LAMPIRAN 1
DAFTAR RIWAYAT HIDUP

Data Pribadi

Nama	: Aldi Suryadi
Jenis Kelamin	: Laki-Laki
Tempat/Tanggal Lahir	: Bandung 8 Oktober 2000
Alamat Rumah	: Perumahan Daan Mogot Estate Blok JA No.14
Telepon/HP	: 087780680008
Email	: aldisuryadi16@gmail.com
NIM	: 20180801288
Program Studi	: Teknik Informatika

Pendidikan Formal

Periode	Sekolah	Jurusan
2012-2015	Seraphine Bakti Utama	-
2015-2018	SMAN 96 Jakarta	IPA

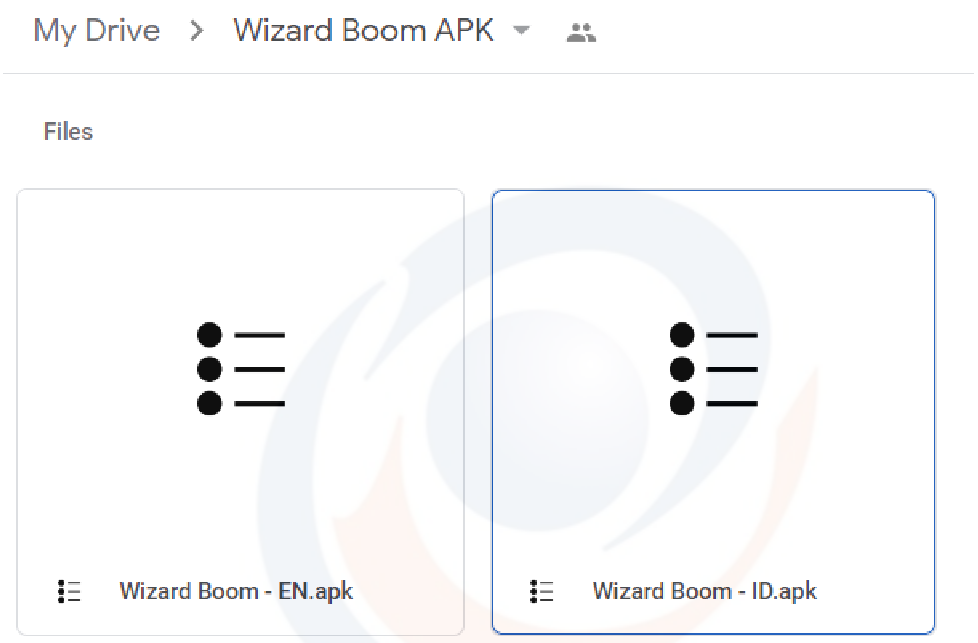
Riwayat Pekerjaan

Juni 2018 – Agustus 2018	: Game Operation Staff di Prodigy Infinitech
Agustus 2018 – Mei 2019	: Quality Assurance di PT Graha Karindo Interactive
Juni 2019 – Sekarang	: Game Operation Specialist di PT Indofun Digital Technology

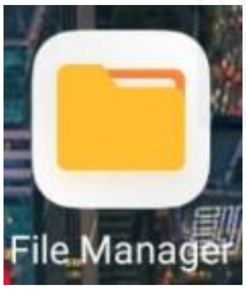
LAMPIRAN 2
USER GUIDE

Tahap Instalasi APK

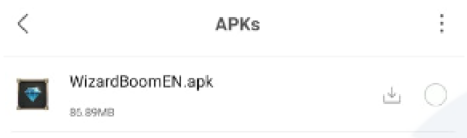
1. Pemain dapat melakukan download pada APK di google drive berikut: https://drive.google.com/drive/folders/1vKv3jVixNwDGRjU2QsDWNc9EdgAt4Or4?usp=sharing
   atau download di playstore dengan link berikut: https://play.google.com/store/apps/details?id=com.AldiSuryadi.WizardBoom



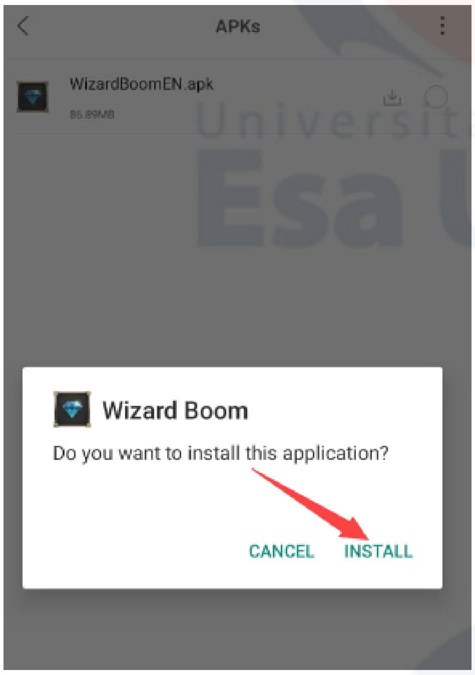
2. Kemudian setelah download selesai pemain dapat membuka file manager



