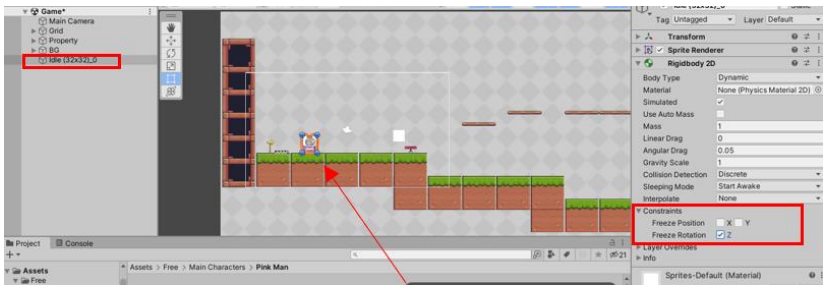
TUGAS PERTEMUAN: 8

Camera & Character Movement

NIM	:	2118101
Nama	:	Marie Pangestu
Kelas	:	C
Asisten Lab	:	Rifal Rifqi Rhomadon (2218106)

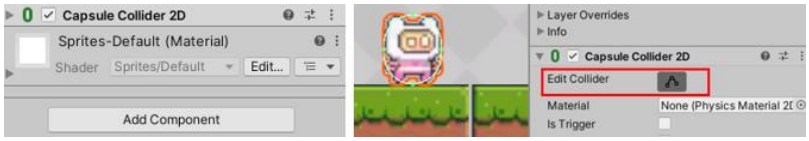
1.1 Tugas 1 : Membuat Character Movement

1. Buka project bab 7, kemudian *drag* and *drop* karakter yang akan dipakai kedalam *project* unity, jika tidak bisa bisa di *slice* dan sesuaikan *pixel* terlebih dahulu. Lalu tekan karakter pada *hierarchy*>*inspector*>add component 'Rigidbody 2D'>centang *freeze rotation* Z.



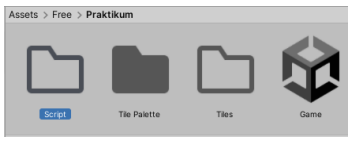
Gambar 8.1 *Checklist Freeze Rotation* Z

2. Pada inspector karakter, tambahkan komponen 'Capsule Collider 2D' lalu tekan tombol '*edit collider*' untuk memunculkan bulatan di area karakter lalu sesuaikan ukuran lingkaran dengan ukuran karakter.



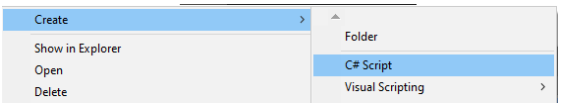
Gambar 8.2 Add Component Capsule Collider 2D On Character

3. Tambahkan folder baru didalam folder Praktikum dengan nama 'Script'.



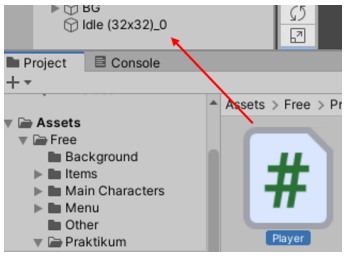
Gambar 8.3 New Folder Script

4. Tambahkan script baru didalam folder script dengan nama ‘Player’.



Gambar 8.4 New Player Script

5. Drag and drop script player ke karakter pada hierarchy dan tanda jika script berhasil ditambahkan pada karakter akan muncul pada inspector.



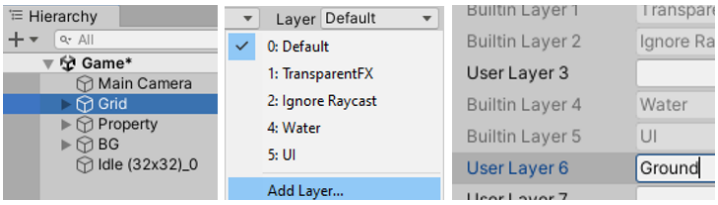
Gambar 8.5 Drag and Drop Script Player

6. Masukkan source code berikut ke dalam script player

```
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)
```

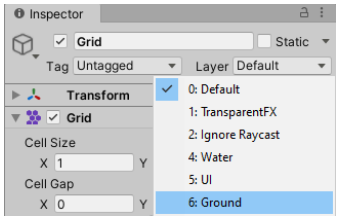
```
    {
      // ukuran player
      transform.localScale = new Vector3(1, 1, 1);
      facingRight = true;
    }
    #endregion
  }
}
```

7. Tekan grid pada hierarchy, beralih pada inspector tekan layer>add layer>beri nama layer baru ‘Groud’.



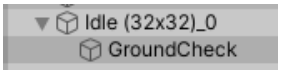
Gambar 8.6 New Layer Ground

8. Kemudian ubah layer grid dari default ke ground dan akan muncul message box tekan ‘yes. Change the children’.



Gambar 8.7 Ubah Layer Grid Default ke Ground

9. Kemudian klik kanan pada karakter di hierarchy > create empty ‘GroundCheck’.



Gambar 8.8 Create Empty GroundCheck

10. Tekan komponen GroundCheck, lalu tekan movetool dan ubah posisi panah hijau dan merah seperti pada gambar pada karakter.

