Sistema de Misil Teledirigido

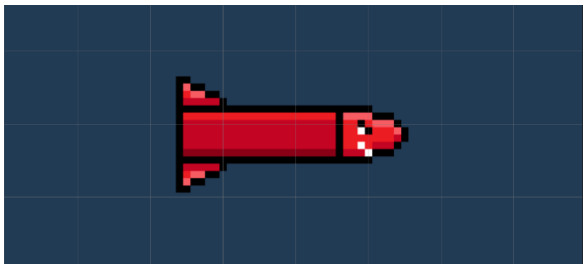
Por Ciro Sebastián Hernández López

¿Qué es?

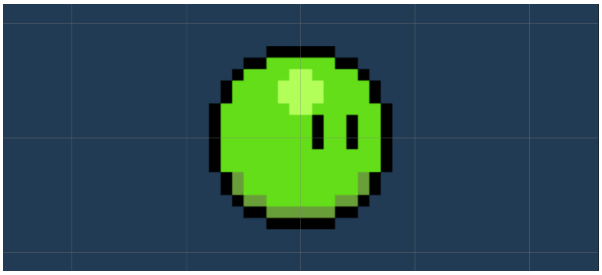
Este es un sistema que crea proyectiles que siguen a un objeto especifico en el juego, en este caso el jugador y se destruyen si entran en contacto con él o tras pasar un lapso especifico de tiempo.

¿Cómo funciona?

Primero tenemos un Prefab del misil, este cuenta con un sprite, un RigidBody y un BoxColLider.



Tambien necesitamos un objeto para que el misil lo siga, en este caso creamos el objeto “Player”, cuenta con su propio RigidBody y Collider. Pero lo más importante es que cuenta con un tag.



Antes de empezar a seguir al objetivo este se lanza hacia arriba para luego empezar a seguir al objetivo. Esto se logra aplicando una fuerza al RigidBody del Misil usando un valor publico llamado “launchForce”, luego se inicia una co-rutina que cambia el valor del booleano a “IsLaunching” a cierto tras

un tiempo determinado. Esto hace que se inicie el método para seguir al objetivo.

```
rb.AddForce(new Vector2(rb.velocity.x, launchForce));
StartCoroutine(Launch());
```

```
void Update()
{
    if (IsLaunching == true)
    {
        Chase();
    }
}
```

```
IEnumerator Launch()
{
    yield return new WaitForSeconds(wait);
    IsLaunching = true;
}
```

Para determinar el objetivo se usa un variable publica del tipo string llamada target, en esta se escribe el tag del objetivo deseado. El misil busca un objeto con este tag y lo empieza a seguir, cambiando de dirección para que siempre este apuntando al objetivo.

```
private void Chase()
{
    distance = Vector2.Distance(transform.position, Player.transform.position);
    Vector2 direction = Player.transform.position - transform.position;
    direction.Normalize();
    float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;

    transform.position = Vector2.MoveTowards(this.transform.position, Player.transform.position, speed * Time.deltaTime);
    transform.rotation = Quaternion.Euler(Vector3.forward * angle);
}
```

Se utiliza un chequeo de la colisión para determinar si el misil ha hecho contacto con el objetivo, revisando el tag del objeto colisionado para determinar si es el objetivo.

```
    private void OnCollisionEnter2D(Collision2D other)
{

    if (other.gameObject.CompareTag(target))
    {
        Destroy(gameObject);
    }
}
```

También se añadió una co-rutina para que el misil se destruya tras un tiempo determinado si no ha tocado al objetivo. Este tiempo de define por una variable llamada "lifeTime".

```
IEnumerator Destroy()
{
    yield return new WaitForSeconds(lifeTime);
    Destroy(gameObject);
}
```

Esta co-rutina empieza desde que el misil entra en la escena.

Para crear varios misiles instanciamos varias el prefab del misil en una subrutina.

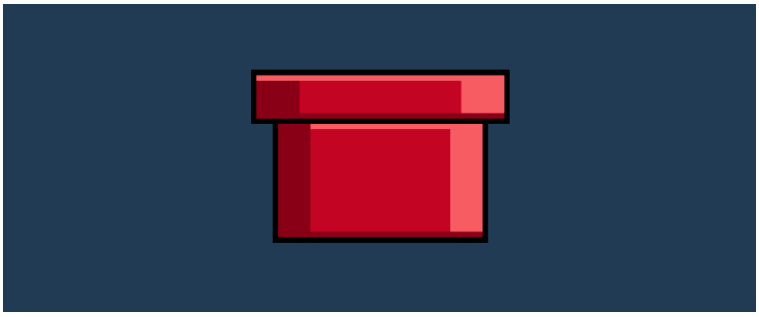
```
private IEnumerator Spawn(float rate, GameObject Projectile)
{
    yield return new WaitForSeconds(rate);
    GameObject newMisil = Instantiate(Projectile, new Vector3(spawn.transform.position.x, spawn.transform.position.y + 2), Quaternion.identity);
    StartCoroutine(Spawn(rate, Projectile));
}
```