3. Cari APK bernama WizardBoomEN yang telah didownload pada file manager
4. Klik pada APK



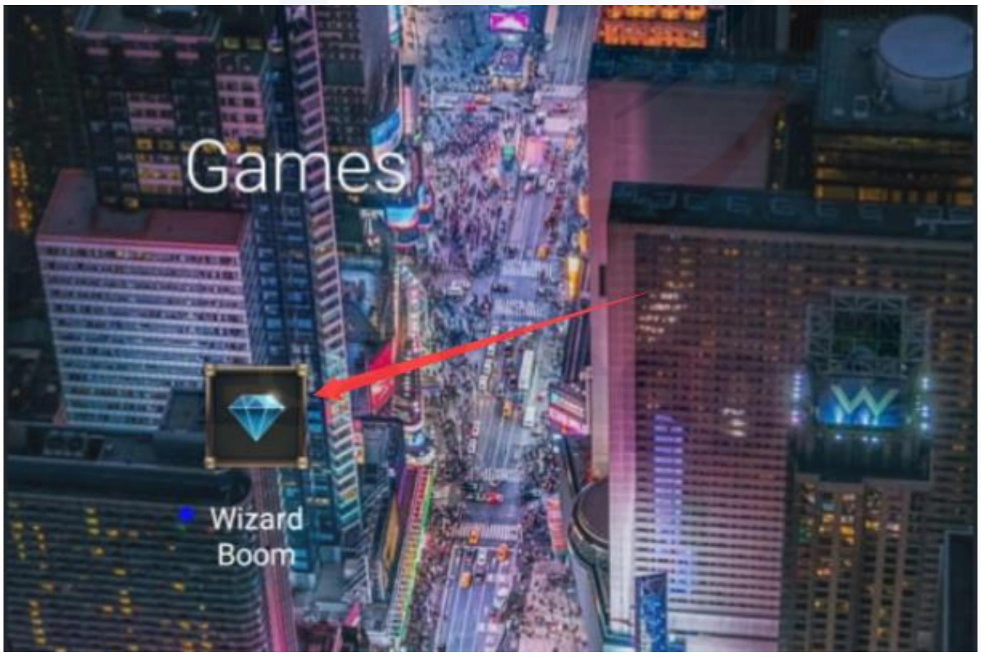
5. Kemudian klik install untuk melakukan installasi



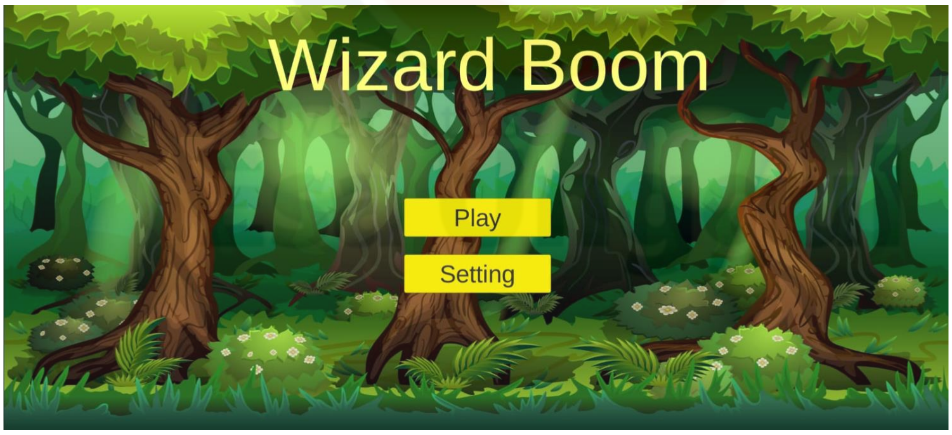
Do you want to install this application?

Tahap melakukan permainan

6. Setelah APK berhasil terinstall maka tahap selanjutnya adalah menekan tombol icon game wizard boom untuk memasuki permainan

