TUGAS PERTEMUAN: 8

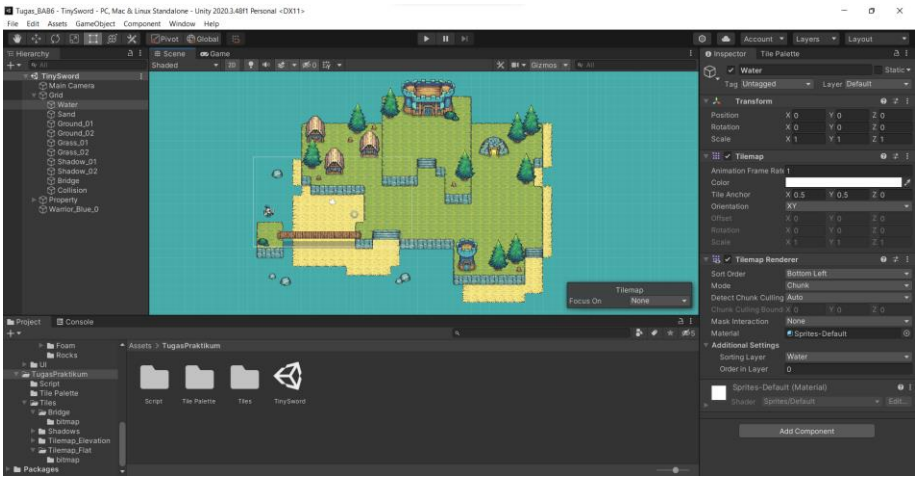
CAMERA & CHARACTER MOVEMENT

NIM	:	2118064
Nama	:	Ilham Maulana Prasetyo
Kelas	:	B
Asisten Lab	:	Bagas Anardi Surya W (2118004)

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

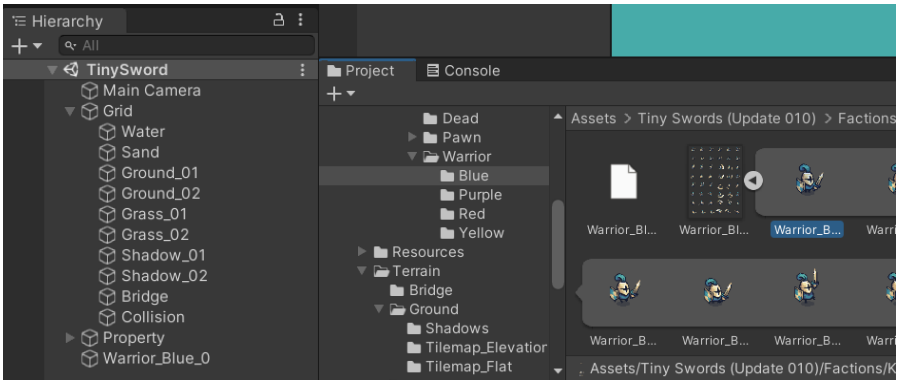
A. Membuat Character Movement, Detect Ground, & Jumping

