Tabla de contenido

[1. Hoja de sprites 2](#_Toc6682735)

[2. Project Settings 2](#_Toc6682736)

[2.1. Input 2](#_Toc6682737)

[2.1.1. Eventos de teclado 2](#_Toc6682738)

[3. Interactuar con las partes de un objeto: GetComponent 2](#_Toc6682739)

[4. Animator 4](#_Toc6682740)

[4.1. Blend Trees 4](#_Toc6682741)

[5. Colisiones 5](#_Toc6682742)

[6. Capas 5](#_Toc6682743)

[7. UI interfaces y Canvas 5](#_Toc6682744)

[7.1. Texto flotante 5](#_Toc6682745)

[7.2. Barra de vida 5](#_Toc6682746)

[8. Sistema de partículas 6](#_Toc6682747)

[8.1. Sangre 6](#_Toc6682748)

[9. NPC 6](#_Toc6682749)

[9.1. Movimiento 6](#_Toc6682750)

[10. Composición (POO) en MonoBehaviour 7](#_Toc6682751)

[11. Triggers 7](#_Toc6682752)

[12. Audio 7](#_Toc6682753)

[13. Escenas 7](#_Toc6682754)

[13.1. Cambio de escenas 7](#_Toc6682755)

[14. Persistencia de datos 7](#_Toc6682756)

[15. Comunicación entre scripts con SendMessage 7](#_Toc6682757)

[16. Desactivación en lugar de destrucción 8](#_Toc6682758)

[17. Transform 8](#_Toc6682759)

[18. Búsqueda de objetos dentro del juego 8](#_Toc6682760)

[19. Timers 8](#_Toc6682761)

[20. Números aleatorios 8](#_Toc6682762)

[21. Instanciación de objetos 8](#_Toc6682763)

[22. Rigidbody y fuerzas que interactúan con él 8](#_Toc6682764)

[23. GUI 9](#_Toc6682765)

[24. Disparos con raycast 10](#_Toc6682766)

[24.1. Grappling hook basic 10](#_Toc6682767)

[25. Line renderer 3d 12](#_Toc6682768)

[26. Cursor 14](#_Toc6682769)

[27. Luces 15](#_Toc6682770)

[28. Draw Lines y Gizmos 15](#_Toc6682771)

**Conceptos Unity**

# Hoja de sprites

Al agregar una hoja de sprites en la parte de edición se puede indicarle de cuantos pixeles está conformada cada sub-imagen; las imágenes también se pueden recortar de modo automático en la herramienta de “Sprite Editor” además es posible agregar márgenes y un padding al rededor. Si la imagen es borrosa se debe cambiar el “Filter Mode” a point en lugar de bilinear.

# Project Settings

## Input

GetAxisRaw permite obtener el valor que el input por defecto cambia según presionemos las flechas

Todo esto definido en input

### Eventos de teclado

if(Input.GetKey (KeyCode.UpArrow))

# Interactuar con las partes de un objeto: GetComponent

GetComponent<>()

public Color color1 = Color.red;  
    public Color color2 = Color.blue;  
    public float duration = 3.0F;  
  
    Camera cam;  
    bool ortho = false;  
  
    public Transform objetivo;  
  
    void Start()  
    {  
        cam = GetComponent<Camera>();  
        cam.clearFlags = CameraClearFlags.SolidColor;  
  
        *//distancia del renderizado*  
        *//cam.farClipPlane = 20.0f;*  
    }  
  
    void Update()  
    {  
  
        *//sigue con la mirada al objetivo debe ser asignado luego*  
        transform.LookAt (objetivo);  
  
  
  
  
        if (Input.GetKeyDown (KeyCode.Alpha3))   
        {  
            ortho=!ortho; *// este es un flag*  
            cam.orthographic = ortho;  
  
        }  
  
        if (Input.GetKeyDown (KeyCode.Alpha4))   
        {  
            *//distancia del renderizado aumenta*  
            cam.farClipPlane++;  
        }  
  
        if (Input.GetKeyDown (KeyCode.Alpha5))   
        {  
            cam.farClipPlane--;  
        }  
  
        if (Input.GetKeyDown (KeyCode.KeypadPlus))   
        {  
            *//la suma va en grados para la apertura de la vista de la caamara*  
            cam.fieldOfView += 5;  
        }  
  
        if (Input.GetKeyDown (KeyCode.KeypadMinus))   
        {  
            cam.fieldOfView -= 5;  
        }  
  
  
        */\**  
*if (Input.GetKeyDown (KeyCode.Alpha1))*  
*{*  
*cam.backgroundColor = color1;*  
*}*  
  
*if (Input.GetKeyDown (KeyCode.Alpha2))*  
*{*  
*cam.backgroundColor = color2;*  
*}*  
*//float t = Mathf.PingPong(Time.time, duration) / duration;*  
*//cam.backgroundColor = Color.Lerp(color1, color2, t);\*/*

# Animator

Controla los clips de animaciones con máquinas de estado.

