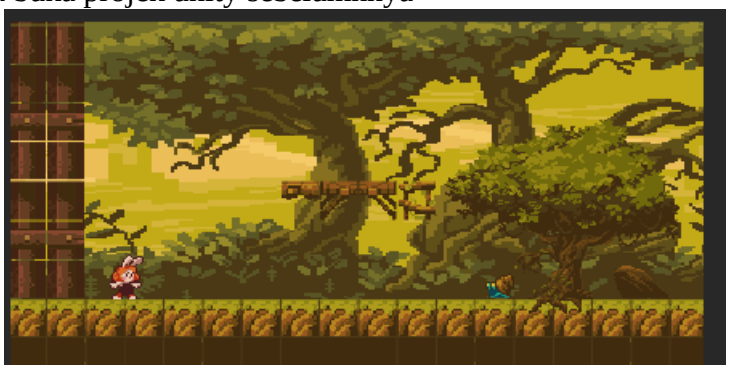
TUGAS PERTEMUAN: 8

CAMERA & CHARACTER MOVEMENT

NIM	:	2118021
Nama	:	Irfani Muhamad Al Rizqi
Kelas	:	A
Asisten Lab	:	NATASYA OCTAVIA(2118034)

1 Tugas 1 : Membuat Character Movement, Detect Ground, Jumping, & Camera Movement

1. Pertama buka projek unity sebelumnnya

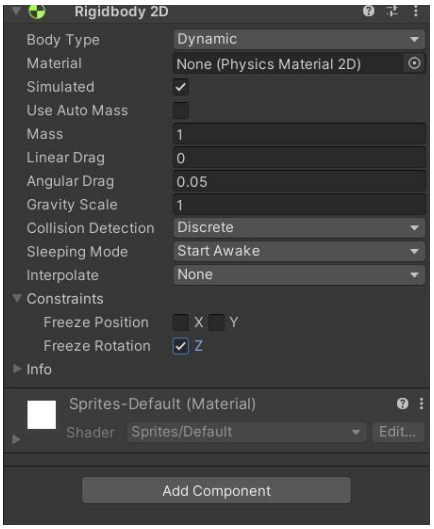


Gambar 8.1 Membuka Projek Unity

2. Tambahkan karakter playe idle 1 kedalam *hierarchy*, untuk menambahkannya seret *asset* ke dalam *hierarchy*.

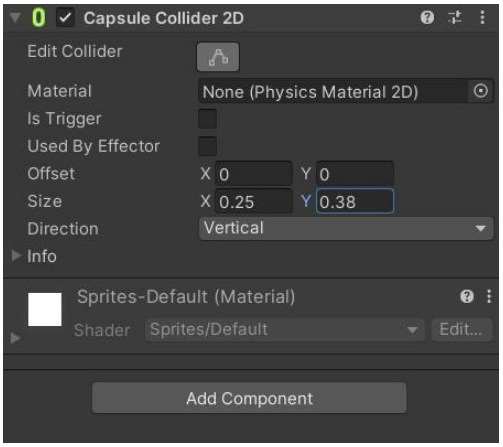
Gambar 8.2 Menambahkan Karakter

3. Lalu klik "*Player-idle-1*" kemudian tambahkan *component* "*Rigidbody* 2D", lalu pada *constraints* ceklis bagian "*Freeze Rotation Z*".



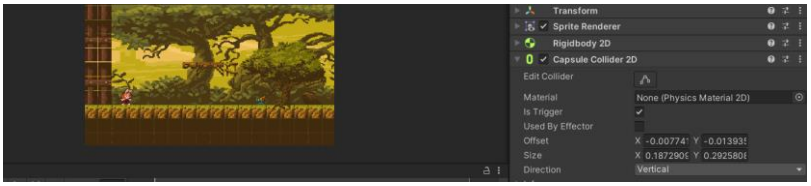
Gambar 8.3 Menambahkan Component

4. Tambahkan *component* lagi yaitu "*Capsule Collider* 2D", kemudian pada *menu* klik *icon edit collider*.



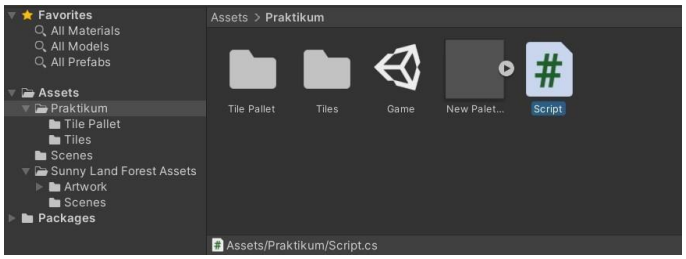
Gambar 8.4 Menambahkan Capsule Collider 2D

