

# Unity - Tema 6

## Scriptable Objects

Scripting III

Javier Alegre Landáburu  
[javier.alegre@u-tad.com](mailto:javier.alegre@u-tad.com)



# Resumen del tema

- ¿Qué es un scriptable object?
- ¿Cómo se programa?
- ¿Para qué sirve?

# Scriptable Object

- Hasta ahora, creábamos componentes con variables publicas para guardar datos.
- Por ejemplo para crear las armas tenemos un montón de variables (ammo, fireRate, isAMachineGun) en el script que controla el uso del arma.
- En el script que estamos usando, estamos mezclando datos con lógica de funcionamiento.

```
public class SimplePistol : MonoBehaviour
{
    public Transform m_raycastSpot;
    public float m_damage = 80.0f;
    public float m_forceToApply = 20.0f;
    public float m_weaponRange = 9999.0f;
    public int m_ammoCapacity = 3;
    public float m_rateOfShot = 10;
    public float m_accuracy = 70.0f;
```

```
private void Update()
{
    m_shotTimer += Time.deltaTime;

    m_currentAccuracy = Mathf.Lerp(m_currentAccuracy, m_accuracy, m_accuracyRecoverPerSecond * Time.deltaTime);

    m_canShot = (m_isMachineGun) ? true : m_canShot;

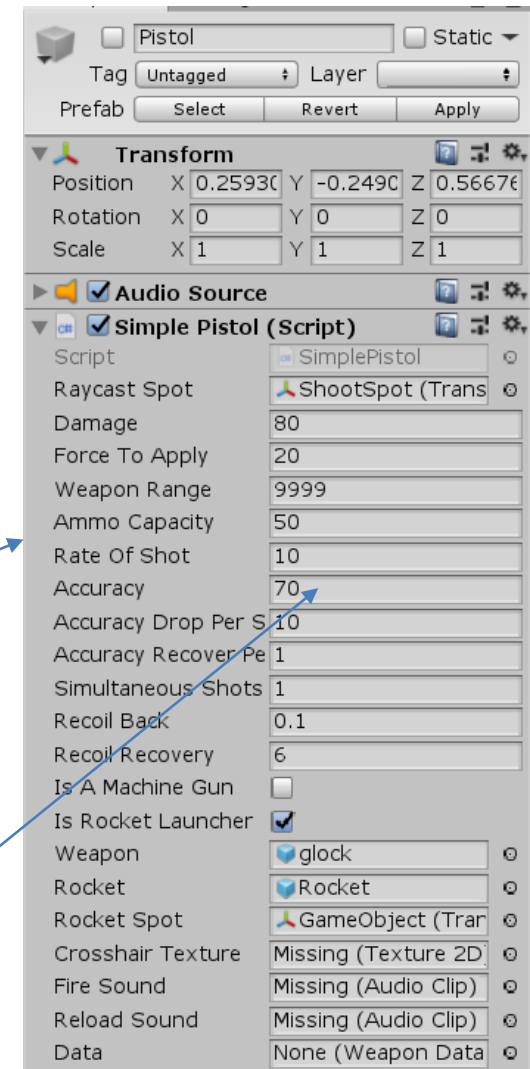
    if (m_shotTimer >= m_timeBetweenShots && m_canShot)
    {
        if (Input.GetButton("Fire1"))
        {
            if (m_isRocketLauncher)
            {
                ShotRocket ();
            }
            else
            {
                Shot();
            }
        }

        if (Input.GetButtonDown("Fire2"))
        {
            Reload();
        }
    }
    else if (Input.GetButtonUp("Fire1"))
    {
        m_canShot = true;
    }

    m_weapon.transform.position = Vector3.Lerp(m_weapon.transform.position, transform.position, m_recoilRecovery * Time.deltaTime);
}
```

# Scriptable Object

- Al hacerlo de esta manera, tenemos que rellenar los datos del arma en el propio componente asociado a un *gameobject* en la escena.
- Si queremos guardar esos datos para poder usarlos en otras escenas, tenemos que crear un *prefab* con ese *gameobject* para hacerlo.
- Si queremos editar esos datos, tenemos que abrir la escena, buscar el arma y editarlos.




- En caso de querer cambiar el comportamiento del arma por culpa de un ítem por ejemplo, nos toca cambiar los valores de esas variables.

