PRAKTIKUM ANIMASI DAN GAME PERTEMUAN :9

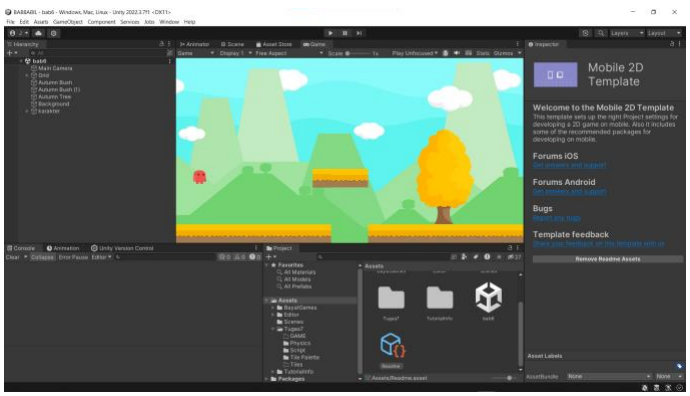
Game Animation

NIM	**:**	2018048
NAMA	**:**	Mustaqdimin Salsabil Haq
KELAS	**:**	D
ASISTEN LAB	**:**	Difa Fisabilillah (2118052)

9.1 Tugas 1: Membuat Game Animation

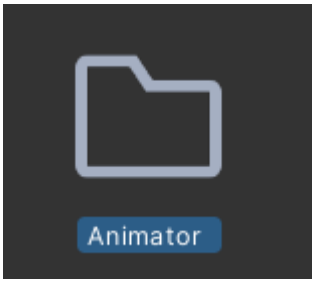
A. Membuat Game Animation

1. Buat file *projek Unity* Tugas 8



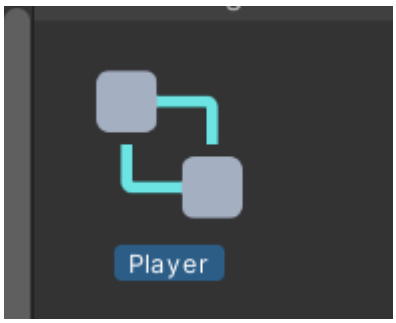
Gambar 9.1 Buka projek tugas 8

2. Klik karakter, pergi ke *Inspector*, pilih *Add Component*, cari Animator ,Buat sebuah folder baru dengan nama “Animator”