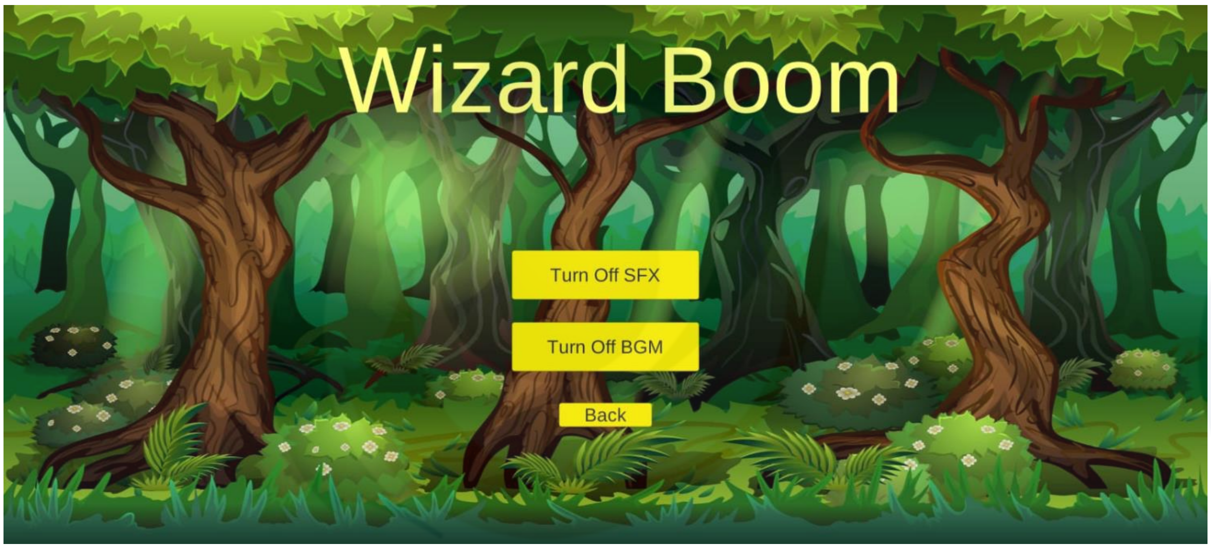


7. Tampilan awal dari permainan adalah sebagai berikut :

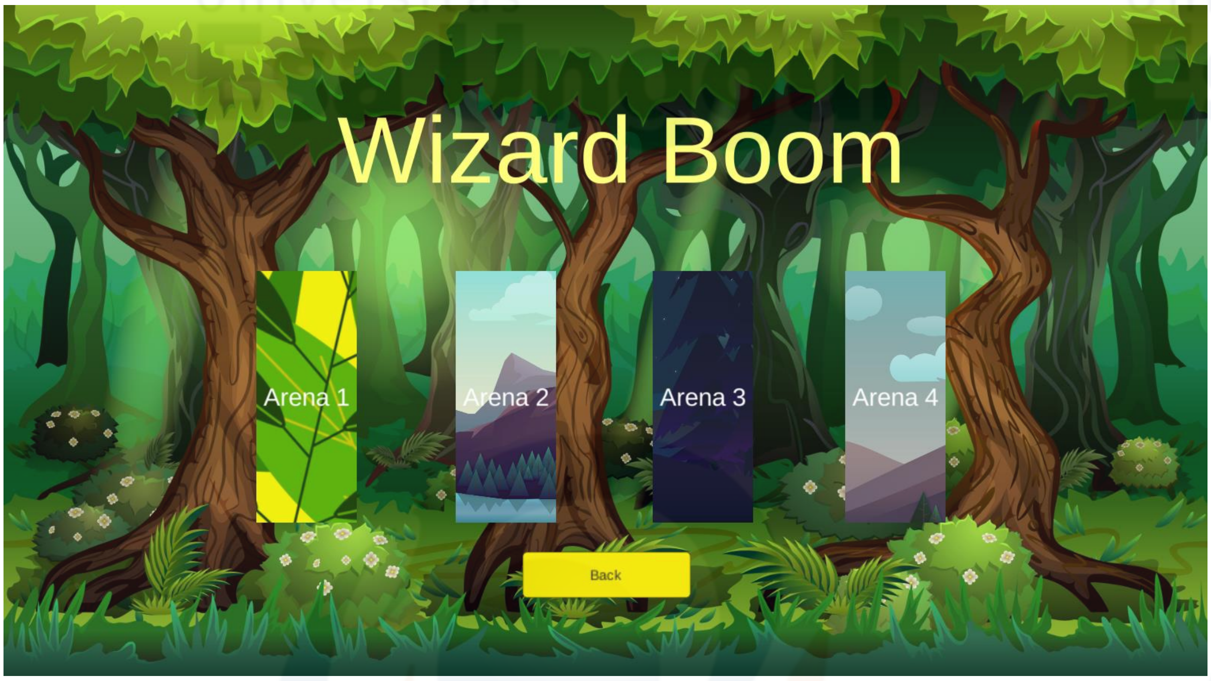


8. Pemain dapat menekan tombol play untuk memilih arena yang akan dimainkan atau tombol settings untuk melakukan pengaturan

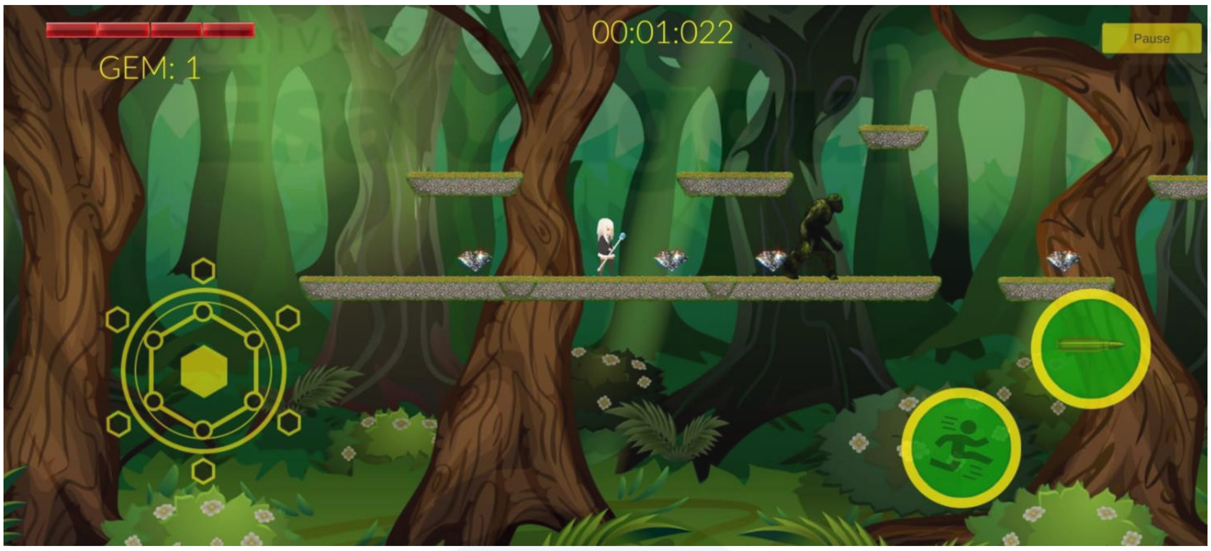
Tampilan Menu Settings



Tampilan Menu Play



9. Tersedia 4 arena pada game wizard boom yang dapat dipilih, sebagai contoh arena 1 pada wizard boom sebagai berikut :



Fungsi-fungsi yang ada pada game dijelaskan sebagai berikut

a. Joystick – untuk melakukan gerakan pada karakter utama
b. Icon peluru – untuk melakukan serangan pada karakter utama
c. Icon lompat – untuk melakukan lompatan pada karakter utama
d. Pause – untuk memberhentikan permainan sementara
e. Resume – untuk melanjutkan permainan
f. Main Menu – untuk kembali ke menu tampilan awal
g. Restart – untuk mengulang kembali permainan

