TUGAS PERTEMUAN: 9

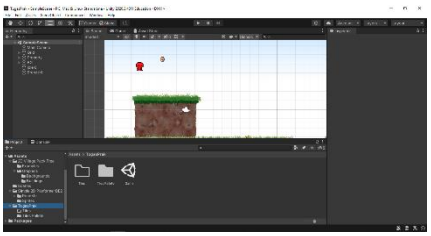
GAME ANIMATION

NIM	:	2119139
Nama	:	Farhan Maulana Ridho
Kelas	:	C
Asisten Lab	:	Bagas Anardi Surya W (2119004)

9.1 Tugas 9 : Mengimplementasikan Game Animation

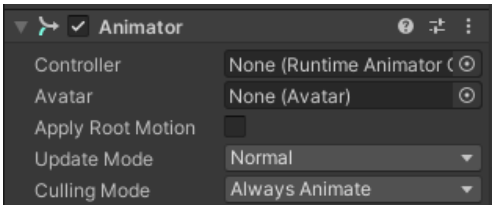
A. Membuat Character Animation

1. Pertama buka project Unity yang sudah dibuat.



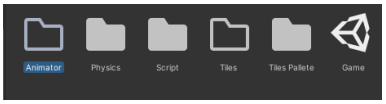
Gambar 9.1 Tampilan *Project Unity*

2. Pada hierachy karakter di inspector klik *add component* dan tambahkan *animator*.



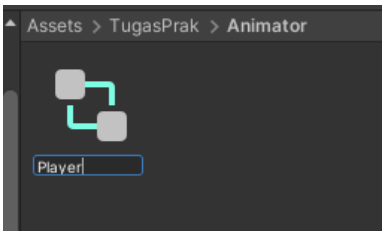
Gambar 9.2 Add Component Animator

3. Buat folder baru pada TugasPrak beri nama Animator.



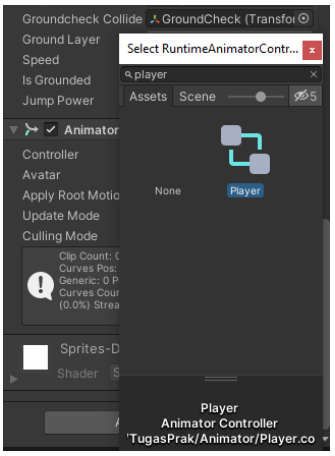
Gambar 9.3 *New Folder*

4. Di dalam folder *Animator* yang telah dibuat, buatlah *file Animator Controller* dengan cara klik kanan, kemudian pilih *Create* dan pilih *Animator Controller*. Beri nama *file* tersebut *Player*.



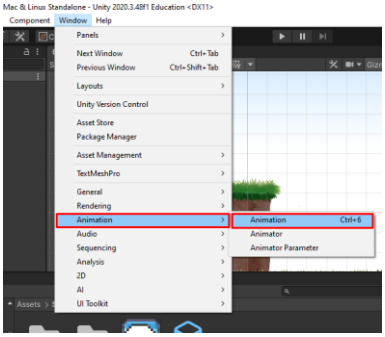
Gambar 9.4 *Animator Controller*

5. Pada hierarchy klik karakter, pada inspector cari komponen animator, ubah controller menjadi Player.



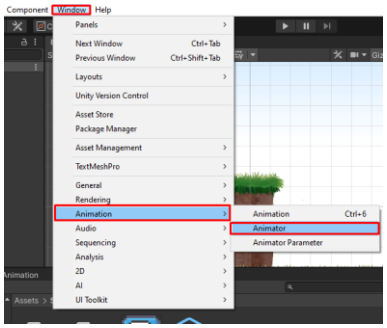
Gambar 9.5 *Component Animator*

6. Selanjutnya menambahkan menu panel Animation dengan klik menu *Windows>Animation>Animator* atau dengan menekan *Ctrl* + 6.