1. Buka Project Unity Tugas Bab 7 sebelumnya.



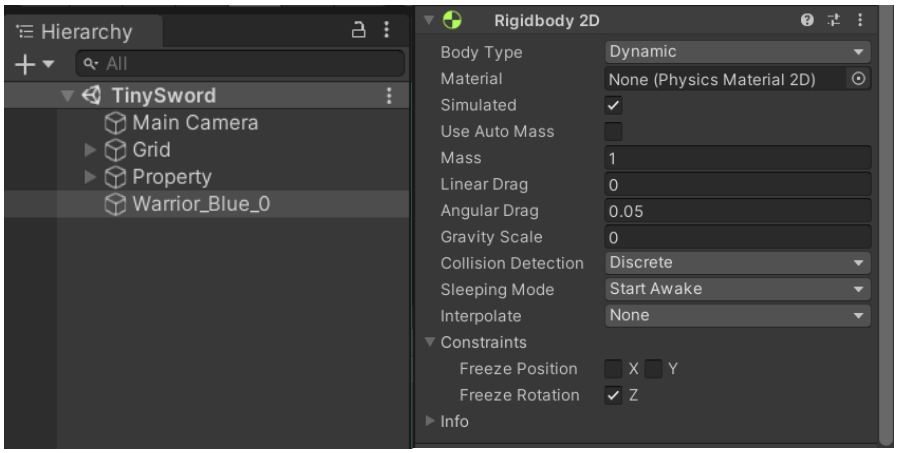
Gambar 8.1 Membuka Project Unity

2. Tambahkan karakter player bernama Warrior_Blue_0, kemudian import ke dalam Hirarki.



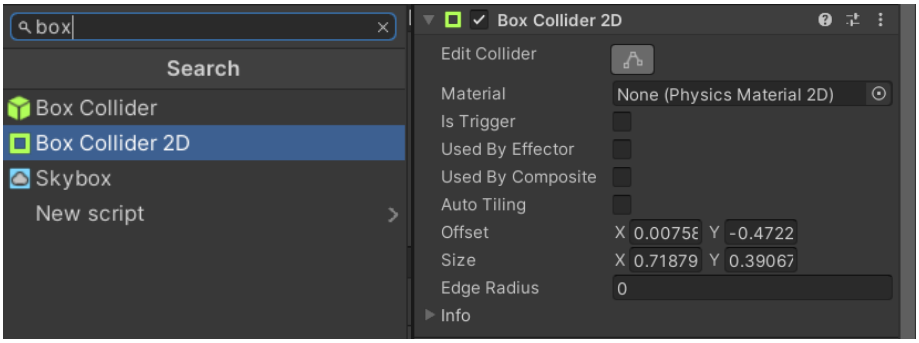
Gambar 8.2 Import Karakter Player

3. Klik Warrior_Blue_0 tambahkan komponen Rigidbody 2D, ubah Gravity Scale menjadi 0, lalu centang pada Freeze Rotation Z.



Gambar 8.3 Menambahkan RigidBody 2D

4. Kemudian tambahkan komponen Box Collider 2D di Warrior_Blue_0, lalu klik icon di sebelah kanan Edit Collider untuk mengatur ukurannya.

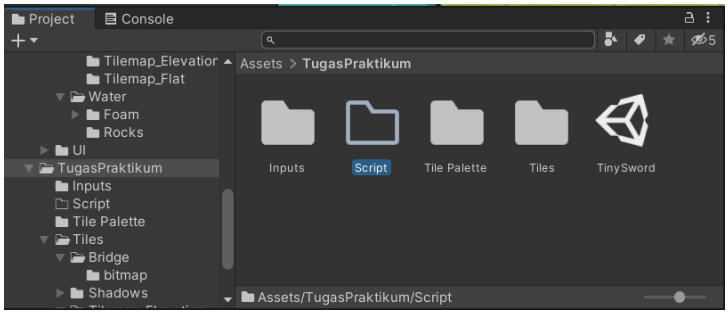


Gambar 8.4 Menambahkan Box Collider 2D

5. Lalu sesuaikan garis persegi dengan bagian kaki karakter.

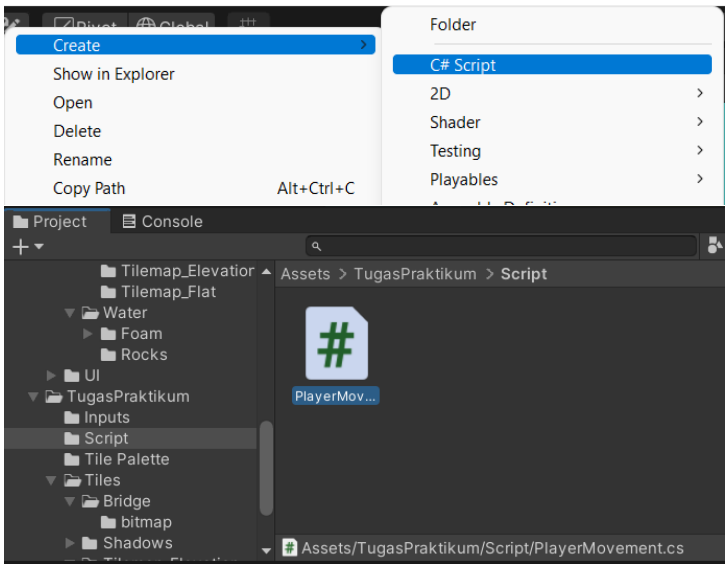
Gambar 8.5 Menyesuaikan Ukuran Box Collider 2D