# Scriptable Object


- Y si pudiéramos crear un archivo en un directorio del proyecto que nos permitiera guardar esos datos sin necesidad de que estén en un componente asociado a un *gameObject* del cual hemos hecho un *prefab*.
- Para eso existen los *scriptableobjects*.
- Tipo de archivo de Unity que nos permite guardar datos en un fichero desde el editor para luego usarlos en tiempo de ejecución.
- Necesitamos un script que contenga los tipos de datos que queremos guardar (variables) y que herede de *scriptableObject*.

```
public class SimpleWeaponInfo : ScriptableObject
{
    public string      m_name;
    public float       m_damage      = 80.0f;
    public float       m_forceToApply = 20.0f;
    public float       m_weaponRange = 9999.0f;
    public int         m_ammoCapacity = 3;
    public float       m_rateOfShot  = 10;
    public AudioClip   m_fireSound;
    public AudioClip   m_reloadSound;
}
```



# Scriptable Object

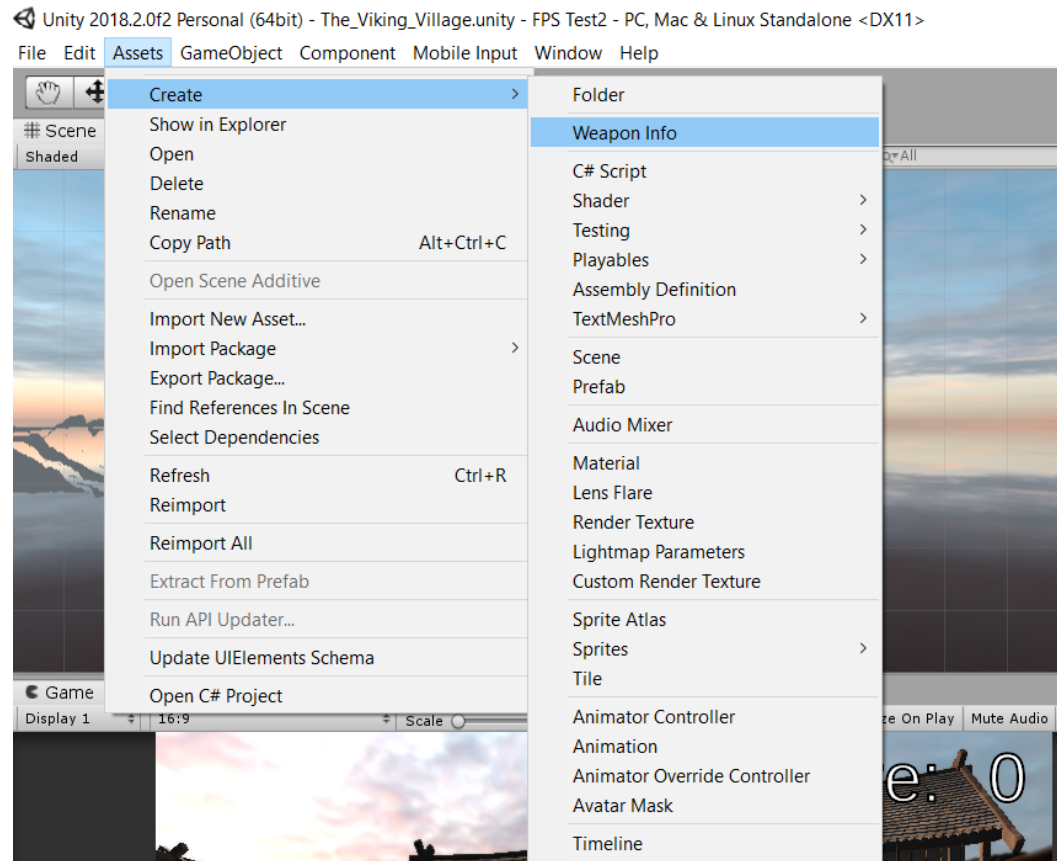
- El *scriptableObject* nos define los datos que vamos a querer guardar en el fichero.
- Necesitamos “algo” que cree esos ficheros basándose en lo que hemos añadido al *scriptableObject*.
- Vamos a añadir una opción en Unity que nos permita generar estos fichero.
- Para ello tenemos que añadir unas cuantas palabras reservadas justo antes de la declaración de la clase.



```
[CreateAssetMenu(fileName = "New Weapon Info", menuName = "Weapon Info", order = 51)]
public class SimpleWeaponInfo : ScriptableObject
{
    private string m_name;
    private float m_damage = 80.0f;
    private float m_fireRate = 30.0f;
}
```