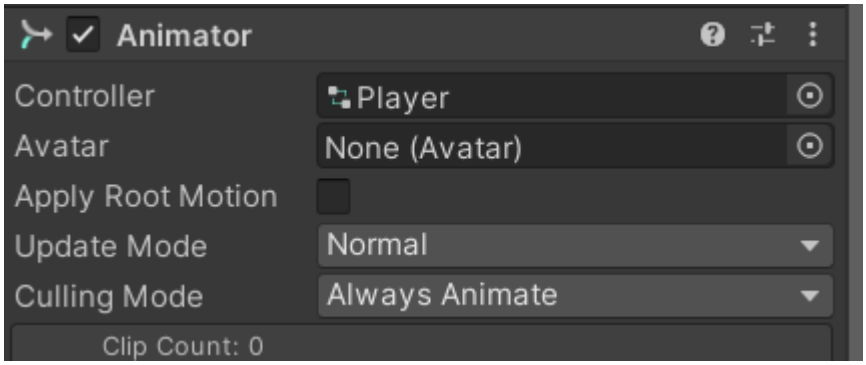
Gambar 9.2 Membuat folder Animator

3. Klik kanan folder Animator>*Create*>*Animation Controller*, dan ubah namanya menjadi “Player”



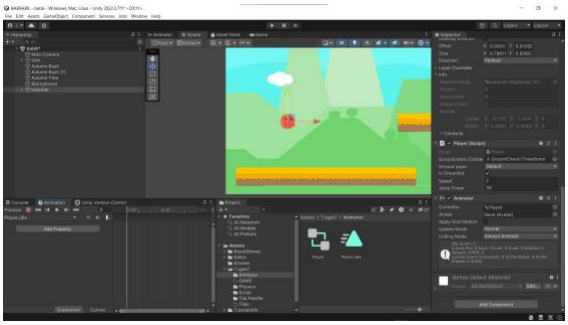
Gambar 9.3 Membuat Karakter Animator

4. Drag karakter, kemudian masukkan kedalam Controller komponen Animator



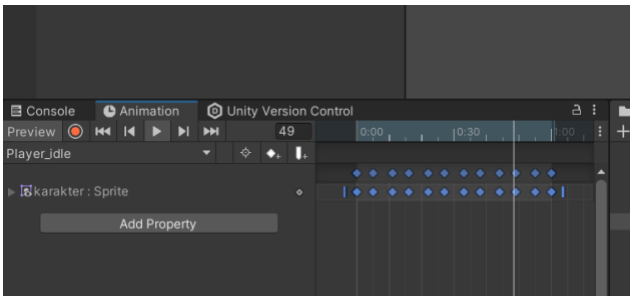
Gambar 9.4 Mendrag karakter Animator

5. Klik karakter, kemudian pergi ke tab Animation, pilih Create



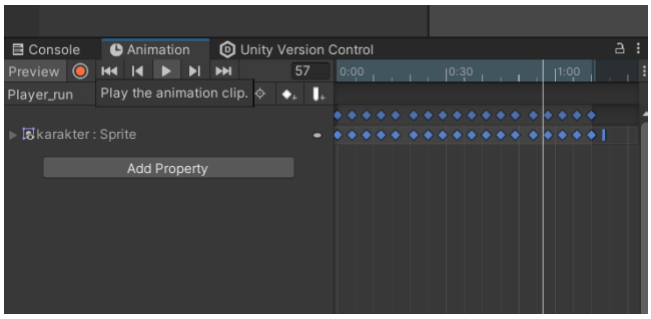
Gambar 9.5 Membuat suatu *Animation*

6. Drag and Drop karakter lalu regangkan, pada *timeline* tekan *Ctrl+A* di *keyboard*, klik bagian kotak kecil disamping *keyframe* terakhir dan geser sampai waktu 01:00



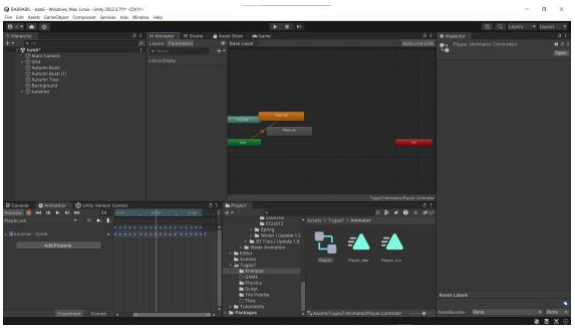
Gambar 9.6 *Drag and Drop* karakter

7. Buat animasi baru, Klik pada “Player_idle” kemudian pilih Create New Clip, dan beri nama “Player_run”, Simpan pada Folder Animator Pilih karakter *run* kemudian drag kedalam *timeline* Player_run *Animation* lalu geser ke *keyframe* waktu 1:10



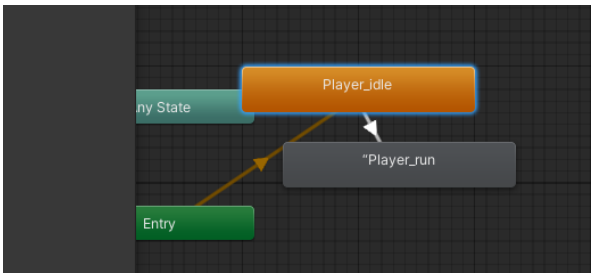
Gambar 9.7 Menambahkan karakter *playerrun*

8. Klik kanan pada area sekitar *Animator*, pilih *Create State>From New Blend Tree*



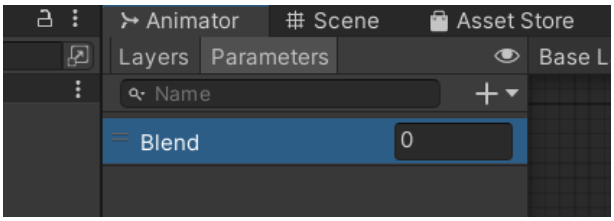
Gambar 9.8 Membuat *Blend Tree*

9. Kemudian buat transisi antara player_idle dan player_run dengan cara klik kanan pada player_idle dan pilih Make Transition dan tarik ke player_run