6. Buat folder baru bernama “Script” di folder TugasPraktikum.



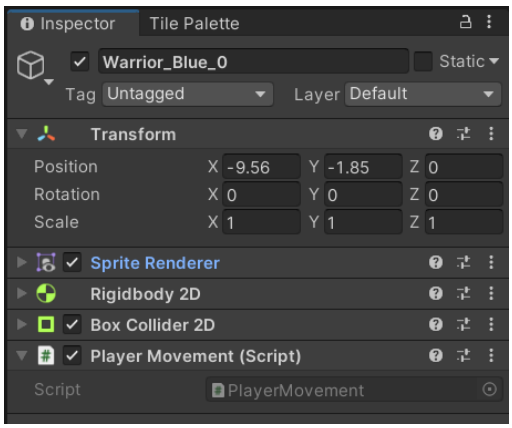
Gambar 8.6 Membuat Folder Script

7. Kemudian masuk kedalam folder Script, lalu klik kanan, Create > C# Script, beri nama “PlayerMovement”.



Gambar 8.7 Membuat Script PlayerMovement

8. Drag & drop script PlayerMovement kedalam Hirarki Warrior_Blue_0, kemudian klik 2x pada script PlayerMovement maka akan masuk kedalam text editor.

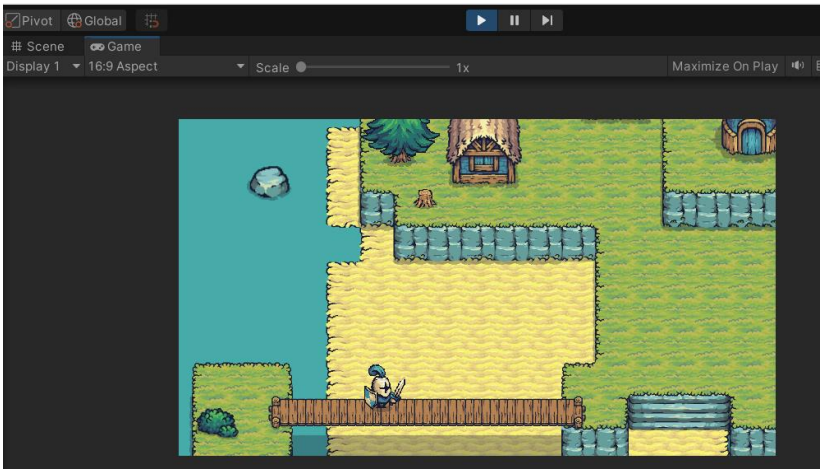


Gambar 8.8 Drag & Drop Script PlayerMovement

9. Masukan source code dibawah ini dan pastikan nama public class harus sama dengan nama file yang dibuat.

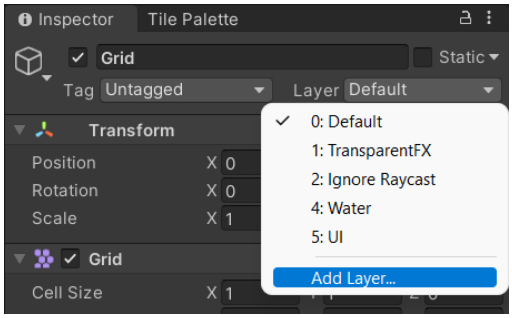
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
  Rigidbody2D rb;

  [SerializeField] float speed = 3;
  float horizontalValue;
  float verticalValue;
  bool facingRight;

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

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

  void FixedUpdate()
  {
    Move(horizontalValue, verticalValue);
  }

  void Move(float horizontalDir, float verticalDir)
  {
    #region gerak atas bawah kiri kanan
    float xVal = horizontalDir * speed * 100 *
Time.fixedDeltaTime;
    float yVal = verticalDir * speed * 100 *
Time.fixedDeltaTime;
    Vector2 targetVelocity = new Vector2(xVal, yVal);
    rb.velocity = targetVelocity;