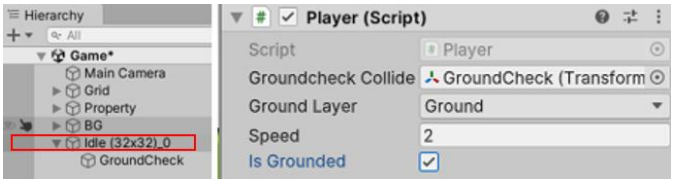
Gambar 8.9 Move GroundCheck

11. Tambahkan source code ber-highlight kuning pada code sebelumnya untuk membuat fungsi agar karakter berpijak pada tanah/latar.

```
[SerializeField] Transform groundcheckCollider;
  [SerializeField] LayerMask groundLayer;
  const float groundCheckRadius = 0.2f; // +
  [SerializeField] float speed = 1;
  float horizontalValue;
  [SerializeField] bool isGrounded; // +
  bool facingRight;
```

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

12. Tekan karakter pada hierarchy, pada inspector ubah groundcheck dan layer script seperti pada gambar dan isi checklist serta speed sesuai keinginan.



Gambar 8.10 Hasil Slice Asset

13. Tambahkan source code bermarka kuning untuk membuat fungsi melompat menggunakan space pada keyboard.

```
    const float groundCheckRadius = 0.2f; // +
    [SerializeField] float speed = 1;
    [SerializeField] float jumpPower = 100;

    float horizontalValue;
    [SerializeField] bool isGrounded; // +

    bool jump;
    bool facingRight;
```

```
  void Update ()
  {
    horizontalValue = Input.GetAxisRaw("Horizontal");
    if (Input.GetButtonDown("Jump"))
    jump = true;
    else if (Input.GetButtonUp("Jump"))
    jump = false;
  }
```

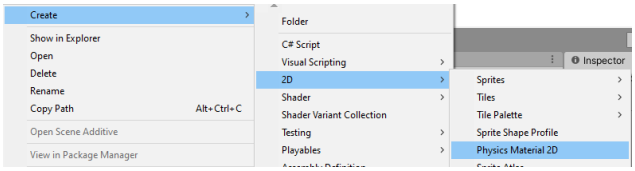
```
void FixedUpdate()
{
  GroundCheck();
  Move(horizontalValue, jump);
}
```

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

14. Buat folder baru didalam folder Praktikum dengan nama ‘Physics’.

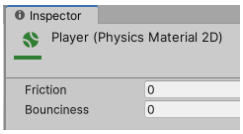
Gambar 8.11 New Folder Physics

15. Masuk kedalam folder physics, buat physics material 2D dengan nama ‘Player’.



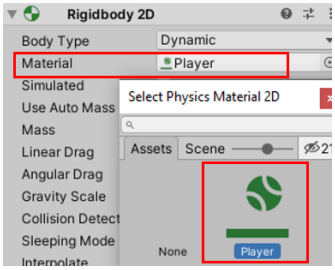
Gambar 8.12 New Physics Material 2D Player

16. Kemudian ubah nilai friction pada inspector menjadi ‘0’.



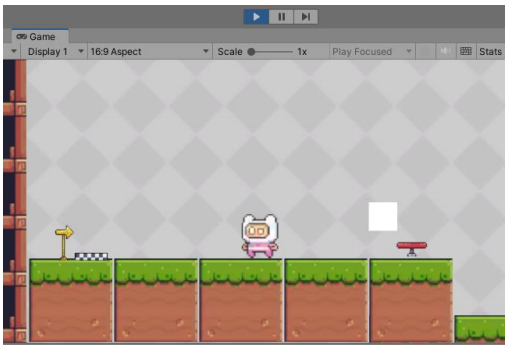
Gambar 8.13 Ubah Nilai Friction

17. Tekan karakter pada hierarchy > inspector > rigidbody 2D > ubah material dari none ke player yang sudah dibuat sebelumnya.



Gambar 8.14 Ubah Material Rigidbody 2D Karakter

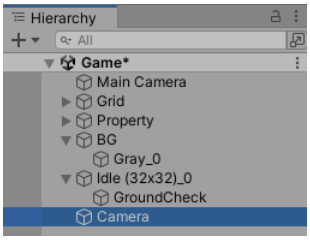
18. Uji coba play game apakah seluruh fungsi yang ditambahkan bisa berjalan dengan benar.