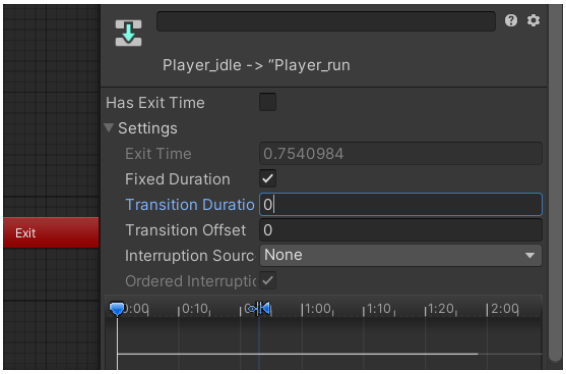
Gambar 9.9 Menambahkan *Motion field*

10. Masuk ke tab parameter, tambahkan tipe data float dengan cara tekan icon tambah dan ubah namanya menjadi “Blend”



Gambar 9.10 Menambahkan sebuah *motion*

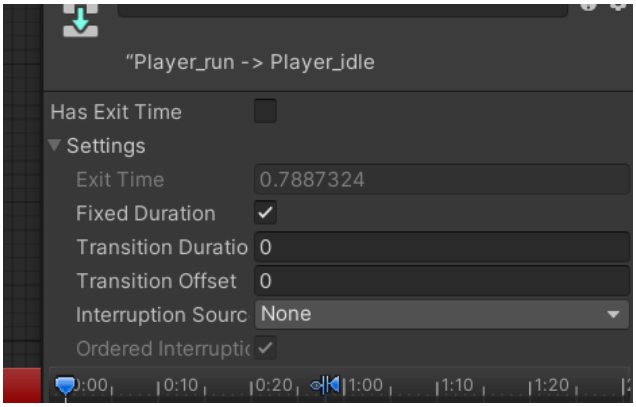
11. Klik panah putih tersebut, pada bagian conditions klik icon tambah kemudian atur menjadi “Blend”, Atur nilai conditions blend tersebut menjadi 0.01 Pada bagian Settings, hilangkan centang pada Has Exit Time dan atur nilai Transition Duration menjadi 0



Gambar 9.11 Menambahkan sebuah *motion*

12. Buat transisi juga dari player_run ke player_idle dengan cara klik kanan pada player_run dan pilih Make Transition. Tambahkan parameter transisi dengan tipe data Float. Klik ikon tambah dan rename menjadi “Blend”. Setelah itu, ubah operator dari Greater menjadi Less dan atur

nilainya menjadi 0.01. Pada bagian Settings, hilangkan centang pada Has Exit Time dan atur nilai Transition Duration menjadi 0



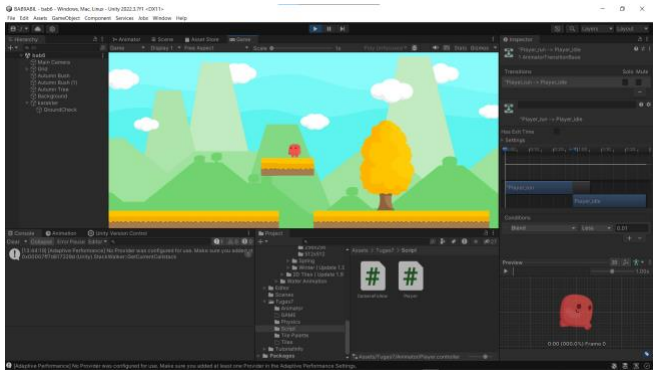
Gambar 9.12 Menambahkan sebuah *motion*

13. Agar animasi dapat sesuai ketika berjalan, buka script Player dan tambahkan source code berikut pada class Player

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Player : MonoBehaviour
{
    public Animator animator;
    Rigidbody2D rb;
    [SerializeField]
    Transform groundcheckCollider;
    [SerializeField]
    LayerMask groundLayer;
    const float groundCheckRadius = 0.2f;
    [SerializeField]
    bool isGrounded;
    [SerializeField]
    float speed = 1;
    [SerializeField]
    float jumpPower = 100;
    float horizontalValue;
    bool facingRight;
    bool jump;
    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
    }
    // Update is called once per frame
    void Update()
    {
        horizontalValue =
Input.GetAxisRaw("Horizontal");
        if (Input.GetButtonDown("Jump"))
            jump = true;
```

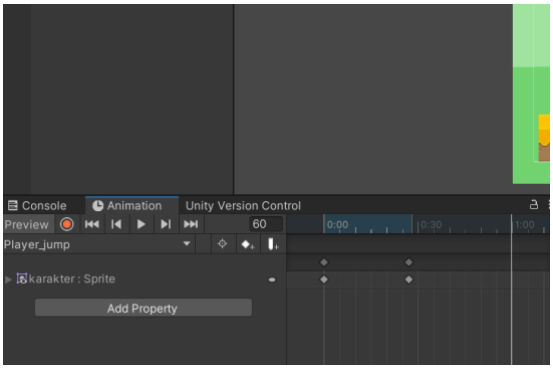
```
        else if (Input.GetButtonUp("Jump"))
            jump = false;
    }
    void FixedUpdate()
       {
            GroundCheck();
            Move(horizontalValue, jump);

