TUGAS PERTEMUAN: 8

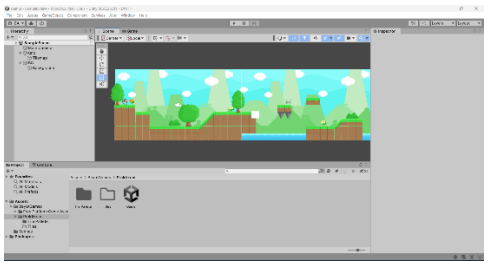
CAMERA & CHARACTER MOVEMENT

NIM	:	2118115
Nama	:	Ellok Ananda Madya Pratiwi
Kelas	:	C
Asisten Lab	:	Nayaka Apta Nayottama (2218102)

1.1 Tugas 1 : Camera dan Camera Movement

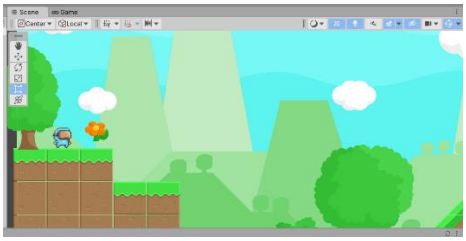
A. Membuat Pergerakan Player

1. Buka file project unity sebelumnya



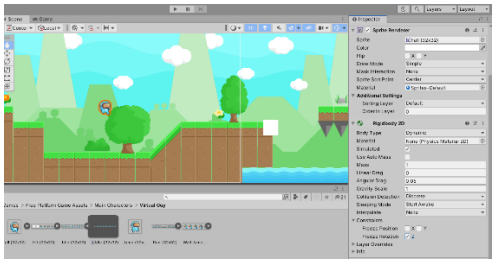
Gambar 8. 1 Membuka Projek Sebelumnya

2. Tambahkan player dengan nama idle dengan drag ke hirarki



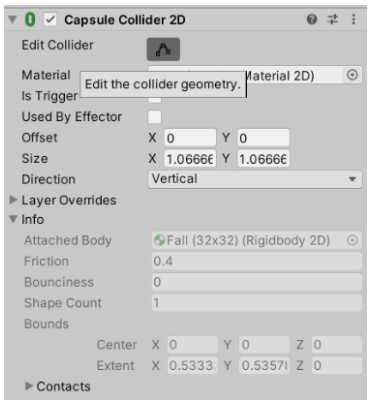
Gambar 8. 2 Menambahkan Player

3. Tambahkan component rigidbody pada inspector idle lalu centang pada freeze rotation



Gambar 8. 3 Menambahkan Component RigidBody

4. Tambahkan component dengan nama capsule collider lalu klik ikon di sebelah edit collider

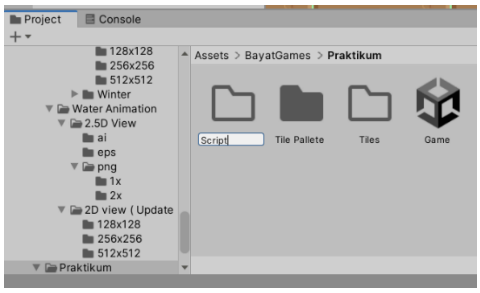


Gambar 8. 4 Menambahkan Component Capsule Collider

5. Sesuaikan posisi garis oval dengan karakter player

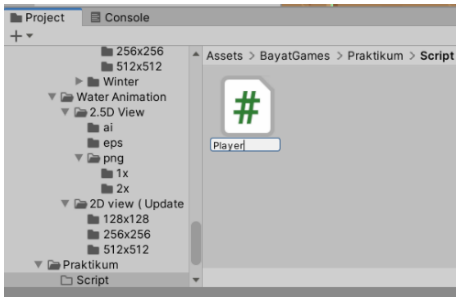
Gambar 8. 5 Menyesuaikan Karakter Dengan Garis Oval

