



TUGAS PERTEMUAN: 9

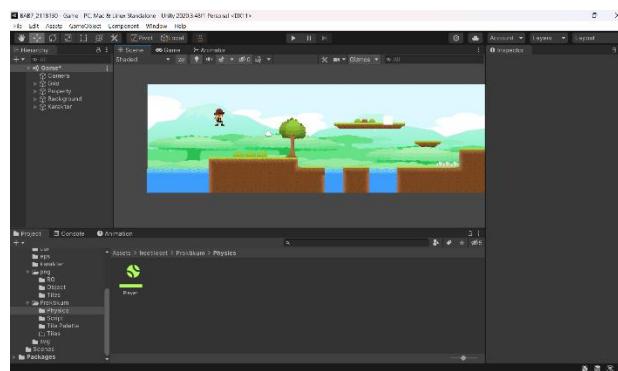
GAME ANIMATION

NIM	:	2118130
Nama	:	Nurul Faizah
Kelas	:	D
Asisten Lab	:	WISANDO BERLIAN PANDENSOLANG (2218095)

9.1 Tugas 9 : Mengimplementasikan Game Animation

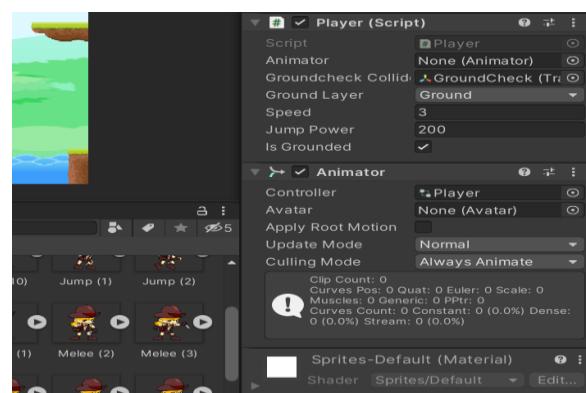
A. Membuat Character Animation

1. Pertama buka project Unity sebelumnya yang telah diimpor.



Gambar 9.1 Tampilan Project Unity

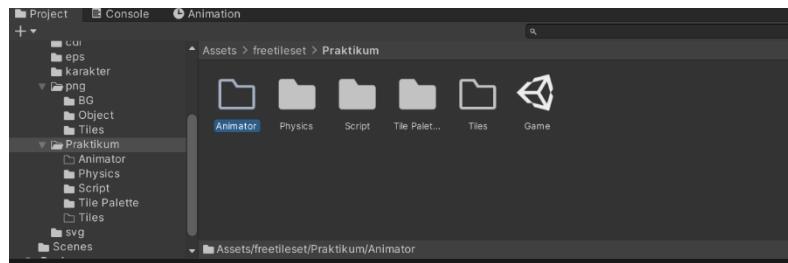
2. Berikutnya adalah pada karakter di inspector nya klik *add component* dan tambahkan *animator*.



Gambar 9.2 Add Component Animator

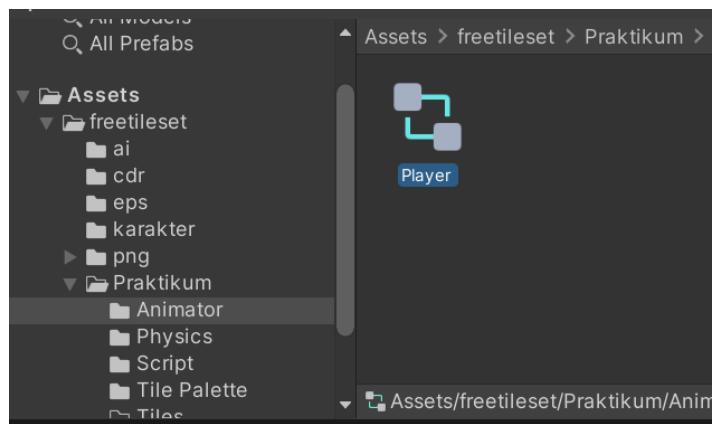


3. Buat folder baru pada TugasPraktikkum berikan nama Animator.



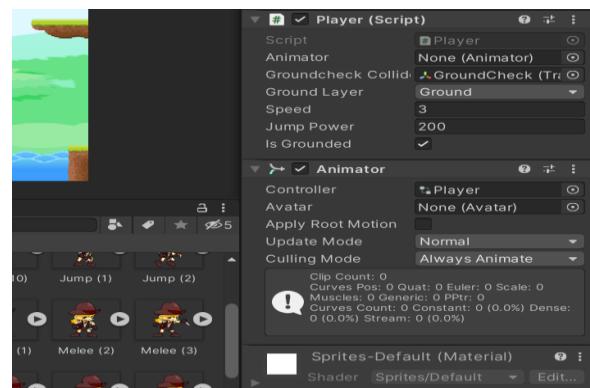
Gambar 9.3 Folder Animator

4. Di dalam *Folder Animator* yang sudah dibuat, buat *file Animator Controller* dengan cara klik kanan kemudian *create* pilih *Animator Controller* berikan nama Player.