    if (facingRight && horizontalDir < 0)
    {
        //ukuran karakter player
        transform.localScale = new Vector3(-1, 1, 1);
        facingRight = false;
    }
    else if (!facingRight && horizontalDir > 0)
    {
        //ukuran karakter player
        transform.localScale = new Vector3(1, 1, 1);
        facingRight = true;
    }
    #endregion
  }
}
```

10. Untuk mengecek apakah source code diatas berfungsi, play game, tekan keyboard “W” untuk ke arah atas dan tekan “W” untuk ke arah bawah. Tekan keyboard “A” untuk ke arah kiri dan tekan “D” untuk ke arah kanan.



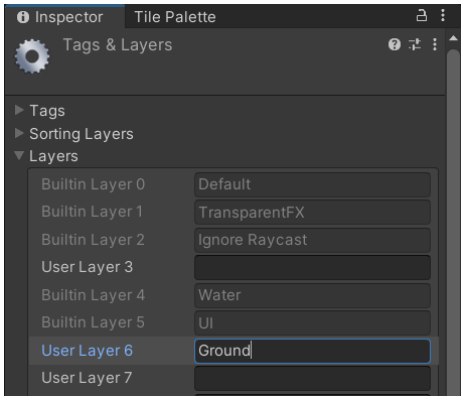
Gambar 8.9 Menguji Fungsi Pergerakan Karakter

11. Untuk membuat player dapat meloncat menggunakan spasi, buat GorundCheck dengan cara, klik Grid pada Hierarchy, pergi ke Inspector, pilih Layer, Klik Add Layer.



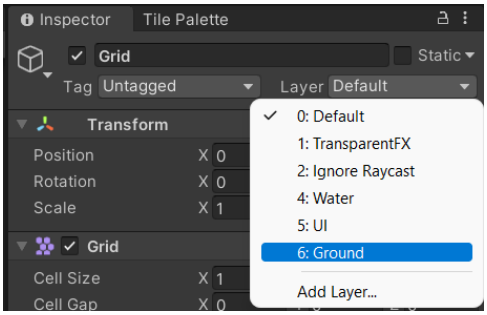
Gambar 8.10 Menambahkan Layer

12. Kemudian isi “Ground” pada User Layer 6.



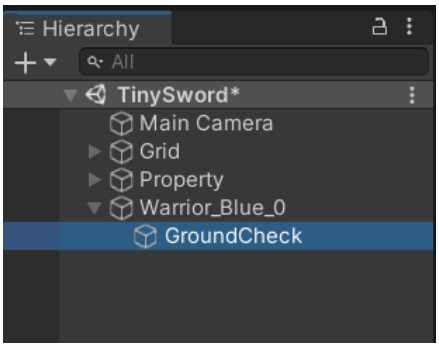
Gambar 8.11 Membuat User Layer Ground

13. Ubah Layer menjadi Ground.



Gambar 8.12 Mengubah Layer Pada Grid

14. Klik kanan pada Warrior_Blue_0, lalu Create Empty, beri nama GroundCheck.



Gambar 8.13 Membuat GroundCheck

15. Klik GroundCheck pada Hirarki, lalu gunakan "Move Tools" untuk memindahkan ke bagian bawah Player seperti gambar berikut.

Gambar 8.14 Memindahkan GroundCheck

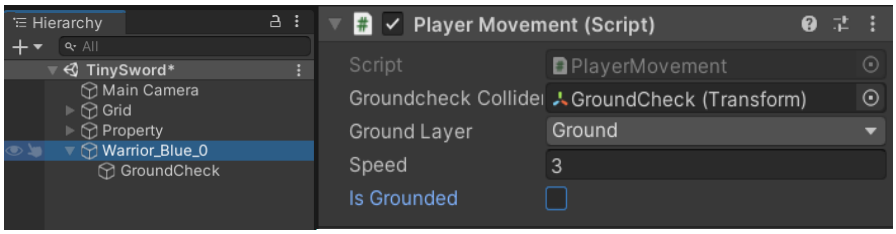
16. Kembali ke script PlayerMovement, tambahkan source code berikut.

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

const float groundCheckRadius = 0.2f; // +
[SerializeField] float speed = 3;
float horizontalValue;
float verticalValue;

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

17. Buat void GroundCheck dibawah void FixedUpdate & tambahkan GroundCheck(); pada void FixedUpdate.

```
void Update()
{
  GroundCheck(); // +
  horizontalValue = Input.GetAxisRaw("Horizontal");
  verticalValue = Input.GetAxisRaw("Vertical");
}

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

18. Klik Warrior_Blue_0, pada Inspector komponen PlayerMovement script, ubah "Grouncheck Collider" caranya tekan icon lingkaran lalu pilih yang GroundCheck Transform, dan pada Ground Layer pilih Ground.



