TUGAS PERTEMUAN: 10

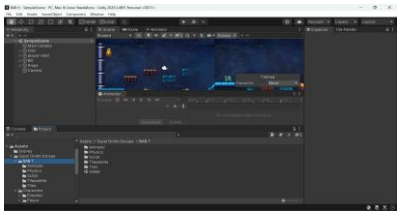
ENEMY AI & ATTACK

NIM	:	2118023
Nama	:	Ckristina Candra Dewi
Kelas	:	D
Asisten Lab	:	Natasya Octavia (2118034)

10.1 Tugas 1 : Membuat Enemy Ai & Attack

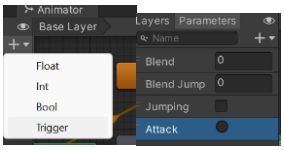
A. Membuat Mekanisme Attack

1. Buka project bab 9



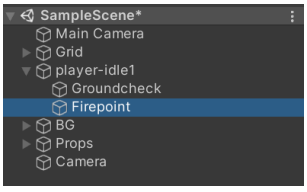
Gambar 10.1Tampilan Bab 9

2. Pada tab animator, tambahkan parameter dengan tipe data *Trigger* dan beri nama dengan



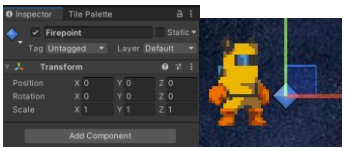
Gambar 10.2Tampilan Parameter Baru

3. Buat *Layer Game object* baru didalam *player-idle*-1, Klik kanan pilih *Create Empty* lalu *Rename* menjadi *Firepoint*.



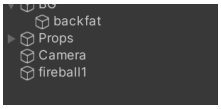
Gambar 10.3 Tampilan *Layer Game object*

4. Pada inspector dari Firepoint, ubah icon menjadi titik dan letakkan di depan p*layer*



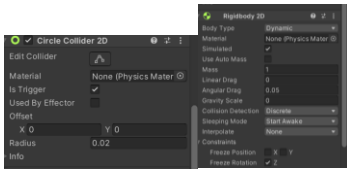
Gambar 10.4Tampilan *Firepoint*

5. Tambahkan item fireball1 yang berasal dari folder fx lalu fireball ke *hierarchy*.



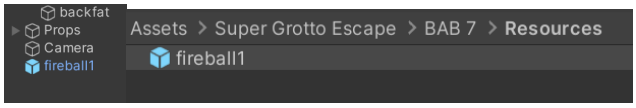
Gambar 10.5Tampilan *Hiearchy*

6. Tambahkan *Component* Circle Collider 2d dan Riggidbody 2D pada fireball1, lalu sesuaikan *setting* seperti pada gambar.



Gambar 10.6Tampilan *Component* Fireball1

7. Buat folder baru dengan nama *Resources*, lalu drag and drop fireball1 kedalam folder Res*ource*, dan hapus fireball pada *Hierarchy*



Gambar 10.7Tampilan *Resources*

8. Pada *script* p*layer* tambahkan *source code* berikut pada public class

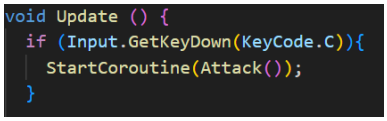
```
public class Player : MonoBehaviour
{
  public Animator animator;
  public GameObject bullet;
  public Transform firepoint;
```

Gambar 10.8 Tampilan *Source Code*

9. Tambahkan source code berikut dibawah fungsi fixedUpdate

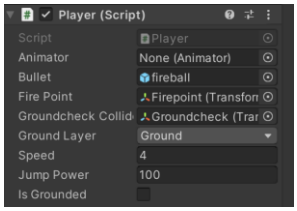
Gambar 10.9Tampilan *Source Code*

10. Tambahkan *source code* berikut pada fungi Update



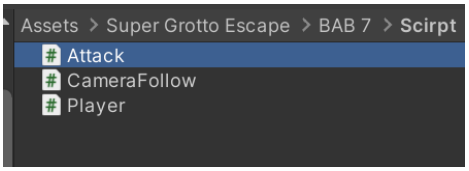
Gambar 10.10 Tampilan *Source Code*

11. Klik p*layer*-idle1 lalu ubah bullet menjadi fireball dan Fire Point menjadi firepoint.



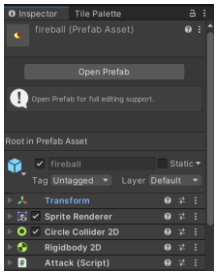
Gambar 10.11 Tampilan P*layer Script*

12. Buat *script* baru dengan nama *Attack*



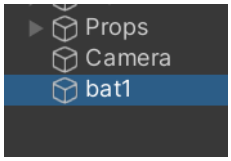
Gambar 10.12 Tampilan Membuat *Script Attack*