* animator.SetFloat ("moveX",Input.GetAxisRaw("Horizontal"));

Según las variables se producen transiciones ya que desde el código se puede cambiar la máquina de estado a ejecutarse

* “Fixed duration” marcado indica que no sea en bucle el movimiento, es necesario desmarcarlo en la mayoría de ocasiones.
* “Transition duration” es el tiempo que se demora en pasar de una animacion a otra, de preferencia para el 2d en movimiento ese valor debe ser 0

Animator animator = GetComponent<Animator> ();

animator.Play ("saltar",-1,0f);

## Blend Trees

Es una forma organizacional de estados

# Colisiones

Para que los objetos colisionen no basta con las “box colider 2D” es necesario “Rigidbody 2D”

* void OnTriggerEntered2D(Collider2D other){}
* void OnCollisionEntered2D(Collision2D other){}

También hay la versión Exit, Enter y la persistida

public void OnCollisionEnter(Collision obj\_col){

if (obj\_col.gameObject.name == objetivo\_desaparicion)

Destroy (obj\_paracaidas.gameObject, 2f);

if (obj\_col.gameObject.name == "barco\_juego")

Destroy (obj\_paracaidas.gameObject); }

void OnTriggerEnter(Collider obj) *//OnTriggerExit,OnTriggerStay*  
    {   
        *//transform.Translate (10, 0, 0);*  
        *//Debug.Log ("Trigger {0}"+obj.gameObject.name);*  
    }  
  
    void OnCollisionEnter(Collision obj)  
    {   
          
        if (obj.gameObject.name == "cubo3")   
        {   
            *// para muchos objetos es mejor usar tags*  
            Debug.Log ("Colisiono con el tercero");  
            Destroy (obj.gameObject);  
        }      
              
    }  
  
    void OnControllerColliderHit(ControllerColliderHit obj\_col)  
    {   
        if (obj\_col.gameObject.name == "cubo1")  
            Destroy (obj\_col.gameObject);  
    }

# Capas

“Sorting layers” indica que objeto irá encima de otro.

Podemos apoyarnos en el vector z para esto pero no es muy aconsejable.

1. Instanciación de un objeto especifico a partir de uno genérico (prefab)

* var clone = (GameObject) Instantiate (danio\_flotante, col.gameObject.transform.position, Quaternion.Euler (Vector3.zero));

clone.GetComponent<NumerosFlotantes> ().numero = damage;

El código de arriba clona el prefab para permitir la modificación de sus atributos

# UI interfaces y Canvas

## Texto flotante

Primero se lo aparta del canvas agregándole la opción “world space” de esa manera no se ata al UI. El texto se lo crea en una posición y se aumenta su eje y.

## Barra de vida

Con un slider se puede crear una barra de vida, eliminando las partes innecesarias

Se necesitan dos sprites:

1.-Marco

2.-Barra con los colores

Si son imágenes necesitan ser convertidas a sprites de manera interna

Se agrega la jerarquía:

Canvas

Imagen (con ancho y alto del marco/Barra, componente -> scrollbar y agregamos la mascara de abajo en handle rect y modificamos Size y Value)

Imagen(Marco)

Mascara (con el componente Mask y sin la casilla marcada, center -> stretch)

Imagen(Imagen de la vida llena)

Lo que está más abajo es loque se muestra al frente, es por eso que marco debe ir debajo de la más cara asi se muestra de frente

private float vida\_base = 100;

private float vida\_actual;

public Scrollbar barra\_vida;

public void reducirVida(float reducción){

if (vida\_actual >= reducción) {

vida\_actual -= reducción;

barra\_vida.size = vida\_actual / vida\_base;}}

# Sistema de partículas

## Sangre

Se usó la instanciación esférica, de color rojo, con tamaño reducido y creación de cuadros

# NPC

## Movimiento

Para limitar el movimiento de un npc en 2d lo encerramos en un box collider que solo detectará el mismo. También se puede usar la mecánica del 3D donde se usa EmptyObjects como target de una ruta (mirar el script del oso)

# Composición (POO) en MonoBehaviour

Para realizar composición en lugar de “new” debemos agregar componentes

* PlayerStats estadisticas = gameObject.AddComponent <PlayerStats>();

# Triggers

Los objetos vacíos pueden servir para activar cosas en el juego, esto es importante.

