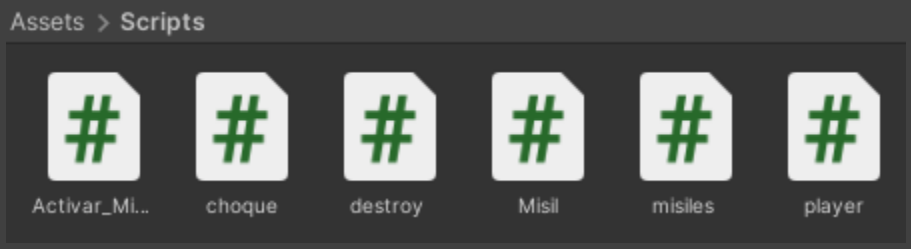
Sistema de misiles

Creador: Oskar Dimas Carbajal

Este programa esta echo para hacer un lanzamisiles estático, de una sola ráfaga, otorgando un control de personaje para testear los misiles, rango, tiempo de vida y Angulo de misil. Siendo seis scripts los encargados de ejecutar todo



Tenemos como primero y mas importante el Script **Activar_Misil**, siendo el más importante a la hora de iniciar el programa, teniendo una variable

```
//variable
public GameObject objectToActiveAndDesactivate;
```

Siendo la encargada de activar y desactivar el spawn de misiles

```
Mensaje de Unity | 0 referencias
void Update()
{
    if (Input.GetKeyDown(KeyCode.E))//cada que se precione la tecla e se desactivara el objeto
    {
        objectToActiveAndDesactivate.SetActive(false); // se pondra en falso para desactivarlo
    }
    if (Input.GetKeyDown(KeyCode.R))//cada que se precione la letra R se activara el obejeto
    {
        objectToActiveAndDesactivate.SetActive(true); // se pondra en verdadero el objeto piniendolo a funcionar
    }
}
```

Teniendo en Update las funciones asignadas para activar y desactivar el misil en cuestión, siendo la letra **E** la encargada de desactivar el misil y la letra **R** la encargada de activar el spawn de misil, ambas funciones ejecutándose solo cuando la tecla sea presionada (no pulso mantenido).

Siendo el segundo código **player** el encargado del movimiento del jugador. Siendo las teclas de control **W,A,S,D.** Teniendo solo dos variables encargadas de la velocidad y la rotación del personaje (Teniendo ambos en público si se tiene en mente que la velocidad sea mayor o menor, así como la rotación, siendo esto más enfocado al apartado estético del mismo).

```
//variables

public float speed = 0.0f; //velocidad a la que se va a mover el personaje

public float rotationSpeed = 0.0f; //velocidad a la que va a rotar el peronaje

//ambos son publicos para poder modificarlos desde la consola
```

Teniendo a continuación las funciones

```
// variables horizontales y verticales
float  horizontalInput = Input.GetAxis("Horizontal");
float verticalInput = Input.GetAxis("Vertical");
```

Teniendo las variables a modificar siendo vertical y horizontal

```
//con esto prodremos controlar la velocidad de movimiento
Vector3 movementDirection = new Vector3(horizontalInput, 0, verticalInput);
movementDirection.Normalize();
```

Siendo este proceso el que se encargara de la velocidad a la que se mueva el personaje, detectando por las mismas variables en qué dirección los movemos y suavizando el movimiento del mismo player.

```
// gira el objeto y hacerlo mirar en la direccion en la que nos movemos, ademas de suabizar el movimiento de giro que tendra al player u objeto que controlemos
transform.position = transform.position + movementDirection * speed * Time.deltaTime;
if (movementDirection != Vector3.zero) transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(movementDirection),rotationSpeed * Time.deltaTime);
```

Teniendo esta función como la encargada de rotar el modelo en una sola dirección dando sentido al movimiento

Siendo el tercer script Misiles el encargado de duplicar los misiles, así como llevar acabo el temporizador de cuanto tiempo se duplicarán los misiles, además de llevar el contador y contador máximo.

```
[SerializeField] //variable "papá"
private GameObject _misil;

[SerializeField]//variables hijas
private float _timer = 0.1f; //temporizador acargo de lanzar los misiles
private float timerCount = 0.4f; //Contador  de tiempo

[SerializeField]//variables hijas
private int _counter; //cnatdor
private int _maxCounter = 5000; //contador maximo
```

Teniendo el proceso de **Disparar_Cr** como el encargado de duplicar los misiles y detener todo cuando el contador llegue al número establecido en el maxCounter, así como también se encarga de dar la dirección al misil dependiendo del misil anterior

```
//proceso encargado de generar las copias del misil original
1 referencia
IEnumerator Disparar_CR()
{
    Debug.Log("inicio 7u7"); //al iniciar este proceso en la consola se vera el mensaje inicio 7u7
    for (int i=0; i<_maxCounter; i++)
    {
        _counter++;
        Instantiate(_misil, transform.position, transform.rotation);
        yield return new WaitForSeconds(_timer);
    }
    Debug.Log("Fin del atentado"); //al terminar este proceso se vera el mensaje Fin del atentado
}
```