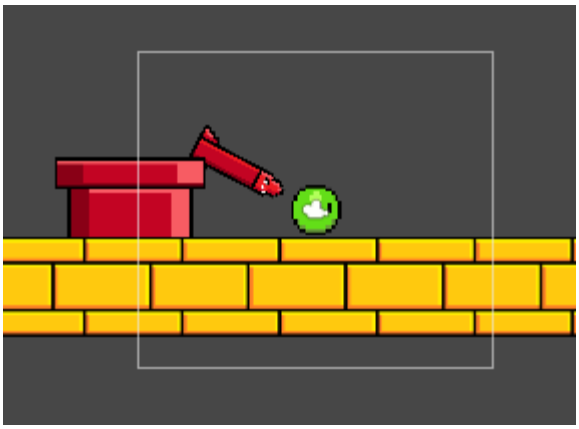
Para determinar la posición donde se crean usamos un prefab con el tag "Spawner", el código revisa este tag y lo utiliza para determinar la locación donde se crea el misil.

```
void Start()
{
    StartCoroutine(Spawn(misliRate, MisilPrefab));
    spawn = GameObject.FindGameObjectWithTag("Spawner");
}
```

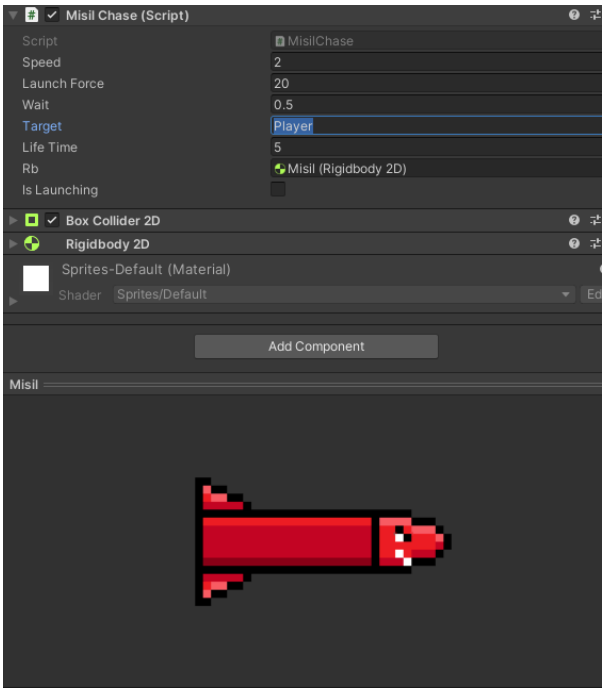
¿Cómo usarlo?

Para que empiecen a aparecer misiles una solamente debe poner el prefab "Spawner" en la escena, este empezara a aparecer los misiles automáticamente.

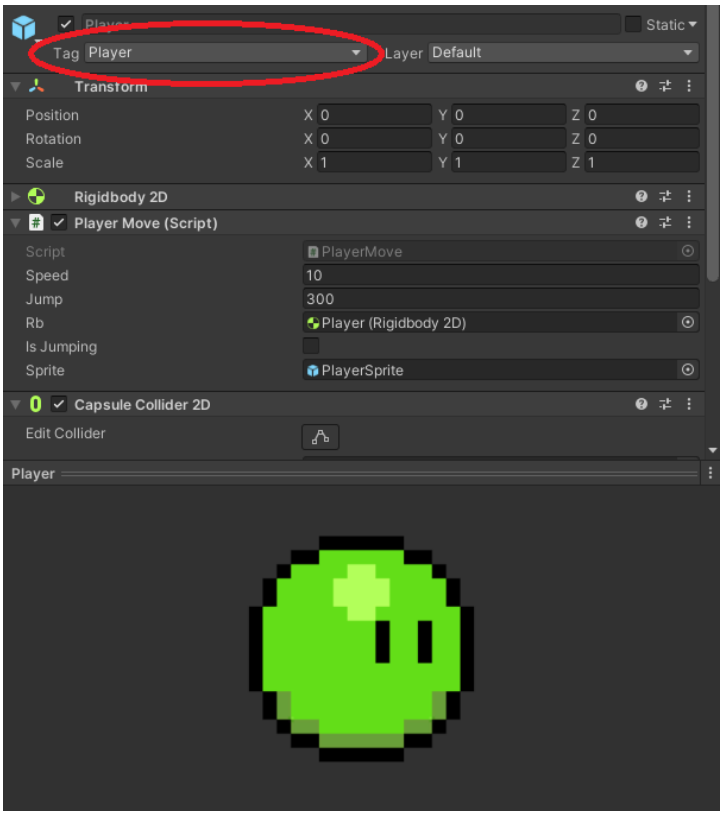




Para que los misiles empiecen a seguir el objetivo se debe configurar el prefab del misil. Asignando un tag para que los objetos con este sean el objetivo.

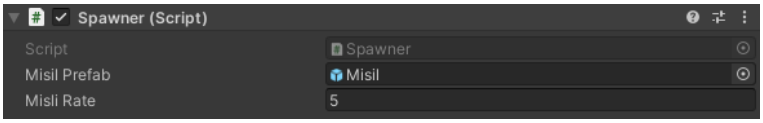


Este debe coincidir con el tag de algún objeto en la escena para que lo empiece a seguir.



Se pueden configurar varios aspectos de los misiles y del "Spawner":

- Velocidad de los misiles
- La fuerza con la que son lanzados
- El tiempo que esperan antes de seguir al objetivo
- El objetivo del misil
- Que tanto dura en pantalla
- Que tan seguido aparece



Conclusión

Esta herramienta es muy útil para juegos en 2D que requieran un obstáculo más interesante, es fácil de implementar pues solo se debe importar los prefabs del misil y el spawner.

También es fácil de configurar los aspectos esenciales de la herramienta.