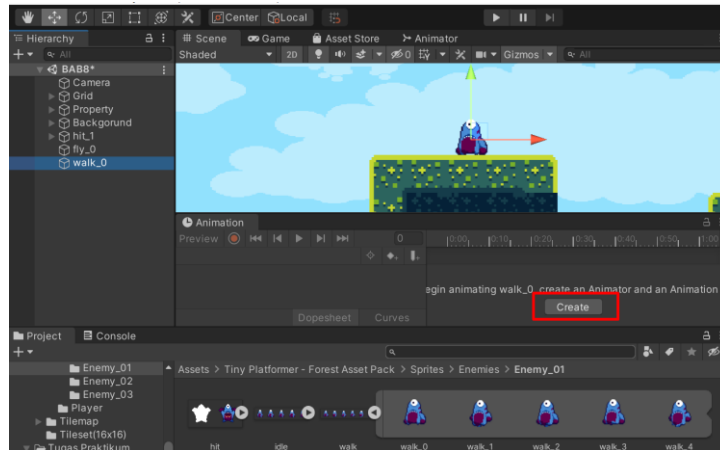
1. Tambahkan *Enemy2* pada *hirarky*.



Gambar 10.20 Enemy 2

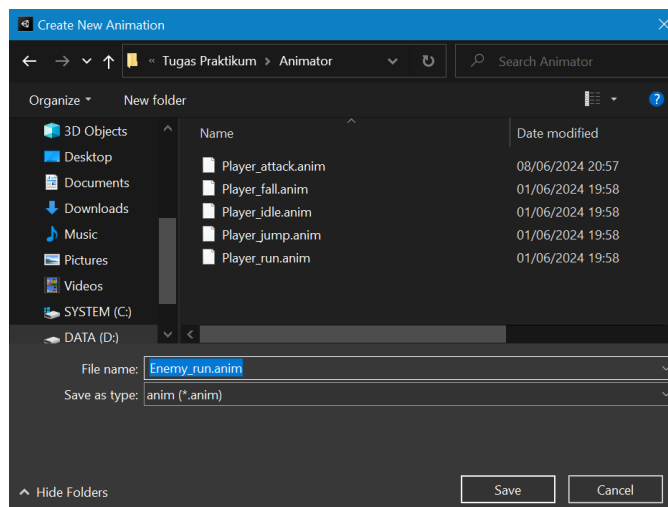


2. Buat animasi dengan cara klik *Create* pada *Animation*.



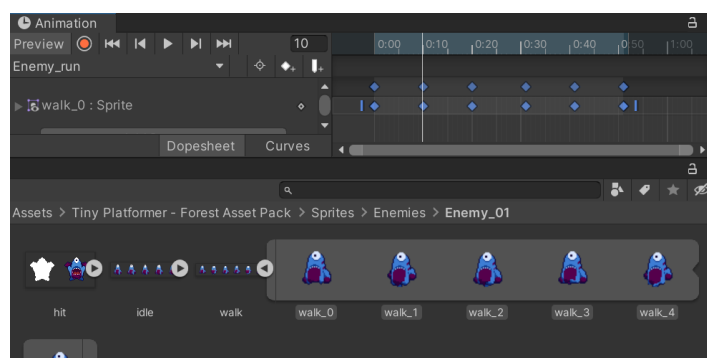
Gambar 10.21 Create Animation

3. Dan beri nama *Enemy_run* simpan pada *foder animator*.



Gambar 10.22 Save Enemy Run

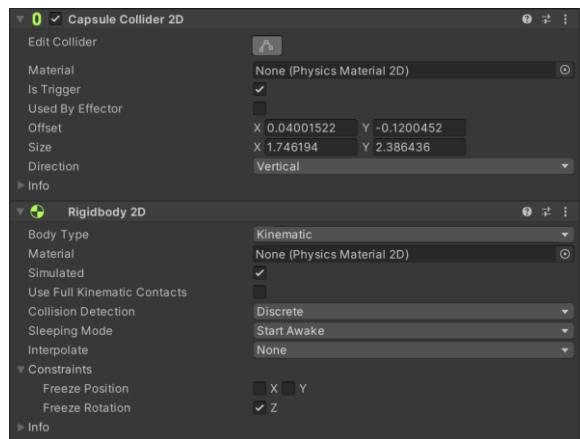
4. Tambahkan *asset* berikut pada *timeline* untuk membuat animasi *Enemy2* berjalan.