13. Tambahkan script Attack ke fireball



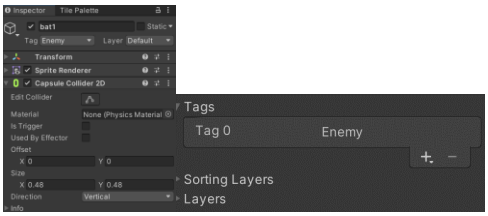
Gambar 10.13 Tampilan *Fireball*

14. Tambahkan *Enemy* bat 1 dari folder *enemy* ke *hierarchy*.



Gambar 10.14 Tampilan Menambahkan *Enemy*

15. Tambahkan Component capsule collider 2D pada Enemy bat1, lalu tambahkan tag Enemy



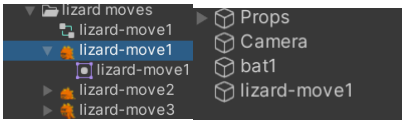
Gambar 10.15 Tampilan Inspector *Enemy*

16. Jalankan *Game* untuk melihat hasilnya

Gambar 10.16 Tampilan Hasil

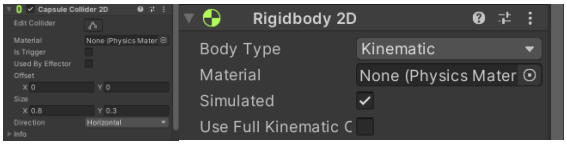
B. Enemy AI

1. *Enemy* Behavior NPC
   a. Cari folder *Character*s lalu masuk ke folder *Enemy*, setelah itu pilih folder lizard-moves dan tambahkan lizard-move1 ke *hierarchy*.



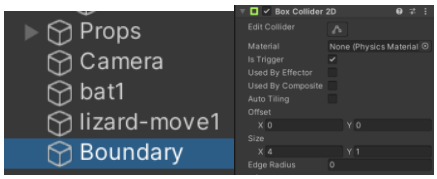
Gambar 10.17 Tampilan Menambahkan *Enemy*

   b. Atur seperti pada gambar



Gambar 10.18 Tampilan *Component*

c. *Create emphty Game* pada *hierarchy* lalu ubah namanya menjadi Boundary. Setelah itu tambahkan Box Collider 2d.



Gambar 10.19 Tampilan Boundary

d. Tammbahkan *Enemy_Behavior* pada folder *script*, lalu tambahkan *source code* berikut dan tambahkan ke lizard-move1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ENEMY_Behavior : MonoBehaviour{
    [SerializeField] float moveSpeed = 1f;
    Rigidbody2D rb; void Start(){
rb = GetComponent<Rigidbody2Dlalu();   }
    void Update(){  if (isFacingRight()){
 rb.velocity   =   new   Vector2(moveSpeed,   0f);
}else{
  rb.velocity = new Vector2(-moveSpeed, 0f);} }
    private bool isFacingRight() {
return transform.localScale.x lalu Mathf.Epsilon;
}
    private    void    OnTriggerExit2D(Collider2D
collision){    transform.localScale = new
Vector2(-transform.localScale.x,
transform.localScale.y);   }}
```

e. Jalankan *Game* untukmelihat hasilnya

Gambar 10.20 Tampilan Hasil

2. *Enemy* AI

a. Tambahkan *Enemy*_AI pada folder *script*, lalu tambahkan *source code* berikut.