5. Lalu sesuaikan garis lingkaran dengan karakternya dengan memasukkan nilai pada *offset* X Y dan *size* X Y sesuiakan dengan karakter playernya.



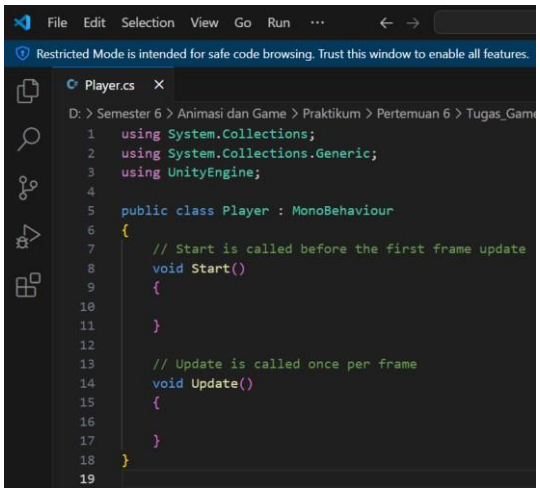
Gambar 8.5 Menyesuaikan Capsule Collider

6. Pada *folder* “Praktikum” buatlah *folder* baru Bernama “*Script*” kemudian isikan *folder* tersebut dengan membuat C# *Script* dengan klik kanan pada *folder* lalu pilih *Create* > C# *Script*, beri nama *file* dengan “*Player*”.



Gambar 8.6 Membuat Script Baru

7. Kemudian seret *file script* kedalam *Hierarchy Player*, kemudian klik dua kali pada *script* untuk masuk kedalam *text editor*.



Gambar 8.7 Masuk Kedalam Text Editor

8. Kemudian masukkan *source code* dibawah ini kedalam *text editor Player* yang sudah dibuat tadi.

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

public class Player : MonoBehaviour
{
    Rigidbody2D rb;

    [SerializeField] float speed = 1;
    float horizontalValue;
    bool facingRight;

    private void Awake()
    {
```

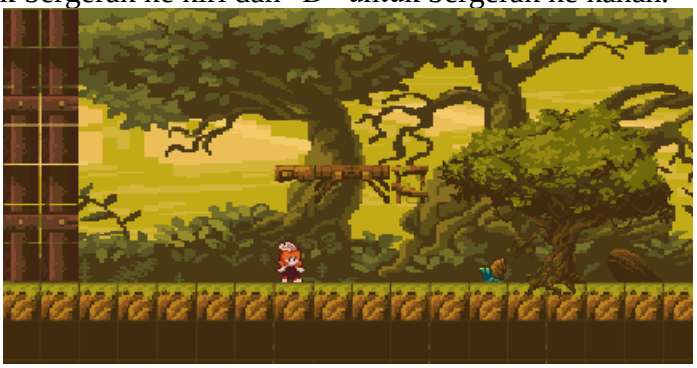
```
        rb = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        horizontalValue                                =
Input.GetAxisRaw("Horizontal");
    }

    void FixedUpdate()
    {
        Move(horizontalValue);
    }

    void Move(float dir)
    {
        #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)
        {
        // ukuran player
        transform.localScale = new Vector3(-1, 1,
1);
        facingRight = false;
        }

        else if (!facingRight && dir > 0)
        {
        // ukuran player
        transform.localScale = new Vector3(1, 1,
1);
        facingRight = true;
        }