10. Pemain sudah dapat melakukan permainan

LAMPIRAN 3
PROGRAM CODE

Main Menu

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

public class MainMenu : MonoBehaviour
{
    // Start is called before the first frame update

    public enum MenuState {  Main_Menu, Arena, Setting};
    public MenuState currentState;

    public GameObject Menu;
    public GameObject ArenaSelection;
    public GameObject Setting;

    public Text bgmTxt;
    public Text sfxTxt;


    void Awake()
    {
        currentState = MenuState.Main_Menu;
    }

    void Update()
    {
        switch (currentState)
        {
            case MenuState.Main_Menu:
                Menu.SetActive(true);
                ArenaSelection.SetActive(false);
                Setting.SetActive(false);
                break;
            case MenuState.Arena:
                ArenaSelection.SetActive(true);
                Menu.SetActive(false);
                Setting.SetActive(false);
                break;
            case MenuState.Setting:
                Setting.SetActive(true);
                Menu.SetActive(false);
                ArenaSelection.SetActive(false);
            break;
    }

    if (MasterManager.bgm == true)
    {
        bgmTxt.text = "Turn Off BGM";
    }
    else
    {
        bgmTxt.text = "Turn On BGM";
    }

    if (MasterManager.sfx == true)
    {
        sfxTxt.text = "Turn Off SFX";
    }
    else
    {
        sfxTxt.text = "Turn On SFX";
    }
}

//when button is pressed


public void OnClickArena()
{
    currentState = MenuState.Arena;
}

public void OnClickSetting()
{
    currentState = MenuState.Setting;
}

public void OnMainMenu()
{
    currentState = MenuState.Main_Menu;
}

public void OnClickArena1()
{
    SceneManager.LoadScene("Arena 1");
}

public void OnClickArena2()
{
    SceneManager.LoadScene("Arena 2");
}
```

```
    public void OnClickArena3()
    {
        SceneManager.LoadScene("Arena 3");
    }

    public void OnClickArena4()
    {
        SceneManager.LoadScene("Arena 4");
    }

    public void toggleBGM()
    {
        MasterManager.Instance.ToggleBGM();