Y ejecutándose todo ese proceso en Start

```
Mensaje de Unity | 0 referencias
void Start()
{
    StartCoroutine(Disparar_CR()); //al momento de inicar el rpograma entonces se iniciara el script y la ejecucion de IEnumerator
}
```

Teniendo en el cuatro script **Destroy** el encargado de dar tiempo de vida teniendo una sola variable para llevarlo acabo

```
//variable encargada del tiempo que estara el misil, para despues ser eliminado
[SerializeField]
public float timeLife = 0.0f;
```

Teniendo un proceso encargado de destruir el misil ya sea que colisione con un objeto (ejecutando a su vez un sistema de partículas) u eliminándose una ves que termina su tiempo de vida

```
//proceso encargado de activar la explocion al momento de colicionar con el objetivo
Mensaje de Unity | 0 referencias
private void OnTriggerEnter(Collider other)
{
    if (other.tag == "player") //cuando se cuampla con la condicional de que el misil colicione con el objetivo entonces se activara este proceso
    {
        Instantiate(_explosionEffect, transform.position, Quaternion.identity); //cuando se llegue a inpactar con el objetivo se activara la explocion y se eliminara el objetivo
        Destroy(gameObject);
    }
}
```

Y al igual que en el scrpit anterior ejecutando dicho proceso en el Start

```
Mensaje de Unity | 0 referencias
void Start()
{
    Destroy(gameObject, timeLife); //este scprit se activara desde el inicio para aseguar borrar toda copia generada al momento de que se acabe el tiempo puesto en _timerLife

}
```

Teniendo como cuarto script **Choque** que se encarga de dar físicas al misil, agregando gravedad, velocidad, tiempo (necesario para el cálculo de gravedad) y ángulo

```
public float speed = 10.0f;
public float angle = 45.0f;
public float gravity = 9.81f;

private Vector3 velocity;
private float time = 0.0f;
```

Teniendo un proceso encargado de hacer que la velocidad se proporcionar a la masa del objeto dependiendo de la posición en la que este. Además de checar si el objeto choco con algo y mandando al ultimo un mensaje a consola de si choco con algo

```
Mensaje de Unity | 0 referencias
private void Update()
{
    time += angle * Mathf.Deg2Rad;
    Vector3 displacement = velocity * time + 0.5f * Vector3.down * gravity * time * time;
    transform.position += displacement;

    Collider[] colliders = Physics.OverlapSphere(transform.position, 0.2f);
    foreach (Collider collider in colliders)
    {
        Debug.Log("El misil choco con"+ collider.gameObject.name);
    }
}
```

Ejecutando en Start un cálculo de la velocidad inicial del objeto

```
Mensaje de Unity | 0 referencias
private void Start()
{
    float radianAngle = angle * Mathf.Deg2Rad;
    velocity = new Vector3(speed * Mathf.Cos(radianAngle), speed * Mathf.Sin(radianAngle), 0.0f);
}
```

Y siendo el ultimo script **Misil** el encargado de lar una ligera variación en el misil usando en base la velocidad del mismo y apoyándose de la gravedad de choque. Teniendo tres variables, la posición del objeto, la velocidad del objetivo y el rango de seguimiento del misil

