Documentation-Jan: WeaponHandling System (v. 17.05.24)

General:

- Currently there are for different Weapon Types (Range Weapons)
    - ‚Handcannon.cs‘
    - ‚Submachinegun.cs’
    - ‚Shotgun.cs‘
    - ‚Energy Launcher.cs‘
    - all derive from ‚BaseWeapon.cs‘ which provides the basic variables and functionality (compare Fig. 1-3)

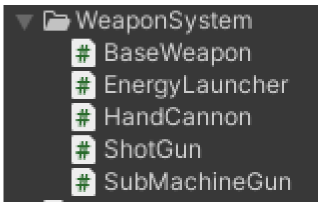


Fig. 1: WeaponType Classes in the Project Folder (currently to be found under Testing/Jan/Scripts/WeaponSystem)

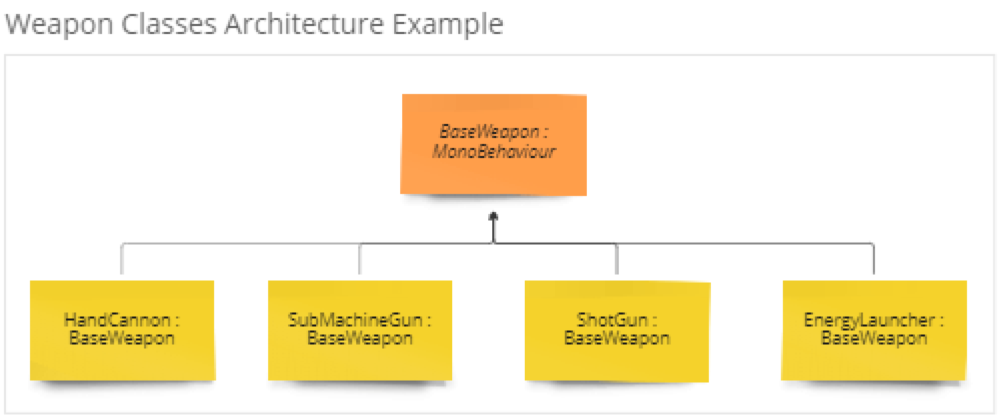

Weapon Classes Architecture Example
BaseWeapon : MonoBehaviour
HandCannon : BaseWeapon
SubMachineGun : BaseWeapon
ShotGun : BaseWeapon
EnergyLauncher : BaseWeapon

Fig. 2: Inheritance Architecture of Weapon Types

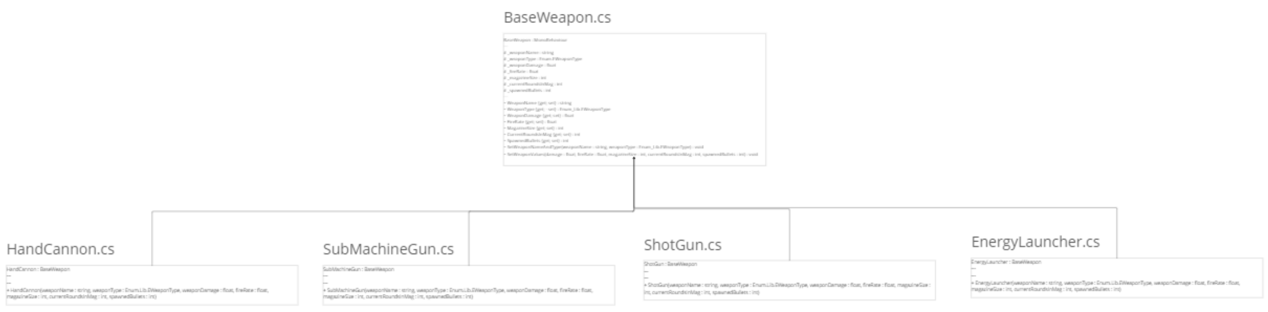


Fig. 3: Class Diagram of the Weapon Types

- Also there currently are two more Classes:
  - **PlayerWeaponHandling.cs**
    - Handles all Weapon and Attack related logic regarding the Player, like Input operations, Shooting, Weapon Swapping and Holstering, Bullet Instantiation on shooting etc. (Compare Fig. 4)
  - **PlayerEquipmentSO.cs**
    - Is a Scriptable Object and handles the deeper Logic behind Weapon Pickup and Swapping for now.
    - It also contains a Struct ‚**WeaponTypeValues**‘
      - this Struct is basically used and necessary for the Inspectorvisualization and permanently saving of the changes made in the Inspector of the ‚PlayerEquipment‘-Asset (compare Fig. 4-6)

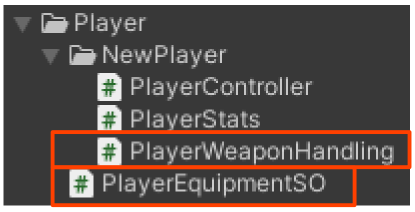


Fig. 4: The PlayerWeaponHandling.cs and PlayerEquipmentSO.cs in the Project folder (currently still under Testing/Jan/Scripts/Player)

