# 1 MEMBUAT KARAKTER

|  |  |  |
| --- | --- | --- |
| **NIM** | : | 1918027 |
| **Nama** | : | Dirgarianda Reydjavic Christian Rumabar |
| **Kelas** | : | D |
| **Asisten Lab** | : | M. Rafi Faddilani (2118144) |
| **Baju Adat** | : |  |
| **Referensi** | : |  |

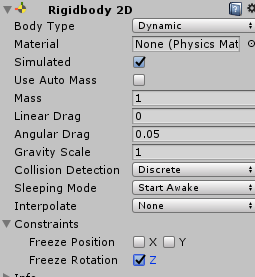
## Langkah-Langkah Membuat Tugas

1. **Membuat Pergerakan Player**
2. Buat file *projek Unity* Tugas 7



### 8.1 Buka projek tugas 7

1. Pada properties *Rigidbody 2D*, klik *Constraint*, dan centang pada bagian Z, *Freeze* *Rotation*, kotak tersebut dicentang agar saat karakter menginjak tanah tidak terguling



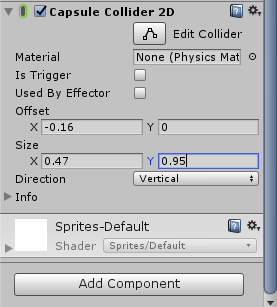
### 8.2 Tampilan Freeze Rotation

1. Tambahkan lagi satu komponen bernama *Capsule Collider 2D*



### 8.3*Capsule Collider 2D*

1. Pada *properties Capsule Collider 2D*, isikan nilai *Offset* dan *Size*



### 8.4 Mengatur nilai *Offset* dan *Size*

1. Klik kanan pada asset lalu buat folder baru dan beri nama *Script*



### 8.5 Membuar folder *Scirpt*

1. Pada folder *script* klik kanan pilih *C# script*

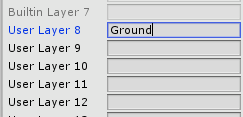


### 8.6 Membuat script untuk warrior

1. Lalu pada script karakter masukkan Source Berikut

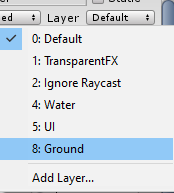
|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class karakter : MonoBehaviour  {  Rigidbody2D rb;  [SerializeField]  Transform groundcheckCollider;  [SerializeField]  LayerMask groundLayer;  const float groundCheckRadius = 0.2f;  [SerializeField]  bool isGrounded;  [SerializeField]  float speed = 1;  [SerializeField]  float jumpPower = 100;  float horizontalValue;  bool facingRight;  bool jump;  private void Awake()  {  rb = GetComponent<Rigidbody2D>();  }  // 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);  }    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  }  } |

1. Isikan *User Layer* 8 dengan nama *Ground*



### 8.7 Mengisi *layer 8* menjadi *Ground*

1. Ganti *Layer Default* menjadi *Ground*



### 8.8 Menggati layer menjadi *Ground*

1. Klik kanan karakter kemudian pilih *Create Empty* dan ubah namanya menjadi *GroundCheck*



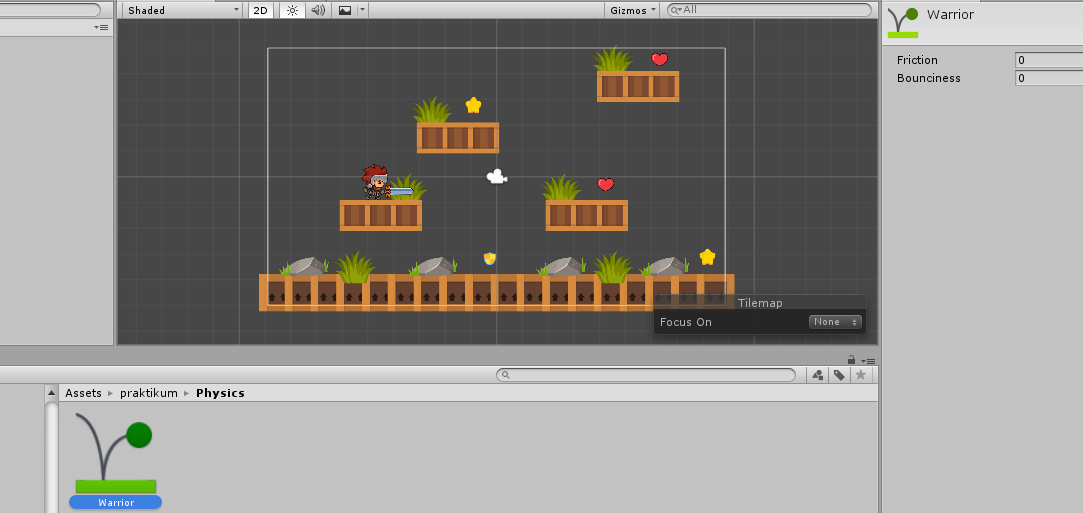
### 8.9 Membuat *GroundCheck*

1. drag *GroundCheck* arahkan kedalam *Groundcheck* Collider yang ada pada *Inspector*



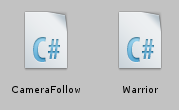
### 8.10 Penyesuaian *GroundCheck*

1. Klik kanan folder “*Physics*” pilih *Create>Physics Material 2D* dan beri nama “karakter” lalu ubah nilai Friction dan Bouncies menjadi 0



### 8.11 Tampilan *Physics* karakter

1. **Camera Movement**
2. Tambahkan script dalam folder Script dan beri nama *“CameraFollow.cs”*



### 8.12 Membuat dan Menambahkan *Source Code*

1. Tambahkan *Source Code* berikut pada Script *CameraFollow*

|  |
| --- |
| 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);  }  } |

1. Pergi ke *Inspector Main* Camera kemudian lakukan setting *Camera Follow* seperti gambar berikut



### 8.13 Mensetting *Camera Follow*

1. Jika di *Play*, maka camera akan mengikuti pergerakan setiap karakter.



### Tampilan game ketika di Run

1. **KUIS BAB 8**

**CameraFollow**

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

Penjelasan :

Pertama-tama, pustaka System.Collections dan System.Collections.Generic diimpor untuk mendukung penggunaan koleksi data. Pustaka UnityEngine juga diimpor untuk memanfaatkan fungsionalitas yang disediakan oleh Unity.

Kelas CameraFollow diturunkan dari kelas MonoBehaviour, yang memungkinkan kelas ini untuk digunakan sebagai skrip pada objek di dalam Unity. Dalam kelas ini, terdapat variabel privat player dari tipe Transform yang diberi atribut [SerializeField], yang berarti variabel ini dapat diatur nilainya melalui editor Unity tanpa harus menjadikannya publik.

Fungsi Update dipanggil setiap frame oleh Unity. Di dalam fungsi ini, posisi kamera diatur ulang sehingga sumbu x kamera mengikuti posisi x pemain, sementara sumbu y dan z kamera tetap konstan. Hal ini dicapai dengan menggunakan konstruktor Vector3 untuk membuat vektor posisi baru untuk kamera, di mana komponen x diambil dari posisi x pemain, dan komponen y serta z diambil dari posisi kamera saat ini.