        #endregion
    }
}
```

Analisa

Script di atas adalah sebuah komponen Unity untuk mengendalikan pergerakan karakter pemain dalam game 2D. Menggunakan Rigidbody2D, karakter dapat bergerak ke kiri atau kanan sesuai input dari pemain melalui sumbu horizontal (diperoleh dari Input.GetAxisRaw("Horizontal")). Dalam metode Move, kecepatan

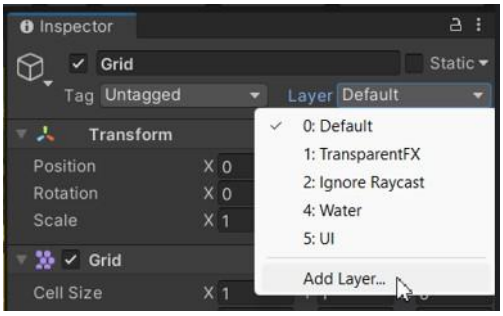
horizontal dihitung dan diterapkan ke Rigidbody2D untuk mengatur kecepatan karakter. Script ini juga menangani flipping karakter untuk memastikan bahwa sprite selalu menghadap ke arah gerakan.

9. Kemudian lakukan *test* dengan cara menekan tombol pada *keyboards* "A" untuk bergerak ke kiri dan "D" untuk bergerak ke kanan.



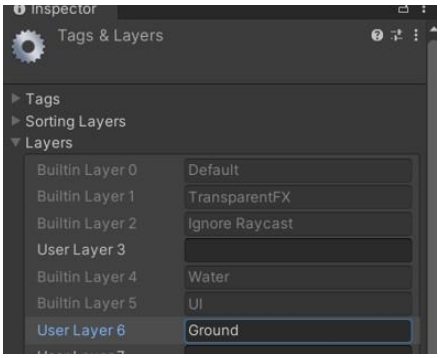
Gambar 8.8 Melakukan Test Pergerakan

10. Kemudian membuat karakter untuk dapat melompat menggunakan tombol *space* pada *keyboards*, klik *Grid* pada *hierarchy* pada *inspector* pilih *layer* kemudian *add layer*.



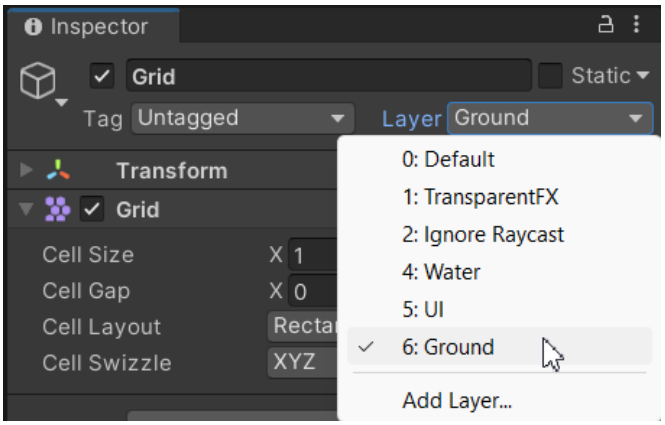
Gambar 8.9 Membuat Layer Baru

11. Ketika sudah menekan *Add layer* ketikkan "*Ground*" pada *User Layer* 6.



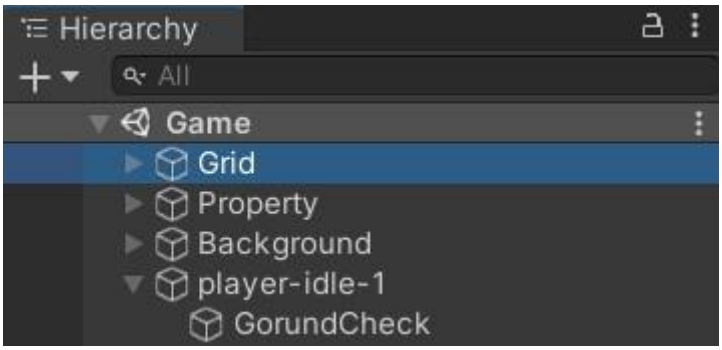
Gambar 8.10 Mengisi Layer

12. Ubah *layer* dari *layer default* menjadi *layer ground* yang telah dibuat tadi, jika muncul *pop up change layer* klik *yes*.



Gambar 8.11 Mengubah Layer

13. Kemudian buat *Hierarchy* baru pada *player* dengan klik kanan lalu pilih *Create empty* ubah nama menjadi "*GroundCheck*".



Gambar 8.12 Menambahkan Hierarchy Baru

14. Lalu Kembali ke *script player*, tambahkan *source code* dibawah ini.

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

const float groundCheckRadius = 0.2f; // +
[SerializeField] float speed = 1;
float horizontalValue;

[SerializeField] bool isGrounded; // +
bool facingRight;
```