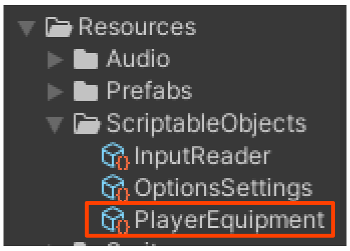
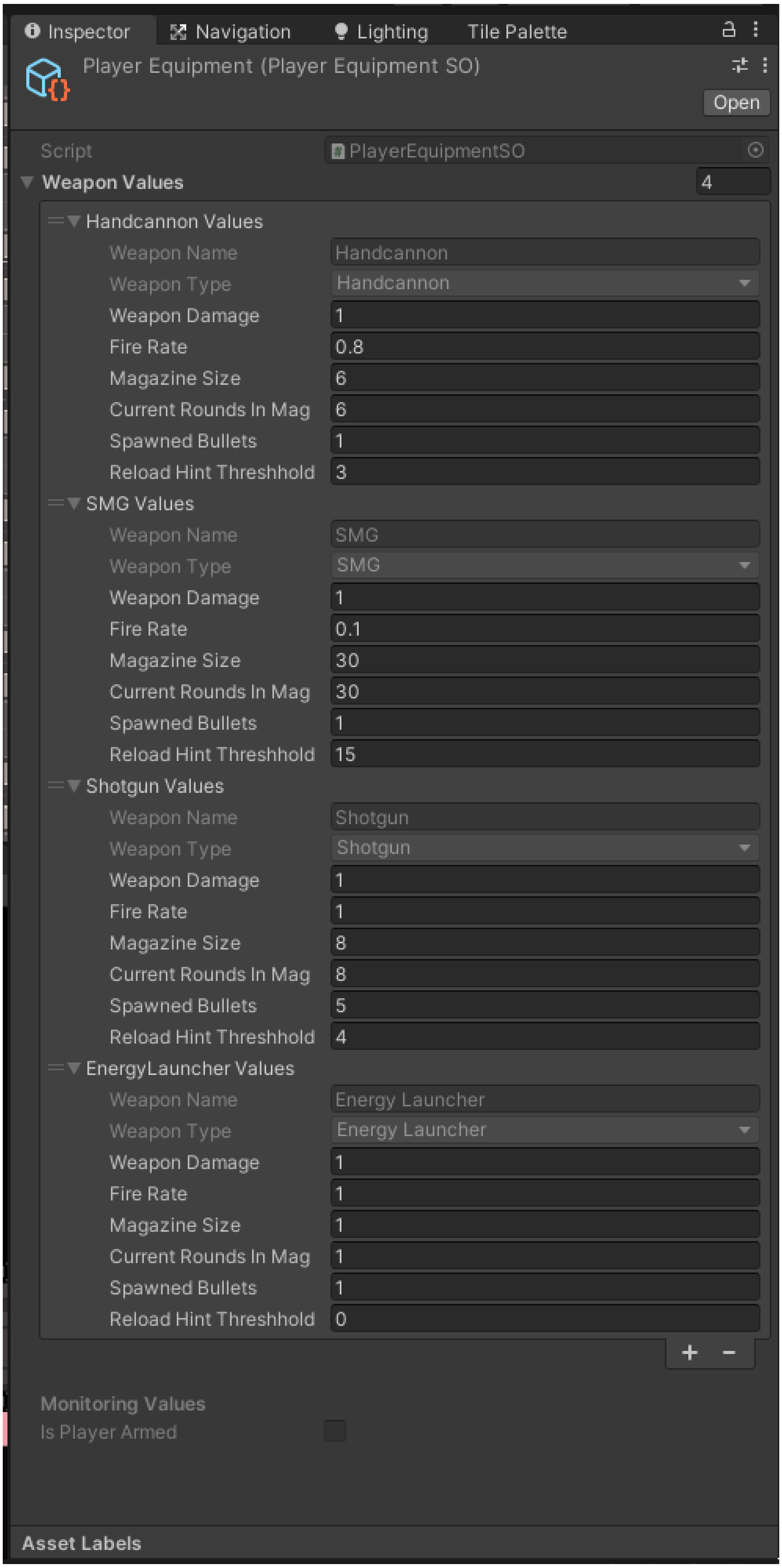


Fig. 5: The PlayerEquiptment-Asset (found currently in the Project Folder under Resources/ScriptableObjects)



Player Equipment (Player Equipment SO)

Script	PlayerEquipmentSO
Weapon Values	4
Handcannon Values	
Weapon Name	Handcannon
Weapon Type	Handcannon
Weapon Damage	1
Fire Rate	0.8
Magazine Size	6
Current Rounds In Mag	6
Spawned Bullets	1
Reload Hint Threshhold	3
SMG Values	
Weapon Name	SMG
Weapon Type	SMG
Weapon Damage	1
Fire Rate	0.1
Magazine Size	30
Current Rounds In Mag	30
Spawned Bullets	1
Reload Hint Threshhold	15
Shotgun Values	
Weapon Name	Shotgun
Weapon Type	Shotgun
Weapon Damage	1
Fire Rate	1
Magazine Size	8
Current Rounds In Mag	8
Spawned Bullets	5
Reload Hint Threshhold	4
EnergyLauncher Values	
Weapon Name	Energy Launcher
Weapon Type	Energy Launcher
Weapon Damage	1
Fire Rate	1
Magazine Size	1
Current Rounds In Mag	1
Spawned Bullets	1
Reload Hint Threshhold	0

Monitoring Values

Is Player Armed	☐

Fig. 6: The Inspector view of the PlayerEquipment-Asset. Here the Values for the specific Weapon Types can be set. Currently they are set to the Standard Values.