Gambar 8.15 Hasil Tampilan Character Movement

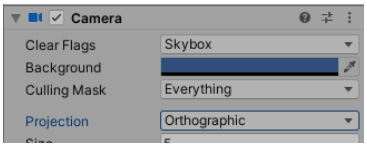
1.2 Tugas 2 : Camera Movement

1. Klik kanan pada hierarchy > create empty dengan nama ‘Camera’.



Gambar 8.16 Create Empty Camera

2. Beralih pada inspector camera, ubah projection menjadi ;Orthographic’.



Gambar 8.17 Projection Orthographic

3. Buat script baru pada folder script dengan nama ‘’CameraFollow’.

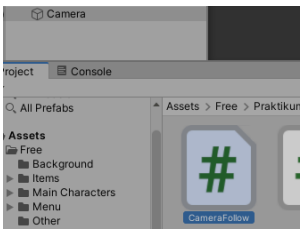
Gambar 8.18 New Script CameraFollow

4. Masukkan source code berikut ke dalam script CameraFollow.

```
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);
    }
}
```

5. Drag and drop script camerafollow pada hierarchy camera.

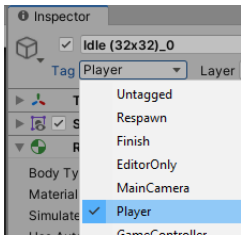


Gambar 7.19 Ubah Mask Interaction Background

6. Masuk ke inspector camera dan ubah nilai MAX X dan Y script camerafollow seperti pada gambar 8.20

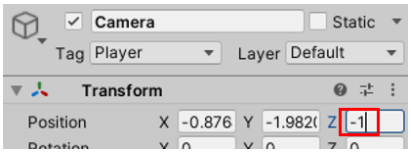
Gambar 8.20 Ubah Nilai Max X dan Y Script CameraFollow

7. Kemudian ubah tag karakter pada inspector menjadi player.



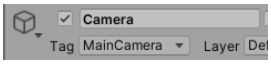
Gambar 8.21 Tag Character

8. Tekan camera pada hierarchy, ubah position transform seperti pada gambar.



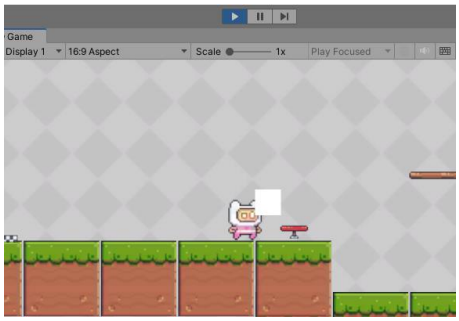
Gambar 8.22 Ubah Nilai Max X dan Y Script CameraFollow

9. Ubah tag camera menjadi main camera, dan hapus komponen main camera pada hierarchy yang sudah dibuat otomatis dari projectnya (agar camera tidak double).



Gambar 8.23 Tag Camera

10. Lalu play project dan gerakkan karakter untuk tes camera movement.



Gambar 8.24 Hasil Tampilan Camera Movement

1.3 Kuis

1. Source code Player

```
public class Player : MonoBehaviour
{
    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 jump;
    bool facingRight;

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

  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)
    {
```

```
        // 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
    }
}
```

Penjelasan :

Kode berikut dimulai dengan **Transform groundcheckCollider**: Referensi ke objek Transform yang digunakan untuk memeriksa apakah pemain menyentuh tanah dan layermask ground layer dimana layer yang diangkap tanah/pijakan dicek apakah karakter berada di tanah. Lalu nilai 0.2f untuk jarak lingkaran untuk pengecekan tanah. Dan pemberian kecepatan berjalan/horizontal pemain dengan nilai 1. Pemberian kekuatan lompatan yaitu 100. Float horizontal value untuk menyimpan nilai input pemain. bool isGrounded: Menyimpan status apakah pemain berada di tanah. bool jump: Menyimpan status apakah tombol lompat ditekan. bool facingRight: Menyimpan status apakah pemain menghadap ke kanan. Lalu saat dijalankan, fungsi awake akan dipanggil yang menyimpan komponen rigidbody2D. Lalu fungsi Update() yang akan dipanggil setiap karakter mengalami perubahan tiap frame dengan masukkan nilai HorizontalValue dan percabangan fungsi jump. Lalu fungsi FixedUpdate() untuk memanggil fungsi GroundCheck untuk memeriksa apakah pemain menyentuh tanah. Dan fungsi move dengan parameter nilai input horizontal dan status lompat.

2. Source code CameraFollow

```
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);
    }
}
```

Penjelasan :

Dilakukan deklarasi nilai pada sumbu x dan x untuk batas gerak kamera dan batas maksikmum gerak kamera. Lalu membuat variable privat untuk player dimana jika project dijalankan akan memanggil fungsi awake untuk memanggil game object “player” lalu dilakukan pada nilai X dan Y margin apakah posisi melebihi margin. Kemudian dijalankan fungsi FixedUpdate untuk mengupdate posisi kamera mengikuti player dengan memanggil fungsi trackplayer(). Dimana isi dari fungsi track player sendiri yaitu inisialisasi posisi x dan y dan terdapat percabangan jika posisi x melebihi margin maka hitung target x dan sebaliknya untuk nilai y. lalu batasi target x dan y agar tidak melebihi batas minimum dan maksimum dengan fungsi clamp. Dari nilai target x dan y terbentuknya posisi kamera terbaru.