Gambar 9.4 Animator Controller

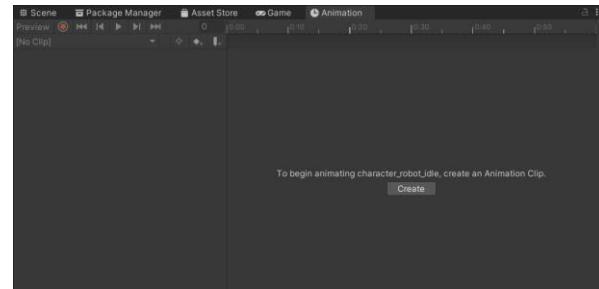
5. Selanjutnya adalah pada inspector cari komponen animator, ubah controller menjadi Player.



Gambar 9.5 Component Animator

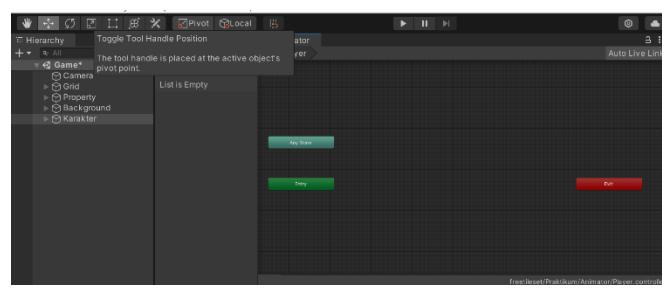


6. Langkah berikutnya adalah menambahkan menu panel Animation atau dengan menekan Ctrl + 6.



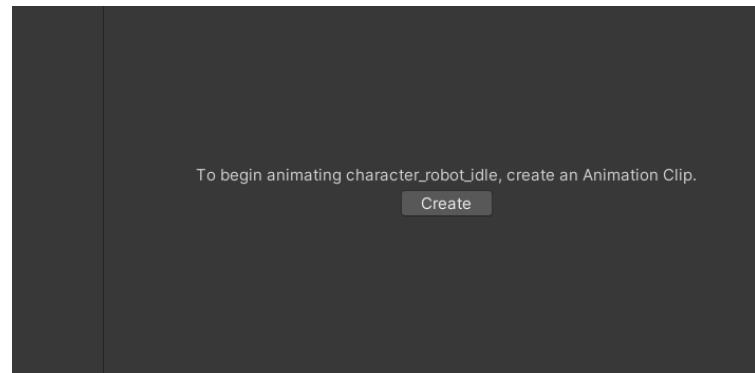
Gambar 9.6 Menu Panel *Animation*

7. Tambahkan juga menu panel Animator.



Gambar 9.7 Menu Panel *Animator*

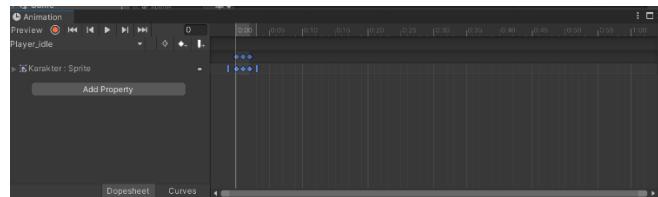
8. Selanjutnya adalah pembuatan animasi pada *asset* karakter, pertama pada *menu panel animation* klik *Create*.



Gambar 9.8 *Create Animation Clip*

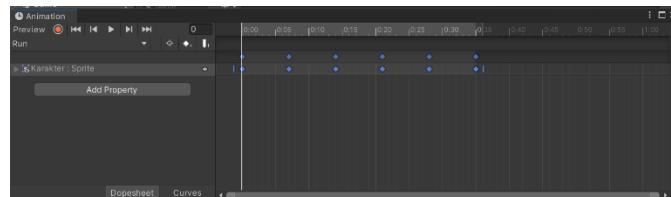


9. Kemudian yang dilakukan pertama adalah menyimpan *Folder Animation* dan berikan nama Player_idle, setelah itu di dalam menu *project asset* masukkan *asset character_idle* dengan cara *drag and drop* ke tab *animation*.