6. Pada folder praktikum buat folder baru lalu beri nama Script



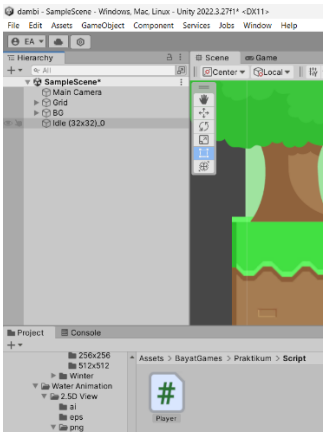
Gambar 8. 6 Membuat Folder Script

7. Dalam folder Script tersebut buat file C# dengan nama Player



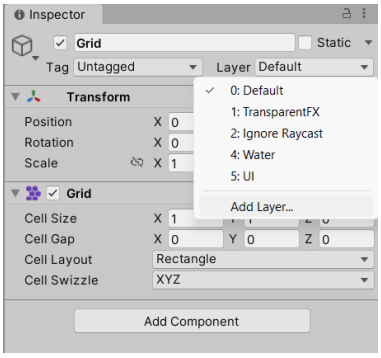
Gambar 8. 7 Membuat File Player

8. Drag file C# dengan nama player ke dalam hirarki Idle



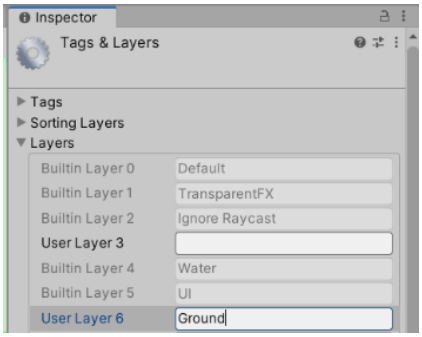
Gambar 8. 8 Meletakkan File Player Ke Hirarki

9. Pada hirarki klik grid lalu pada inspector bagian layer pilih Add Layer



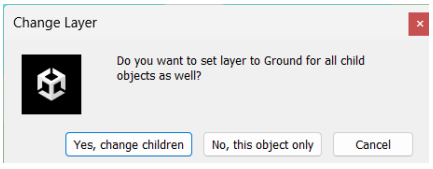
Gambar 8. 9 Menambahkan Layer

10. Pada User Layer 6 ketikkan Ground



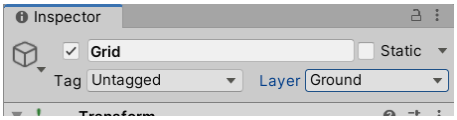
Gambar 8. 10 Memberi Nama Layer

11. Jika muncul pop up seperti ini klik yes



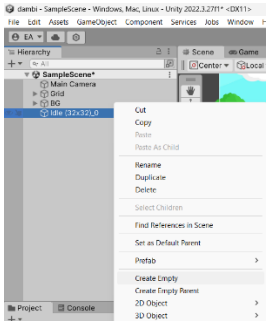
Gambar 8. 11 Tampilan Pop Up

12. Ubah layer menjadi ground, jika muncul pop up change layer klik yes



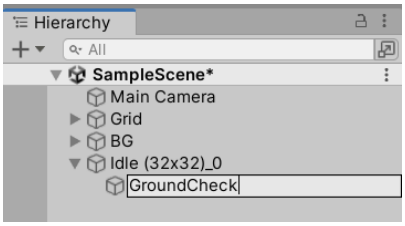
Gambar 8. 12 Mengubah Layer

13. Klik kanan pada idle lalu pilih create empty



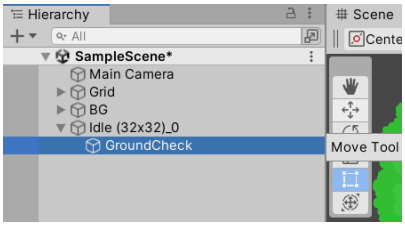
Gambar 8. 13 Menambahkan File Kosong Pada Hirarki

14. Ubah nama GameObject menjadi GroundCheck



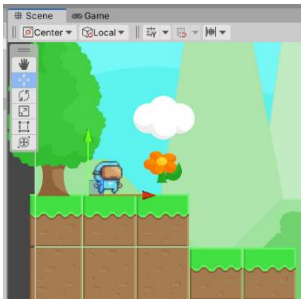
Gambar 8. 14 Mengubah Nama GameObject

15. Klik pada hirarki GroundCheck lalu pada scene klik move tool



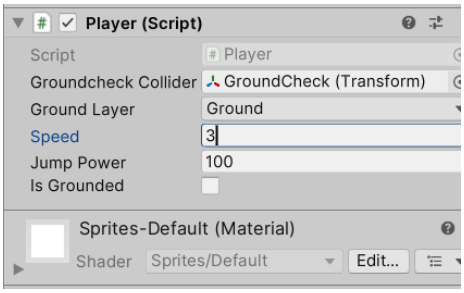
Gambar 8. 15 Meklik Hirarki GroundCheck

16. Posisikan karakter seperti gambar berikut



Gambar 8. 16 Memosisikan Karakter

17. Pada inspector GroundCheck bagian player, ubah Groundcheck Collider menjadi GroundCheck dan Ground Layer menjadi Ground. Lalu ubah speed menjadi 3