# Audio

Para agregar un sonido rápidamente se puede usar un prefab que cree algo y arrastrando el clip de sonido al mismo sonara al ser instanciado.

Otra manera es crear un controlador (con varios audios sources) con la música ya cargada. Agregamos esos controladores a nuestra clase donde habrá sonido y simplemente con la función .Play podemos reproducir el efecto cuando deba suceder

# Escenas

## Cambio de escenas

Primero se cargan en BuildSettings todas las escenas a utilizar

Luego en código dentro de un objeto trigger se importa using UnityEngine.SceneManagement; y se usa SceneManager.LoadScene (level2load); siendo level2load el nombre de la escena

# Persistencia de datos

DontDestroyOnLoad (transform.gameObject);

# Comunicación entre scripts con SendMessage

//el objeto dañino

if (obj\_col.gameObject.name == "jugador") {

obj\_col.gameObject.SendMessageUpwards ("reducirVida", 10,SendMessageOptions.DontRequireReceiver);

}

# Desactivación en lugar de destrucción

Con esto evitamos errores y simplemente lo hacemos no jugable

col.gameObject.SetActive (false);

# Transform

Para cambiar la posición local o rotación usamos

* this.transform.localPosition = new Vector3 (0.004f,-0.292f,0f);
* this.transform.eulerAngles = new Vector3(0f, 180f, -179.13f);

Importante en el rotation no es necesario usar Quaterniones

# Búsqueda de objetos dentro del juego

GameObject.Find ("plataforma");

# Timers

Uso de Time.deltaTime

# Números aleatorios

Random.Range(a,b);

# Instanciación de objetos

Instantiate (objeto, new Vector3(x,y,z), transform.rotation);

# Rigidbody y fuerzas que interactúan con él

RigidBody rb = GetComponent<Rigidbody> ();

//Hay 4 distintas fuerzas que se pueden agregar a un rigidbody

rb.AddForce (new Vector3 (x\*Time.deltaTime,0,0), ForceMode.Force);

relativeForce

# GUI

//mensajes en pantalla

void OnGUI(){

GUI.Label (new Rect (10, 60, 140, 20), "Cajas recogidas: " + cajas\_recogidas.ToString ());

GUI.Label (new Rect (150, 60, 150, 20), "Vidas: " + vidas\_barco.ToString ());}

void OnGUI() {

float x = transform.position.x, y = transform.position.y, z = transform.position.z;

float xr = transform.rotation.eulerAngles.x, yr = transform.rotation.eulerAngles.y, zr = transform.rotation.eulerAngles.z;

GUI.Label (new Rect (10, 10, 200, 30), "Hola " + texto);

GUI.Label (new Rect (10, 50, 120, 30), x.ToString () + "," + y.ToString () + "," + z.ToString ());

GUI.Box (new Rect (10, 90, 120, 30), xr.ToString () + "," + (Mathf.Round (yr)).ToString () + "," + zr.ToString ());

if (GUI.Button (new Rect (40, 140, 60, 30), "Arriba"))

transform.Translate (0, avance, 0);

if (GUI.Button (new Rect (10, 170, 60, 30), "Izquierda"))

transform.Translate (-avance, 0, 0);

if (GUI.Button (new Rect (70, 170, 60, 30), "Derecha"))

transform.Translate (avance, 0, 0);

if (GUI.Button (new Rect (40, 200, 60, 30), "Abajo"))

transform.Translate (0, -avance, 0);

texto = GUI.TextField (new Rect (250, 10, 200, 30), texto, 25);

rot = GUI.HorizontalSlider (new Rect (10, 230, 100, 30), rot, -45.0f, 45.0f);

transform.Rotate (0, rot, 0);

movimiento = GUI.Toggle (new Rect (10, 250, 100, 30), movimiento, "Permitir avance");

if (movimiento)

avance = 1;

else

avance = 0;