Analisa

Penambahan variabel `groundcheckCollider`, `groundLayer`, `groundCheckRadius`, dan `isGrounded` dalam script ini memberikan kemampuan untuk mendeteksi apakah karakter berada di tanah, yang penting untuk mengatur aksi seperti melompat. `groundcheckCollider` menunjukkan posisi dari collider yang digunakan untuk deteksi tanah, sementara `groundLayer` menentukan lapisan mana yang dianggap sebagai tanah. `groundCheckRadius` adalah ukuran lingkaran deteksi, dan `isGrounded` menyimpan status apakah karakter sedang menyentuh

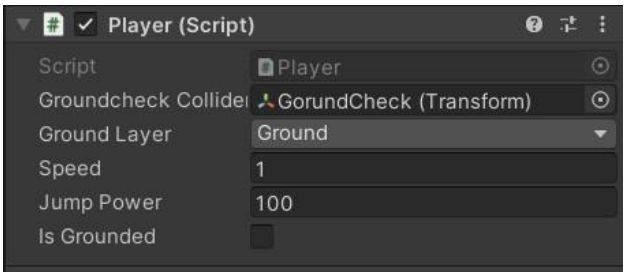
tanah. Setelah itu membuat *void ground check* dan tambahkan *GroundCheck*() pada *void fixedUpdate*.

```
   void FixedUpdate()
   {
   GroundCheck();
   Move(horizontalValue);
   }
   void GroundCheck()
   {
   isGrounded = false;
   Collider2D[] colliders =
Physics2D.OverlapCircleAll(groundcheckCollider.position
, groundCheckRadius,
   groundLayer); if
   (colliders.Length > 0)
   isGrounded = true;
   }
```

Analisa

Pada *void FixedUpdate* digunakan untuk memeriksa karakter menyentuh tanah dan menggerakkan pemain berdasarkan input *horizontal*. Kemudian pada *void Groundcheck* digunakan untuk pengecekan karakter menyentuh tanah dengan memeriksa *collider* dalam *radius* tertentu, jika *collider* terdeteksi maka pemain dianggap menyentuh tanah.

15. Lalu klik *Hierarchy Player* pada *inspector* bagian *Player* "*Script*" rubah bagian *Groundcheck Collider* menjadi "*GroundCheck* (*Transform*)" dan pada *Ground Layer* pilih "*Ground*".



Gambar 8.13 Mengatur Ground

16. Lalu Kembali *Player Script* untuk membuat *player* melompat tambahkan *script* berikut.

```
[SerializeField] float jumpPower = 100;

bool jump;
```

Analisa

Pada *code* diatas digunakan untuk mengatur kekuatan karakter untuk melompat, pada *code* kekuatan melompat diatur dengan nilai 100. Lalu pada *boolean jump* digunakan untuk menentukan keputusan untuk

karakter melompat

17. Kemudian tambahkan *script* dibawah ini, letakkan *code* pada bagian *void update.*

```
If (Input.GetButtonDown("Jump"))
Jump = true ;
else if (Input.GetButtonUp("Jump"))
Jump = false ;
```

Analisa

Pada *code* diatas digunakan untuk menentukan keputusan ketika tombol *space* ditekan. Jika *GetButtoDown* maka karakter akan melakukan lompatan dan jika *GetButtonUp* maka karakter akan menghentinkan lompatannya.

18. Lalu rubah *code* pada *void FixedUpdate*() dengan *code* dibawah ini.

```
void FixedUpdate()
     {
          GroundCheck();
          Move(horizontalValue, jump);
     }
```

Analisa

Pada *void FixedUpdate* terdapat dua fungsi didalamnya, pertama *GroundCheck*() yang digunakan untuk memeriksa pemain berada diatas tanah, pemeriksaan dilakukan dengan mendeteksi *collider*.