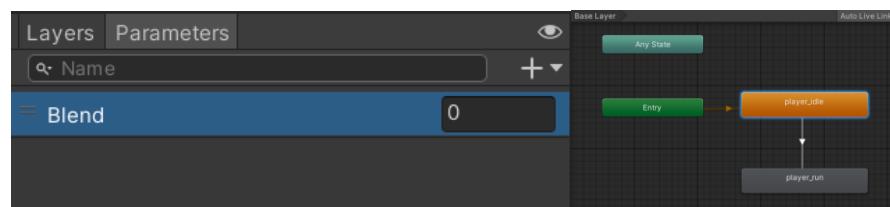
Gambar 9.9 Player_idle Animation

10. Tambahkan juga animasi baru berikan nama Player_run, lakukan hal yang sama seperti langkah sebelumnya.



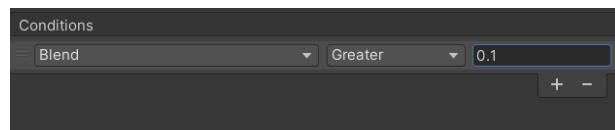
Gambar 9.10 Player_run Animation

11. Setelah kita menambahkan animasi, selanjutnya adalah membuka *Panel Animator*, buat transisi antara player_idle dengan player_run. Untuk caranya adalah klik kanan pada Player_idle pilih *make transition* tarik ke player_run. Jangan lupa masuk ke tab parameter buat tipe data float berikan nama Blend.



Gambar 9.11 Transition Player_idle dengan Player_run

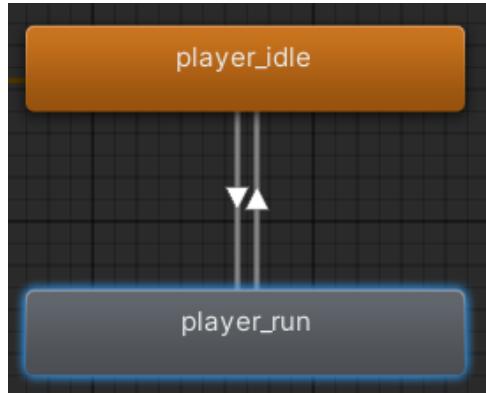
12. Setelah ada transisi, klik tanda panah putih dan pada bagian conditions klik icon + kemudian atur menjadi Blend ubah angka 0 menjadi 0,01.



Gambar 9.12 Conditions Blend



13. Lakukan hal yang sama sebaliknya dari Player_run transisi ke Player_idle. Pada komponen *conditions* tambahkan juga *Blend* dan *setting* ke *Less* ubah ukuran menjadi 0,01.



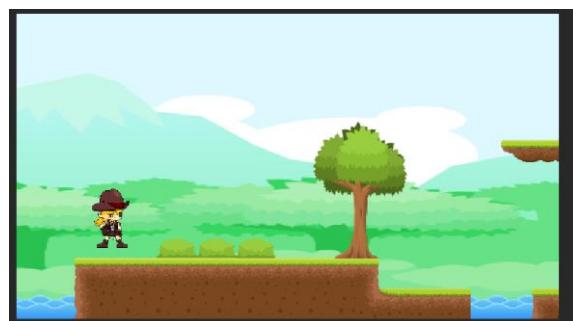
Gambar 9.13 Player_run transition to Player_idle

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

```
public Animator animator;  
  
Rigidbody2D rb;  
  
[SerializeField] Transform groundCheckCollider;  
[SerializeField] LayerMask groundLayer;  
  
const float groundCheckRadius = 0.2f; // +  
[SerializeField] float speed = 1;  
[SerializeField] float jumpPower = 100;  
float horizontalValue;  
  
[SerializeField] bool isGrounded; // +  
bool facingRight;  
bool jump;  
  
private void Awake()  
{  
    rb = GetComponent<Rigidbody2D>();  
    animator = GetComponent<Animator>();  
}  
  
void Update ()  
{  
    horizontalValue = Input.GetAxisRaw("Horizontal");  
    if (Input.GetButtonDown("Jump"))  
    {  
        animator.SetBool("Jumping", true);  
        jump = true;  
    }  
    else if (Input.GetButtonUp("Jump"))  
    {  
        jump = false;  
    }  
}
```