            animator.SetFloat("Blend",
Mathf.Abs(rb.velocity.x));
        }
     void GroundCheck()
    {
        isGrounded = false;
        Collider2D[] colliders =
Physics2D.OverlapCircleAll(groundcheckCollider.position
, groundCheckRadius,
       groundLayer);
        if (colliders.Length > 0)
            isGrounded = true;
    }
    void Move(float dir, bool jumpflag)
    {
        if (isGrounded && jumpflag)
        {
            isGrounded = false;
            jumpflag = false;
            rb.AddForce(new Vector2(0f, jumpPower));
        }
        #region gerak kanan kiri
        float xVal = dir * speed * 100 *
Time.fixedDeltaTime;
        Vector2 targetVelocity = new Vector2(xVal,
rb.velocity.y);
        rb.velocity = targetVelocity;
        if (facingRight && dir < 0)
        {
            transform.localScale = new Vector3(-1, 1,
1);
            facingRight = false;
        }
        else if (!facingRight && dir > 0)
        {
            transform.localScale = new Vector3(1, 1,
1);
            facingRight = true;
        }
        #endregion
    }
}
```

14. Jika dijalankan maka player dapat memiliki animasi ketika berhenti ataupun ketika berjalan



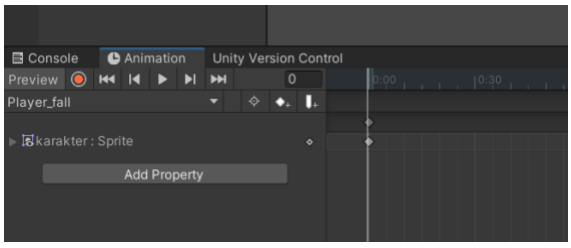
Gambar 9.13 Menambahkan *astrowalk*

15. Kemudian buat animasi baru tekan tulisan “Player_run” kemudian pilih Create New Clip, dan beri nama “Player_jump”, Pada folder player buka jump lalu pilih gambar player-jump-1, kemudian drag ke tab Animation.



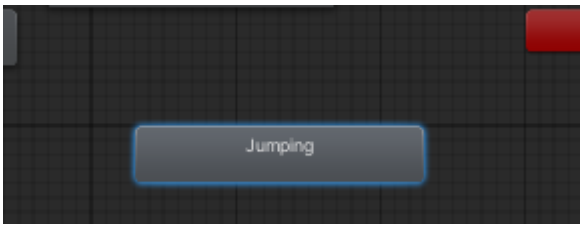
Gambar 9.14 Menambahkan *astrowalk*

16. Buat animasi baru dengan cara tekan tulisan “Player_jump" kemudian pilih Create New Clip, dan beri nama “Player_fall”, Pada tab Project buka folder karakter lalu pilih Idle dan pilih gambar playerfall, kemudian drag ke tab Animation.



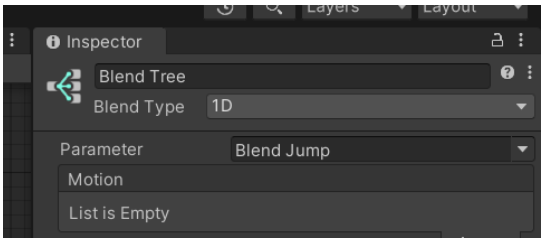
Gambar 9.15 Menambahkan *astrofall*

17. Buat *Blend Tree*, pergi ke *Inspector* dan ubah namanya menjadi *Jumping*



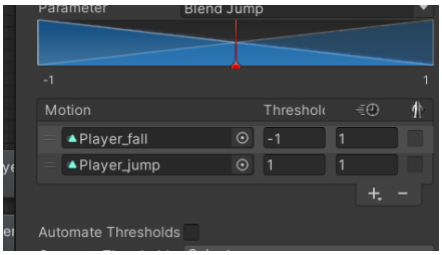
Gambar 9.16 Menambahkan *Jumping*

18. Pada menu Parameteres tambahkan parameter tipe data Float tekan icon + dan ubah namanya menjadi “Blend Jump”, Pada menu Animator, Klik dua kali pada Blend Tree “Jumping”, Tekan pada Blend Tree, Klik 2X Blend Tree “Jumping”, pada inspector ubah parameter menjadi “Blend Jump”



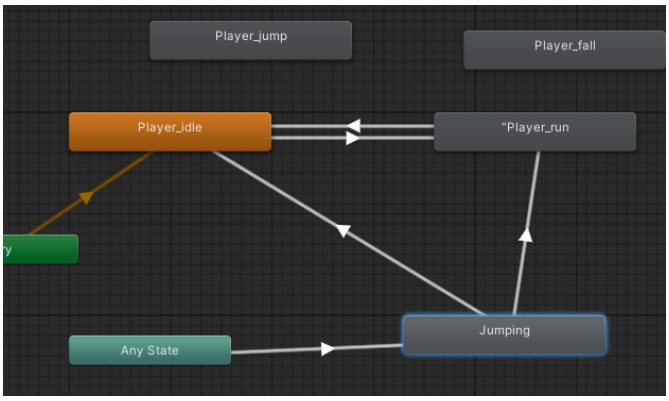
Gambar 9.17 Mengubah parameter *Jumping*

19. Buka menu Inspector, tekan icon + dan pilih Add Motion Field. Tambahkan dua Motion Field, Klik bagian icon None (Motion), maka akan muncul Windows Motion, Tambahkan Sesuai dengan urutan, Hilangkan centang “Automate Thresholds” dan atur nilai Threshold seperti berikut



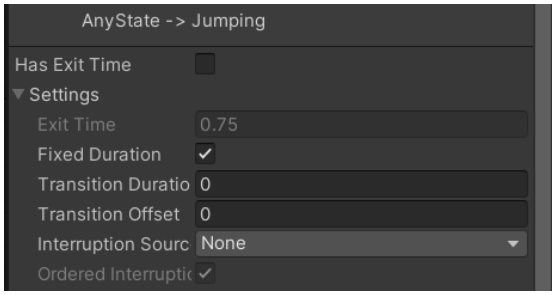
Gambar 9.18 Mengubah parameter *Jumping*

