//Series

https://www.youtube.com/playlist?list=PL2614K6kEvHC37KsgRtqnydn1vQd8uCPy

//Cài IDE để code script

- Cài VSCode vào máy.

- Vào Unity Chọn Edit -> References -> External Tools ->External Script Editor -> chọn VSCode.

// Lỗi Visual Studio Không auto Complete Code Unity

-> Fix lỗi Ferecence đến Visual studio

-> Fix lỗi FrameWork

-> Fix Lỗi IOS-.. (Built -> Advent -Langues)

-> Fix lỗi Tool OF Unity -> TRUE ALL

https://www.google.com/search?ei=jOcDXYGjK5fW-QajnqHQDw&q=visual+studio+code+unity+autocomplete+not+working&oq=ide+autocomplete+d%C3%B9ng+code+unity&gs_l=psy-ab.1.0.0i71l8.0.0..7306...0.0..0.0.0.....0.....gws-wiz.Y1OarHfRaBo

https://stackoverflow.com/questions/42597501/autocompletion-not-working-in-visual-studio

//Unity sẽ khởi tạo từ :

Start()

//Unity sẽ 1 frame

Update()

//Unity sẽ 0.2s

FixedUpdate()

// Phím Tắt

Alt: Bàn Tay

R: Thay đổi kích thước

W: Di Chuyển

Ctrl: Hiện All Vùng Cắt trong Editor Sprite

Ctrl+D : Tạo bảng sao

// Thuộc tính cơ bản Unity

Order in Layer: độ ưu tiên layer hiển thị trước

//Bắt Đầu

Tạo Thư Mục:

- Sprites

- Animations

- Scenes

- Scripts

- Graphics

Đưa Những cái Hình Sheet Vào Sprites.

//Cắt Ảnh.

- Sprite Mode : Multiple.

- Pixels Per Unit: 19

- Sprite Editor -> Slice -> Tự Chỉnh(Dùng chuột kéo) | Auto -> đặt Tên .(Đè Ctrl để hiện cắt).

- Filter Mode -> Point

//Đưa Hình vào khung làm việc

- Kéo thả ảnh vào

//Tạo nền đất lặp lại

- Draw Mode: Tiled

- Tile Mode : Adaptive

-> Kéo Hình dài ra

- Đưa component

#Box Collider 2D

Copy size vào size của collider2d

//Tạo vật lí cho người chơi

#Đưa Box Collider 2D // Set khối hợp

#Rigidbody 2D // Set Vật Lí

//Tạo Animation

- Create -> Save Vào thư mục animation

- Đưa vào 3 hình ảnh đứng yên vào animation idle

#Samples : Thời gian số càng nhỏ càng chậm

- Create new Clip -> Đưa hình ảnh đứng yên vào (Walking)

- Create new Clip -> Đưa hình ảnh nhảy vào (Jump).

//Animator

Chọn Nhân Vật.

-> Idle -> make selection -> walking và ngược lại

- ...

- + Speed (Float)

- + Grounded (bool)

Chọn đường make -> Tích bỏ Has Exit Time : Để cho nó mượt hơn , hình nó không bị trì hoãn

Settings -> Transition Duration (0).

- Conditions ->

Speed (0.1)

Grounded (true)

...

//Tạo Đứng -> Đi -> Nhảy

Idle -> Walking:

+ Speed : Greater > 0.1

+ Grounded: true

Walking -> Idle :

+Speed: Less < 0.1

+ Grounded: true

Walking -> Jump:

+ Speed : Greater > 0.1

+ Grounded: false

Jump -> Walking:

+Speed : Greater > 0.1

+ Ground: True;

Jump -> Idle:

+ Ground: true;

Idle -> Grounded:

+ Ground: false

//Fix lỗi Hình bị lật

Rigidbody2D-> Constaints -> tích vào Z

//Input Manager (Dùng để nhập vào phím điều kiển)

Edit -> Project Settings -> Input ->

- Negative Button : Left

- Positive Button : Right

- ....