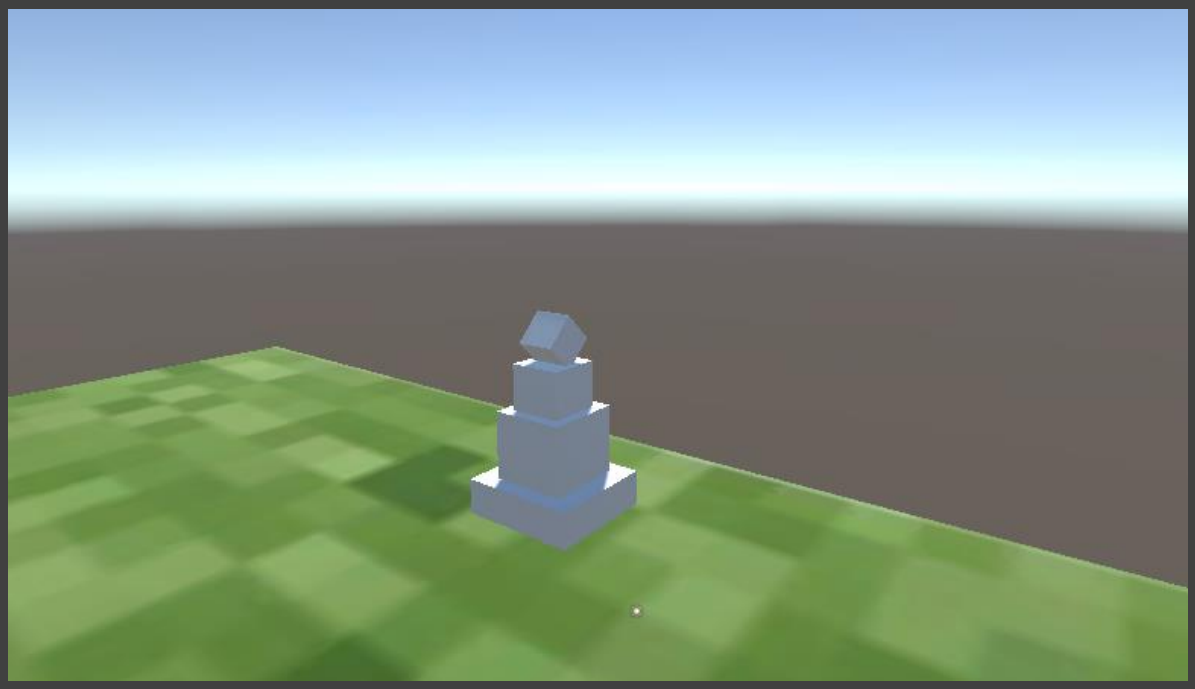
```
//variables cambiantes 7u7
[SerializeField]
private Transform m_objetivo = null; //variable obejtivo o distancia
[SerializeField]
private float m_velocidad = 0.0f;//varable velocidad
[SerializeField]
private float m_seguimiento = 0.0f;//que tanto se va a acercar el misil al objetivo
```

Y ejecutando un cálculo en base a la posición del objetivo y la posición en la que se encuentre el misil y acercándose ligeramente para acertar al objetivo

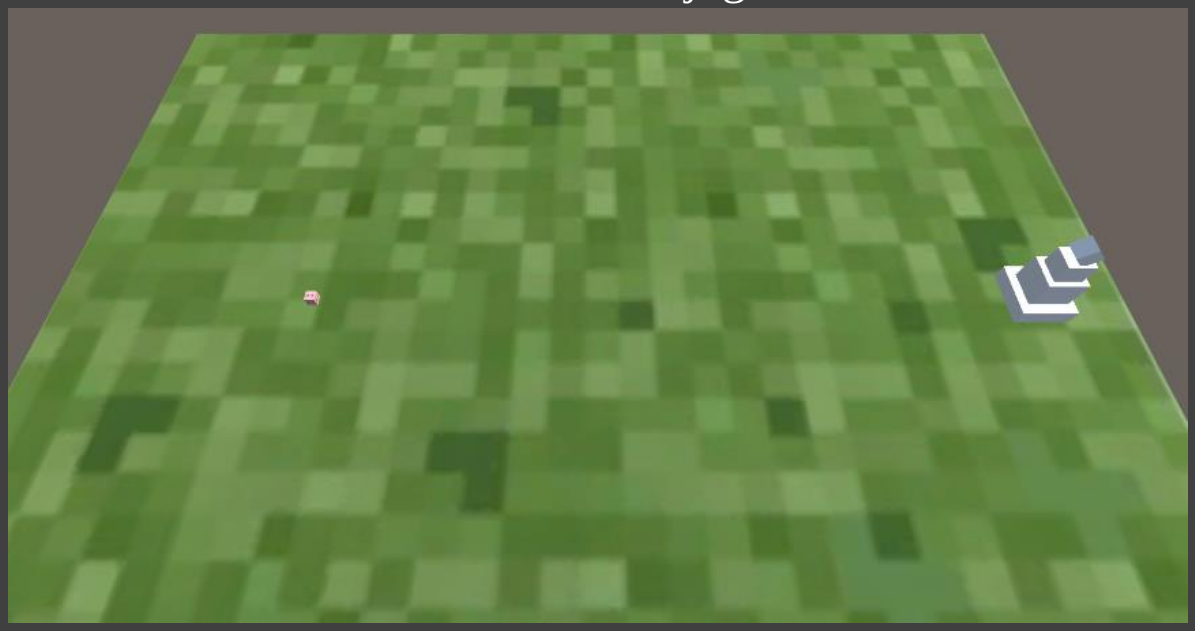
```
// cambia los valores a los que se va a dirigir el misil junto a los valores de la velocidad a la que va (aun que ambos se pueden modificar desde el inspector de unity)
Mensaje de Unity | 0 referencias
private void Update()
{
    Quaternion desireRotation = Quaternion.LookRotation(m_objetivo.position - transform.position);//ajusta la posicion respecto a la posicion del obetivo respecto a la posicion del misil
    transform.rotation = Quaternion.Slerp(transform.rotation , desireRotation , m_seguimiento * Time.deltaTime);//se modificara la rotacion respecto al obejtivo y se miltiplocara respecto al tiempo

    transform.Translate(Vector3.forward * Time.deltaTime * m_velocidad);//se multiplica el tiempo por la veolcidad
}
```