Gambar 8.15 Mengubah Groundcheck Collider

19. Lalu untuk membuat player melompat tambahkan script berikut.

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

const float groundCheckRadius = 0.2f; // +
[SerializeField] float speed = 3;
[SerializeField] float jumpPower = 100; // ++
float horizontalValue;
float verticalValue;

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

20. Tambahkan script berikut di bagian void Update.

```
void Update()
{
  GroundCheck(); // +
  horizontalValue = Input.GetAxisRaw("Horizontal");
  verticalValue = Input.GetAxisRaw("Vertical");
```

```
  if (Input.GetButtonDown("Jump")) // ++
    jump = true;
  else if (Input.GetButtonUp("Jump"))
    jump = false;
}
```

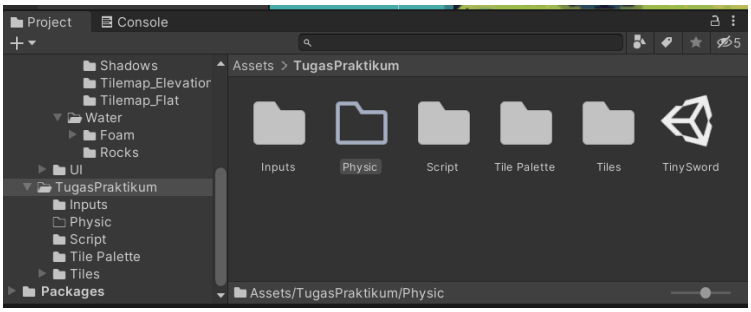
21. Tambahkan juga jump pada parameter Move.

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

22. Lalu tambahkan juga script berikut pada void Move.

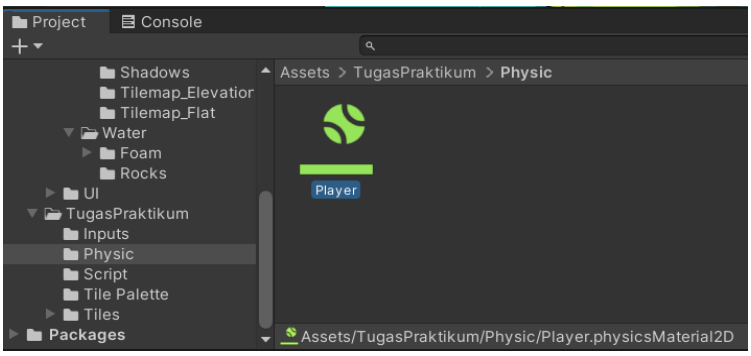
```
void Move(float horizontalDir, float verticalDir, bool
jumpflag)
{
  if(isGrounded && jumpflag)
  {
    isGrounded = false;
    jumpflag = false;
    rb.AddForce(new Vector2(0f, jumpPower));
  }
  else if (!isGrounded && jumpflag)
  {
    jumpflag = false;
    rb.AddForce(new Vector2(0f, jumpPower));
  }
```