Gambar 10.23 Tambahkan Asset Timeline

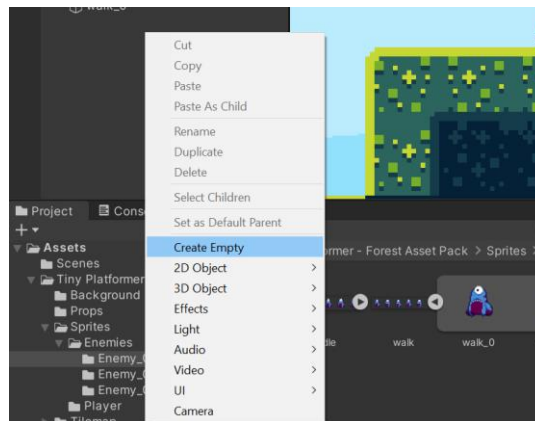


5. Dan beri komponen berikut pada *Enemy2* dan atur juga komponen menjadi seperti berikut.



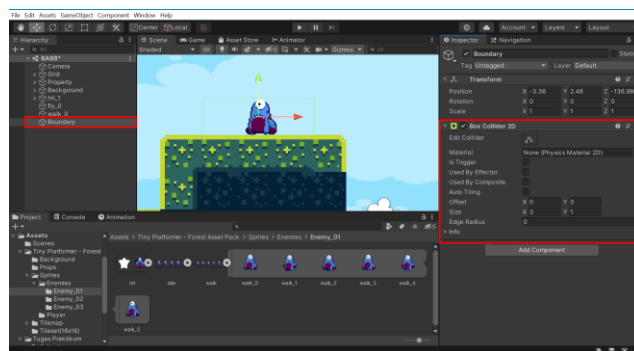
Gambar 10.24 Add Component

6. Klik kanan pada *hirarky* dan pilih *Create Empty* dan beri nama *Boundary*.



Gambar 10.25 Create Empty

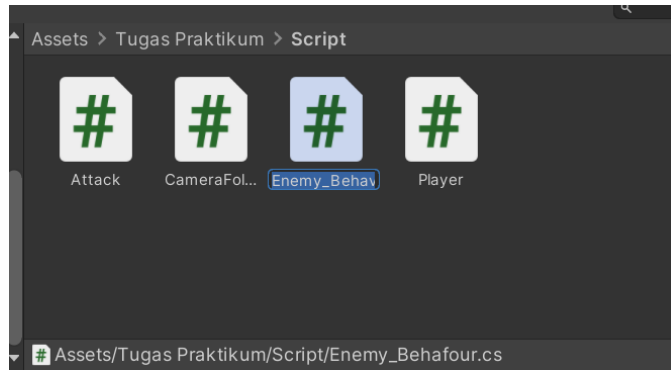
7. Pada *boundary* tambahkan komponen *Box Collider 2D* dan atur seperti berikut.



Gambar 10.26 Box Collider



8. Buat *file script* dengan nama *Enemy_Behavior*.



Gambar 10.27 Script *Enemy_Behavior*

9. 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>();
    }

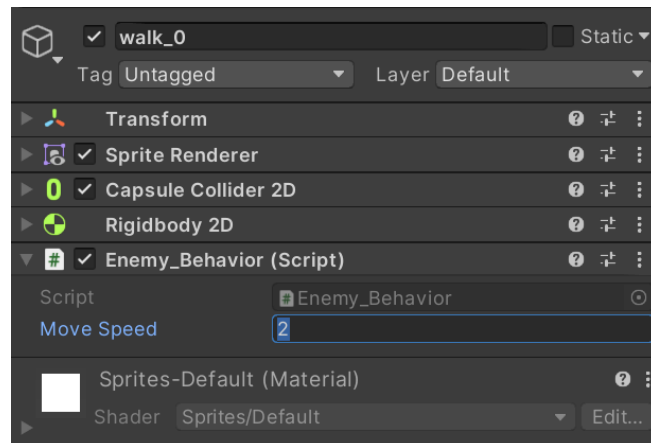
    void Update()
    {
        if (isFacingRight())
        {
            rb.velocity = new Vector2(-moveSpeed, 0f);
        }
        else
        {
            rb.velocity = new Vector2(moveSpeed, 0f);
        }
    }