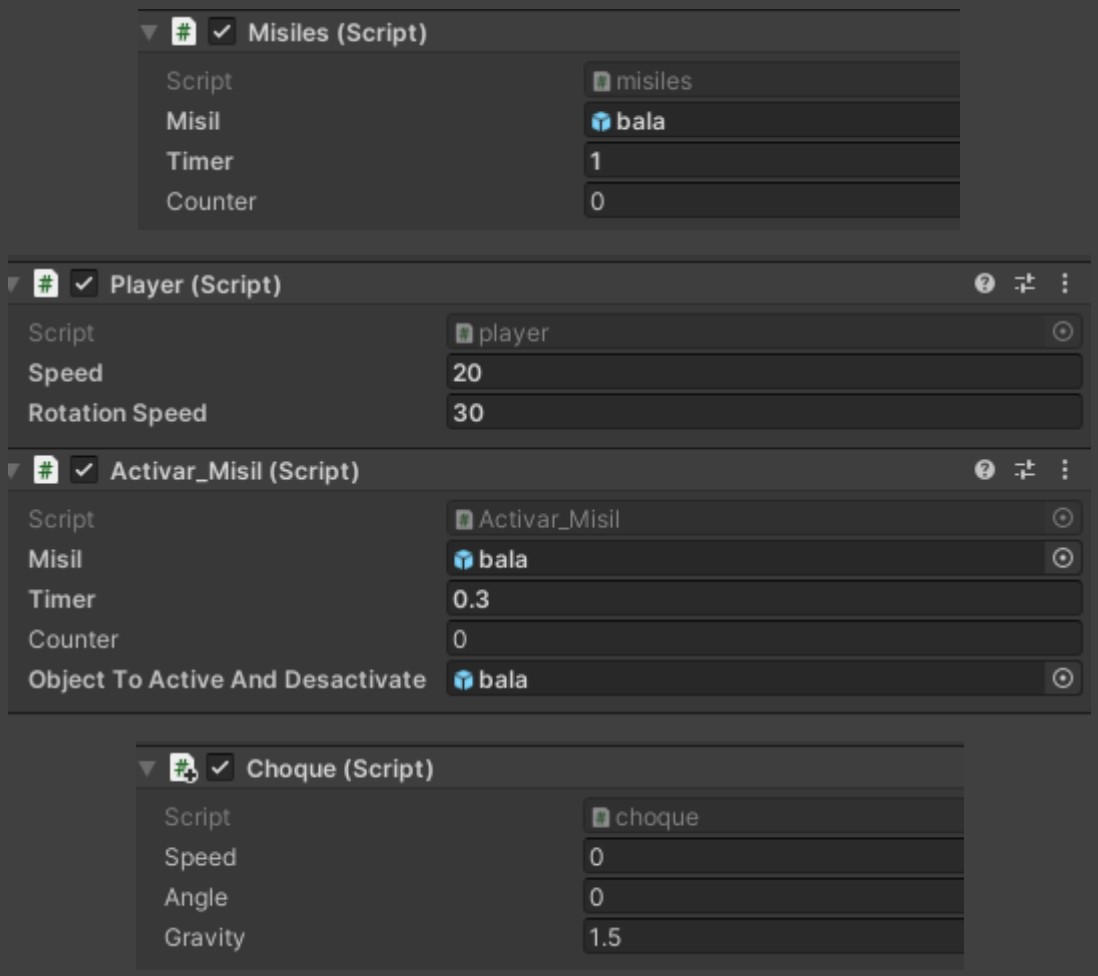
Teniendo una torreta básica para darse una idea de donde está el misil y de donde saldrán



Así como una vista semi ortogonal para dar una mejor idea de donde está el jugador



Además de tener cosas disponibles donde podrás cambiar valores, velocidades, gravedad y tiempo.


Misiles (Script)
Script
misiles
Misil
bala
Timer
1
Counter
0
Player (Script)
Script
player
Speed
20
Rotation Speed
30
Activar_Misil (Script)
Script
Activar_Misil
Misil
bala
Timer
0.3
Counter
0
Object To Active And Desactivate
bala
Choque (Script)
Script
choque
Speed
0
Angle
0
Gravity
1.5