Gambar 9.14 Update Script Players

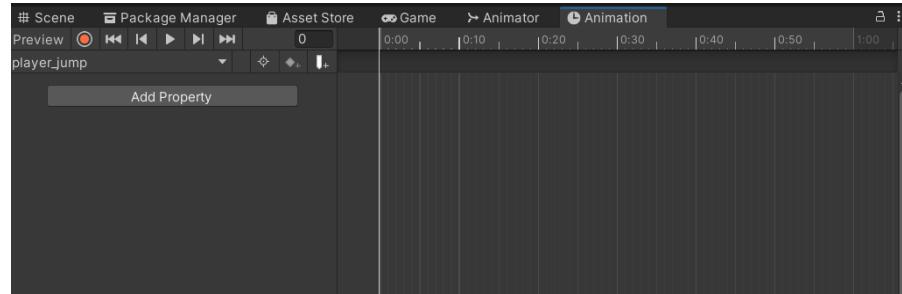
15. Jika dijalankan maka player memiliki animasi ketika berhenti maupun berlari.



Gambar 9.15 Animation run and idle

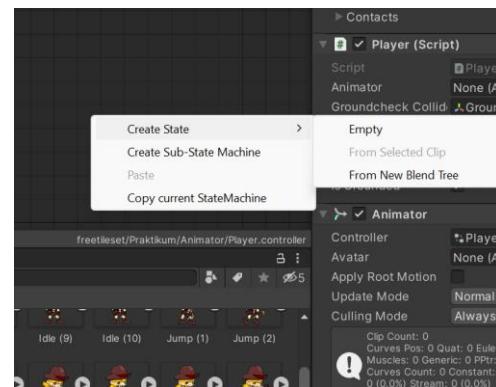


16. Selanjutnya adalah membuat animasi baru yaitu Player_jump, pada panel animasi *create new clip*.



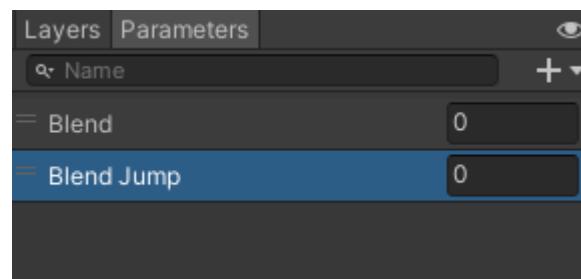
Gambar 9.16 *Animation Player_jump*

17. Kemudian untuk menambahkan animasi lompat, pada window animator buat *From New Blend Tree*.



Gambar 9.17 *From New Blend Tree*

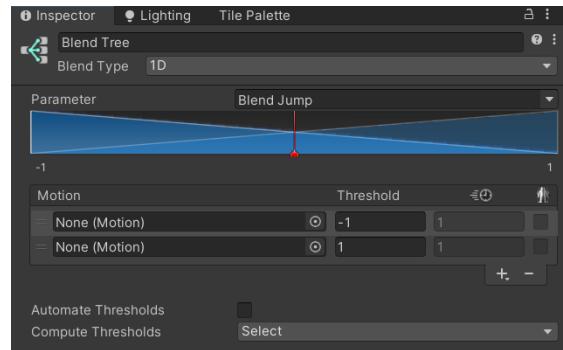
18. Pada menu paramaters tambahkan paramater dengan tipe data *float* tekan icon + dan berikan nama *Blend Jump*.



Gambar 9.18 *Blend Jump*

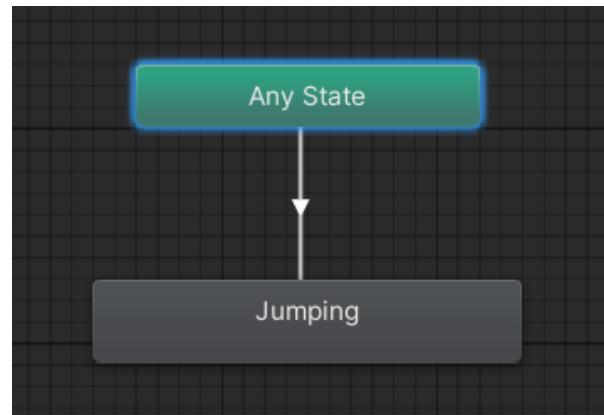


19. Klik dua kali di window Animator pada Blend Tree “Jumping”, tekan pada Blend Tree. Dan pada inspector ubah parameters menjadi Blend Jump, ubah juga motionnya dengan Player Fall dan Run. Sesuaikan settingan dengan gambar dibawah ini.