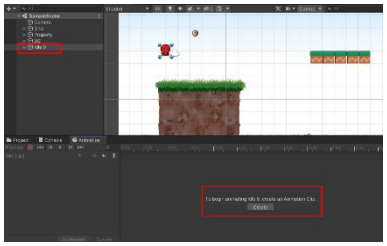
Gambar 9.6 Menu *Animation*

7. Selanjutnya tambahkan juga menu panel Animator pada menu *windows*.



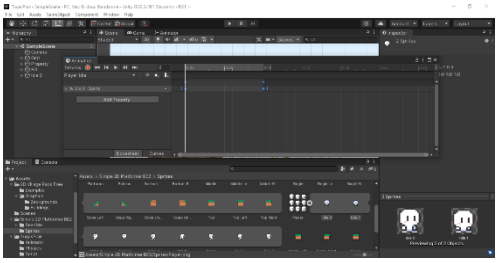
Gambar 9.7 Menu Animator

8. Selanjutnya pada hierarchy klik karakter, kemudian pada *menu panel animation* klik *Create*.



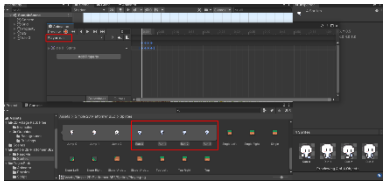
Gambar 9.8 *Create Animation Clip*

9. Simpan file tersebut di dalam folder Animation dan beri nama Player_idle. Setelah itu, di menu Project Assets, masukkan asset character_idle dengan cara drag and drop ke tab Animation.



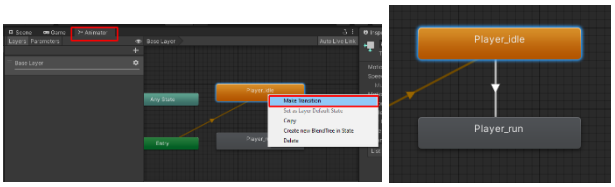
Gambar 9.9 *Animation*

10. Tambahkan animasi baru dan beri nama Player_run. Lakukan langkah yang sama seperti sebelumnya.



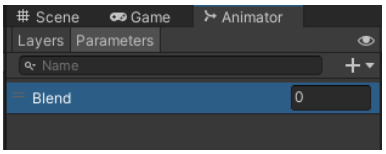
Gambar 9.10 *Animation*

11. Setelah menambahkan animasi, buka Panel Animator dan buat transisi antara Player_idle dan Player_run. Caranya adalah dengan klik kanan pada Player_idle, pilih Make Transition, lalu tarik ke Player_run.



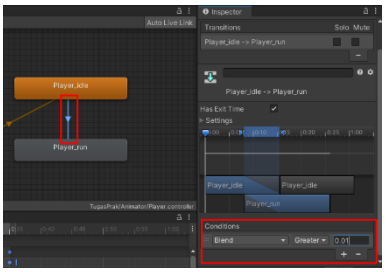
Gambar 9.11 *Transition* Player_idle dengan Player_run

12. Selanjunya tab Parameters, buat tipe data float, dan beri nama Blend.



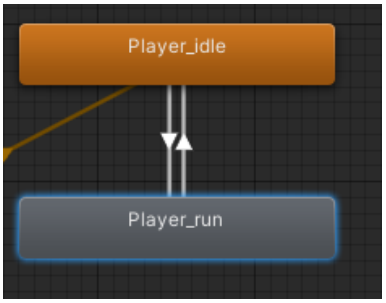
Gambar 9.12 *Parameters*

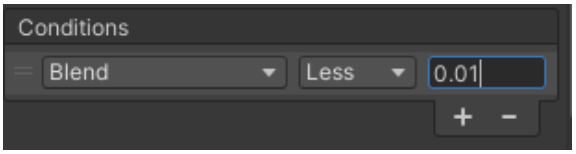
13. Setelah transisi dibuat, klik tanda panah putih. Pada bagian Conditions, klik ikon + dan atur menjadi Blend, lalu ubah angka 0 menjadi 0,01.