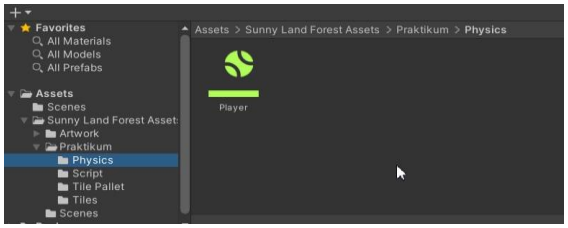
19. Tambahkan juga *code* dibawah ini pada *void Move*().

```
bool jumpflag

if(isGrounded && jumpflag)
{
isGrounded = false;
jumpflag = false;
rb.AddForce(new Vector2(0f, jumpPower));}
```

Analisa

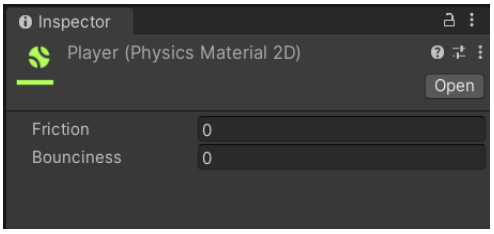
Pada *code* diatas digunakan untuk memastikan bahwa karakter dapat melompat jika berada di atas tanah. Fungsi *jumpflag* digunakan untuk membatasi gerakan melompat karakter sebanyak satu kali, pemain diperbolehkan melakukan lompatan lagi jika karakter berada di atas tanah.

20. Kemudian pada *folder* praktikum buat *folder* baru dengan nama "*Physics*", pada *folder physics* klik kanan pilih *Create* > 2D > *Physical Material* 2D lalu berinama "*Player*".



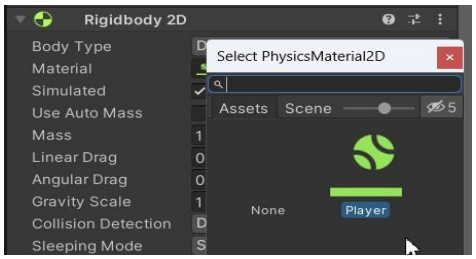
Gambar 8.14 Menambahkan Physical Material 2D

21. Klik *Physical Material* 2D yang baru dibuat, lalu pada menu *inspector* beri nilai 0 untuk *Friction* dan *Bounciness*.