- For the Weapon Pickup system to work an Weapon-GameObject would need to be set in the Scene and needs to implement a ‚WeaponTypeObject‘-component where the Weapon Type would need to be specified (compare Fig 7-8)

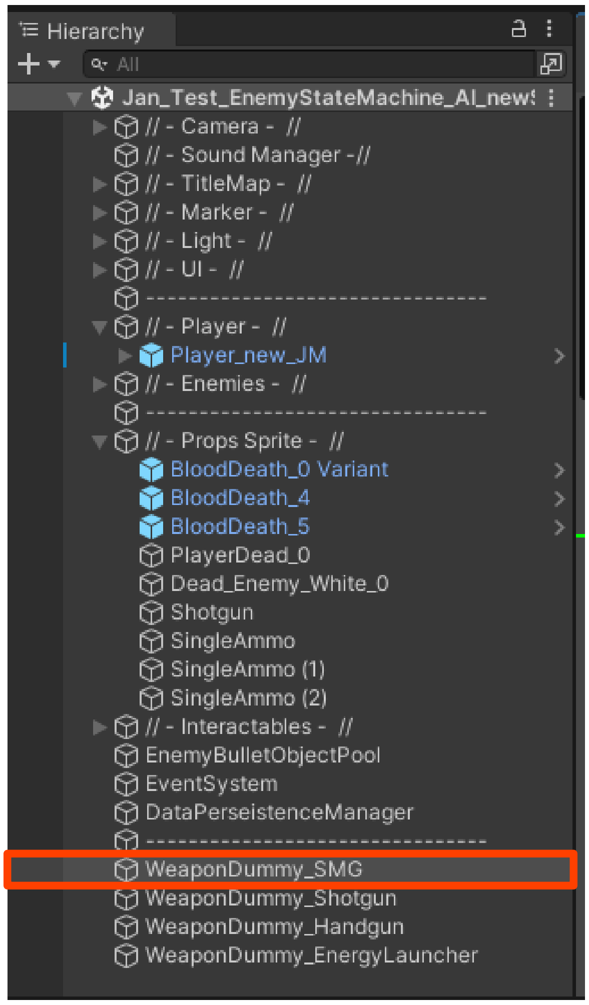
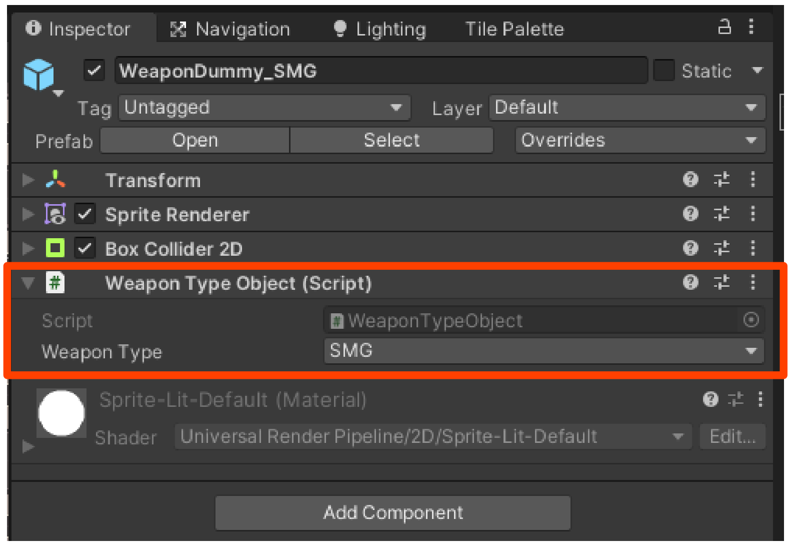