23. Buat folder baru dengan nama “Physics” di folder TugasPraktikum.



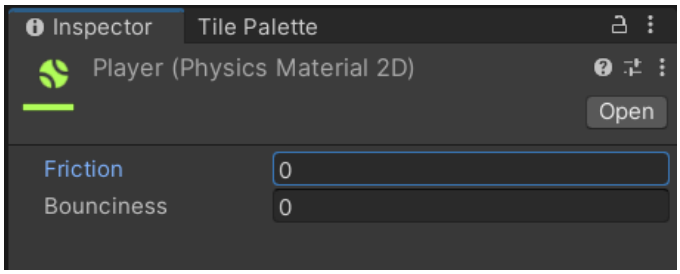
Gambar 8.16 Membuat Folder Physics

24. Kemudian masuk kedalam folder Physics, lalu klik kanan pilih Create > 2D > Physical Material 2D, beri nama “Player”.



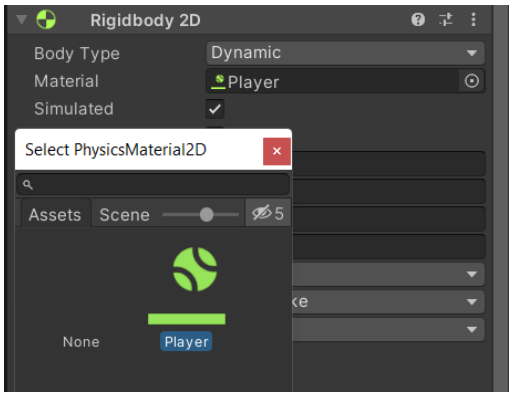
Gambar 8.17 Membuat Physics Material 2D

25. Kemudian klik Player (Physic Material 2D), lalu pada Inpector ubah Friction & Bounciness mencadi 0.



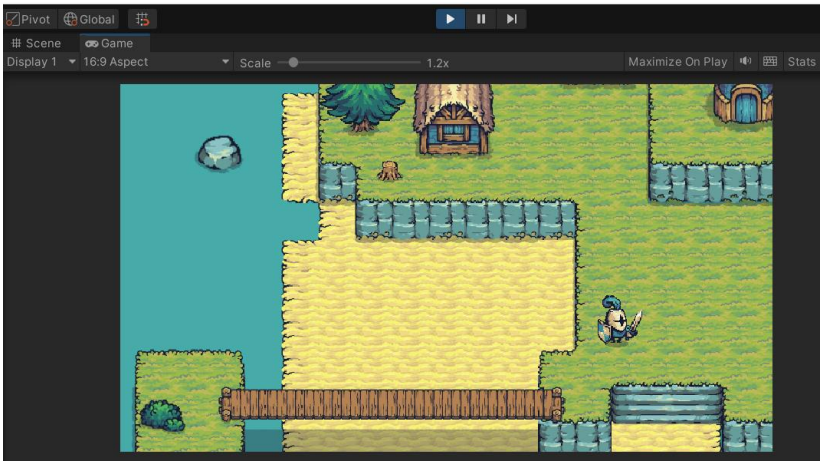
Gambar 8.18 Mengubah Nilai Friction & Bounciness

26. Klik Warrior_Blue_0 pada Hirarki, pada Inspector cari Rigidbody 2D lalu klik icon pada bagian Material untuk membuka box select Physhics Material 2D , lalu pilih asset Player yang sudah di buat sebelumnya.



Gambar 8.19 Menambahkan Physhics Material 2D Player

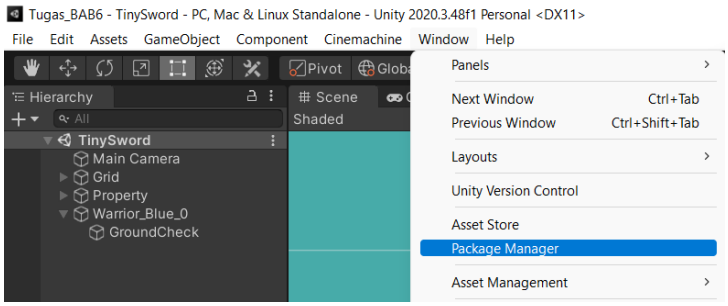
27. Tekan play, maka player bisa melompat dengan menekan spasi.