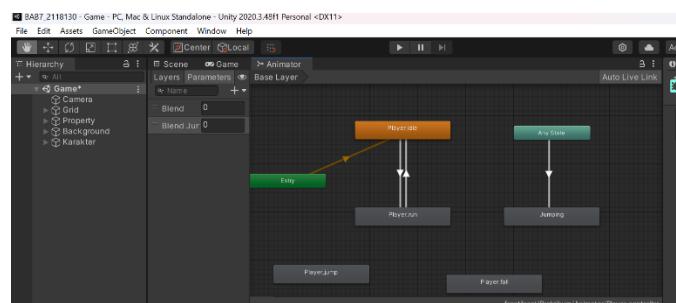
Gambar 9.19 Inspector Blend Tree

20. Kembali ke Base Layer, klik kanan Any State buat transisi dan tarik panahnya ke jumping.



Gambar 9.20 Base Layer

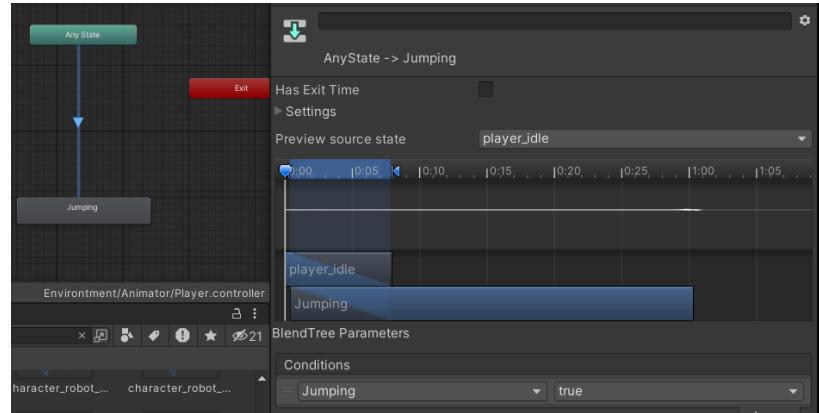
21. Langkah selanjutnya adalah menambahkan parameter transisi dengan tipe data bool dan berikan nama Jumping.



Gambar 9.21 Parameters Jumping

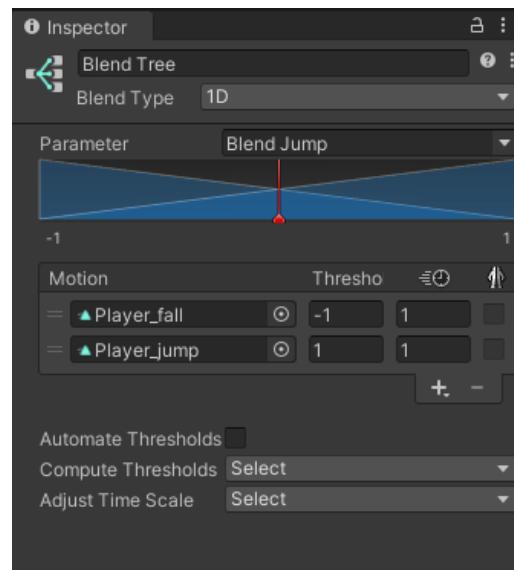


22. Klik panah yang mengarah ke Jumping, pada inspector tambahkan condition Jumping dan ganti nilainya menjadi true.



Gambar 9.22 Jumping Condition

23. Pada *Inspector* juga terdapat *parameters*, ubah *parameters* tersebut ke *Blend Jump*. Dan pada *motion* ganti menjadi Player_fall dan Player_jump, samakan settingan seperti pada gambar dibawah ini.