//Lưu ý Khủng

Nếu sửa mấy biến script toàn cục thì phải cập nhật sửa số lại trên script unity, nếu không đừng trách tại sao lỗi

//Script

Scripts -> create -> C# Script -> Đặt Tên -> Gán Script vào cho layer

//Các Kiểu dữ liệu:

Vector3 v3= new Vector3(x,y);//tạo ra 3 vector,z tự hiểu là 0

public GameObject number; // Phải Đặt File PreFab vào để có thể phân phối

//Ngẫu Nhiên Trong Tọa độ x,y

Random.Range(x,y);

//Vector

Vector2.right: chỉ ảnh hưởng giá trị x(1,0)

```
//Tạo cho Game Object
        Instantiate(Nguyên mẫu,Vị trí,Độ Xoay);
//Hoán tính Velocity
        r2.velocity.x; //lấy giá trị x
        r2.velocity.y; // lấy giá trị y
        r2.AddForce(); Thêm Vào


//using library
        using UnityEngine;
//Hàm Mặc định
        void Start(){}//Khởi đầu
        void Update(){}//Chạy liên tục 1 frame
        void FixUpdate(){} // Update mỗi 0.2s //liên quan vật lí
//Khai báo biến mà unity có thể thấy:
         public float speed = 50f, maxspeed = 3 , jump=220f;
         public bool grounded = true,faceright = true;

//Các Thành Phần
         public Rigidbody2D r2; // Dùng để lấy các biến của component đó.
         public Animator anim;// Dùng lấy animator



//Tạo duy chuyển cơ bản
        #Di Chuyển trái và phải
        -Khởi tạo các thành phần trên
        void Start(){
                r2 = gameObject.GetComponent<Rigidbody2D>();
```

```
        anim = gameObject.GetComponent<Animator>();
    }
    void Update(){
        anim.SetBool("Grounded",grounded);//Set chuyển đổi true
        anim.SetFloat("Speed",Mathf.Abs(r2.veloctity.x));//Chỉ trả về giá trị dương, giá trị thật
sự của người chơi
    }
    void FixedUpdate(){//cẩn thật quên Fixed
        //Đi
        float h = Input.GetAxis("Horizontal");//lấy phím trong InputManager
        r2.AddForce(Vector2.right * speed * h);// h (-1-> 0 -> 1) // Đưa lực vào
        //Giới hạn quán tính
         if (r2.velocity.x > maxspeed)
        {

            r2.velocity = new Vector2(maxspeed, r2.velocity.y);
        }
         if (r2.velocity.x < -maxspeed)
        {
             r2.velocity = new Vector2(-maxspeed, r2.velocity.y);
        }
    }
//Tạo Nhảy
    - Trong hàm Update(){
         if (Input.GetKeyDown(KeyCode.Space))
         {

            if (grounded)
            {
```

```
                    r2.AddForce(Vector2.up * jump);//Vector2.up (0,1) ,
                    grounded = false;
                }

            }
        }
```

//Thay đổi ảnh

-Trong Hàm FixedUpdate(){

```
                if (h > 0 && !faceright)//nhấn > , !mặt phải
                {
                    Flip();
                }
                if (h<0 && faceright)
                {
                    Flip();
                }
```

- Hàm Flip(){

```
                faceright = !faceright;// đổi mặt
                Vector3 scale;
                scale = transform.localScale;//x y z dòng 3 ,
                scale.x *= -1;//đảo lại
                transform.localScale = scale; // gán lại
            }
```

//Kiểm tra Nhảy

Thêm 1 BoxCollider2D và kéo xuống dưới nhân vật bằng x , y và offset

Tích vào IsTriger = true

Tạo Script mới

//Thuộc tính

PlayerManager player;

class GroundCheck:MonoBehaviour{

void Start(){

player = gameObject.GetComponentInParent<PlayerManager>(); // truy Xuất đến đối tượng cha mẹ