Gambar 9.13 *Conditions Blend*

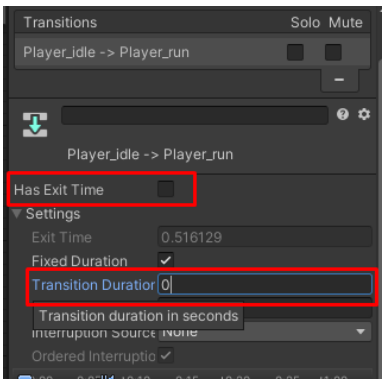
14. Lakukan langkah yang sama untuk transisi dari Player_run ke Player_idle. Pada komponen Conditions, tambahkan juga Blend, atur menjadi Less, dan ubah nilainya menjadi 0,01.





Gambar 9.14 Player_run transition to Player_idle

15. Selanjutnya pada setiap transisi pada seeting matikan HasExit Time, kemudian Transtion Duration ganti angka 0.



Gambar 9.15 *Settings*

16. Kemudian pada folder script player tambahkan juga source code yang sudah diberikan oleh modul.

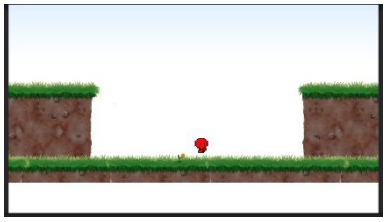
```
public class Player : MonoBehaviour
{
    public Animator animator;

    Rigidbody2D rb;
    [SerializeField] Transform groundcheckCollider;
    [SerializeField] LayerMask groundLayer;
```

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

```
void FixedUpdate()
{
    GroundCheck();
    Move(horizontalValue, jump);
    animator.SetFloat("Blend", Mathf.Abs(rb.velocity.x));
}
```