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

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

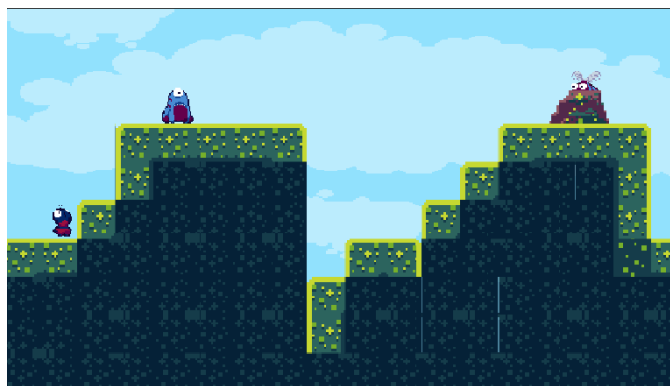


10. Dan tambahkan komponen *Enemy_Behavior* pada *Enemy2*.



Gambar 10.28 Add Component

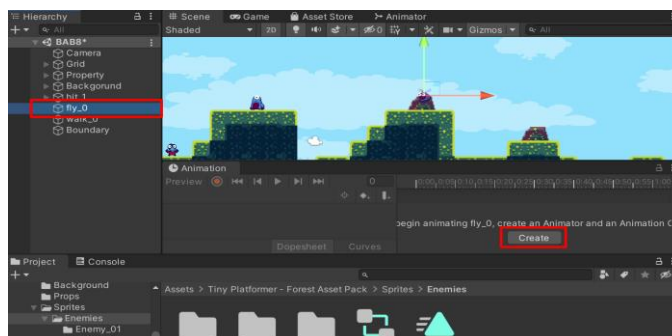
11. Jika di *run* maka *Enemy2* akan berjalan ke kiri dan ke kanan serta memiliki animasi berjalan.



Gambar 10.29 Hasil Enemy Behavior NPC

C. Enemy AI

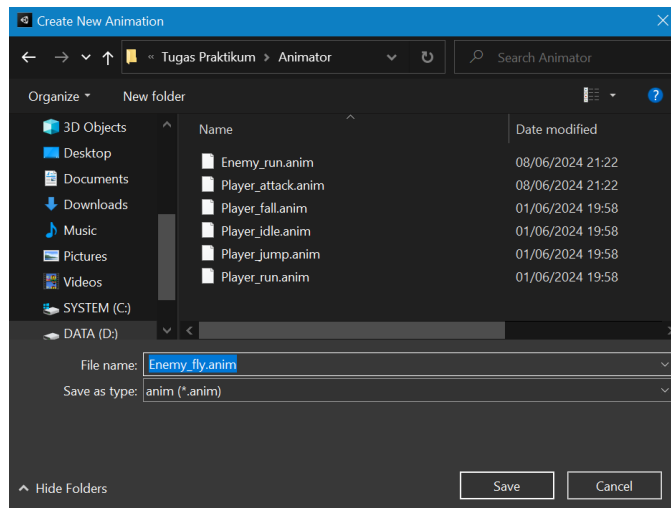
1. Klik *Enemy1* pada *hirarky* dan tambahkan animasi dengan cara pada *menu Animation* pilih *Create*.