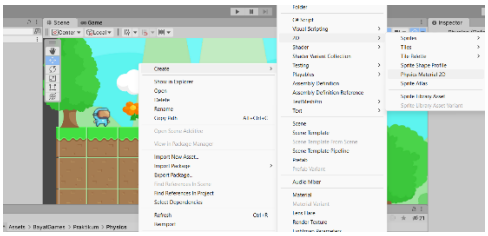


Gambar 8. 17 Mengatur Inspector GroundCheck

18. Buat folder baru dengan nama Physics pada folder Praktikum

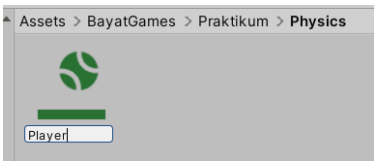
Gambar 8. 18 Membuat Folder Physics

19. Didalam folder Physics, create – 2D – Physical Material 2D



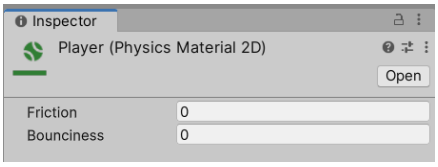
Gambar 8. 19 Menambahkan File Physical Material 2D

20. Beri nama Player



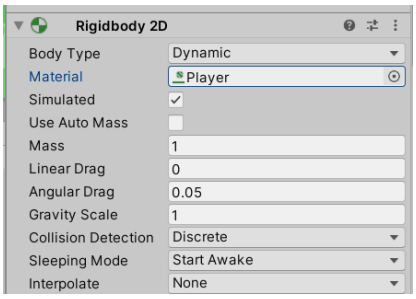
Gambar 8. 20 Memberi Nama File Physical Material 2D

21. Pada inspector Player (Physics Material 2D) ubah friction dan bounces menjadi 0



Gambar 8. 21 Mengubah Aturan File Physical Material 2D

22. Buka inspector dari idle lalu cari component Rigidbody 2D dan ubah material menjadi Player



Gambar 8. 22 Mengubah Material Pada Component RigidBody

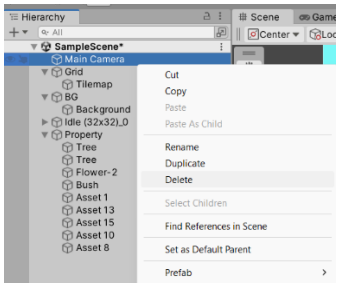
23. Klik 2 kali pada file C# dengan nama Player yang terletak pada folder Scripts lalu ketikkan source code dibawah ini

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

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 facingRight;
  bool jump;

  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(-0.5f, 0.5f,
0.5f);
      facingRight = false;
    }

    else if (!facingRight && dir > 0)
    {
      // ukuran player
       transform.localScale = new Vector3(0.5f, 0.5f,
0.5f);
      facingRight = true;
    }