}

void FixedUpdate(){

}

```
void OnTriggerEnter2D(Collider2D collision)//enter
{
    player.grounded = true;
}

void OnTriggerStay2D(Collider2D collision)//giữ
{
    player.grounded = true;
}

void OnTriggerExit2D(Collider2D collision)//không còn va chạm
{
```

```
        player.grounded = false;
    }



}

//Double Click
//Thêm thuộc tính:
            doubleJump = false;
//hàm trong PlayerManager
        if (grounded)
        {
            r2.AddForce(Vector2.up * jump);
            doubleJump = true;
            grounded = false;
        }
        else
        {
            if(doubleJump){
                doubleJump = false;
                r2.velocity = new Vector2(r2.velocity.x, 0);// y không đổi
                r2.AddForce(Vector2.up * jump*0.7f); // độ đi được
            }
        }

      }
```

```
//Toàn Bộ Code Nhân Vật
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerManager : MonoBehaviour
{
    public float speed = 50f, maxspeed = 3 , jump=220f;
    public bool grounded = true,faceright = true;
    public Rigidbody2D r2;
    public Animator anim;
    void Start()
    {
        //khởi tạo
        r2 = gameObject.GetComponent<Rigidbody2D>();
        anim = gameObject.GetComponent<Animator>();
    }

    void Update()// frame = 0.33..
    {
        //set parameter vào animator
        anim.SetBool("Grounded",grounded);
        anim.SetFloat("Speed",Mathf.Abs(r2.velocity.x));//giá trị dương , vị trí hiện tại


        if (Input.GetKeyDown(KeyCode.Space))
        {

            if (grounded)
```

```
        {
            r2.AddForce(Vector2.up * jump);
            grounded = false;
        }

    }
}

void FixedUpdate()// 2s
{
    //Đi
    float h = Input.GetAxis("Horizontal");
    r2.AddForce(Vector2.right * speed * h);
    //Giới hạn quán tính
    if (r2.velocity.x > maxspeed)
    {

        r2.velocity = new Vector2(maxspeed, r2.velocity.y);
    }
    if (r2.velocity.x < -maxspeed)
    {
        r2.velocity = new Vector2(-maxspeed, r2.velocity.y);
    }

    if (h > 0 && !faceright)
    {
        Flip();
    }
    if (h<0 && faceright)
```

```
        {
            Flip();
        }
    }

    void Flip()
    {
        faceright = !faceright;
        Vector3 scale;
        scale = transform.localScale;//x y z dòng 3
        scale.x *= -1;//đảo lại
        transform.localScale = scale;



    }

}



//Fix lỗi Kẹt vào gốc
            1.Tích vào Used By Effector cho nền đất
            2.Add Component -> Platform Effector 2D ->  Bỏ Tích Use One Way

//Giảm tốc độ 70 %
      if (grounded)
        {
            r2.velocity = new Vector2(r2.velocity.x * 0.7f, r2.velocity.y);
        }
```

// Chỉnh độ khoản cách giữa 2 BoxCollider

Edit -> Project Settings -> Default Contact Offset 0.01 // sát nhưng mà không va chạm nhau

//Rigidbody2D.

Body Type:

+ Dynamic //Có tác động vật lí

+ Static //Tĩnh

+ Kinematic // Bay bồng bền . +tích thêm Use Full Kinematic

//Tạo Vật thể rơi xuống sau 2s

Tạo 1 Script gắn vào cái thứ cần rơi.