Gambar 10.30 Create Animation

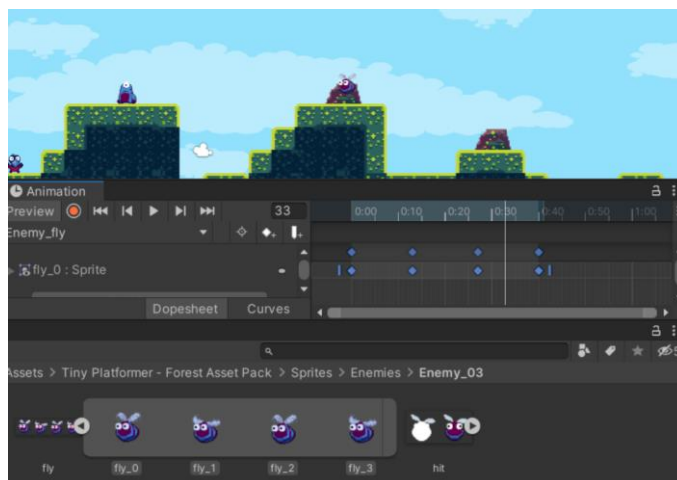


2. Dan beri nama *Enemy_fly* simpan pada *folder Animator*.



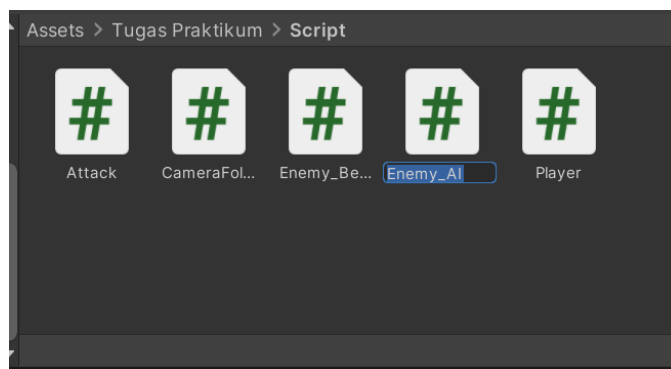
Gambar 10.31 Save Enemy Fly

3. Tambahkan *asset* berikut pada *timeline*.



Gambar 10.32 Tambahkan Asset Pad Timeline

4. Buat *file script* dengan nama *Enemy_AI*.



Gambar 10.33 Script Enemy AI



5. Tambahkan *source code* berikut pada *Enemy_AI*.

```
using System.Collections;
using 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();
    }
}

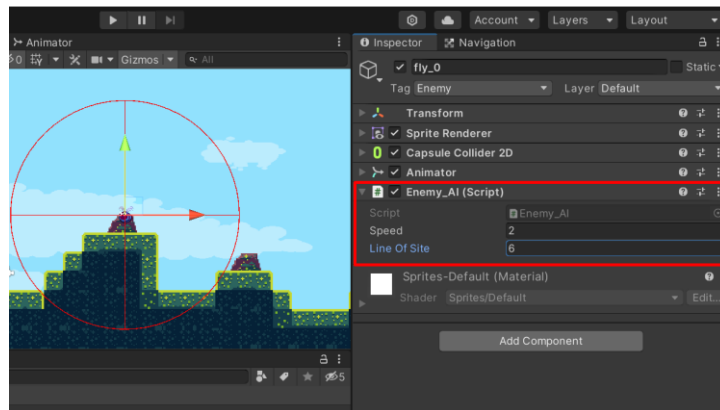
// Menghadapkan musuh ke arah awal
void FlipTowardsInitialPosition()
{
    if (initialPosition.x > transform.position.x &&
    !facingRight)
    {
        Flip();
    }
    else if (initialPosition.x <
    transform.position.x && facingRight)
    {
        Flip();
    }
}

// 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);
}
}
```

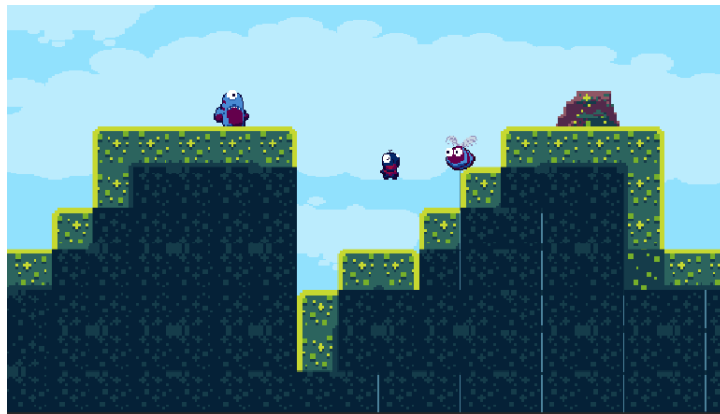



6. Klik *Enemy1* dan pada *inspector* tambahkan komponen *Enemy_AI* dan atur *Speed* juga *Line of Site*.



Gambar 10.34 Setting Inspector

7. Jika di *run* maka *Enemy1* akan mengikuti gerakan *player* dan juga memiliki animasi terbang.



Gambar 10.35 Hasil Enemy AI

D. Respawn

1. Pada *file script Player* tambahkan variabel berikut.

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

2. Dan tambahkan perintah berikut pada fungsi *Awake*.

```
respawn_loc = transform.position;
```

3. Buat fungsi *playagain*.

```
void playagain()  
{  
    if(play_again == true)  
    {  
        nyawa = 3;  
        transform.position = respawn_loc;  
    }  
}
```