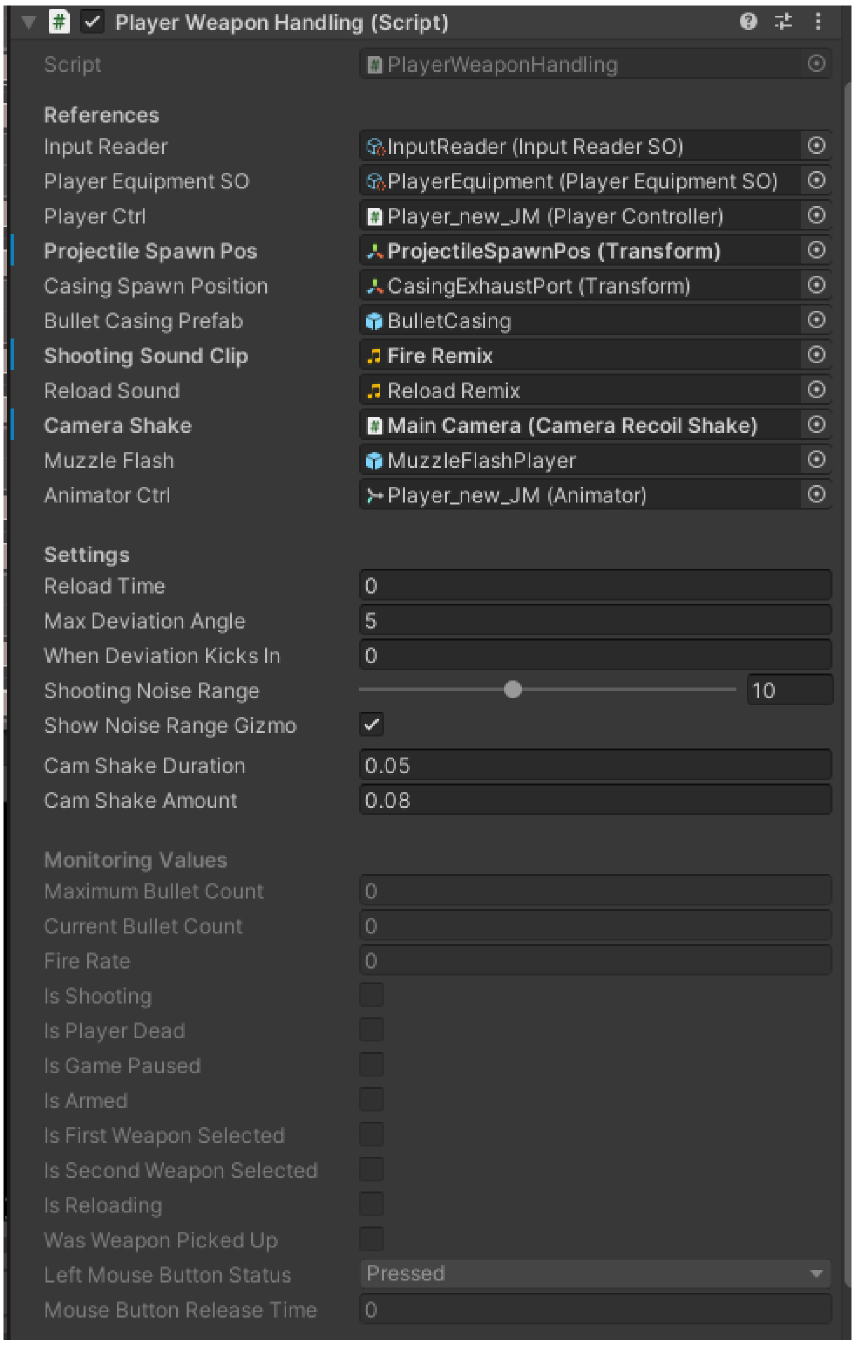
Fig. 7: The Hierarchy with a Weapon Dummy Objec highlighted.



WeaponDummy_SMG
Static
Tag Untagged
Layer Default
Weapon Type Object (Script)
Script WeaponTypeObject
Weapon Type SMG
Sprite-Lit-Default (Material)
Shader Universal Render Pipeline/2D/Sprite-Lit-Default

Fig. 8: The Inspector View of the Weapon Dummy Object. Note in this Case the Weapon Type 'SMG' is selected for this Object.

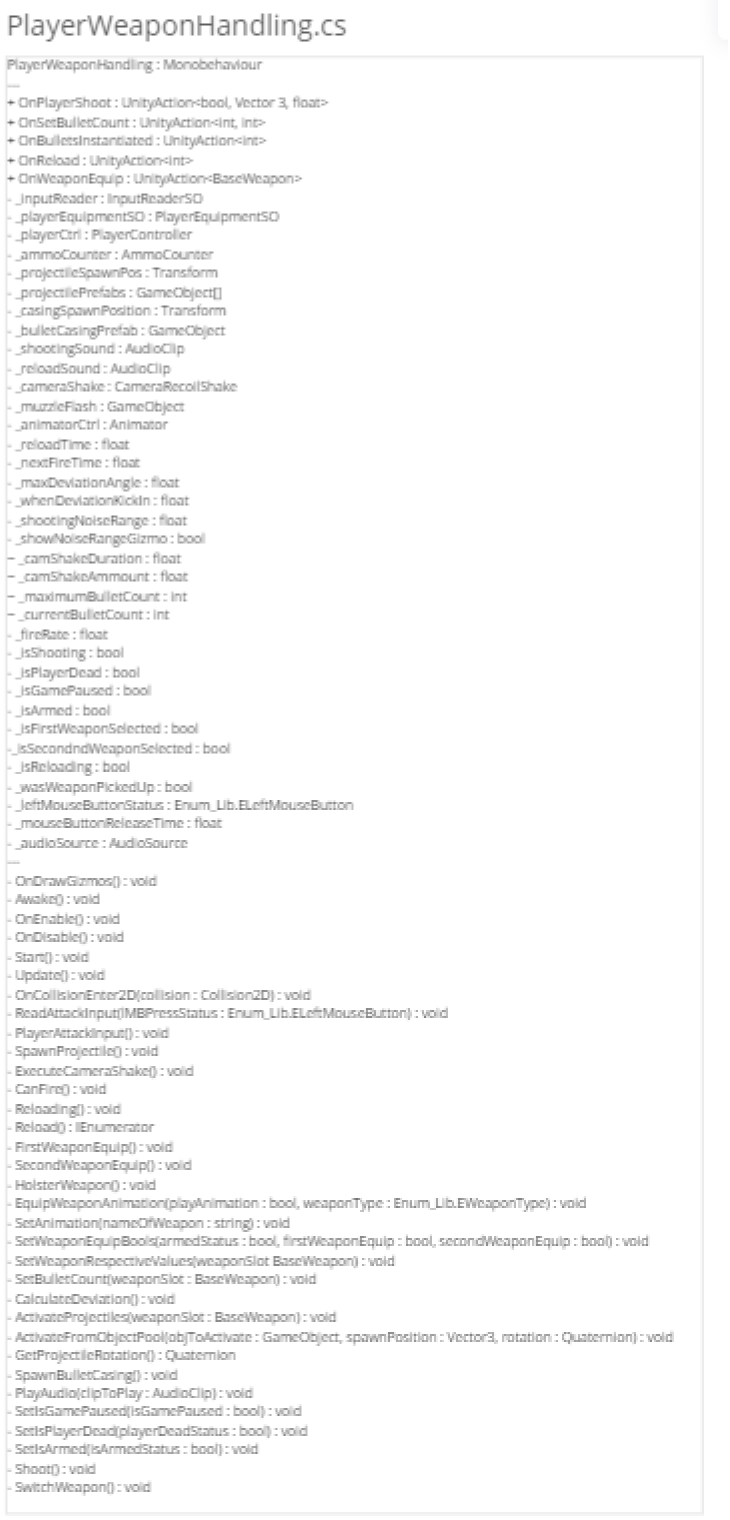
- Also the Player Object needs to implement the ‚PlayerWeaponHandling' for a working Weapon/Attack Behaviour (compare Fig. 9)
    - Note for the Weapon-Pickupsystem to work the Player and the WeaponObject in the Scene also needs to implement a Collider2D-Component(!)