//RepeatButton() Window()}

# Disparos con raycast

//mecánica de disparo

if (Input.GetKeyDown (KeyCode.Alpha1)) {

if(Physics.Raycast( new Vector3(transform.position.x, pos\_y , transform.position.z), -transform.forward , out fin\_rayo, max\_distance)){

line.enabled = true;

line.SetPosition (1, fin\_rayo.point);

if (fin\_rayo.collider.gameObject.name == "tiburon\_prefab(Clone)") {

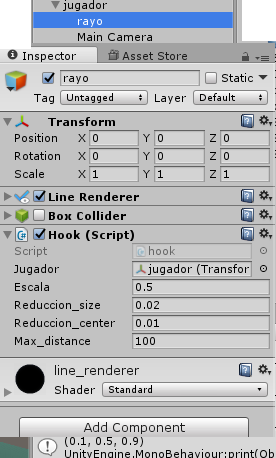
fin\_rayo.collider.transform.Translate (transform.forward);

Destroy (fin\_rayo.collider.gameObject);

line.enabled = false;}}}

## Grappling hook basic

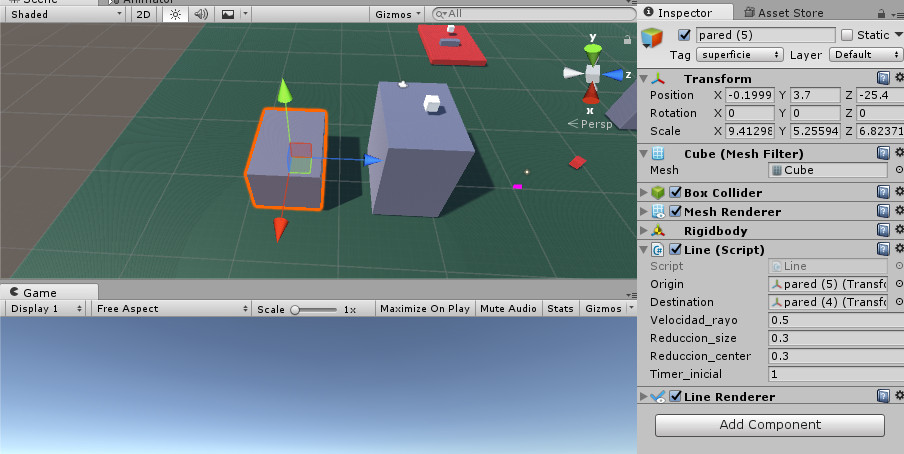
private LineRenderer line\_renderer;  
    private float counter, objetivo\_dist;  
    GameObject cam;  
  
    public Transform jugador;  
  
    public float escala = .5f, reduccion\_size = .3f, reduccion\_center = .3f, max\_distance = 20f;  
  
  
  
  
    *//BoxCollider box\_collider;*  
    public static Vector3 hand, objetivo\_dir, objetivo\_punto;  
    public static RaycastHit hit;  
    public static bool enganche, press=false;  
  
    void Awake()  
    {  
        line\_renderer = GetComponent<LineRenderer> ();  
        cam = GameObject.Find ("Main Camera");   
    }  
  
  
    void Start () {  
          
  
        enganche = false;  
  
    }  
  
  
    void Update () {  
        print (cam.transform.forward);  
  
        if(Input.GetMouseButton(0))  
        {  
            if(enganche == false)  
                buscar\_punto ();  
            if(enganche == true)  
                mostrar\_linea ();  
            press = true;  
  
        }  
        else  
        {  
            press = false;  
            mostrar\_linea ();  
        }  
  
  
  
  
  
  
    }  
          
    float x;  
  
    void buscar\_punto()  
    {  
        print ("buscando punto");  
  
        if ( Physics.Raycast(jugador.position, cam.transform.forward, out hit, max\_distance))   
        {  
            objetivo\_dir = cam.transform.forward;  
            objetivo\_punto = hit.point;  
            enganche = true;  
        }  
        else  
        {  
            print ("gancho perdido");  
        }  
  
  
  
    }  
  
    void mostrar\_linea()  
    {  
        if(press)  
        {  
            hand = new Vector3 (jugador.position.x, jugador.position.y,  
                jugador.position.z + jugador.localScale.z );  
  