//Thuộc tính

```
public Rigidbody2D r2;
public float timeDelay=2f;
```

//Start(){}//khởi tạo Rigidbody2D

//phương thức

```
void OnCollisionEnter2D(Collision2D col) // gọi khi dụng đụng thằng khác
  {
    if (col.collider.CompareTag("Player")) // so sánh với tag của thằng nó đụng
    {
      StartCoroutine(fall()); // mới gọi được Interface IEnumerator
    }
  }

  IEnumerator fall()
  {
    yield return new WaitForSeconds(timeDelay);// chạy đến đây bắt đầu thời gian , khi hết
    r2.bodyType = RigidbodyType2D.Dynamic; //sẽ chạy tiếp ở đây
```

```
    yield return 0;// tắt

  }

//Cách tạo tag
        vào object cần tạo -> tag -> new tag -> nhấn vào dấu +

//Di chuyển qua lại

        // Phải set thêm cái Rigidbody2D
                + sử dụng kinematic
                + sử dụng thêm tag
                + sử dụng thêm platform effect 2D
                + BoxCollider2D thêm cai use effect

        //Thuộc tính
         public float speed = 0.05f, changeDirection = -1;
        Vector3 Move;

        //Phương thức
 void Start()
  {
    Move = transform.position;
  }

  // Update is called once per frame
  void Update()
  {
    Move.x += speed;
```

```
        transform.position = Move;
    }

    void OnCollisionEnter2D(Collision2D col)
    {
        if (col.collider.CompareTag("Ground"))
        {
            speed *= changeDirection;
        }
    }

//Tạo Camera Theo người chơi

            //Các thuộc tính
public float smoothtimeX, smoothtimeY;//thời gian trì hoãn
    public Vector2 velocity;

    public GameObject player;// khởi tạo đối tượng player
    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");//tìm đúng tag player gắn vào để follow
theo
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        float posX = Mathf.SmoothDamp(
```

```
        this.transform.position.x,
        player.transform.position.x,
        ref velocity.x,
        smoothtimeX
        );

    float posY = Mathf.SmoothDamp(
        this.transform.position.y,
        player.transform.position.y,
        ref velocity.y,
        smoothtimeY
        );//hiện tại , mục tiêu tới , dạng gì , thời gian chuyển từ current -> target

    transform.position = new Vector3(posX,posY,transform.position.z);//đổi vị trí theo player

  }

//Giới hạn camera
        //Thêm thuộc tính vào
  public Vector2 minpos, maxpos;
  public bool bound;

        //bổ xung phương thức
        if (bound)
    {
        transform.position =new Vector3(
          Mathf.Clamp(transform.position.x,minpos.x,maxpos.x),//hàm ngày để giới hạn trong khoản đó
          Mathf.Clamp(transform.position.y, minpos.y,maxpos.y),
          Mathf.Clamp(transform.position.z, transform.position.z, transform.position.z)
```

```
            );
        }
//Các UI
            Canvas
                        Panel
                        Button
                        Text
//Tạo Canvas để làm màng hình dừng
            Chuột phải ->UI -> Canvas
//Nhúng vào cammera
            Chọn Canvas Render Mode -> Screen Space -Camera
//Button -> text
            font size: 30
            Horizontal Overflow  : overflow
            Vertical Overflow : overflow
//Thuộc tính của GameObject
            setActive(true); // Ẩn hiện đối tượng đó

//Tạo Menu Dừng
            Đưa tất cả các cái trong canvas vào 1 gameObject
            Vào Camera kéo script vào

            //Thuộc tính
             private bool pause;
              public GameObject PauseUI;


            //Hàm
 // Start is called before the first frame update
```

```
void Start()
{
    PauseUI.SetActive(false);
}

// Update is called once per frame
void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        pause = !pause;
    }
    if (pause)
    {
        PauseUI.SetActive(true);//Ẩn hiện trên GameObject
        Time.timeScale = 0;// dừng hẳng 0.1 thì nó sẽ chạy 10%
    }
    if (!pause)
    {
        PauseUI.SetActive(false);
        Time.timeScale = 1; // 100% chạy
    }

}



public void Resume()
{
    pause = false;
```

```
}

public void Restart()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);// Thay đổi cái scence hiện tại thành active scene
}

public void Quit()
{
    Application.Quit();//đóng không tác dụng trên máy ảo
}
```

//Thêm Onclick vào button

Nhấn vào button cần -> Onclick -> Click + -> Truyền Đối tượng Main Camera vào -> tìm hàm cần click

//Thêm Dừng cho các vật liên quan

Vào MovingFalt

```
public PauseMenu pausep;
```

Bổ xung Phương Thức

```
Start(){
        pausep = GameObject.FindGameObjectWithTag("MainCamera").GetComponentInParent<PauseMenu>();
}
Update(){
    if (pausep.pause)
    {
        this.transform.position = this.transform.position;
```

```
                }


                 if (!pausep.pause)
                {
                        Move.x += speed;
                        transform.position = Move;
                }
        }
```

//Tạo HP

//Thêm gameObjectEmpty vào canvas -> UI -> Image -> Gán hình vào source Image -> tích vào Preserve Aspect

//Thêm vào playerManager

```
        //Thuộc tính
        public int ourHealth;
         public int maxHealth=5;
        //Phương thức .
        Start(){
                ourHealth = maxHealth;//khởi đầu đầy máu
        }


        FixedUpdate(){
                 //kiểm tra máu
                if (ourHealth <= 0)
                {
                     Death();
                  }
        }
```

```
        public void Death()
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        }
    //Tạo Script và kéo vào GameObjectEmpty
        //Thư viện
using UnityEngine.UI;
        //Thuộc tính
public Sprite[] Heartsprite;
public PlayerManager player;
public Image Heart;
// Start is called before the first frame update
void Start()
{
   player = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerManager>();
}

// Update is called once per frame
void Update()
{
   if (player.ourHealth > 5)
   {
      player.ourHealth = 5;
   }
   if (player.ourHealth < 0)
   {
      player.ourHealth = 0;
   }
```

```
    Heart.sprite = Heartsprite[player.ourHealth];
  }

//Tạo Gai
        Đưa Gai Vào -> Tạo BoxCollider2D
                    Tạo Script
        //Thuộc tính
                public PlayerManager player;
        //Phương Thức
  // Start is called before the first frame update
  void Start()
  {
    player = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerManager>();
  }

  // Update is called once per frame
  void OnTriggerEnter2D(Collider2D col)
  {
    if (player.CompareTag("Player"))
    {
      player.Damage(2);
    }
  }

        //Bổ sung thêm hàm trong PlayerManager
        public void Damage(int damage)
        {
                ourHealth -= damage;
        }
```