Player Weapon Handling (Script)

Script	PlayerWeaponHandling
References	
Input Reader	InputReader (Input Reader SO)
Player Equipment SO	PlayerEquipment (Player Equipment SO)
Player Ctrl	Player_new_JM (Player Controller)
Projectile Spawn Pos	ProjectileSpawnPos (Transform)
Casing Spawn Position	CasingExhaustPort (Transform)
Bullet Casing Prefab	BulletCasing
Shooting Sound Clip	Fire Remix
Reload Sound	Reload Remix
Camera Shake	Main Camera (Camera Recoil Shake)
Muzzle Flash	MuzzleFlashPlayer
Animator Ctrl	Player_new_JM (Animator)
Settings	
Reload Time	0
Max Deviation Angle	5
When Deviation Kicks In	0
Shooting Noise Range	10
Show Noise Range Gizmo	✓
Cam Shake Duration	0.05
Cam Shake Amount	0.08
Monitoring Values	
Maximum Bullet Count	0
Current Bullet Count	0
Fire Rate	0
Is Shooting	☐
Is Player Dead	☐
Is Game Paused	☐
Is Armed	☐
Is First Weapon Selected	☐
Is Second Weapon Selected	☐
Is Reloading	☐
Was Weapon Picked Up	☐
Left Mouse Button Status	Pressed
Mouse Button Release Time	0

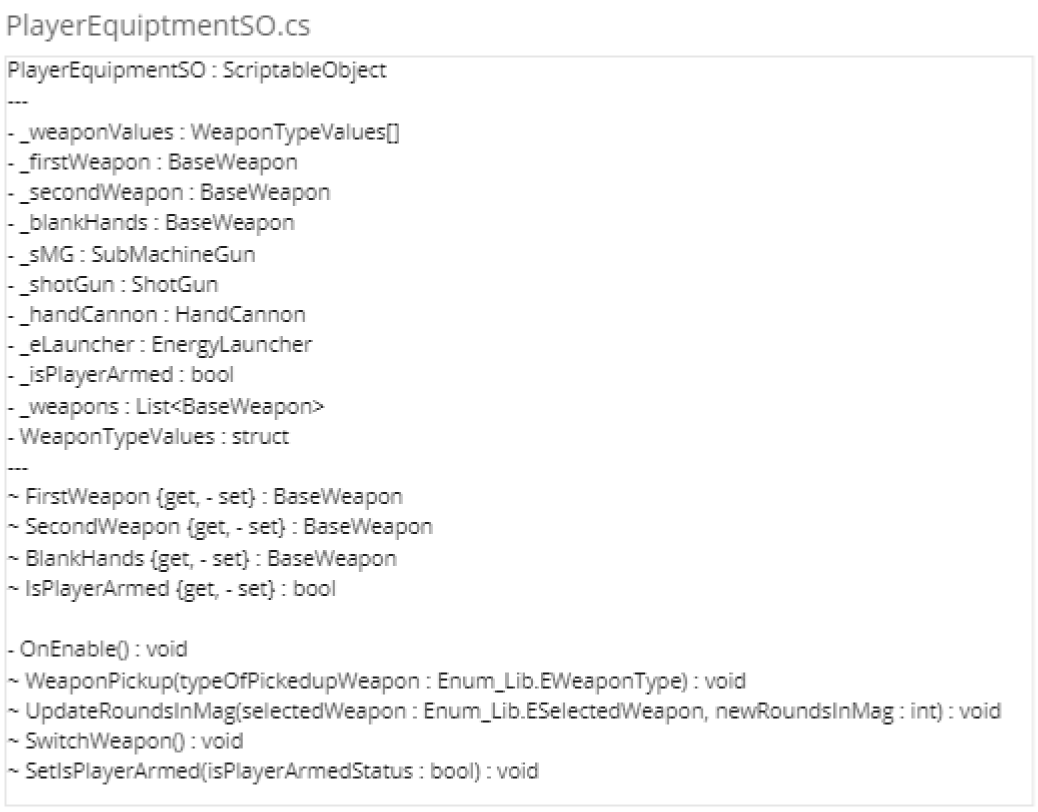
Fig. 9: The Inspector view of the PlayerWeaponHandling.cs when applied on the PlayerObject. With a example Setup (probably might change during further development)