20. Kembali ke Base Layer, klik kanan Any State, pilih Make Transition dan arahkan panahnya ke Jumping, Klik kanan Jumping, pilih Make Transition dan arahkan panahnya ke Player_idle dan Player_run



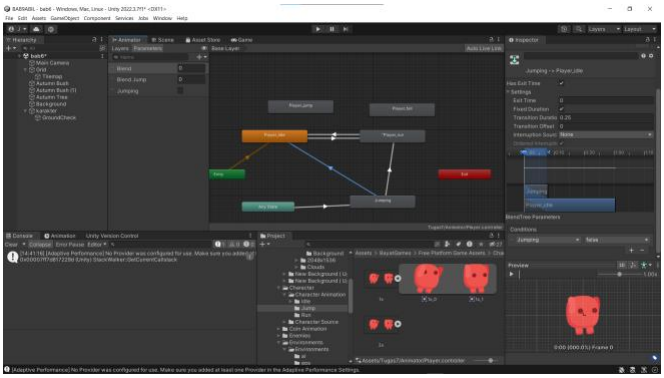
Gambar 9.19 Mengubah parameter *Jumping*

21. Tambahkan parameter transisi dengan tipe data Bool tekan icon + dan ubah namanya menjadi "Jumping", Klik panah yang mengarah ke Jumping, pada inspector tambahkan condition, pilih condition Jumping dan ubah nilainya menjadi true, Klik Settings dan ubah nilai Transition Duration menjadi 0 dan hilangkan centang Has Exit Time



Gambar 9.20 Mengubah parameter *Jumping*

22. Klik panah yang mengarah ke Player_idle dan Player_run, pada inspector tambahkan condition, pilih condition Jumping, pada arah panah ke player_idle ubah menjadi false, pada arah panah ke player_run ubah menjadi true, Klik Settings dan ubah nilai Transition Duration menjadi 0 dan hilangkan centang Has Exit Time



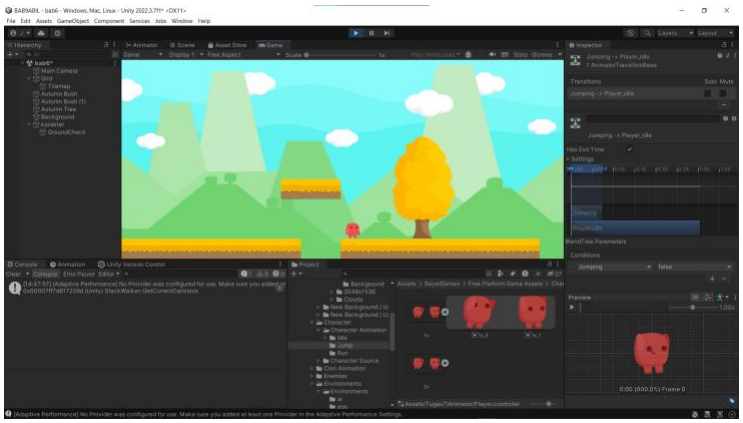
Gambar 9.21 Mengubah parameter *Jumping*

23. Tambahkan Source code seperti ini

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Player : MonoBehaviour
{
    public Animator animator;
    Rigidbody2D rb;
    [SerializeField]
    Transform groundcheckCollider;
    [SerializeField]
    LayerMask groundLayer;
    const float groundCheckRadius = 0.2f;
    [SerializeField]
    bool isGrounded;
    [SerializeField]
    float speed = 1;
    [SerializeField]
    float jumpPower = 100;
    float horizontalValue;
    bool facingRight;
    bool jump;
    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
    }
    // Update is called once per frame
    void Update()
    {
        horizontalValue                                    =
Input.GetAxisRaw("Horizontal");
        if (Input.GetButtonDown("Jump")) {
```

```
            jump = true;
            animator.SetBool("Jumping", true);
        }
        else if (Input.GetButtonUp("Jump")) {
            jump = false;
            animator.SetFloat("Blend              Jump",
rb.velocity.y);
        }}
    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);
    }
    void Move(float dir, bool jumpflag)
    {
        if (isGrounded && jumpflag)
        {
            isGrounded = false;
            jumpflag = false;
            rb.AddForce(new Vector2(0f, jumpPower));
        }
        #region gerak kanan kiri
        float   xVal   =   dir   *   speed   *   100   *
Time.fixedDeltaTime;
        Vector2  targetVelocity  =  new  Vector2(xVal,
rb.velocity.y);
        rb.velocity = targetVelocity;
        if (facingRight && dir < 0)
        {
            transform.localScale  =  new  Vector3(-1,  1,
1);
            facingRight = false;
        }
        else if (!facingRight && dir > 0)
        {
            transform.localScale = new Vector3(1, 1, 1);
            facingRight = true;
        }
        #endregion
    }
}
```