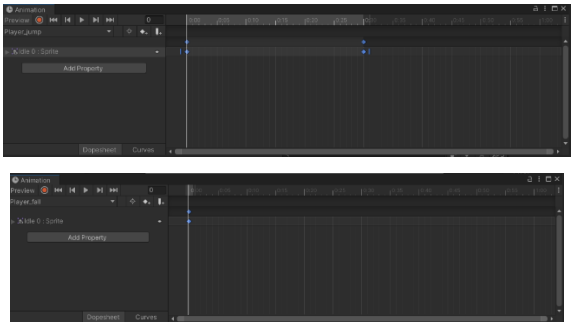
Gambar 9.16 *Update Script Players*

17. Jika sudah maka jalankan maka pastikan animasi sudah berjalan.



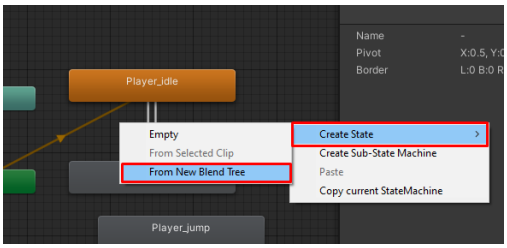
Gambar 9.17 *Animation run and idle*

18. Selanjutnya pada panel animation *create new clip* tambahkan Player_jump dan Player_fall.



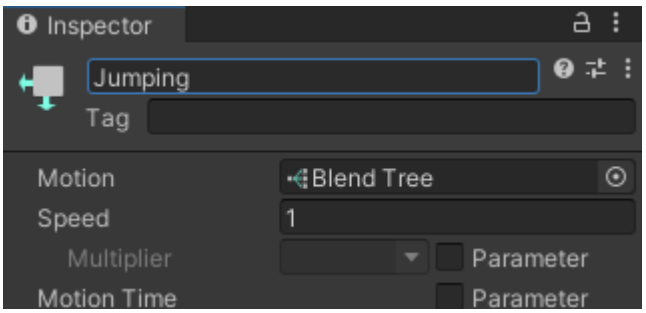
Gambar 9.18 *Animation* Player_jump

19. Kemudian untuk animasi melompat, pada area kosong klik kanan pilih *Create State>From New Blend Tree*.



Gambar 9.19 *From New Blend Tree*

20. Setelah membuat *Blend Tree* baru pada inspector beri nama "*Jumping*".

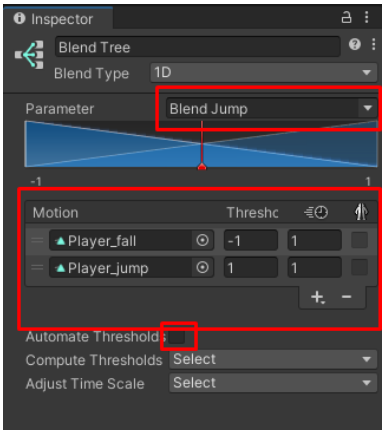


Gambar 9.20 *Blend Tree*

21. Di menu Parameters, tambahkan parameter dengan tipe data *float*, tekan ikon +, dan beri nama *Blend Jump*.

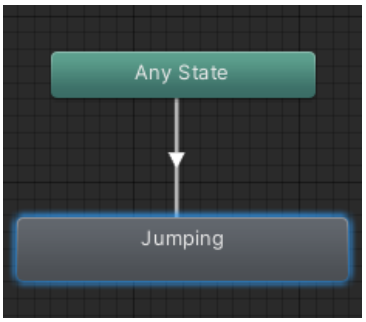
Gambar 9.21 *Blend Jump*

22. Klik dua kali pada *Blend Tree* "*Jumping*" di panel *Animator*, lalu tekan pada *Blend Tree*. Di *inspector*, ubah parameter menjadi *Blend Jump* dan atur *motion*-nya menjadi Player Fall dan Run. Di Automate Threshold dimatikan. Sesuaikan pengaturan threshold dengan gambar di bawah ini.



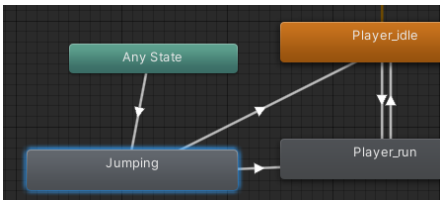
Gambar 9.22 *Inspector Blend Tree*

23. Setelah itu kembali ke base layer. Pada any state buat transisi dengan klik kanan make transion dan tarik ke Jumping.



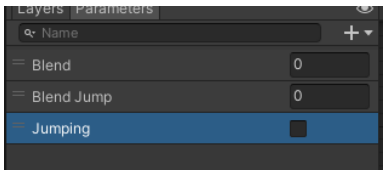
Gambar 9.23 *Base Layer*

24. Pada jumping buat transisi dan tarik ke Player_idle dan Player_run.



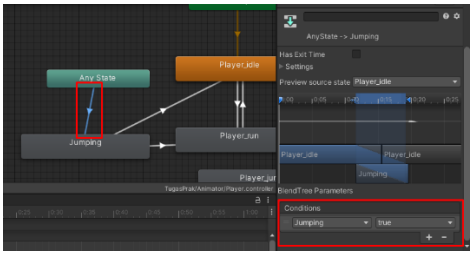
Gambar 9.24 *Base Layer*

25. Selanjutnya menambahkan parameter dengan tipe data bool dan beri nama Jumping.



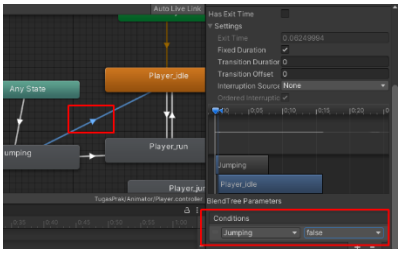
Gambar 9.25 *Parameters Jumping*

26. Pada Any State, klik panah yang mengarah ke Jumping. Di inspector, tambahkan kondisi Jumping dan ubah nilainya menjadi true.