PlayerWeaponHandling.cs

```
PlayerWeaponHandling : Monobehaviour
---
+ OnPlayerShoot : UnityAction<bool, Vector 3, float>
+ OnSetBulletCount : UnityAction<int, int>
+ OnBulletsInstantiated : UnityAction<int>
+ OnReload : UnityAction<int>
+ OnWeaponEquip : UnityAction<BaseWeapon>
- _inputReader : InputReaderSO
- _playerEquipmentSO : PlayerEquipmentSO
- _playerCtrl : PlayerController
- _ammoCounter : AmmoCounter
- _projectileSpawnPos : Transform
- _projectilePrefabs : GameObject[]
- _casingSpawnPosition : Transform
- _bulletCasingPrefab : GameObject
- _shootingSound : AudioClip
- _reloadSound : AudioClip
- _cameraShake : CameraRecoilShake
- _muzzleFlash : GameObject
- _animatorCtrl : Animator
- _reloadTime : float
- _nextFireTime : float
- _maxDeviationAngle : float
- _whenDeviationKickIn : float
- _shootingNoiseRange : float
- _showNoiseRangeGizmo : bool
– _camShakeDuration : float
– _camShakeAmmount : float
– _maximumBulletCount : int
– _currentBulletCount : int
- _fireRate : float
- _isShooting : bool
- _isPlayerDead : bool
- _isGamePaused : bool
- _isArmed : bool
- _isFirstWeaponSelected : bool
-_isSecondndWeaponSelected : bool
- _isReloading : bool
- _wasWeaponPickedUp : bool
- _leftMouseButtonStatus : Enum_Lib.ELeftMouseButton
- _mouseButtonReleaseTime : float
- _audioSource : AudioSource
---
- OnDrawGizmos() : void
- Awake() : void
- OnEnable() : void
- OnDisable() : void
- Start() : void
- Update() : void
- OnCollisionEnter2D(collision : Collision2D) : void
- ReadAttackInput(lMBPressStatus : Enum_Lib.ELeftMouseButton) : void
- PlayerAttackInput() : void
- SpawnProjectile() : void
- ExecuteCameraShake() : void
- CanFire() : void
- Reloading() : void
- Reload() : IEnumerator
- FirstWeaponEquip() : void
- SecondWeaponEquip() : void
- HolsterWeapon() : void
- EquipWeaponAnimation(playAnimation : bool, weaponType : Enum_Lib.EWeaponType) : void
- SetAnimation(nameOfWeapon : string) : void
- SetWeaponEquipBools(armedStatus : bool, firstWeaponEquip : bool, secondWeaponEquip : bool) : void
- SetWeaponRespectiveValues(weaponSlot BaseWeapon) : void
- SetBulletCount(weaponSlot : BaseWeapon) : void
- CalculateDeviation() : void
- ActivateProjectiles(weaponSlot : BaseWeapon) : void
- ActivateFromObjectPool(objToActivate : GameObject, spawnPosition : Vector3, rotation : Quaternion) : void
- GetProjectileRotation() : Quaternion
- SpawnBulletCasing() : void
- PlayAudio(clipToPlay : AudioClip) : void
- SetIsGamePaused(isGamePaused : bool) : void
- SetIsPlayerDead(playerDeadStatus : bool) : void
- SetIsArmed(isArmedStatus : bool) : void
- Shoot() : void
- SwitchWeapon() : void
```