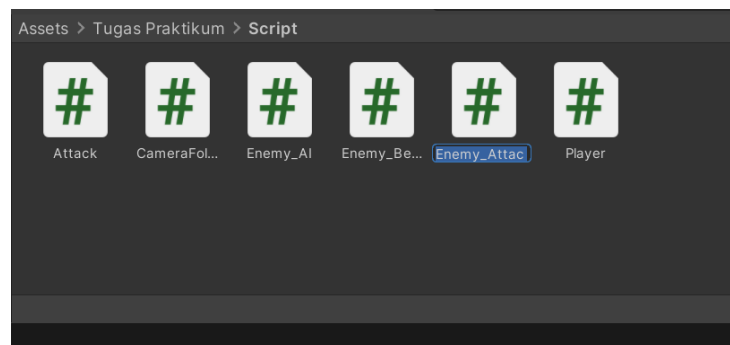


```
        play_again = false;
    }
}
```

4. Tambahkan kondisi berikut pada *void Update*.

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

5. Dan juga buat *file Enemy_Attacked*.



Gambar 10.36 Create Script

6. Beri *source code* berikut.

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

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

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

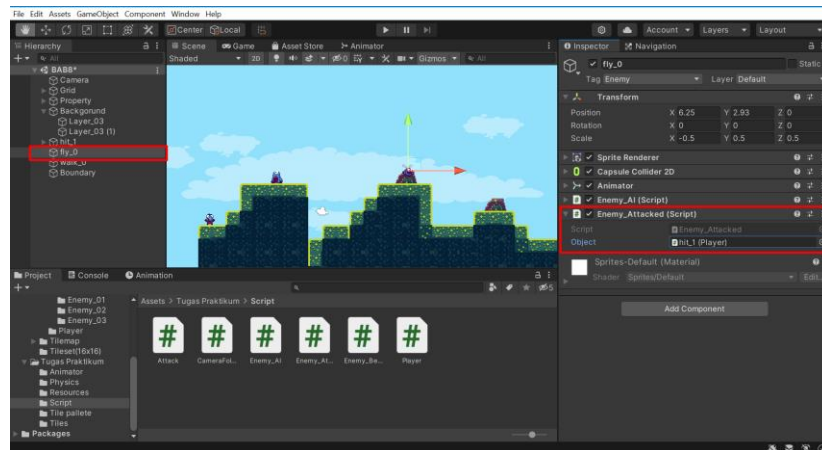
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            Object.nyawa--;