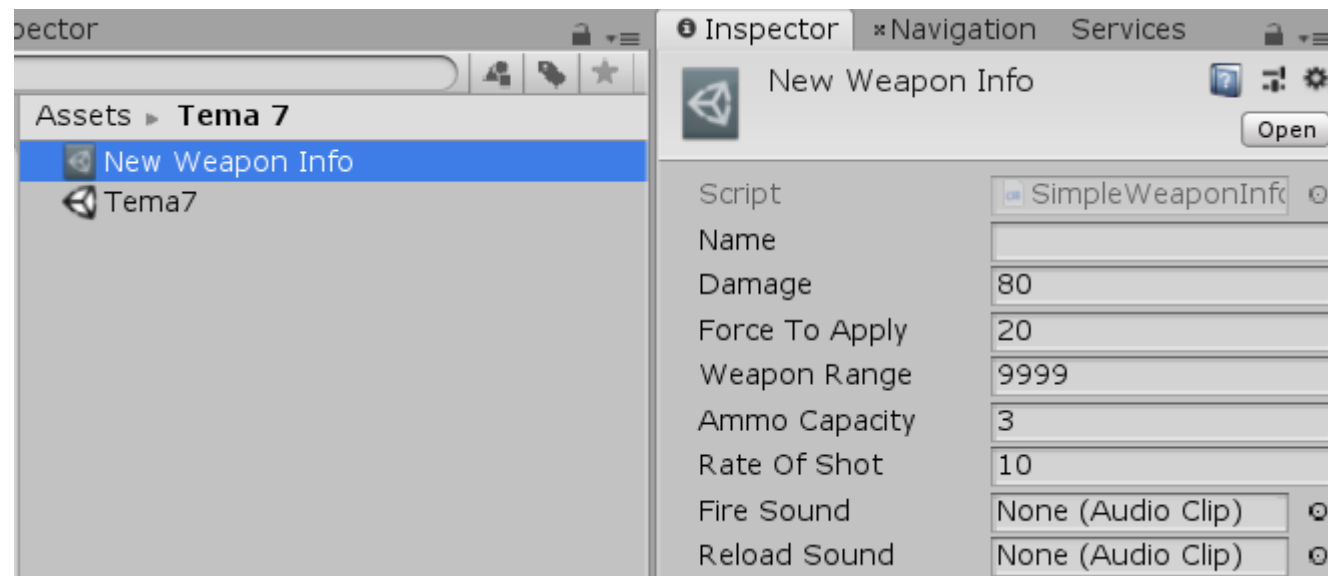
# Scriptable Object

- Si lo hemos hecho bien, nos aparecerá un nuevo elemento en el menú de assets.



# Scriptable Object

- Al pulsar sobre la nueva opción se nos creara un fichero que tiene lo necesario para que podamos darle valor a esos datos desde el editor.





# Scriptable Object

- Dejar las variables publicas no es muy buena idea porque los *scriptableobjects* deberían ser archivos sólo para la lectura de datos.
- Si modificamos el valor de una variable en tiempo de ejecución desde un script en el editor, se modifica el valor en el scriptableobject hasta que lo volvamos a cambiar desde el editor o desde otro script. En tiempo de ejecución no se guardarían los datos
- Vamos a hacer las variables privadas y usaremos propiedades (get y set) para acceder a la lectura de estos valores.
- Tenemos que usar la directiva “SerializeField” para que las variables privadas se vean en el editor.



# Scriptable Object

- 1º Hacemos todas las variables privadas y usamos la directiva "SerializeField" para que las variables privadas se vean en el editor:

```
[CreateAssetMenu(fileName = "New Weapon Info", menuName = "Weapon Info", order = 51)]  
public class SimpleWeaponInfo : ScriptableObject  
{  
    [SerializeField]  
    private string    m_name;  
  
    [SerializeField]  
    private float     m_damage      = 80.0f;  
  
    [SerializeField]  
    private float     m_forceToApply = 20.0f;  
  
    [SerializeField]  
    private float     m_weaponRange = 9999.0f;  
  
    [SerializeField]  
    private int       m_ammoCapacity = 3;  
  
    [SerializeField]  
    private float     m_rateOfShot  = 10;  
  
    [SerializeField]  
    private AudioClip m_fireSound;  
  
    [SerializeField]  
    private AudioClip m_reloadSound;  
}
```

# Scriptable Object

- 2º Añadimos las propiedades para poder acceder a las variables:

```
[CreateAssetMenu(fileName = "New Weapon Info", menuName = "Weapon Info", order = 51)]
public class SimpleWeaponInfo : ScriptableObject
{
    [SerializeField]
    private string m_name;

    public string name
    {
        get
        {
            return m_name;
        }
    }

    [SerializeField]
    private float m_damage = 80.0f;

    public float damage
    {
        get
        {
            return m_damage;
        }
    }

    [SerializeField]
    private float m_forceToApply = 20.0f;
}
```