```
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;
void     Start()               {     player     =
GameObject.FindGameObjectWithTag("Player").transf
orm;
initialPosition
=GetComponent<Transformlalu().position;     }
void Update()     {float distanceToPlayer =
Vector2.Distance(player.position,
ransform.position); // Jika pemain berada dalam
jarak penglihatan musuh
if (distanceToPlayer < lineOfSite)       {
        transform.position =

Vector2.MoveTowards(this.transform.position,
player.position, speed * Time.deltaTime); }else
{// Musuh kembali ke posisi awal
transform.position=
Vector2.MoveTowards(transform.position,
initialPosition, speed * Time.deltaTime); }     }
private void OnDrawGizmosSelected()     {
Gizmos.color             =             Color.red;
Gizmos.DrawWireSphere(transform.position,
lineOfSite);    }}
```

b. Tambahkan *script Enemy*_AI ke bat1 lalu atur Speed juga Line of Site untuk menentukan jarak dan speed pada *Enemy*.

Gambar 10.21 Tampilan bat1

C. Respawn

1. Buka P*layer*,cs, lalu tambahkan *source code* berikut

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

Gambar 10.22 Tampilan *Source Code*

2. Tambahkan kode dibawah ini untuk mengatur posisi respawn sesuai dengan posisi awal permainan dimulai

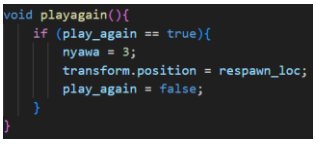
```
private void Awake()
{
    rb = GetComponent<Rigidbody2D>();
    animator = GetComponent<Animator>();

    respawn_loc = transform.position;
}
```

Gambar 10.23 Tampilan *Source Code*

3. Tambahkan kode dibawah ini di dalam void update *Player*.cs agar ketika nyawa *player* dibawah 0 maka akan melakukan respawn dan agar ketika p*layer* jatuh dibawah platform akan melakukan respawn

Gambar 10.24 Tampilan *Source Code*

4. Lalu, tambahkan void playagain



Gambar 10.25 Tampilan *Source Code*

5. Buat *Enemy_Attack*ed pada folder *script*, lalu tambahkan *source code* berikut dan tambahkan *source code* ke bat1

```
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
lalu();  } }
    void OnTriggerEnter2D(Collider2D other){
if (other.CompareTag("Player")){Object.nyawa--;   if
(Object.nyawa < 0){ Object.play_again = true;} } }}
```

6. Lalu, jalankan *Game* untuk melihat hasilnya

Gambar 10.26 Tamplan Hasil

D. KUIS Pertemuan 10

Lengkapi Source code dibawah ini :

```
using UnityEngine;
public class PlayerAttack : MonoBehaviour
{
    public int atackRange = 2.0f;
    public int attacDamage = 10;

    void Update()
    {
        if (InputGetButtonDown("Fire1"))
        {
            PerformMeleeAttack();
        }
    }

    void PerformMeleeAttack()
    {
        RaycastHit hit;
        if          (Physics.Raycast(transform.position,
transform.forward, out hit, attackRange))
        {
            // Lengkapi kode di sini untuk mengenai
musuh dan mengurangi health mereka
        }
    }
}
```

Source code yang sudah dilengkapi yang kurang

```
using UnityEngine;
public class PlayerAttack : MonoBehaviour{
    public float attackRange = 2.0f; // Ubah tipe data
ke float
    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))
        {
            // Mendapatkan komponen Health dari objek
yang terkena
            EnemyHealth          enemyHealth          =
hit.collider.GetComponent<EnemyHealth>();
            if (enemyHealth != null)
            {
                // Mengurangi health musuh
                enemyHealth.TakeDamage(attackDamage);
            }
```

```
        }
    }
}
// Skrip EnemyHealth untuk menangani health musuh
public class EnemyHealth : MonoBehaviour
{
    public int maxHealth = 100;
    private int currentHealth;
    void Start()
    {
        currentHealth = maxHealth;
    }
    public void TakeDamage(int damage)
    {
        currentHealth -= damage;
        Debug.Log("Enemy Health: " + currentHealth);