//Fix lỗi ở gai và addForce

//Thêm Thuộc tính vào player

,maxJump =4;

//Phương thức:

FixUpdate()

if (r2.velocity.y > maxJump)

{

r2.velocity = new Vector2(r2.velocity.x, maxJump);

}

if (r2.velocity.y < -maxJump)

{

r2.velocity = new Vector2(r2.velocity.x, -maxJump);

}

Bổ xung hàm KnowBack Khi va chạm

public void KnockBack(float knockpow , Vector2 knowdir)//lực , vector

{

r2.velocity = new Vector2(0, 0);

r2.AddForce(new Vector2(knowdir.x * -100 , knowdir.y * knockpow));

}

//Gọi hàm bên Gai chỗ Trigger

player.KnockBack(400f,player.transform.position);

//Tạo thêm hiệu ứng chóp đỏ khi đụng phải

+Tạo Ra 1 Animation cho player -> new Clip -> Add Property -> Sprine Renderer -> Color +

+Ghi Record

+ Vào Player -> add component -> thêm animation vào

+ Animations -> size: 1

+ Kéo cái File RedFlash.anim lúc tạo vào ô Element 0 :

+ Bỏ Check Play Automaticcally

+ Muốn cho nó chạy thì -> Nhấn vào animation RedFlash -> Nhấn vào debug -> Check vào Legacy

//Bổ xung hàm vào

```
void Damage(int damage){
    gameObject.GetComponent<Animation>().Play("RedFlash");
}
```

//Tạo Nhân Vật có tấn công

+Tạo Ra 1 animation Tấn công

+ Tạo ra 1 gameObjectEmpty trong Player

+ Thêm BoxCollider vào, Check Is Trigger

+ Chỉnh Pivot lại không nó lệch , vào sprine Editor để chỉnh lại

+ Tạo ra 2 Script . +1 cho player

+1 Cho GameObjectEmpty

+ Make Selection cho PlayerAttack

+ Idle -> PlayerAttack :

Attackiing : true

Speed : lest < 0.1

Grounded: true

+ PlayerAttack -> Idle

Attacking : false

Grounded: true

+ Jump -> PlayerAttack

Attacking : true

Grounded: false

+PlayerAttack -> Jump

Attacking: false:

Grounded: false

+ Walking -> PlayerAttack

Attacking: true

Speed: Greates > 0.1

+PlayerAttack -> Walking

Attacking :false

Speed:Greates > 0.1

```
//Trong Script PlayerAttack
    public float attackdelay = 0.3f;//0.3s
public bool attacking = false;

public Animator anim;
public Collider2D trigger;


// Start is called before the first frame update
void Awake()//Awake chạy trước start và không ảnh hưởng bởi enable
{
   anim = gameObject.GetComponent<Animator>();
   trigger.enabled = false;//không kích hoạt triger
}

// Update is called once per frame
```

```
  void Update()
  {
    if (Input.GetKeyDown(KeyCode.Z) && !attacking)//khi nhấn và chưa tấn công
    {
      attacking = true;
      trigger.enabled = true;
      attackdelay = 0.3f; // gắn lại để không nó giảm luôn
    }

    if (attacking)
    {
      if (attackdelay > 0)
      {
        attackdelay -= Time.deltaTime;//Thời gian 1s, 2s... thực không bị ảnh hưởng bởi Frame
      }
      else
      {
        attacking = false;
        trigger.enabled = false;
      }

      anim.SetBool("Attacking", attacking);//thay đổi attacking trong animator
    }
  }

        //Lưu ý : Nhớ kéo AttackTrigger vào Trong Script trigger

//Tạo Enemy

        //Script AttackTrigger
```

```
        public int dmg = 20;
    // Start is called before the first frame update
    private void OnTriggerEnter2D(Collider2D col)
    {
        if (col.isTrigger != true && col.CompareTag("enemy"))
        {
            col.SendMessageUpwards("Damage", dmg);// Gọi Xuống hàm void Damage(int damage)//dưới
kia
        }
    }




        //Script Box
        public int Health = 100;
    // Start is called before the first frame update



    // Update is called once per frame
    void Update()
    {
        if (Health <= 0)
        {
            Destroy(gameObject);//xóa bỏ đối tượng hiện tại
        }
    }

    void Damage(int damage)
    {
```

```
    Health -= damage;
  }
```