# Scriptable Object

- Por último borramos las variables del script del arma y añadimos una variable publica de tipo *scriptableObject* donde asignaremos el fichero que acabamos de crear y configurar.

```
public class SimplePistol : MonoBehaviour
{
    public Transform m_raycastSpot;
    public float m_damage = 80.0f;
    public float m_forceToApply = 20.0f;
    public float m_weaponRange = 9999.0f;
    public int m_ammoCapacity = 3;
    public float m_rateOfShot = 10;
    public float m_accuracy = 70.0f;
    public float m_accuracyDropPerShot = 10.0f;
    public float m_accuracyRecoverPerSecond = 1.0f;
    public int m_simultaneousShots = 1;
    public float m_recoilBack = 0.1f;
    public float m_recoilRecovery = 0.01f;
    public bool m_isAMachineGun = false;
    public bool m_isRocketLauncher = false;
    public GameObject m_weapon;
    public GameObject m_rocket;
    public Transform m_rocketSpot;
    public Texture2D m_crosshairTexture;
    public AudioClip m_fireSound;
    public AudioClip m_reloadSound;
    private bool m_canShot;
    private int m_currentAmmo;
    private float m_timeBetweenShots;
    private float m_shotTimer;
    private float m_currentAccuracy;

    void Start()
    {
        m_currentAmmo = m_ammoCapacity;

        m_timeBetweenShots = 1.0f / m_rateOfShot;

        m_shotTimer = 0.0f;
    }
}
```

```
public class WeaponBehaviour : MonoBehaviour
{
    private bool m_canShot;
    private int m_currentAmmo;
    private float m_timeBetweenShots;
    private float m_shotTimer;
    private float m_currentAccuracy;

    public GameObject m_weapon;
    public Transform m_raycastSpot;
    public Transform m_rocketSpot;

    [SerializeField]
    private WeaponData data;

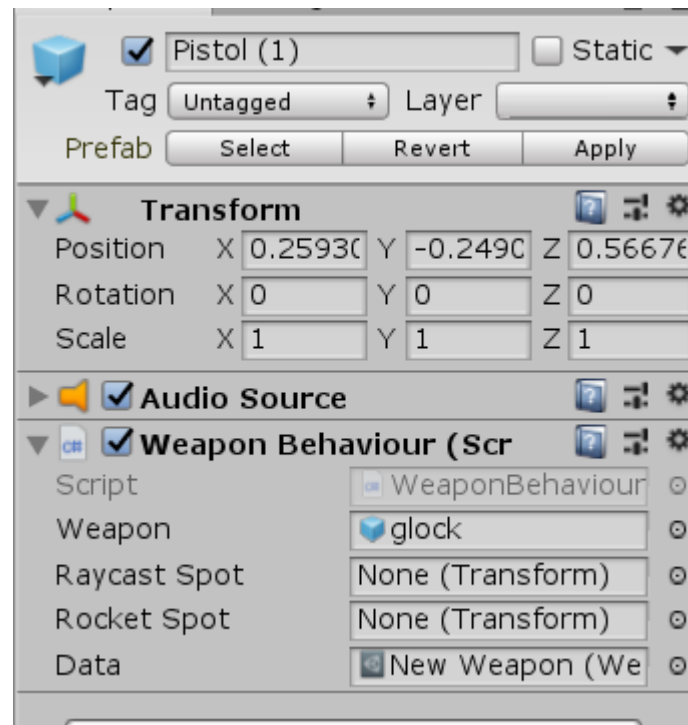
    void Start()
    {
        m_currentAmmo = data.ammoCapacity;

        m_timeBetweenShots = 1.0f / data.rateOfShot;

        m_shotTimer = 0.0f;
    }
}
```

# Scriptable Object

- Ahora podemos asignar el *scriptableObject* en la escena al script del arma.



# Scriptable Object

- Desde el script accederemos a las variables del *scriptable* en vez de a las variables de la clase (que ya no existen).

```
void Start()
{
    m_currentAmmo = m_ammoCapacity;
    m_timeBetweenShots = 1.0f / m_rateOfShot;
    m_shotTimer = 0.0f;
}
```

```
void Start()
{
    m_currentAmmo = data.ammoCapacity;
    m_timeBetweenShots = 1.0f / data.rateOfShot;
    m_shotTimer = 0.0f;
}
```