        if (currentHealth <= 0)
        {
            Die();
        }
    }
    void Die()
    {
        // Menangani kematian musuh
        Debug.Log("Enemy died!");
        Destroy(gameObject);
    }
}
```

Penjelasan :

pada source code diatas ditambahkan komponen Enemyhealth yang digunakan untuk mendapatkan komponen health dari objek yang terkena tembakan. Lalu, ditambahkan juga statement if untuk enemyhealth jika dia tidak null maka enemy akan mengurangi health musuh. Pada PlayerAttack menangani input pemain untuk serangan. Dalam metode Update, ketika tombol "Fire1" ditekan, metode PerformMeleeAttack dipanggil. Metode ini menggunakan Raycast untuk mendeteksi objek di depan pemain dalam jarak tertentu (attackRange). Jika raycast mengenai objek yang memiliki komponen EnemyHealth, maka metode TakeDamage pada komponen tersebut dipanggil untuk mengurangi kesehatan musuh sebesar attackDamage.

Berikan Tanda Merah yang menyebabkan Source code Eror

```
void HandleJumpInput()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            animator.SetBool("isJumping", );
            rb.AddForce(Vector2.up     *     jumpForce,
ForceMode2D.Impulse);
        }
        else if (Input.GetKey(KeyCode.Space))
        {
            animator.SetBool("isJumping",);
        }
    }

    void HandleMovementInput()
    {
        float move = Input.GetAxis("Horizontal");

        if (move != 1)
        {
            animator.SetBool("isIdle", true);
            transform.Translate(Vector3.left  *  move  *
Time.deltaTime);
        }
        else
        {
            animator.SetBool("isWalking", false);
        }

        if (move != 0)
        {
            transform.localScale  =  new  Vector3(-4,  1,
1);
        }
        else if (move > 0)
        {
            transform.localScale = new Vector3(1, 2, 1);
        }
    }
```

Source code yang sudah diperbaiki

```
void HandleJumpInput()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        animator.SetBool("isJumping",     true);     //
Diperbaiki: Menambahkan argumen boolean yang sesuai
        rb.AddForce(Vector2.up       *       jumpForce,
ForceMode2D.Impulse);
    }
    else if (Input.GetKey(KeyCode.Space))
    {
        animator.SetBool("isJumping",     true);     //
Diperbaiki: Menambahkan argumen boolean yang sesuai
    }
}
```

```
void HandleMovementInput()
{
    float move = Input.GetAxis("Horizontal");

    if (move != 0)
    {
        animator.SetBool("isIdle", false);
        animator.SetBool("isWalking", true);
        transform.Translate(Vector3.right * move *
Time.deltaTime);

        if (move < 0)
        {
            transform.localScale = new Vector3(-1, 1,
1); // Diperbaiki: Menggunakan skala yang sesuai
        }
        else if (move > 0)
        {
            transform.localScale = new Vector3(1, 1, 1);
// Diperbaiki: Menggunakan skala yang sesuai
        }
    }
    else
    {
        animator.SetBool("isIdle", true);
        animator.SetBool("isWalking", false);
    }
}
```

Analisa:

Kesalahan yang terdapat pada source code adalah pada animator.SetBool kurang kondisi true atau false. Karena memakai tipe data bool harusnya ditambakan kondisi true atau false. Contohnya seperti pada animator.SetBool(“isWalking”, false), script tersebut benar karena sudah menambahkan kondisi true atau false. Kesalahan selanjutnya adalah pada script transform.localScale yang mana nilanya harus konsisten, jika nilainya 1 maka semua nilainya juga harus 1. Setelah kode diperbaiki, jika ada gerakan (nilai move tidak nol), animasi idle dimatikan (isIdle diatur ke false) dan animasi berjalan dihidupkan (isWalking diatur ke true), lalu pemain bergerak ke kanan atau kiri berdasarkan nilai input. Skala lokal dari pemain juga diubah untuk mencerminkan arah gerakan: negatif untuk kiri dan positif untuk kanan. Jika tidak ada gerakan, animasi idle dihidupkan dan animasi berjalan dimatikan. Fungsi HandleJumpInput memeriksa apakah tombol spasi ditekan. Jika ya, animator akan mengatur parameter isJumping menjadi

true, dan kekuatan ditambahkan ke Rigidbody (rb) pemain untuk menyebabkan lompatan. Jika tombol spasi masih ditekan, animator tetap mengatur isJumping menjadi true.

E. Link Github Pengumpulan

https://github.com/kristinacandra/2118023_PRAK_ANIGAME.git