Fig.10: Current Code-Set up of the PlayerWeaponHandling.cs

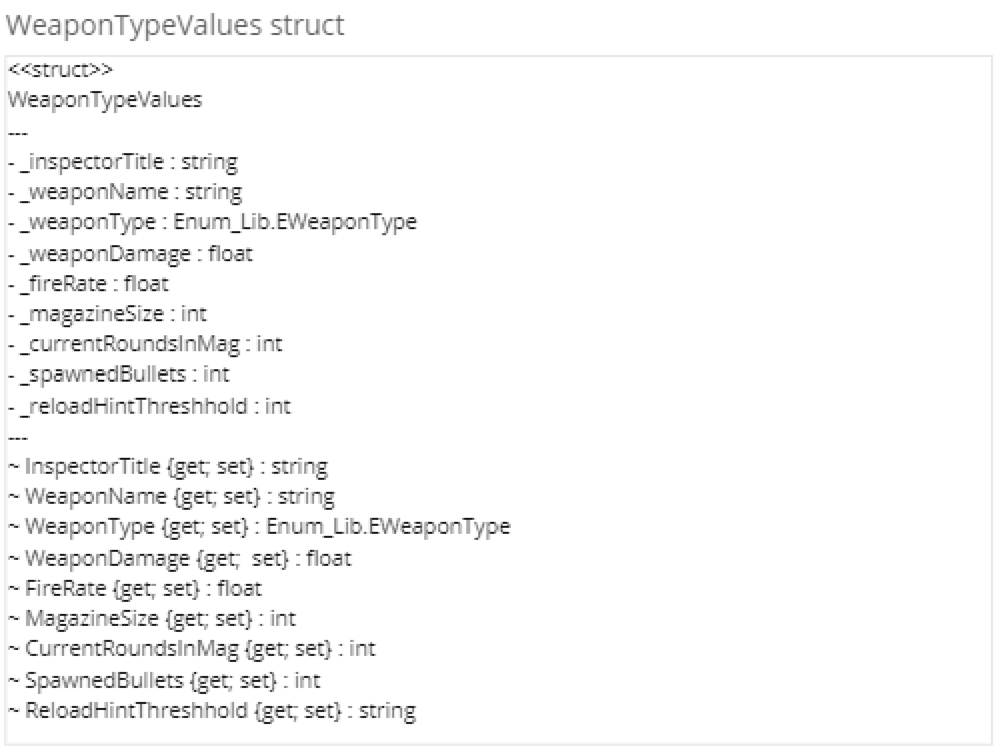


PlayerEquiptmentSO.cs

```
PlayerEquipmentSO : ScriptableObject
---
- _weaponValues : WeaponTypeValues[]
- _firstWeapon : BaseWeapon
- _secondWeapon : BaseWeapon
- _blankHands : BaseWeapon
- _sMG : SubMachineGun
- _shotGun : ShotGun
- _handCannon : HandCannon
- _eLauncher : EnergyLauncher
- _isPlayerArmed : bool
- _weapons : List<BaseWeapon>
- WeaponTypeValues : struct
---
~ FirstWeapon {get, - set} : BaseWeapon
~ SecondWeapon {get, - set} : BaseWeapon
~ BlankHands {get, - set} : BaseWeapon
~ IsPlayerArmed {get, - set} : bool

- OnEnable() : void
~ WeaponPickup(typeOfPickedupWeapon : Enum_Lib.EWeaponType) : void
~ UpdateRoundsInMag(selectedWeapon : Enum_Lib.ESelectedWeapon, newRoundsInMag : int) : void
~ SwitchWeapon() : void
~ SetIsPlayerArmed(isPlayerArmedStatus : bool) : void
```

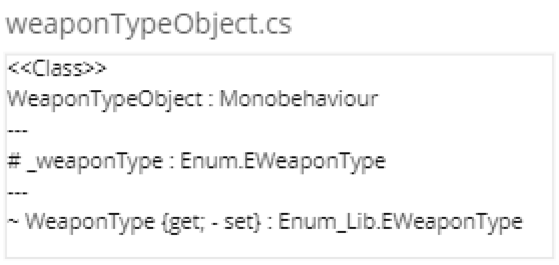
Fig. 11: current Code-Setup of the PlayerEquipmentSO.cs



WeaponTypeValues struct

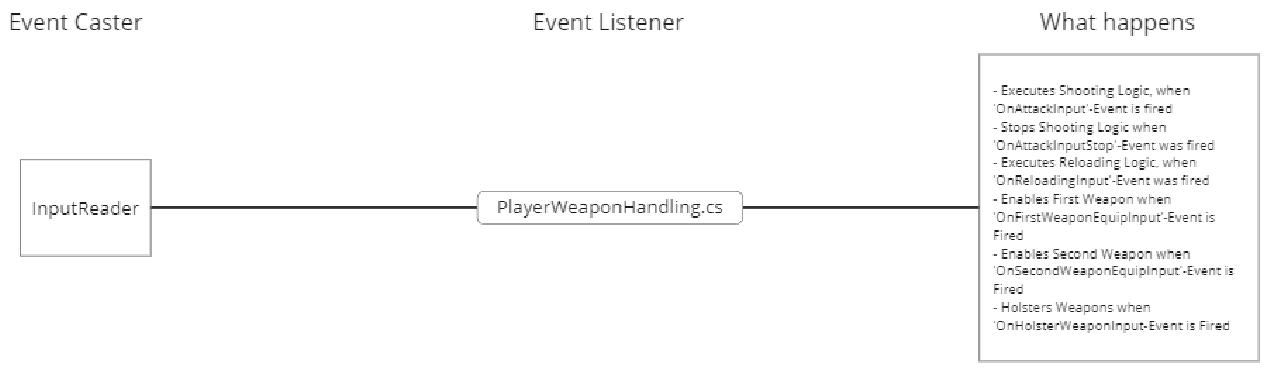
```
<<struct>>
WeaponTypeValues
---
- _inspectorTitle : string
- _weaponName : string
- _weaponType : Enum_Lib.EWeaponType
- _weaponDamage : float
- _fireRate : float
- _magazineSize : int
- _currentRoundsInMag : int
- _spawnedBullets : int
- _reloadHintThreshhold : int
---
~ InspectorTitle {get; set} : string
~ WeaponName {get; set} : string
~ WeaponType {get; set} : Enum_Lib.EWeaponType
~ WeaponDamage {get;  set} : float
~ FireRate {get; set} : float
~ MagazineSize {get; set} : int
~ CurrentRoundsInMag {get; set} : int
~ SpawnedBullets {get; set} : int
~ ReloadHintThreshhold {get; set} : string
```

Fig. 12: Current Setup of the Struct 'WeaponTypeValues' wich is declared and defined in the 'PlayerEquipmentSO.cs'; This Struct is used to create a proper Inspector View of the 'PlayerEquipmentSO.cs'