            if (Object.nyawa < 0)
```



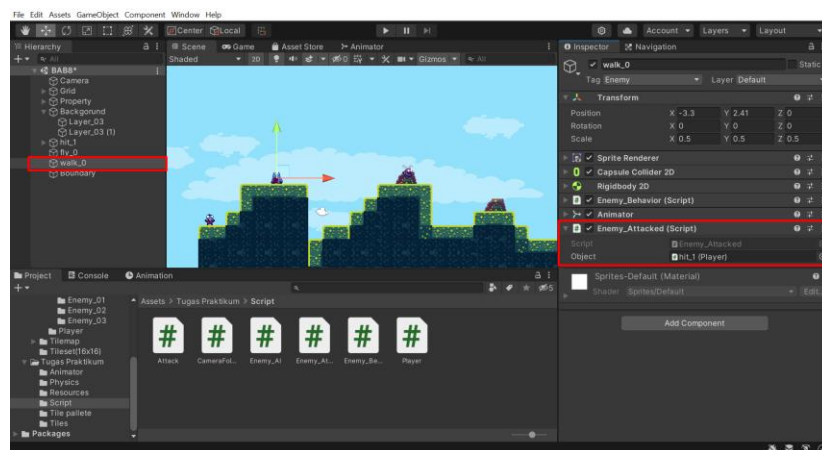
```
{  
    {  
        Object.play_again = true;  
    }  
}  
}
```

7. Lalu pada *Enemy1* tambahkan komponen *Enemy_Attacked* dan atur *Object* menjadi *player*.



Gambar 10.37 Setting Inspector

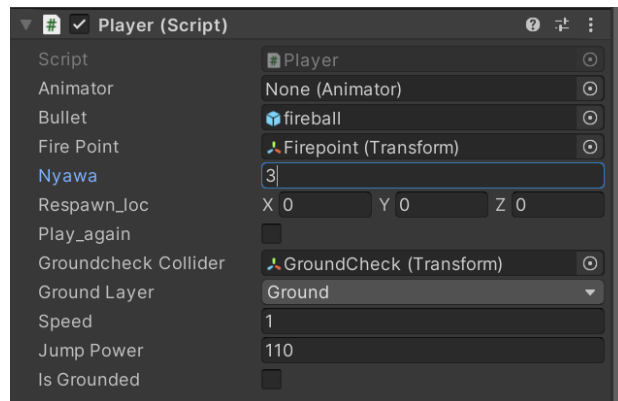
8. Tambahkan juga pada *Enemy2* dan atur *Object* menjadi *player*.



Gambar 10.38 Create Enemy 2

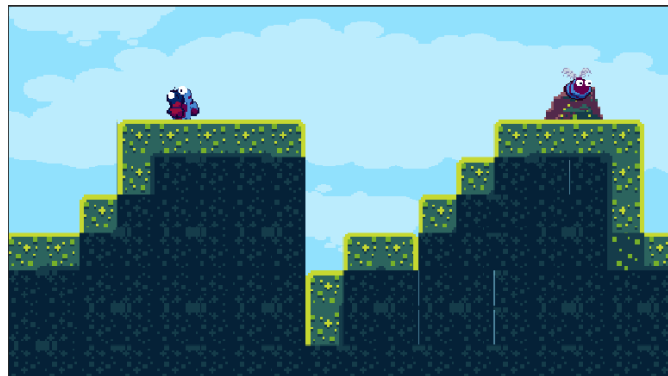


9. Klik karakter dan beri *value* nyawa pada *script player*.



Gambar 10.39 Setting Player Script

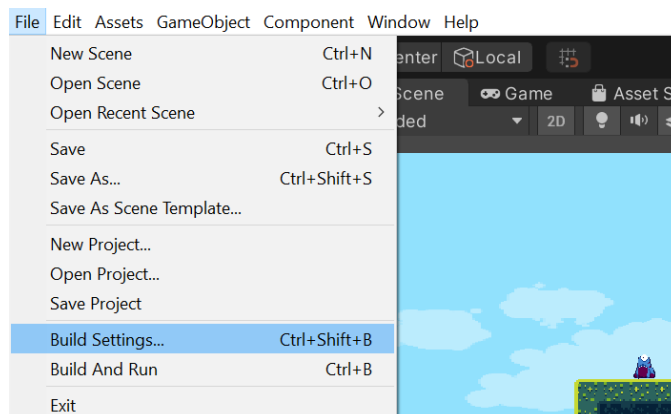
10. Jika di *run* ketika karakter menyentuh *Enemy* nyawa akan berkurang, jika nyawa 0 maka kembali ke posisi awal.



Gambar 10.40 Hasil Enemy AI

E. Render

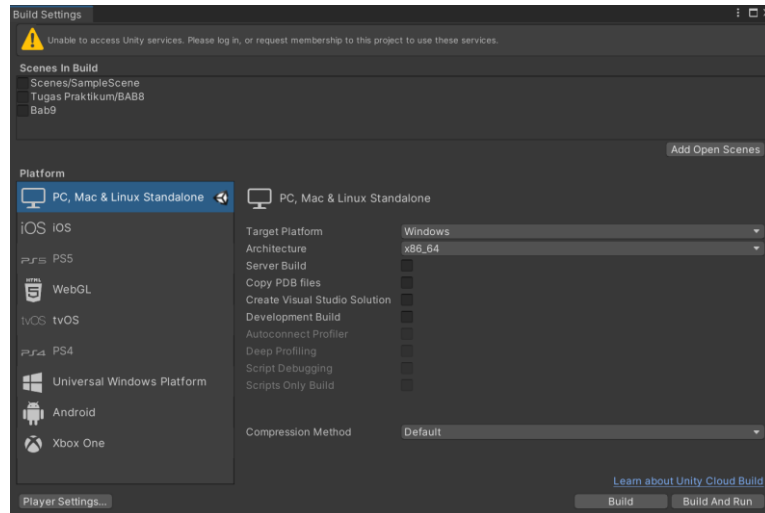
1. Klik *menu file* dan pilih *Build Settings*.



Gambar 10.41 Build Setting

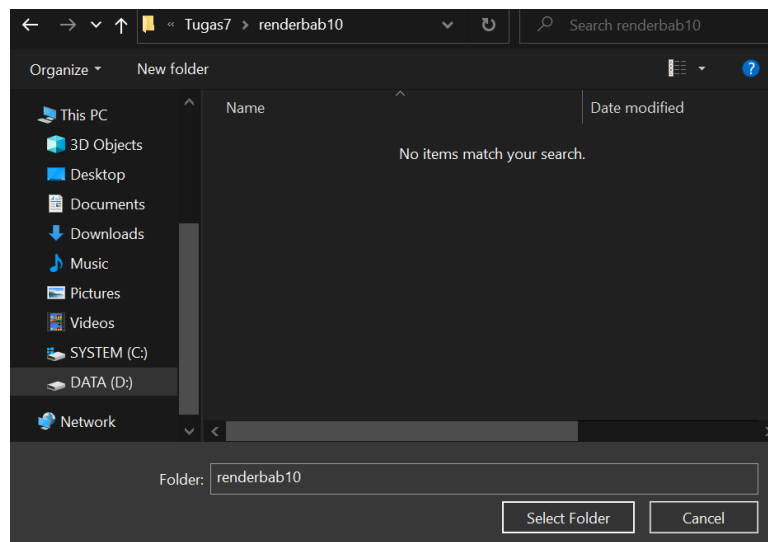


2. Dan atur menjadi seperti berikut lalu klik *Build*.



Gambar 10.42 Setting Build

3. Pilih *folder* untuk menyimpan dan klik *Select Folder*.



Gambar 10.43 Folder Penyimpanan

F. Link Github

https://github.com/arikarisma15/2118059_PRAK_ANIMASIGAME.git



KUIS