            *//box\_collider.center = new Vector3(box\_collider.center.x, box\_collider.center.y,*  
            *//    box\_collider.center.z+(x \* reduccion\_center));*  
            *//box\_collider.size = new Vector3(box\_collider.size.x, box\_collider.size.y,*  
            *//    box\_collider.size.z+(x \* reduccion\_size));*  
  
            line\_renderer.enabled = true;  
            line\_renderer.SetPosition (0, jugador.position);  
            line\_renderer.startWidth = .1f;  
            line\_renderer.endWidth = .1f;  
  
  
            counter += (escala \* Time.deltaTime);  
  
  
            *//x = Mathf.Lerp (0.1f,  Vector3.Distance( jugador.position , hit.point), counter);*  
  
            line\_renderer.SetPosition (1, objetivo\_punto );  
  
  
  
            *//Physics.Raycast (hand, objetivo\_dir, max\_distance);*  
        }  
  
        if(Vector3.Distance( jugador.position , hit.point) < 5f || !press)  
        {  
            enganche = false;  
            line\_renderer.enabled = false;  
            counter = 0.1f;  
        }



# Line renderer 3d

CREAR ANIMACION DE linea script

private LineRenderer line\_renderer;  
    private float counter,dist;  
  
    public Transform origin, destination;  
  
    private float line\_draw\_speed;  
    public float velocidad\_rayo = .5f, reduccion\_size = .3f, reduccion\_center = .3f;  
    float x;  
    bool linea\_visible = false;  
    BoxCollider caja;  
    Vector3 origen\_collider\_center;  
  
    float timer;  
    public float timer\_inicial = 6f;  
    void Start () {  
        line\_renderer = GetComponent<LineRenderer> ();  
  
        line\_draw\_speed = velocidad\_rayo \* Time.deltaTime;  
        caja = GetComponent<BoxCollider> ();  
        timer = timer\_inicial;  
    }  
      
  
    void Update () {  
        if(timer > 0)  
            timer -= Time.deltaTime;  
        if(timer < 0 )*//Input.GetMouseButtonDown(0))*  
        {  
            timer = 0;  
            origen\_collider\_center = caja.center;  
            linea\_visible = true;  
        }  
  
  
  
        if (linea\_visible)  
            mostrar\_linea ();  
              
    }  
  
  
  
          
      
  
    void mostrar\_linea()  
    {  
  
        caja.center = new Vector3(caja.center.x, caja.center.y, caja.center.z+(x \* reduccion\_center));  
        caja.size = new Vector3(caja.size.x, caja.size.y, caja.size.z+(x \* reduccion\_size));  
      
        line\_renderer.enabled = true;  
        line\_renderer.SetPosition (0, origin.position);  
        line\_renderer.startWidth = .1f;  
        line\_renderer.endWidth = .1f;  
        dist = Vector3.Distance (origin.position, destination.position);  
  
        if (counter <= 1f)   
        {  
            counter += (line\_draw\_speed);  
  
            x = Mathf.Lerp (0, dist, counter);  
  
            Vector3 point\_a = origin.position;  
            Vector3 point\_b = destination.position;  
  
            Vector3 point\_a\_long\_line = x \* Vector3.Normalize (point\_b - point\_a) + point\_a;  
  
            line\_renderer.SetPosition (1, point\_a\_long\_line);  
        }   
        else  
        {  
            linea\_visible = false;  
            line\_renderer.enabled = false;  
            counter = 0;  
            caja.center = origen\_collider\_center;  
            caja.size = new Vector3(0.5f,0.5f,0f);  
            timer = timer\_inicial;  
        }  
              
  
    }



# Cursor

CursorLockMode wantedMode;  
  
    *// Apply requested cursor state*  
    void SetCursorState()  
    {  
        Cursor.lockState = wantedMode;  
        *// Hide cursor when locking*  
        Cursor.visible = (CursorLockMode.Locked != wantedMode);  
    }  
  
    void OnGUI()  
    {  
        GUILayout.BeginVertical();  
        *// Release cursor on escape keypress*  
        if (Input.GetKeyDown(KeyCode.Escape))  
            Cursor.lockState = wantedMode = CursorLockMode.None;  
  