//Tạo Trụ Nhìn trái nhìn phải bắn , và khi có người đi vào phạm vi thì bắn

Has Exit Time : sau khoảng thời gian đó nếu như chưa animation xong thì bỏ luôn frame còn lại

.....

-Tạo ra

TurretAwake(Awake:true)

TurretAsSleep(Awake:false)

TurretSleep(Mặc định Awake:false)

TurretLookLeft(LookRight:false)

TurretLookRight(LookRight:true)

TurretToLeft(LookRight:false)

TurretToRight(LookRight:true)

- Nối lại thành 1 vòng chỉ cần 1 đường là đủ. khi muốn đứng lại và dừng ở chỗ nào thì true 1 bên thì bên kia cũng qua luôn.

//Script TurretAI

```
public float curHealth = 100;
  public float distance;
  public float wakerange;
  public float shootinterval;
  public float bulletspeed = 5;
  public float butlettimer;

  public bool awake = false;
  public bool lookingRight = true;
```

```
public GameObject bullet; // Viên đạn
public Transform target; //vị trí của người chơi
public Animator anim;
public Transform shootpointL, shootpointR;//vị trí 2 điểm bắn

// Start is called before the first frame update
void Awake()
{
    anim = GetComponent<Animator>();
}

// Update is called once per frame
void Update()
{
    // Kiểm tra 2 cái biến trong animator
    anim.SetBool("Awake",awake);
    anim.SetBool("LookRight",lookingRight);

    RangeCheck();//Gọi



    if (target.transform.position.x > transform.position.x)
    {
        lookingRight = true;
    }

    if (target.transform.position.x < transform.position.x)
    {
```

```
            lookingRight = false;
        }


        if (curHealth < 0)
        {
            Destroy(gameObject);
        }


    }



    void RangeCheck()
    {
        distance = Vector2.Distance(transform.position, target.transform.position);//tính khoảng cách giữa trụ và người chơi
        if (distance < wakerange)
        {
            awake = true;
        }
        if (distance > wakerange)
        {
            awake = false;
        }
    }


    public void Attack(bool attackright)
    {
        butlettimer += Time.deltaTime;//Thời gian thực
        if (butlettimer >= shootinterval)
```

```
    {
        Vector2 direction = target.transform.position - transform.position;
        direction.Normalize();//Độ Nhân Bình thường hóa lại
        if (attackright) // bắn tay phải
        {
            //bị gọi nhiều lần nên phải clone để ko tốn thòi gian,clone lại
            GameObject bulletclone;
            bulletclone = Instantiate(bullet, shootpointR.transform.position,
shootpointR.transform.rotation) as GameObject;//đối tượng , vị trí bắt đầu , xoay
            bulletclone.GetComponent<Rigidbody2D>().velocity = direction * bulletspeed;

            butlettimer = 0; // reset lại

        }
        if (!attackright) // bắn tay phải
        {
            //bị gọi nhiều lần nên phải clone để ko tốn thòi gian,clone lại
            GameObject bulletclone;
            bulletclone = Instantiate(bullet, shootpointL.transform.position,
shootpointL.transform.rotation) as GameObject;//đối tượng , vị trí bắt đầu , xoay
            bulletclone.GetComponent<Rigidbody2D>().velocity = direction * bulletspeed;

            butlettimer = 0; // reset lại

        }
    }
```

```
//AttackCone(Vùng  Tấn công)

public TurretAI turret;
   public bool isLeft = false;



   // Start is called before the first frame update
   void Awake()
   {
      turret = gameObject.GetComponentInParent<TurretAI>();
   }

   // Update is called once per frame
   void OnTriggerStay2D(Collider2D col)
   {
      if (col.CompareTag("Player"))
      {
         if (isLeft)
         {
            turret.Attack(false);
         }

         if (!isLeft)
         {
            turret.Attack(true);
         }
      }
   }
```

```
//Script AttackTrigger
 public int dmg = 20;
    // Start is called before the first frame update
    private void OnTriggerEnter2D(Collider2D col)
    {
        if (col.isTrigger != true && col.CompareTag("enemy"))
        {
            col.SendMessageUpwards("Damage", dmg);
        }
    }



//Đạn  Script Bullet
public PlayerManager player;
    public float lifetime = 2;



    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerManager>();
    }

    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.isTrigger != true)
        {
            if (col.CompareTag("Player"))
```

```
        {
            col.SendMessageUpwards("Damage",1);
        }
        Destroy(gameObject);
    }
}

// Update is called once per frame
void Update()
{
    lifetime -= Time.deltaTime;
    if (lifetime <= 0)
    {
        Destroy(gameObject);
    }
}
```