Gambar 9.23 Inspector Blend Jump

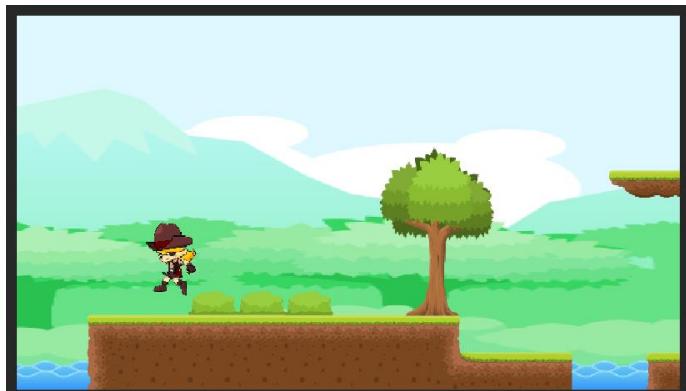


24. Selanjutnya adalah mengubah script dengan mengganti yang sama seperti pada modul.

```
void Update ()  
{  
    horizontalValue = Input.GetAxisRaw("Horizontal");  
    if (Input.GetButtonDown("Jump"))  
    {  
        animator.SetBool("Jumping", true);  
        jump = true;  
    }  
    else if (Input.GetButtonUp("Jump"))  
    {  
        jump = false;  
    }  
  
    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 9.24 *Update Script*

25. Setelah script di update, kita play project game kita dan lakukan jumping, run dan fall maka akan muncul animasinya.

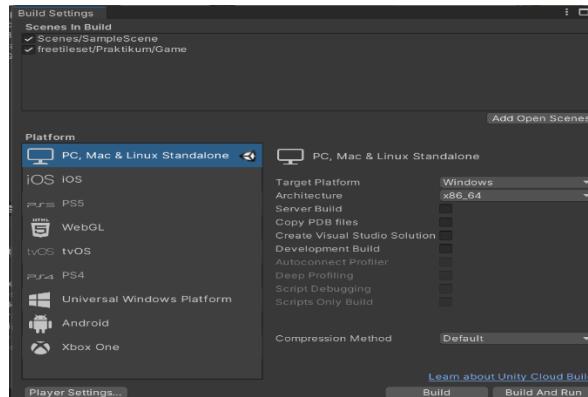


Gambar 9.25 *Play Project*



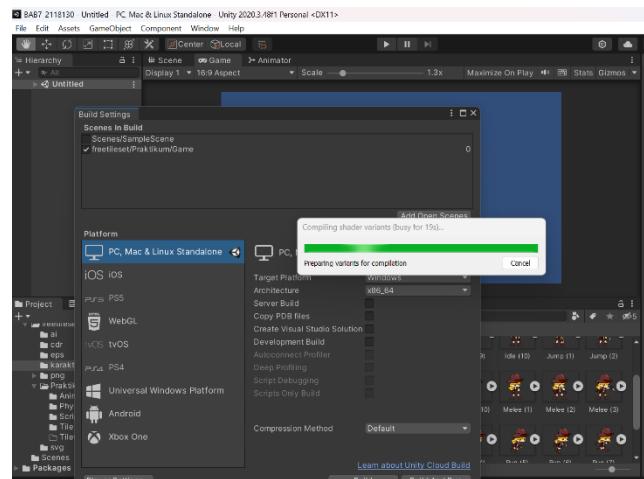
B. Render

- Untuk proses merender pada file kemudian pilih Build Settings atau menekan Ctrl + Shift + B, setelah masuk Setting Build pilih PC, Mac dan Linux. Jangan lupa pada scenes in build mencentang project kita, kalo belum ada tekan Add Open Scenes.



Gambar 9.26 Render Settings

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



Gambar 9.27 Building Render



KUIS

KUIS PERTEMUAN 9

```
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);
    }
}
```

Analisa :

Untuk penjelasan kode diatas merupakan 2 metode yang menangani input pengguna untuk melompat dan gerakan horizontal dalam permainan yaitu pada baris animator.SetBool("isJumping",); dan animator.SetBool("isJumping",);, nilai boolean tidak diberikan. Fungsi SetBool membutuhkan dua argumen: nama parameter dan nilai boolean, sehingga kita harus memberikan nilai true atau false. Selain itu, penggunaan rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse); salah karena variabel jumpForce tidak didefinisikan dalam. Seharusnya menggunakan jumpPower, sesuai dengan deklarasi variabel sebelumnya. Terakhir, terdapat kesalahan logika di HandleMovementInput, di mana kondisi else if (move > 0) tidak akan pernah tercapai karena kondisi if (move != 0) sudah menangani semua kasus di mana move tidak sama dengan 0. Seharusnya, kita menggunakan if terpisah untuk menangani move > 0 dan



move < 0. Inti dari Script ini mengatur lompatan dan gerakan horizontal pemain, termasuk animasi yang sesuai, berdasarkan input dari pemain. Jika pemain menekan atau menahan tombol spasi, pemain akan melompat, dan jika pemain menekan tombol arah kiri atau kanan, pemain akan bergerak ke arah tersebut dan animasi serta skala lokal akan disesuaikan.