    #endregion
  }
}
```

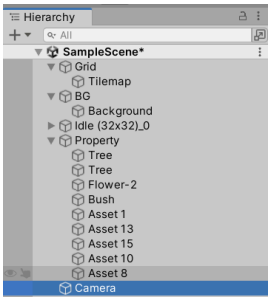
B. Camera Movement

1. Hapus Main Camera yang ada pada hirarki



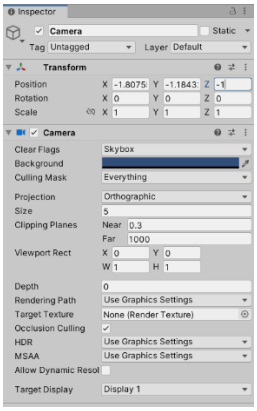
Gambar 8. 23 Menghapus Main Camera

2. Klik kanan pada hirarki lalu create empty dan ubah menjadi Camera



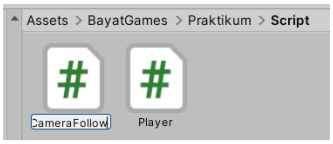
Gambar 8. 24 Menambahkan Camera Pada Hirarki

3. Sesuaikan inspector dari Camera berdasarkan gambar berikut



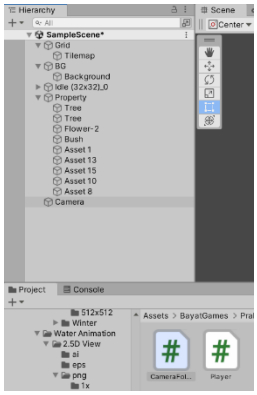
Gambar 8. 25 Mengatur Inspector Camera

4. Buat file C# baru dengan nama CameraFollow pada folder Script



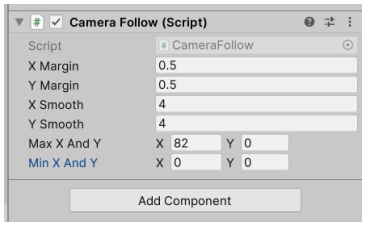
Gambar 8. 26 Membuat File CameraFollow

5. Drag file C# CameraFollow ke dalam hirarki Camera



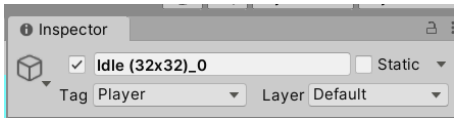
Gambar 8. 27 Meletakkan File CameraFollow Pada Hirarki

6. Ubah Max X dan Y pada Inspector Camera



Gambar 8. 28 Mengubah Pengaturan Pada Inspector Camera

7. Pada inspector Idle ubah tag menjadi Player



Gambar 8. 29 Mengubah Tag

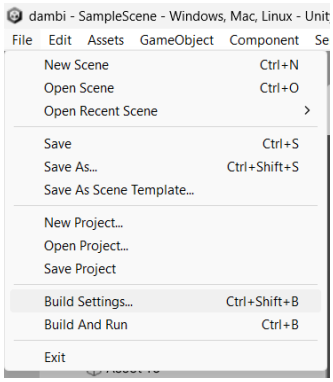
8. Klik 2 kali pada file C# dengan nama CameraFollow yang terletak pada folder Script lalu ketik source code berikut

```
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);
    }
}
```

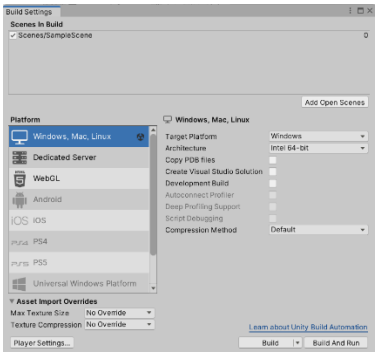
C. Render

1. Pada menu file pilih Build Settings



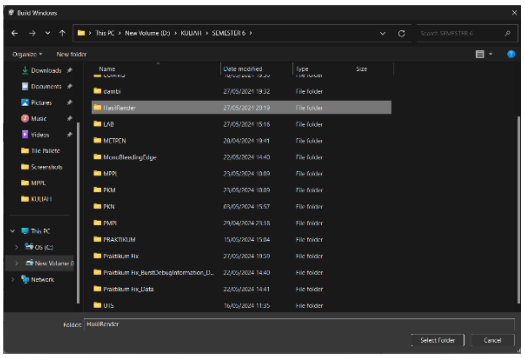
Gambar 8. 30 Memilih Menu File

2. Lalu pada platform pilih Windows, Mac, Linux



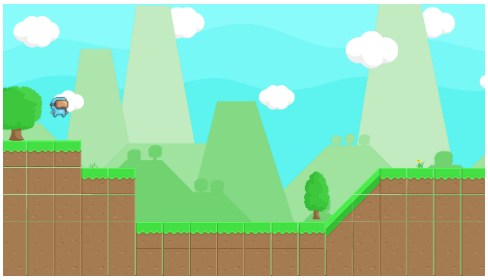
Gambar 8. 31 Memilih Platform

3. Klik button Build and Run lalu pilih direktori folder untuk menyimpan hasil render



Gambar 8. 32 Memilih Direktori Penyimpanan

4. Jalankan hasil render



Gambar 8. 33 Hasil Render

1.2 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.