//Lưu Ý:

Trước khi làm Tạo ra

1.GameObject Collider

1. AttackTriggerLeft

1.AttackTriggerRight

1.ShootPointLeft

1.ShootPointRight

+ Tự Căn chỉnh nhé

```
//Fix lỗi Khi Nhân vật dụng vào đâu thì trigger chỗ đó nhảy max cao
        void OnTriggerEnter2D(Collider2D collision)//enter
```

```
    {
        if (!collision.isTrigger)
        {
            player.grounded = true;
        }

    }

    void OnTriggerStay2D(Collider2D collision)//giữ
    {
        if (!collision.isTrigger)
        player.grounded = true;
    }

    void OnTriggerExit2D(Collider2D collision)//không còn va chạm
    {
        if (!collision.isTrigger)
        player.grounded = false;
    }
```

//Tạo Vùng Die ở phí dưới .

Tái sử dụng lại gai vào chuyển hình ảnh thành none

Kéo xuống dưới Tăng Damage = 5 die

//Tạo Score

Tao 1Text Trong Canvas

Tạo 1 GameObject -> Tên GameMaster

Viết Script

```
        public int Points = 0;
    public Text PointText;



    // Update is called once per frame
    void Update()
    {
        PointText.text=("Score:"+Points);
    }

        // Kéo Text Vào Trong Script của GameMaster
//Tạo Coins
        Vào Script PlayerManager Bổ xung
        public GameMaster gm;
        //Hàm Start
        gm = GameObject.FindGameObjectWithTag("GameMaster").GetComponent<GameMaster>();

        //Thêm Hàm
        public void OnTriggerEnter2D(Collider2D col)
    {
        if (col.CompareTag("Coins"))
        {
            Destroy(col.gameObject);
            gm.Points += 1;
        }
    }

//Tạo Door và cộng điểm
         public int LevelLoad = 1;
```

```
public GameMaster gm;

// Start is called before the first frame update
void Start()
{
    gm = GameObject.FindGameObjectWithTag("GameMaster").GetComponent<GameMaster>();
}

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

}
private void OnTriggerEnter2D(Collider2D col)
{
    if (col.CompareTag("Player"))
    {
        saveScore();
        gm.InputText.text = ("Press E to enter");
    }
}
private void OnTriggerStay2D(Collider2D col)
{
    if (col.CompareTag("Player"))
    {
        if (Input.GetKey(KeyCode.E))
        {
            saveScore();
            SceneManager.LoadScene(LevelLoad);
```

```
        }
    }
}
private void OnTriggerExit2D(Collider2D col)
{
    if (col.CompareTag("Player"))
    {
        gm.InputText.text = ("");
    }
}
void saveScore()
{
    PlayerPrefs.SetInt("points", gm.Points);
}

//Bổ xung bên GameMaster
        public int Points = 0;
public int hightscore = 0;

public Text PointText;
public Text HightText;
public Text InputText;


void Start()
{
    HightText.text = ("HighScore:" + PlayerPrefs.GetInt("highscore"));
    hightscore = PlayerPrefs.GetInt("highscore",0);
    if (PlayerPrefs.HasKey("points"))
```

```
    {
        Scene ActiveScreen = SceneManager.GetActiveScene();
        if (ActiveScreen.buildIndex ==0)
        {
            PlayerPrefs.DeleteKey("points");
            Points = 0;
        }
        else
        {
            Points = PlayerPrefs.GetInt("points");
        }
    }

  }
  // Update is called once per frame
  void Update()
  {
     PointText.text=("Score:"+Points);
  }

//Bổ xung thêm PlayerManager . Hàm Death();
          if (PlayerPrefs.GetInt("highscore") < gm.Points)
        PlayerPrefs.SetInt("highscore", gm.Points);


//Tạo Audio
          Tạo 1 GameObjectEmpty Quản lí âm thanh
          Thêm Component audiosource vào.
          Tạo Thêm Thực mục Resources -> bỏ audio .wav.
```