    }

    public void toggleSFX()
    {
        if (MasterManager.sfx == true)
        {
            MasterManager.sfx = false;
        }
        else
        {
            MasterManager.sfx = true;
        }
    }
}
```

Enemy

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

public class EnemyScript : MonoBehaviour
{
    public GameObject gemsPrefab;

    [SerializeField]
    protected Transform pointA;
    [SerializeField]
    protected Transform pointB;
    [SerializeField]
    protected Vector3 currentTarget;
protected Animator v_anim;
protected Rigidbody2D v_rb;
protected Collider2D v_collider;


[SerializeField]
protected int v_health = 3;
[SerializeField]
protected float v_speed = 5;


protected GameObject v_player;
[SerializeField]
protected float v_attackRange = 5f;
protected bool v_playerChase = false;
protected bool v_inIdlePos = true;
protected bool v_isDead = false;
protected bool v_isHit = false;
protected bool v_onTrack = false;

[SerializeField]
protected int gemsValue = 1;

[SerializeField]
public LayerMask playerLayer;
public LayerMask groundlayer;
public LayerMask wallLayer;
protected bool v_playerDetected;


//movement variable

public Vector2 v_face_direction = Vector2.right;
protected bool movingRight = true;
public Transform detectionPoint;


public bool v_normalAttack = false;
public GameObject projectilePrefab;

private AudioSource audioSource;

public AudioClip AttackSound;
public AudioClip hitSound;


protected SpriteRenderer v_spriteRenderer;
```

```
  // Start is called before the first frame update
  void Start()
  {
     Init();
  }

  // Update is called once per frame
  public virtual void Update()
  {

     if (MasterManager.sfx == false)
     {
        audioSource.volume = 0.0f;
     }

     v_playerDetected = DetectPlayer();

       if (v_anim.GetCurrentAnimatorStateInfo(0).IsName("Idle")  &&  v_anim.GetBool("InCombat")  ==
false && v_playerDetected == false)
     {
        //v_playerDetected = DetectPlayer();

        return;
     }

     if (v_isDead == false)
     {
        EnemyMove();

     }
     else
     {
        // v_player.GetComponent<PlayerScript>().AddEnemyKill();
        StartCoroutine(UpdateScore(3f));
     }




  }

  public virtual void Init()
  {
    // Debug.Log("init registered");
     v_spriteRenderer = GetComponentInChildren<SpriteRenderer>();
     v_player = FindObjectOfType<PlayerScript>().gameObject;
     v_anim = GetComponentInChildren<Animator>();
     v_collider = GetComponentInChildren<Collider2D>();
     v_rb = GetComponentInChildren<Rigidbody2D>();
```

```
        audioSource = GetComponent<AudioSource>();

        if(audioSource == null)
        {
            //Debug.Log("audiosource not found");
        }

    }



    public virtual void EnemyMove()
    {

        if(v_isHit == false && v_normalAttack == false)
        {
            transform.Translate(Vector2.right * v_speed * Time.deltaTime);
        }


            RaycastHit2D  detectGround  =  Physics2D.Raycast(detectionPoint.position,  Vector2.down,  2f,
groundlayer);
            RaycastHit2D  detectWall  =  Physics2D.Raycast(detectionPoint.position,  v_face_direction,  2f,
wallLayer);

        Debug.DrawRay(detectionPoint.position, Vector2.down, Color.green);
        Debug.DrawRay(detectionPoint.position, v_face_direction, Color.green);


        //        Debug.Log(detect.collider.name);

        if (detectGround.collider == false || detectWall.collider == true)
        {
//            Debug.Log("collider not detected");
            if(movingRight == true)
            {

              //  Debug.Log("moving left");
                v_anim.SetTrigger("Idle");
                transform.eulerAngles = new Vector3(0, -180, 0);
                movingRight = false;
                v_face_direction = Vector2.left;

            }
            else
            {
//                Debug.Log("moving right");
                v_anim.SetTrigger("Idle");
               transform.eulerAngles = new Vector3(0, 0, 0);
                movingRight = true;
                v_face_direction = Vector2.right;
            }
        }


        float distance = Vector2.Distance(v_player.transform.localPosition, transform.localPosition);

        //Debug.Log(distance);

        if (distance > v_attackRange)
        {
            v_isHit = false;
            v_playerChase = false;
            v_anim.SetBool("InCombat", false);

        }
        if (v_playerDetected == true)
        {

            v_normalAttack = true;
            v_playerChase = true;
            v_anim.SetBool("InCombat", true);

        }
        else if (v_playerDetected == false && v_isHit == false)
        {
            v_normalAttack = false;
            v_playerChase = false;
            v_anim.SetBool("InCombat", false);
        }
        else
        {
            v_normalAttack = false;
            v_anim.SetBool("InCombat", true);
        }


        //flip enemy sprite towrds player

        Vector2 v_direction = v_player.transform.localPosition - transform.localPosition;

        if (v_direction.x < 0f && v_anim.GetBool("InCombat") == true)
        {
            //v_spriteRenderer.flipX = true;
            //facing left

            transform.eulerAngles = new Vector3(0, -180, 0);
            v_face_direction = Vector2.left;
```

```
}
else if (v_direction.x > 0f && v_anim.GetBool("InCombat") == true)
{
    //v_spriteRenderer.flipX = false;
    //facing right
    transform.eulerAngles = new Vector3(0, 0, 0);
    v_face_direction = Vector2.right;

}

/*
if(v_onTrack == false)
{
    currentTarget = pointA.position;
    //Debug.Log("here");
}

if(pointA != null && currentTarget.x == pointA.position.x)
{
   // v_spriteRenderer.flipX = false;
   //facing right
    transform.localRotation = Quaternion.Euler(0, 0, 0);


    v_face_direction = Vector2.right;


}
else
{
    //v_spriteRenderer.flipX = true;
    transform.localRotation = Quaternion.Euler(0, 180, 0);


    v_face_direction = Vector2.left;




}

if (transform.position.x == pointA.position.x && v_playerChase == false && v_inIdlePos == true)
{
   //Debug.Log("masuk a");
    currentTarget = pointB.position;
    v_onTrack = true;
    v_anim.SetTrigger("Idle");
}
```

```
        else if (transform.position.x == pointB.position.x && v_playerChase == false && v_inIdlePos == true
&& v_onTrack == true)
        {
           //Debug.Log("masuk b");
            currentTarget = pointA.position;
            v_anim.SetTrigger("Idle");

        }



        //Debug.Log("jalan");



        if(v_isHit == false && v_playerChase == false && v_normalAttack == false)
        {
            //Debug.Log("iswalking");

                    transform.position = Vector3.MoveTowards(transform.position, currentTarget, v_speed *
Time.deltaTime);
        }

        //Debug.Log(v_isHit);

        float distance = Vector2.Distance(v_player.transform.localPosition, transform.localPosition);

        //Debug.Log(distance);

       if (distance > v_attackRange)
        {
             v_isHit = false;
             v_playerChase = false;
             v_anim.SetBool("InCombat", false);

        }
        if(v_playerDetected == true)
        {

            v_normalAttack = true;
            v_playerChase = true;
            v_anim.SetBool("InCombat", true);

        }
        else if(v_playerDetected == false && v_isHit == false)
        {
            v_normalAttack = false;
            v_playerChase = false;
            v_anim.SetBool("InCombat", false);
        }
```

```
    //flip enemy sprite towrds player

   Vector2 v_direction = v_player.transform.localPosition - transform.localPosition;

    if(v_direction.x < 0f && v_anim.GetBool("InCombat") == true)
    {
        //v_spriteRenderer.flipX = true;
        //facing left
        transform.localRotation = Quaternion.Euler(0, 180, 0);
        v_face_direction = Vector2.left;

    }
    else if(v_direction.x > 0f && v_anim.GetBool("InCombat") == true)
    {
        //v_spriteRenderer.flipX = false;
        //facing right
        transform.localRotation = Quaternion.Euler(0, 0, 0);
        v_face_direction = Vector2.right;

    }

    */

}

public void EnemyAttack()
{
    if(v_normalAttack == false)
    {
       // audioSource.PlayOneShot(AttackSound, 0.7f);
        Instantiate(projectilePrefab, transform.position, Quaternion.identity);
    }


}



public bool DetectPlayer()
{


        RaycastHit2D hit = Physics2D.Raycast(detectionPoint.position, v_face_direction, 1f, playerLayer);

    Debug.DrawRay(detectionPoint.position, v_face_direction, Color.green);
```

```
        if (hit.collider != null)
        {
//            Debug.Log("detect player");

            return true;
        }
        else
        {
            // Debug.Log("terbang");
            return false;
        }

    }

    IEnumerator UpdateScore(float pauseTime)
    {
        yield return new WaitForSeconds(pauseTime);
        v_player.GetComponent<PlayerScript>().enemyKilledCount += 1;
    }


}
```

Camera

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

public class CameraScript : MonoBehaviour
{
    public Transform playerTransform;
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
            this.transform.position = new Vector3(playerTransform.position.x, playerTransform.position.y,
this.transform.position.z);

    }
}
```

Enemyattack

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

public class EnemyAttack : MonoBehaviour
{
    public float hitDelay = 0.5f;
    // variable to determine if the damage function can be called
    private bool v_canDamage = true;


    void OnTriggerEnter2D(Collider2D other)
    {

        iDamage hit = other.GetComponent<iDamage>();

        if (hit != null)
        {
            // if can attack
            if (v_canDamage == true)
            {
                //Debug.Log("player attacked");
                hit.Damage();
                v_canDamage = false;
                StartCoroutine(damagePause());
            }
        }
    }

    // Coroutine to switch variable back to true after 0.5 seconds
    IEnumerator damagePause()
    {

        yield return new WaitForSeconds(hitDelay);
        v_canDamage = true;
    }
}
```

Gem

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

public class GemScript : MonoBehaviour
{
    public int gemsValue = 1;

    private void OnTriggerEnter2D(Collider2D other)
    {
        //collect

        if(other.tag == "Player")
        {
            PlayerScript player = other.GetComponent<PlayerScript>();

            if (player != null)
            {
                player.AddGems(gemsValue);
                Destroy(this.gameObject);
            }