```
using UnityEngine;

public class PlayerAttack : MonoBehaviour
{
    public float attackRange = 2.0f; // Mengubah tipe data menjadi float untuk mendukung desimal
    public int attackDamage = 10; // Memperbaiki typo dari attacDamage menjadi attackDamage

    void Update()
    {
        if (Input.GetButtonDown("Fire1")) // Memperbaiki typo dari InputGetButtonDown menjadi Input.GetButtonDown
        {
            PerformMeleeAttack();
        }
    }

    void PerformMeleeAttack()
    {
        RaycastHit hit;
        if (Physics.Raycast(transform.position, transform.forward, out hit, attackRange))
        {
            // Mengambil komponen EnemyHealth dari objek yang terkena
            EnemyHealth enemyHealth = hit.collider.GetComponent<EnemyHealth>();

            if (enemyHealth != null)
            {
                // Mengurangi health musuh
                enemyHealth.TakeDamage(attackDamage);
            }
        }
    }
}
```

Analisa :

Pada kelas PlayerAttack, variabel attackRange diubah menjadi tipe float dan typo pada attackDamage diperbaiki. Fungsi PerformMeleeAttack menggunakan RaycastHit untuk memeriksa apakah serangan mengenai sesuatu dalam jangkauan. Jika mengenai sesuatu, dicek apakah objek tersebut memiliki komponen EnemyHealth. Jika memiliki komponen EnemyHealth, fungsi TakeDamage dipanggil untuk mengurangi kesehatan musuh. Pada kelas EnemyHealth, fungsi TakeDamage mengurangi nilai currentHealth dan memeriksa apakah kesehatan sudah habis untuk menghancurkan objek.