Viết script

```
public AudioClip coins, swords, destroy;

  public AudioSource adisrc;



  // Start is called before the first frame update
  void Start()
  {
     coins = Resources.Load<AudioClip>("Game coin");
     swords = Resources.Load<AudioClip>("Sword");
     destroy = Resources.Load<AudioClip>("Rock Crash");
     adisrc = GetComponent<AudioSource>();
  }

  // Update is called once per frame
  public void Playsound(string clip)
  {
     switch (clip)
     {
        case "coins":
              adisrc.clip = coins;
              adisrc.PlayOneShot(coins,0.6f);
           break;
        case "destroy":
               adisrc.clip = destroy;
               adisrc.PlayOneShot(destroy,1f);
           break;
```

```
        case "sword":
            adisrc.clip = swords;
            adisrc.PlayOneShot(swords, 1f);
            break;
    }
}
```

//Cách dùng

//Khởi tạo

```
public SoundManager sound;
```

//Hàm Start

```
sound = GameObject.FindGameObjectWithTag("sounds").GetComponent<SoundManager>();
```

Hàm OnTriggerEnter2D thêm

```
sound.Playsound("coins");
```

//Lưu ý : Nhớ tạo thư mục Resources phải đúng , không cần kéo audio vào mấy cái khởi tạo mà khi chạy nó sẽ tự xuất hiện nhớ kiểm ta

//Tạo Nhiều âm thanh cùng lúc

Thêm Component audiosource vào vật cần phát.

Viết script

//Khởi tạo

```
public AudioSource audiosrc;
public AudioClip audioClip;
```

//Hàm start

```
audiosrc = gameObject.GetComponent<AudioSource>();
```

//Dùng

```
audiosrc.PlayOneShot(audioClip,0.6f);//clip , volume
```

//Nhớ kéo audio . wav vào thuộc tính audioClip

//Tạo Vùng Nước

Tìm 1 hình nước nhỏ rồi chỉnh lớn

..

Thêm Component

+Rigidbody2D : Kime...

+BoxCollider2D : Thêm IsTrigger , Use By Effector

+Are Effector 2D:

-Force Angle: 90 , Đẩy lên theo hướng 90*.

-Force Magnitude: 10

Viết Script Thêm Vào PlayerGroundCheck

```
void OnTriggerEnter2D(Collider2D collision)//enter
{
   if (!collision.isTrigger || collision.CompareTag("water"))
   {
      player.grounded = true;
   }

}

void OnTriggerStay2D(Collider2D collision)//giữ
{
   if (!collision.isTrigger || collision.CompareTag("water"))
   player.grounded = true;
}

void OnTriggerExit2D(Collider2D collision)//không còn va chạm
```

```
    {
        if (!collision.isTrigger || collision.CompareTag("water"))
        player.grounded = false;
    }
```

//Tạo BackGround

Chỉnh hình ảnh:

TextTure Type : Default.

Wrap Mode: Repeat

Filter Mode: Point

Vào Camera -> New -> 3D Object -> Quad.

Kéo ảnh vào , Căn chỉnh hết màng hình

Chỉnh position : z: 100

Chỉnh độ sáng:

Vào Exterior- para... -> Sharder -> StandDard Chuyển thành UI -> Unlit -> text -> Xong

//Tạo Script Scroll

```
    public PlayerManager player;
    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerManager>();
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        Vector2 offset = GetComponent<MeshRenderer>().material.mainTextureOffset;
```

```
    offset.x = player.transform.position.x;
    GetComponent<MeshRenderer>().material.mainTextureOffset = offset * Time.deltaTime;
}
```