        switch (Cursor.lockState)  
        {  
        case CursorLockMode.None:  
            GUILayout.Label("Cursor is normal");  
            if (GUILayout.Button("Lock cursor"))  
                wantedMode = CursorLockMode.Locked;  
            if (GUILayout.Button("Confine cursor"))  
                wantedMode = CursorLockMode.Confined;  
            break;  
        case CursorLockMode.Confined:  
            GUILayout.Label("Cursor is confined");  
            if (GUILayout.Button("Lock cursor"))  
                wantedMode = CursorLockMode.Locked;  
            if (GUILayout.Button("Release cursor"))  
                wantedMode = CursorLockMode.None;  
            break;  
        case CursorLockMode.Locked:  
            GUILayout.Label("Cursor is locked");  
            if (GUILayout.Button("Unlock cursor"))  
                wantedMode = CursorLockMode.None;  
            if (GUILayout.Button("Confine cursor"))  
                wantedMode = CursorLockMode.Confined;  
            break;  
        }  
  
        GUILayout.EndVertical();  
  
        SetCursorState();

Cursor.lockState = CursorLockMode.Locked;  
        print ("width: "+ Screen.width.ToString()+", height: "+Screen.height.ToString());

# Luces

public Light luz\_point;  
    Color mi\_color1 = Color.red;  
    Color mi\_color2 = Color.green;  
    public float duration = 3.0F;  
    *// Use this for initialization*  
    void Start ()   
    {  
          
    }  
      
    *// Update is called once per frame*  
    void Update ()   
    {  
          
        float t = Mathf.PingPong(Time.time, duration) / duration;  
        if (Input.GetKeyDown (KeyCode.Alpha1))   
        {  
            Debug.Log (t);  
        }  
        luz\_point.color = Color.Lerp (mi\_color1, mi\_color2, t);  
  
    }

# Draw Lines y Gizmos

void OnDrawGizmosSelected()  
    {   
        if(objetivo)  
        {  
            Gizmos.color = Color.blue;  
            Gizmos.DrawLine (transform.position, objetivo.position);  
  
            Gizmos.color = Color.white;  
            Gizmos.DrawWireSphere (transform.position, 1.5f);  
                  
        }  
  
  
    }

**Funciones en Unity**

1. Lerp

Vector3.Lerp (transform.position, target\_position, move\_speed\*Time.deltaTime);

Lerp interpola los valores de inicio a fin de manera lenta y continua.

Usado en una cámara no va tan pegada al objetivo, va lentamente.

1. Velocity en RigidBody

* rb.velocity = new Vector2(axis.x\*velocidad\_movimiento,rb.velocity.y);

Modifica la velocidad del personaje y lo hace caminar sin detención en una dirección

1. Clamp

Para anclar el valor de una variable como de la cámara usamos clamp

Mecánica BOUNDS

cam = GetComponent<Camera> ();

alto\_vision\_mitad = cam.orthographicSize;

ancho\_vision\_mitad = alto\_vision\_mitad \* Screen.width/Screen.height;

float clampedX = Mathf.Clamp (transform.position.x, min\_bound.x + ancho\_vision\_mitad, max\_bound.x - ancho\_vision\_mitad);

float clampedY = Mathf.Clamp (transform.position.y, min\_bound.y + alto\_vision\_mitad, max\_bound.y - alto\_vision\_mitad);

transform.position = new Vector3 (clampedX, clampedY, transform.position.z);

clamp no retorna un valor menor al mínimo o mayor al máximo

1. Translate

translate -> le avisa al motor que queremos cambiar las posiciones x,y,z de transform

transform.Translate (new Vector3 (Input.GetAxisRaw("Horizontal")\*velocidad\_movimiento\*Time.deltaTime,0,0));

1. Slerp para hacer que un enemigo persiga con la mirada

transform\_nueva.rotation = Quaternion.Slerp(transform\_nueva.rotation, Quaternion.LookRotation(-target.position + transform\_nueva.position), rotationSpeed\*Time.deltaTime);

Vector3 direction = transform.TransformDirection (Vector3.forward);

Quaternion rotation = Quaternion.FromToRotation (Vector3.up,shoot.normal);

Mathf.SmoothDamp(machine\_gun.transform.localPosition.x,posxy\_next.x,ref velxy.x,0.04f)

Mathf.Lerp();

Vector3.Angle(transform.forward, targetDir);

transform.RotateAround (Vector3.zero, Vector3.up, velAngular \* Time.deltaTime);

Anclar la vision de la cámara

En funciones privadas no es necesario agregar private. En los datos tampoco, pero en los datos es recomendable para facilitar la lectura