Gambar 8.20 Tampilan Hasil Character Movement

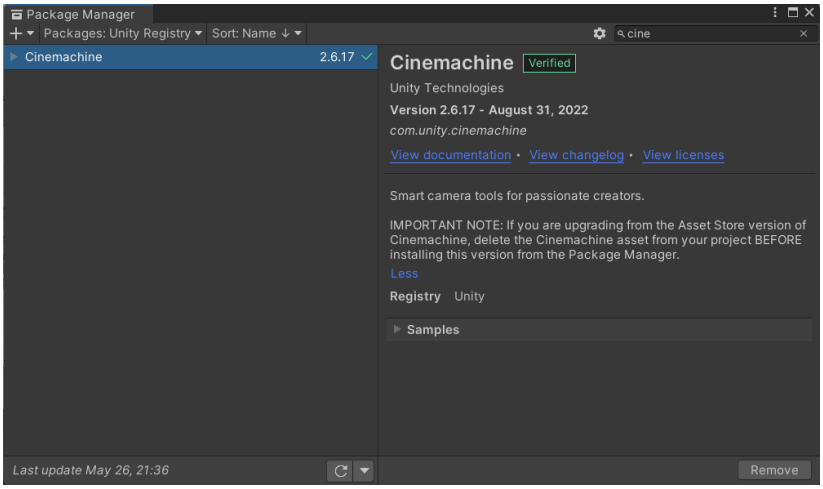
B. Membuat Camera Movement

1. Pada menu atas, klik menu Window pilih Package Manager.



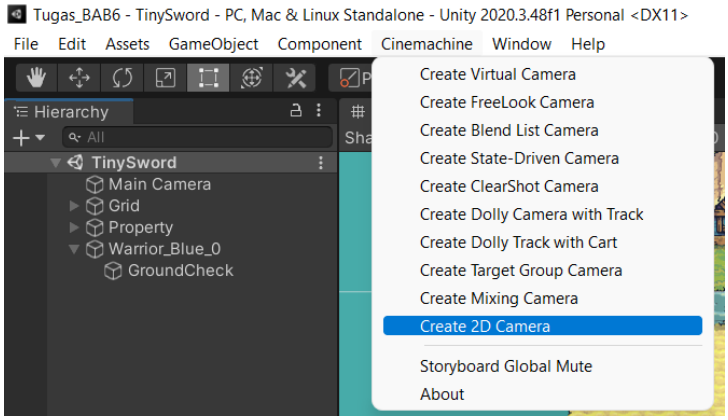
Gambar 8.21 Membuka Package Manager

2. Ketika windows Package Manager terbuka, pada Packages pilih Unity Registry, lalu cari Cinemachine dan install.



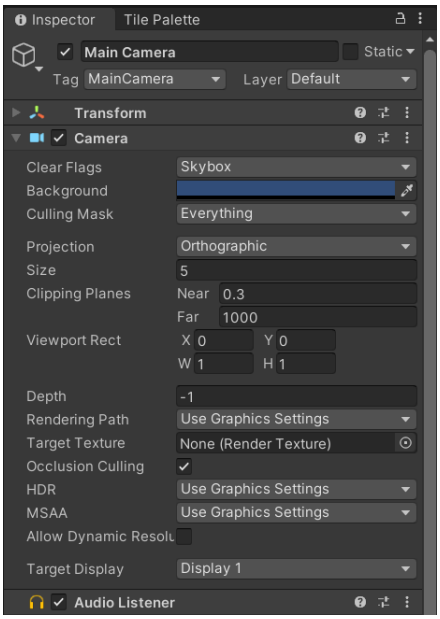
Gambar 8.22 Install Cinemachine

3. Setelah selesai install Cinemachine, akan muncul menu Cinemachine dibagian atas. Klik Cinemachine pilih Create 2D Camera untuk membuat Camera Follow.