        }

    }
```

Player

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

public class PlayerScript : MonoBehaviour, iDamage
{
    public float speed = 5f;
    public int p_health = 4;
    public float jumpForce = 10f;
    public bool facingRight = true;
    public Animator anim;

    [SerializeField]
    private Transform spawnPoint;
    [SerializeField]
    private GameObject projectile;
    [SerializeField]
    private GameObject FireBeam;

    public int gemCount = 0;
    public int enemyKilledCount = 0;
    public int secretAreaCount = 0;
    [SerializeField]
    private LayerMask groundLayer;
    private bool b_isGrounded = false;
    private bool resetJump = false;

    public int health {get; set;}

    public bool defaultWeapon = true;
    public bool fireBeamWeapon = false;
    public int fireBeamAmmo = 0;

    protected float horizontal = 0;
    protected float vertical;

    public GameObject GameOverPanel;
    protected bool gameOver = false;

    private AudioSource audioSource;

    public AudioClip footStepSound;
    public AudioClip hitSound;
    public AudioClip fireBallSound;
    public AudioClip attackSound;
    public AudioClip gemSound;
    public AudioClip jumpSound;

    //cheat stuff
    public static bool cheatEnabled = false;
    public static bool jumpCheatEnabled = false;

    //secret area achievement stuff
    public static bool secretArea1 = false;
    public static bool secretArea2 = false;
    public static bool secretArea3 = false;
    public static bool secretArea4 = false;

    public static bool extraWeapon = false;
    // Start is called before the first frame update

    Rigidbody2D rb;
    void Start()
    {
       audioSource = GetComponent<AudioSource>();

       rb = this.GetComponent<Rigidbody2D>();
       anim = this.GetComponent<Animator>();
       health = p_health;
       //transform.Rotate(new Vector3(0, 180, 0));
    }
```

```
  // Update is called once per frame
  void Update()
  {
      Movement();


      //Debug.Log(health);

      //SFX Stuff
      if (MasterManager.sfx == false)
      {
          audioSource.volume = 0.0f;
      }

      /*
                                          if (Input.GetButtonDown("Player1_Shoot") &&
anim.GetCurrentAnimatorStateInfo(0).IsName("Attack") == false)
      {
           //horizontal= 0;
          anim.SetTrigger("attack");


      }
      */

      // android code attack

                                if (CrossPlatformInputManager.GetButtonDown("A_Button") &&
anim.GetCurrentAnimatorStateInfo(0).IsName("Attack") == false)
      {
          //horizontal= 0;
          anim.SetTrigger("attack");


      }
      if (health < 1)
      {


          anim.SetTrigger("die");
//           Debug.Log("player dead ");

          if (gameOver == false)
          {
              audioSource.PlayOneShot(hitSound, 0.7f);
              gameOver = true;
              Time.timeScale = 0;
              GameOverPanel.SetActive(true);
```

```
        }



    }
}

void PlayFootstepSound()
{
    if(b_isGrounded == true)
    {
        audioSource.PlayOneShot(footStepSound, 0.3f);
    }

}



void Attack()
{

    if(fireBeamAmmo > 0)
    {
        audioSource.PlayOneShot(fireBallSound, 0.7f);

        Instantiate(FireBeam, spawnPoint.position, transform.rotation);
        fireBeamAmmo--;
    }
    else
    {
        //          Debug.Log("spawn bullet");
        audioSource.PlayOneShot(fireBallSound, 0.7f);
        audioSource.PlayOneShot(attackSound, 0.7f);

        Instantiate(projectile, spawnPoint.position, transform.rotation);
    }
}

void Movement()
{
    //pc cpde
    // horizontal = Input.GetAxis("Horizontal");
     //vertical = Input.GetAxis("Vertical");

    //android code
    horizontal = CrossPlatformInputManager.GetAxis("Horizontal");
    vertical = CrossPlatformInputManager.GetAxis("Vertical");
    // Debug.Log(anim.GetBool("isJump"));
```

```
        b_isGrounded = isGrounded();
        //isGrounded();
                    if    ((horizontal   <   0   &&   facingRight||   horizontal   >   0   &&   !facingRight)   &&
anim.GetCurrentAnimatorStateInfo(0).IsName("Attack") == false)
        {
//              Debug.Log("flip");
            facingRight = !facingRight;
            transform.Rotate(new Vector3(0, 180, 0));
        }

        if(jumpCheatEnabled == false)
        {
            //pc code
            /*
            if (Input.GetKeyDown(KeyCode.Space) && isGrounded())
            {
                anim.SetBool("isJump", true);


                audioSource.PlayOneShot(jumpSound, 0.7f);

                rb.velocity = new Vector2(rb.velocity.x, jumpForce);
                StartCoroutine(ResetJumpCouroutine());

            }
            */

            if (CrossPlatformInputManager.GetButtonDown("B_Button") && isGrounded())
            {
                anim.SetBool("isJump", true);


                audioSource.PlayOneShot(jumpSound, 0.7f);

                rb.velocity = new Vector2(rb.velocity.x, jumpForce);
                StartCoroutine(ResetJumpCouroutine());

            }
        }
        else
        {
            //pc code
            /*
            if (Input.GetKeyDown(KeyCode.Space))
            {
                anim.SetBool("isJump", true);