Gambar 9.26 *Jumping Condition*

27. Pada *Jumping* klik panah yang menuju ke Player_idle dan Player_run, lalu di inspector tambahkan kondisi. Pilih kondisi Jumping. Pada panah yang menuju ke Player_idle, ubah menjadi false, dan pada panah yang menuju ke Player_run, ubah menjadi true.



Gambar 9.27 *Inspector Blend Jump*

28. Selanjutnya mengubah script dengan mengganti yang sama seperti pada modul.

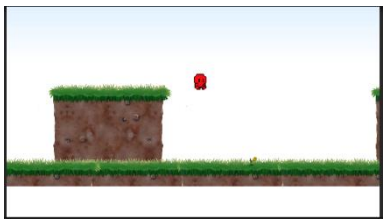
```
if (Input.GetButtonDown("Jump"))
{
    animator.SetBool("Jumping", true);
    jump = true;
}
```

```
void FixedUpdate()
{
    GroundCheck();
    Move(horizontalValue, jump);
    animator.SetFloat("Blend", Mathf.Abs(rb.velocity.x));
    animator.SetFloat("Blend Jump", rb.velocity.y);
}
```

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

Gambar 9.28 *Update Script*

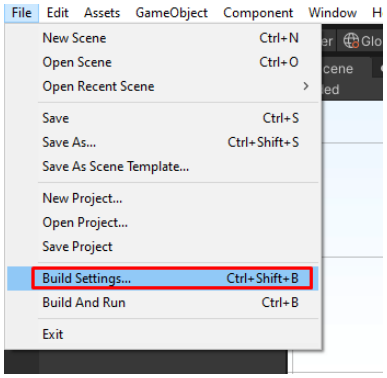
29. Setelah skrip diperbarui, jalankan proyek game dan lakukan aksi melompat, berlari, dan jatuh, sehingga animasi akan aktif.



Gambar 9.29 *Play Project*

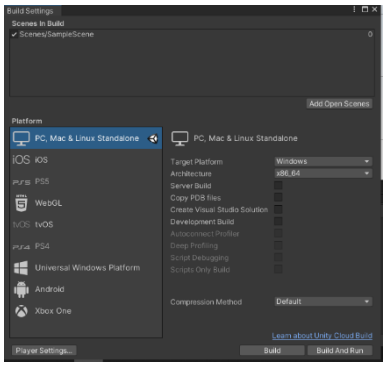
B. Render

1. Untuk proses merender, buka file kemudian pilih Build Settings. Setelah masuk ke Build Settings, pilih PC, Mac, dan Linux. Jangan lupa mencentang proyek kita pada bagian Scenes in Build. Jika belum ada, tekan Add Open Scenes.



Gambar 9.30 Render Settings

2. Setelah itu kita klik Build dan kita pilih project jadinya akan disimpan dimana. Dan tunggu hasilnya.



Gambar 9.31 Building Render

C. Link Pengumpulan Github

Link : https://github.com/farhan2153/2118139_PRAK_ANIGAME.git

KUIS

KUIS PERTEMUAN 9 👍

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

Analisa :

Dalam metode HandleJumpInput(), parameter boolean ditambahkan pada SetBool untuk menandakan apakah karakter sedang melompat. Ini memungkinkan animator mengatur transisi animasi secara akurat, sehingga karakter akan melompat saat tombol ruang ditekan dan tetap dalam animasi melompat jika tombol tersebut ditekan terus-menerus. Selain itu, penambahan gaya dorongan ke atas dilakukan saat tombol ruang ditekan pertama kali menggunakan rb.AddForce agar karakter melompat. Pada metode HandleMovementInput(), logika pergerakan karakter disesuaikan. Jika nilai move tidak nol, karakter dianggap bergerak dan isWalking diatur menjadi true, sehingga karakter bergerak sesuai arah move. Ketika tidak ada input gerakan, isWalking diatur menjadi false sehingga animasi berhenti. Selain itu, perubahan skala karakter disesuaikan dengan arah geraknya menggunakan Mathf.Sign(move), sehingga karakter menghadap ke arah yang benar sesuai input geraknya.