weaponTypeObject.cs

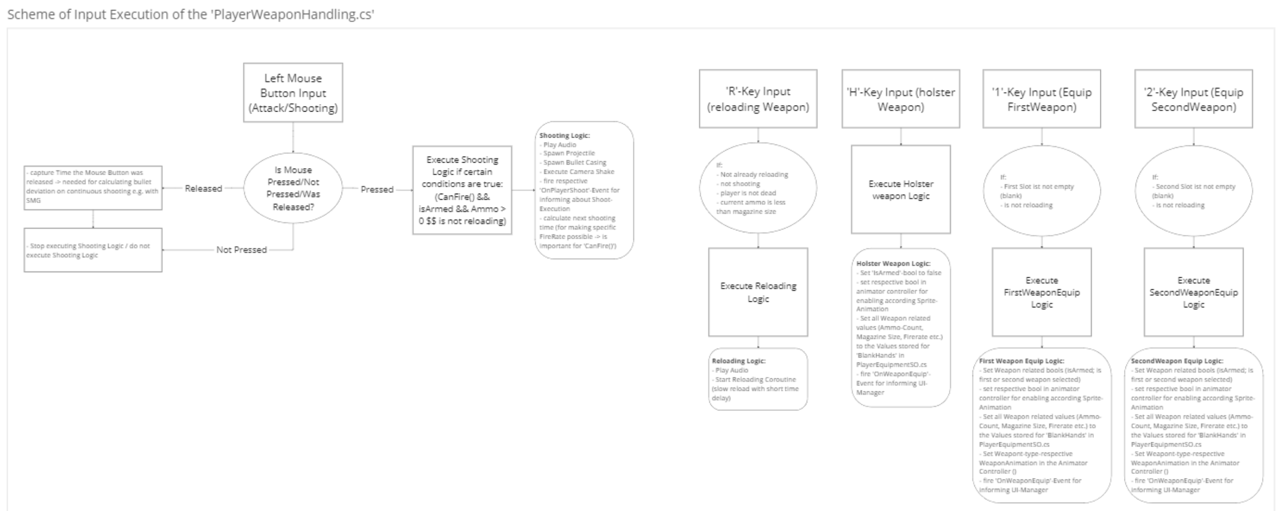
```
<<Class>>
WeaponTypeObject : Monobehaviour
---
# _weaponType : Enum.EWeaponType
---
~ WeaponType {get; - set} : Enum_Lib.EWeaponType
```

Fig. 13: Current Code-Setup of the 'WeaponTypeObject.cs' which is used (and necessary) as a component for actual Weapon-Game-Objects in the Scene that can be picked up by the player.



Event Caster
Event Listener
What happens
InputReader
PlayerWeaponHandling.cs
- Executes Shooting Logic, when 'OnAttackInput'-Event is fired
- Stops Shooting Logic when 'OnAttackInputStop'-Event was fired
- Executes Reloading Logic, when 'OnReloadingInput'-Event was fired
- Enables First Weapon when 'OnFirstWeaponEquipInput'-Event is Fired
- Enables Second Weapon when 'OnSecondWeaponEquipInput'-Event is Fired
- Holsters Weapons when 'OnHolsterWeaponInput-Event is Fired

Fig. 14: scheme of the input-related event-chain for the WeaponHandling



Scheme of Input Execution of the 'PlayerWeaponHandling.cs'
Left Mouse Button Input (Attack/Shooting)
Is Mouse Pressed/Not Pressed/Was Released?
Released
- capture Time the Mouse Button was released -> needed for calculating bullet deviation on continuous shooting e.g. with SMG
Not Pressed
- Stop executing Shooting Logic / do not execute Shooting Logic
Pressed
Execute Shooting Logic if certain conditions are true: (CanFire() && isArmed && Ammo > 0 $$ is not reloading)
Shooting Logic:
- Play Audio
- Spawn Projectile
- Spawn Bullet Casing
- Execute Camera Shake
- fire respective 'OnPlayerShoot'-Event for informing about Shoot-Execution
- calculate next shooting time (for making specific FireRate possible -> is important for 'CanFire()')
'R'-Key Input (reloading Weapon)
If:
- Not already reloading
- not shooting
- player is not dead
- current ammo is less than magazine size
Execute Reloading Logic
Reloading Logic:
- Play Audio
- Start Reloading Coroutine (slow reload with short time delay)
'H'-Key Input (holster Weapon)
Execute Holster weapon Logic
Holster Weapon Logic:
- Set 'IsArmed'-bool to false
- set respective bool in animator controller for enabling according Sprite-Animation
- Set all Weapon related values (Ammo-Count, Magazine Size, Firerate etc.) to the Values stored for 'BlankHands' in PlayerEquipmentSO.cs
- fire 'OnWeaponEquip'-Event for informing UI-Manager
'1'-Key Input (Equip FirstWeapon)
If:
- First Slot ist not empty (blank)
- is not reloading
Execute FirstWeaponEquip Logic
First Weapon Equip Logic:
- Set Weapon related bools (isArmed; is first or second weapon selected)
- set respective bool in animator controller for enabling according Sprite-Animation
- Set all Weapon related values (Ammo-Count, Magazine Size, Firerate etc.) to the Values stored for 'BlankHands' in PlayerEquipmentSO.cs
- Set Weapont-type-respective WeaponAnimation in the Animator Controller ()
- fire 'OnWeaponEquip'-Event for informing UI-Manager
'2'-Key Input (Equip SecondWeapon)
If:
- Second Slot ist not empty (blank)
- is not reloading
Execute SecondWeaponEquip Logic
SecondWeapon Equip Logic:
- Set Weapon related bools (isArmed; is first or second weapon selected)
- set respective bool in animator controller for enabling according Sprite-Animation
- Set all Weapon related values (Ammo-Count, Magazine Size, Firerate etc.) to the Values stored for 'BlankHands' in PlayerEquipmentSO.cs
- Set Weapont-type-respective WeaponAnimation in the Animator Controller ()
- fire 'OnWeaponEquip'-Event for informing UI-Manager

Fig. 15: Scheme on how the Input related logic works for the WeaponHandling System