            audioSource.PlayOneShot(jumpSound, 0.7f);

            rb.velocity = new Vector2(rb.velocity.x, jumpForce);
            StartCoroutine(ResetJumpCouroutine());

         }
         */

         if (CrossPlatformInputManager.GetButtonDown("B_Button"))
         {
            anim.SetBool("isJump", true);


            audioSource.PlayOneShot(jumpSound, 0.7f);

            rb.velocity = new Vector2(rb.velocity.x, jumpForce);
            StartCoroutine(ResetJumpCouroutine());

         }
      }



      //transform.position = transform.position + new Vector3(horizontal * speed * Time.deltaTime, vertical
* speed  * Time.deltaTime, 0);

//       Debug.Log(horizontal);

                                 if((Input.GetButton("Horizontal")        ||       horizontal        !=       0)        &&
anim.GetCurrentAnimatorStateInfo(0).IsName("Attack") == false)
      {
        // Debug.Log("move1");
         anim.SetBool("isRun", true);
         rb.velocity = new Vector2(horizontal * speed, rb.velocity.y);
         anim.SetFloat("speed", Mathf.Abs(horizontal));
      }
      else if(anim.GetCurrentAnimatorStateInfo(0).IsName("Attack") == false)
      {
//          Debug.Log("move2");
         anim.SetBool("isRun", false);
         rb.velocity = new Vector2(horizontal * speed, rb.velocity.y);
         anim.SetFloat("speed", Mathf.Abs(horizontal));
      }
      else if(anim.GetCurrentAnimatorStateInfo(0).IsName("Attack") == true && b_isGrounded == true)
      {
         horizontal = 0;
         rb.velocity = new Vector2(horizontal * speed, rb.velocity.y);
//          Debug.Log("move3");
```

```
    }
    else
    {
        rb.velocity = new Vector2(horizontal * speed, rb.velocity.y);
    }
    //output to log the position change
   // Debug.Log(transform.position);
}

bool isGrounded()
{
    RaycastHit2D hit = Physics2D.Raycast(transform.position, Vector2.down, 0.6f, groundLayer);

   // Debug.Log(hit.collider.gameObject.name);

    Debug.DrawRay(transform.position, Vector2.down * 0.6f, Color.green);

    if(hit.collider != null)
    {
      // Debug.Log("FALSE JUMP");
      if(resetJump == false)
       {
           anim.SetBool("isJump", false);
           return true;
       }

    }
    return false;


}

IEnumerator ResetJumpCouroutine()
{
    resetJump = true;
    yield return new WaitForSeconds(0.1f);
    resetJump = false;


}

public void Damage()
{
    if(cheatEnabled == false)
    {
        if (health < 1)
        {
            return;
        }
        health--;
        audioSource.PlayOneShot(hitSound, 0.7f);

        UIManager.Instance.UpdateHP(health);
        anim.SetTrigger("hurt");

    }
    else
    {
        audioSource.PlayOneShot(hitSound, 0.7f);
        anim.SetTrigger("hurt");
    }



}

void OnTriggerEnter2D(Collider2D other)
{
    // Debug.Log("touch");

    if(other.gameObject != null)
    {
      // Debug.Log("hit smthgn");
    }

    if (other.gameObject.tag == "ExtraWeapon")
    {
        audioSource.PlayOneShot(gemSound, 0.7f);
        fireBeamAmmo += 20;
        Destroy(other.gameObject);
        if(extraWeapon == false)
        {
            extraWeapon = true;
            UIManager.Instance.showAchievement("Extra Weapon Unlocked!");
        }
    }
    if (other.gameObject.tag == "DeadZone")
    {
        Debug.Log("player dead");
        health = health - 10;
         UIManager.Instance.UpdateHP(health);


    }
    if(other.gameObject.tag == "SecretArea")
    {

        if(secretArea1 == false)
        {
            secretAreaCount += 1;
            secretArea1 = true;
            UIManager.Instance.showAchievement("Secret Area 1 Discovered!");
        }




    }

    if (other.gameObject.tag == "SecretArea2")
    {

        if (secretArea2 == false)
        {
            secretAreaCount += 1;
            secretArea2 = true;
            UIManager.Instance.showAchievement("Secret Area 2 Discovered!");
        }

    }

    if (other.gameObject.tag == "SecretArea3")
    {

        if (secretArea3 == false)
        {
            secretAreaCount += 1;
            secretArea3 = true;
            UIManager.Instance.showAchievement("Secret Area 3 Discovered!");
        }

    }

    if (other.gameObject.tag == "SecretArea4")
    {

        if (secretArea4 == false)
        {
            secretAreaCount += 1;
            secretArea4 = true;
            UIManager.Instance.showAchievement("Secret Area 4 Discovered!");
        }

    }
```

```


     if (other.gameObject.tag == "FinishArea" && MasterManager.Instance.currentAreaBossKilled == true)
      {

         UIManager.Instance.UpdateScoreBoard(gemCount, enemyKilledCount,secretAreaCount);
         UIManager.Instance.ScoreBoard.SetActive(true);
         Time.timeScale = 0;

      }

      if (other.gameObject.tag == "FinishAreaNoBoss" )
      {

         UIManager.Instance.UpdateScoreBoard(gemCount, enemyKilledCount, secretAreaCount);
         UIManager.Instance.ScoreBoard.SetActive(true);
         Time.timeScale = 0;

      }
   }

   public void AddGems(int gemAmount)
   {
      audioSource.PlayOneShot(gemSound, 0.7f);

      gemCount += gemAmount;
      UIManager.Instance.UpdateGemCount(gemCount);

   }

   public void AddEnemyKill()
   {
      enemyKilledCount += 1;

   }
}
```