# Scriptable Object

- Hemos visto cómo crear un *scriptableObject* para poder proporcionarle datos a un elemento de la escena.
- Imaginemos ahora que queremos crear un montón de armas y que el *scriptableObject* sea una lista de armas y no la propiedades del arma en sí.
- Creamos la clase de datos con las variables publicas (para que sean editables en el futuro desde una herramienta):

```
[System.Serializable]
public class WeaponItem
{
    public string      m_name;
    public float       m_damage      = 80.0f;
    public float       m_forceToApply = 20.0f;
    public float       m_weaponRange = 9999.0f;
    public int         m_ammoCapacity = 3;
    public float       m_rateOfShot  = 10;
    public float       m_accuracy    = 70.0f;
    public float       m_accuracyDropPerShot = 10.0f;
    public float       m_accuracyRecoverPerSecond = 1.0f;
    public int         m_simultaneousShots = 1;
    public float       m_recoilBack   = 0.1f;
    public float       m_recoilRecovery = 0.01f;
    public bool        m_isAMachineGun = false;
    public bool        m_isRocketLauncher = false;
    public GameObject  m_rocket;
    public Texture2D   m_crosshairTexture;
    public AudioClip   m_fireSound;
    public AudioClip   m_reloadSound;
}
```

# Scriptable Object

- Ahora creamos el *scriptableObject* que contiene la lista de “*WeaponItems*”

```
public class WeaponList : ScriptableObject
{
    public List<WeaponItem> weaponList;
}
```



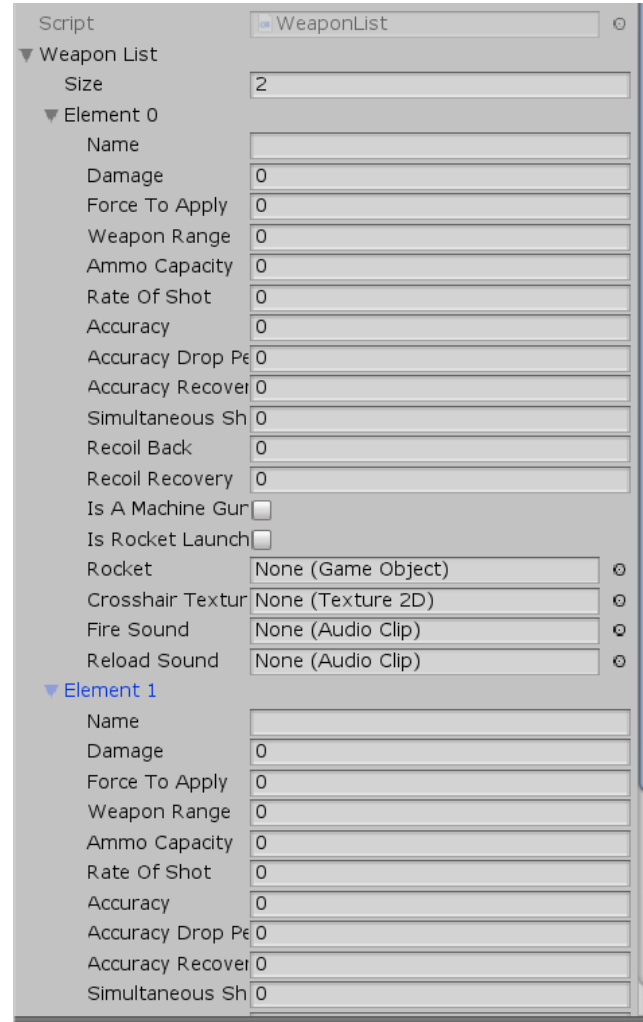
# Scriptable Object

- Por último, creamos un script que nos instancie la lista de armas desde una opción del editor.

```
public class CreateWeaponList
{
    [MenuItem("Assets/Create/Weapon List")]
    public static WeaponList Create()
    {
        WeaponList asset = ScriptableObject.CreateInstance<WeaponList>();
        AssetDatabase.CreateAsset(asset, "Assets/WeaponList.asset");
        AssetDatabase.SaveAssets();
        return asset;
    }
}
```

# Scriptable Object

- Al crear el *scriptableObject* de la lista, tiene esta pinta en el editor:



# Scriptable Object

- En próximos temas veremos cómo crear una herramienta para que modificar esta lista sea algo más cómodo que hacerlo directamente desde el editor.
- Para cargar estos datos, metemos el *scriptableObject* que hemos creado en la carpeta *resources* y ya podemos cargarlo directamente:

```
private WeaponList m_weaponList;

public void LoadWeaponList ()
{
    m_weaponList = Resources.Load<WeaponList>("WeaponList");

    for (int i = 0; i < m_weaponList.weaponList.Count; i++)
    {
        Debug.Log(m_weaponList.weaponList[i].m_name);
    }
}
```

# Scriptable Object

- Ejercicio:
  - Modificar el script del arma para que sus datos estén en un scriptableObject.