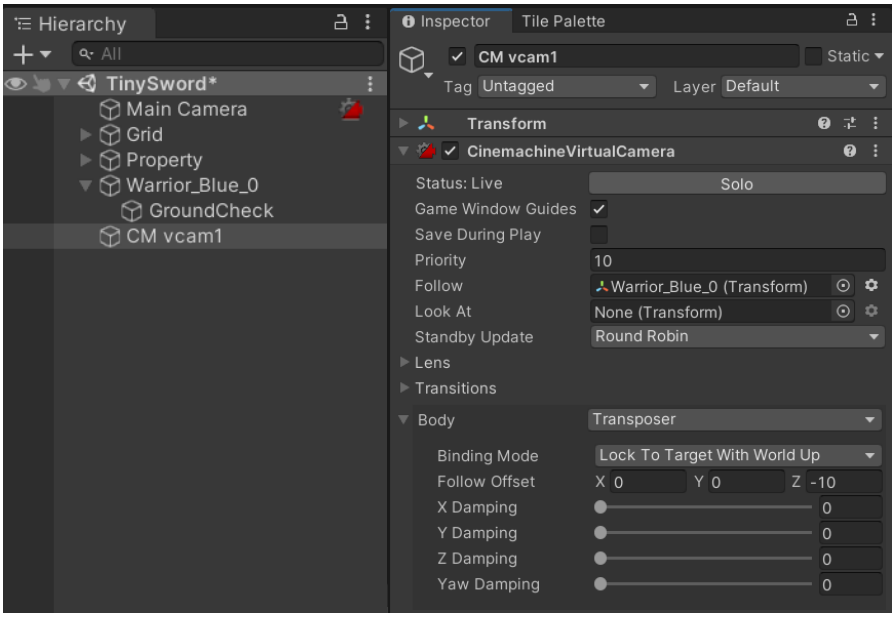
Gambar 8.23 Membuat Camera Follow

4. Pada Hirarki pilih Main Camera, pada Inspector komponen Camera ubah Projection menjadi Orthographic.



Gambar 8.24 Setting Komponen Camera

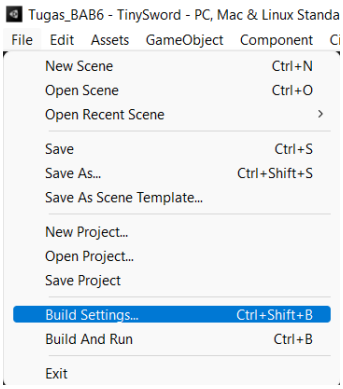
5. Pada Hirarki pilih CM vcam1, pergi ke Inspector komponen CinemachineVirtualCamera, pada bagian Follow pilih Player dan ubah nilai damping pada bagian Body menjadi 0.



Gambar 8.25 Setting CM vcam1

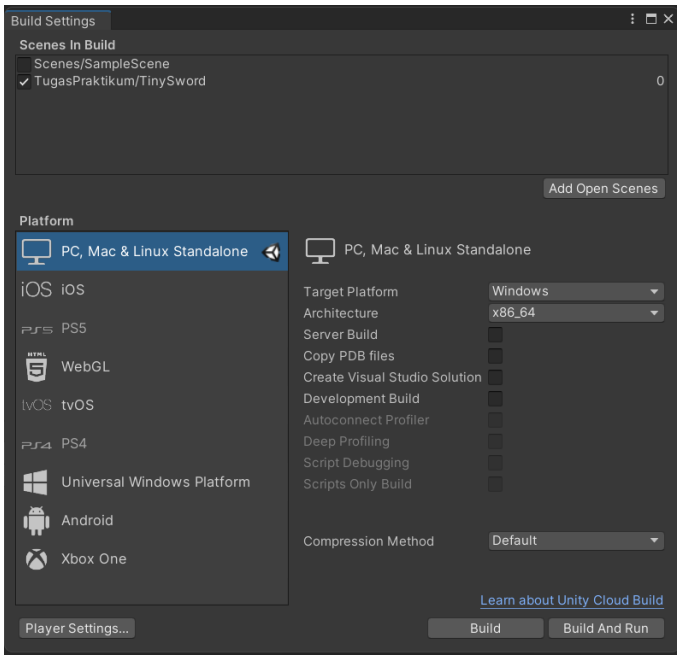
C. Render

1. Pergi ke menu File kemudian pilih Build Settings (Ctrl + Shift + B).



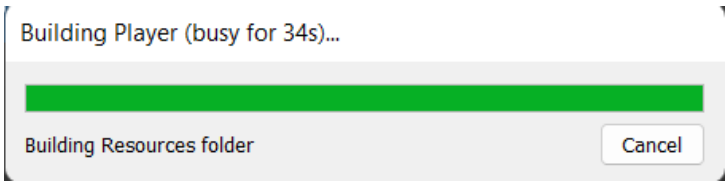
Gambar 8.26 Menu Build Setting

2. Pada Build Settings pilih PC, Mac & Linux, pastikan pada menu Scene in Build berada pada project Tugas, lalu Tekan Build.



Gambar 8.27 Tampilan Build Settings

3. Pilih dimana Project disimpan, dan tunggu hasilnya.



Gambar 8.28 Tampilan Proses Menyimpan

4. Lalu pilih poject yang sudah di render, klik 2x untuk melihat hasilnya.

Gambar 8.29 Tampilan Hasil Project

D. Link Github

https://github.com/ilham-pras/2118064_PRAK_ANIGAME.git

8.2 Kuis

Menjelaskan source code dibawah ini:

```
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:

Source code diatas berfungsi untuk membuat kamera mengikuti pergerakan pemain dalam game. Pada source code mendeklarasikan variabel player bertipe Transform sebagai private, variabel player diatur menggunakan atribut [SerializeField] yang memungkinkan variabel ini diatur melalui Inspector di Unity Editor tanpa menjadikannya publik.

Di dalam metode Update, komponen x dari posisi kamera diubah untuk mengikuti pergerakan pemain di sepanjang sumbu x, sementara komponen y dan z dari posisi kamera tetap sama. Yang artinya kamera hanya mengikuti pemain bergerak secara horizontal.