Timer

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

public class TimerScript : MonoBehaviour
{
    private static TimerScript _instance;

    bool timerActive = true;
    public float currentTime;
    public Text timerTxt;

    public static TimerScript Instance
    {
        get
        {
            if (_instance == null)
            {
                Debug.Log("timer is error");
            }
            return _instance;

        }
    }
    void Awake()
    {
        _instance = this;
        currentTime = 0;
    }


    void Update()
    {
        if(timerActive == true)
        {
            currentTime += Time.deltaTime;
        }
        TimeSpan time = TimeSpan.FromSeconds(currentTime);
        timerTxt.text = time.ToString(@"mm\:ss\:fff");
    }
```

MasterManager

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

public class MasterManager : MonoBehaviour
{
    private static MasterManager _instance;

    private AudioSource audioSource;

    public AudioClip bgMusic1;
    public AudioClip bgMusic2;
    public bool bossTrigger = false;

    public static bool bgm = true;
    public static bool sfx = true;

    public bool currentAreaBossKilled = false;

    public static MasterManager Instance



    {
        get
        {
            if(_instance == null)
            {
                Debug.Log("game manager is error");
            }

            return _instance;
        }
    }


    private void Awake()
    {
        _instance = this;

        audioSource = GetComponent<AudioSource>();

        if(bgm == true)
        {
            audioSource.clip = bgMusic1;
            audioSource.Play(); // play music
        }


    }




    public void ToggleBGM()
    {
        if(bgm == true)
        {

            audioSource.Pause(); // play music
            bgm = false;
        }
```

```
        else
        {
            audioSource.Play();
            bgm = true;
        }
    }

    public void ActivateBossMusic()
    {
        if(bossTrigger == false && bgm == true)
        {
            audioSource.Stop();
            bossTrigger = true;
            audioSource.clip = bgMusic2;
            audioSource.volume = 0.5f;
            audioSource.Play();
        }
    }
    public void Update()
    {
        //Debug.Log(currentAreaBossKilled);
    }


}
```

UIManager

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


public class UIManager : MonoBehaviour
{
    private static UIManager _instance;
    public float timeStart = 0;

    public GameObject ScoreBoard;
    public GameObject AchievementBoard;

    public GameObject PauseGameState;

    protected GameObject timer;
```

```
public static UIManager Instance
{
    get
    {
        if(_instance == null)
        {
            Debug.Log("UIManager is error");
        }
        return _instance;

    }
}

private void Update()
{

}


public Text gemCountTxt;

public Image[] healthBar;
public Image[] enemyHealthBar;

//scoreboard details
public Text currentTime;
public Text gemCountScore;
public Text enemyKilledScore;
public Text secretAreaScore;
public Text timerScore;

//scoreboard points
public Text timePoints;
public Text enemyPoints;
public Text secretPoints;
public Text gemPoints;


public int timerPoints;

public Text totalPoints;



public static GameObject Player;
//achievement stuff
public Text achievementText;
//public Image gameOver;
```

```
private void Awake()
{
    _instance = this;
    Player = GameObject.FindGameObjectWithTag("Player");
}

public void RestartGame()
{

    SceneManager.LoadScene("Arena 1");
    Time.timeScale = 1;




}

public void showAchievement (string achText)
{
    AchievementBoard.SetActive(true);
    achievementText.text = achText;
    StartCoroutine(PauseAchievement(3f));
}


public void LoadMainMenu()
{
    SceneManager.LoadScene("MainMenu");
    Time.timeScale = 1;
}



public void UpdateGemCount(int count)
{
    gemCountTxt.text = "GEM: " + count;
}

public void UpdateScoreBoard(int gemCount, int enemyKilledCount, int secretAreaCount)
{
    gemCountScore.text = "" + gemCount;
    enemyKilledScore.text = "" + enemyKilledCount;
    secretAreaScore.text = "" + secretAreaCount;
    timerScore.text = currentTime.text;

     if(TimerScript.Instance.currentTime < 60)
     {
           timerPoints = 200;
        timePoints.text = "< 1 min: " + timerPoints + " Points";
       }
       else if(TimerScript.Instance.currentTime > 60 && TimerScript.Instance.currentTime < 120)
      {
         timerPoints = 100;
         timePoints.text = "< 2 min: " + timerPoints + " Points";

      }
      else
      {
         timerPoints = 50;

         timePoints.text = "> 2 min: " + timerPoints + " Points";

      }




      enemyPoints.text = "" + enemyKilledCount * 100 + " Points";
      secretPoints.text = "" + secretAreaCount * 100 + " Points";
      gemPoints.text = "" + gemCount * 100 + " Points";
       totalPoints.text = "" + ((enemyKilledCount * 100) + (secretAreaCount * 100) + (gemCount * 100) +
timerPoints) + " Points";


   }
   public void UpdateHP(int hpCount)
   {
      for(int i = 0; i <= hpCount; i++)
      {
         if(i == hpCount)
         {
            healthBar[i].enabled = false;
         }
      }
   }

   public void UpdateEnemyHP(int hpCount)
   {
      for (int i = 0; i <= hpCount; i++)
      {
         if (i == hpCount)
         {
            enemyHealthBar[i].enabled = false;
         }
      }
   }
```

```
public void PauseGame()
{
   Time.timeScale = 0;
   PauseGameState.SetActive(true);
}

public void ResumeGame()
{
   PauseGameState.SetActive(false);
   Time.timeScale = 1;
}

public void OnClickArena2()
{
   SceneManager.LoadScene("Arena 2");
   Time.timeScale = 1;

}


public void OnClickArena3()
{
   SceneManager.LoadScene("Arena 3");
   Time.timeScale = 1;

}

public void OnClickArena4()
{
   SceneManager.LoadScene("Arena 4");
   Time.timeScale = 1;

}



public void ToggleCheat()
{
   if(PlayerScript.cheatEnabled == false)
   {
      PlayerScript.cheatEnabled = true;


   }
   else
   {
      PlayerScript.cheatEnabled = false;
```

```
        }

    }

    public void ToggleJumpCheat()
    {
        if(PlayerScript.jumpCheatEnabled == false)
        {
            PlayerScript.jumpCheatEnabled = true;


        }
        else
        {
            PlayerScript.jumpCheatEnabled = false;

        }
    }
    IEnumerator PauseAchievement(float pauseTime)
    {
        yield return new WaitForSeconds(pauseTime);
        AchievementBoard.SetActive(false);
    }

}
```