Gambar 8.15 Merubah Nilai *Friction*

22. Klik *Hierarchy Player* lalu pada *Rigidbody* 2D pada *Material* pilih *asset Player* yang telah dibuat tadi.

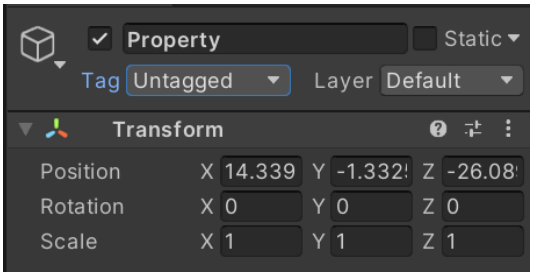


Gambar 8.16 Merubah Material

23. Lalu lakukan *test* apakah karakter dapat melompat Ketika di tekan *space* pada *keyboards*.

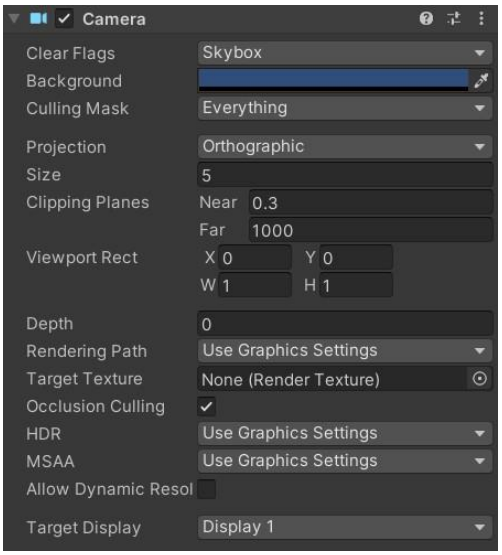
Gambar 8.17 Melakukan Test Melompat

24. Lalu untuk *Camera Movement* klik *Hierarchy Property* lalu pada *inspector* ubah *tag* menjadi *untagged*.



Gambar 8.18 Merubah Tag

25. Buat *Hierarchy* baru dengan nama "*Camera*" kemudian tambahkan *component camera*, lalu sesuaikan *setting*-nya.



Gambar 8.19 Mengatur Setting Camera

26. Kemudian buat *file script* baru pada *folder script* lalu beri nama "*CameraFollow*", kemudian masukkan *code* berikut pada *script*.

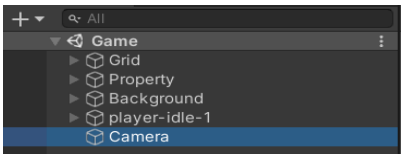
```
    using System.Collections;
    using
    System.Collections.Generic;
    using UnityEngine;

    public class CameraFollow : MonoBehaviour
    {
    public float xMargin = 0.5f;
    public float yMargin = 0.5f;
    public float xSmooth = 4f;
    public float ySmooth = 4f;
    public Vector2 maxXAndY; public
    Vector2 minXAndY; private
    Transform player;
    void Awake()
{
player      =
GameObject.FindGameObjectWithTag("Player").transform;
}

bool CheckXMargin()
{
```

```
return      Mathf.Abs(transform.position.x      -
   player.position.x) > xMargin;
}
bool CheckYMargin()
{
return      Mathf.Abs(transform.position.y      -
   player.position.y) > yMargin;
}
void FixedUpdate()
{
TrackPlayer();
}
void TrackPlayer()
{
float targetX = transform.position.x; float targetY =
   transform.position.y; if (CheckXMargin())
targetX = Mathf.Lerp(transform.position.x,
   player.position.x,
xSmooth * Time.deltaTime); if (CheckYMargin())
targetY = Mathf.Lerp(transform.position.y,
   player.position.y,
ySmooth * Time.deltaTime);
targetX = Mathf.Clamp(targetX, minXAndY.x, maxXAndY.x);
   targetY =
Mathf.Clamp(targetY,    minXAndY.y, maxXAndY.y);
   transform.position = new
Vector3(targetX,  targetY, transform.position.z);
}
   }
```

Analisa

Pada *code* diatas merupakan dasar untuk kamera 2D yang mengikuti pergerakan dari karakter dengan batas yang ditentukan. Menggunakan interpolasi *linear* untuk membuat pergerakan kamera yang halus dan memastikan pergerakan kamera tdak diluar batas yang telah ditentukan.

27. Kemudian *drag & drop file script* "*CameraFollow*" ke dalam *Hierarchy camera*.



Gambar 8.21 Meletakkan Script CameraFollow

28. Pada bagian *inspector camera* ubah nilai dari *Max X and Y* dan *Min X and Y*.

Gambar 8.22 Merubah Nilai Max dan Min

29. Lalu rubah *tag* pada *Hierarchy Player* menjadi "*Player*".

Gambar 8.23 Merubah Tag

30. Lalu jalankan *game* dengan menekan *play* untuk menguji hasil dari *Camera Movement.*

Gambar 8.24 Melakukan Uji Camera

2 Kuis CameraFollow

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraFollow : MonoBehaviour
{
[SerializeField] private Transform player;
void Update()
{
transform.position = new Vector3(player.position.x,
transform.position.y, transform.position.z);
}
}
```

Analisa

Pada *code* diatas Scrip *CameraFollow* pada Unity ini digunakan untuk membuat kamera mengikuti posisi pemain pada sumbu x (horizontal), sementara mempertahankan posisi y dan z kamera tetap. Dengan menggunakan *SerializeField* untuk *Transform player*, objek pemain diatur melalui editor Unity. Pada metode *Update*(), setiap frame, posisi x kamera diubah agar sesuai dengan posisi x pemain, namun posisi y dan z kamera tetap tidak berubah. Ini menghasilkan efek kamera yang mengikuti gerakan horizontal pemain tanpa mengubah ketinggian atau kedalaman pandangan kamera.