24. Jika di play maka karakter sudah bisa bergerak dengan animasi



Gambar 9.22 Mengubah parameter *Jumping*

B. KUIS

Source Code

```
    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);
```

```
            }
        }
```

Penjelasan

Source code diatas yang diberikan warna merah merupakan *source code* yang terdapat kesalahan pada source code tersebut yang nantinya akan menyebabkan *error*. Pada source code *error* 1 dan 2 disebabkan karena *syntax* yang kurang lengkap, pada *source code* digunakan metode *SetBool* tetapi tidak diberikan kondisi *Boolean true* atau *false* pada kondisi boolean. Pada source code *Error* 3 terdapat kesalahan pada nilai *transform* yang seharusnya menggunakan nilai *vektor3.right* karena pergerakan karakter adalah *horizontal*, jika menggunakan nilai kiri, maka hasilnya harus dilakukan konversi terlebih dahulu karena terdapat nilai *negative* sehingga harus dilakukan. Pada source code *Error* 4 kondisi *isWalking* harusnya adalah *true*, karena pada kondisi *if* diberikan kondisi *move* != 0 yang akan menjalankan *idle* atau kondisi diam maka, harusnya *else* adalah nilai *move* akan sama dengan 1, sehingga kondisi *isWalking* harus bernilai *true*. Pada source code *Error* 5 dan 6 terdapat kesalahan dalam memberikan nilai *vector* yang berbeda di antara keduanya dimana nilai *vector* dari kedua *line* tersebut harus bernilai sama karena akan berpengaruh pada ukuran karakter pada saat *game* dimainkan