Weapon Management System

Version 1.0

Parvesh Yadav

December 11, 2025

Contents

1 Introduction

This document describes the Weapon Management System developed in Unity for a simple first-person shooter (FPS) environment. The system allows a player to fire weapons, reload, switch weapons, drop weapons, and pick up new weapons from the scene.

The project uses C# scripts, Unity prefabs, a CharacterController-based movement system, and ScriptableObjects to define weapon data. The focus of this documentation is to explain the structure of the system, how each feature works, and how to extend the system by adding new guns.

The system was designed to be simple, modular, and readable for beginners.

2 System Overview and Problem Description

The goal of the system is to provide the basic functionality required for a weapon framework inside a Unity FPS game. The system solves the following problems:

- Players must be able to hold a weapon and fire it.
- Weapons must have ammo, reload logic, and fire-rate limits.
- Players must be able to swap between multiple guns using the mouse scroll wheel.
- Weapons in the world must be pickable using the E key.
- Players must be able to drop their current weapon using the G key.
- New guns must be easy to add without editing core gameplay scripts.

This system supports two shooting types:

- **Hitscan** weapons (instant raycast shots)
- **Projectile** weapons (e.g., spheres or bullets fired using physics)

The system starts with a **single gun equipped**. A second pistol model (a black cube) can be picked up in the scene, after which the player can scroll to swap between the two.

3 Feature List

- **Weapon firing**: Left mouse button shoots.
- **Reloading**: Press R to refill magazine.
- **Weapon swapping**: Use mouse scroll wheel to switch between available weapons.
- **Weapon dropping**: Press G to drop the currently equipped weapon into the scene.
- **Weapon pickup**: Walk over a pickup object and press E to add it to your weapon list.
- **Pickup UI message**: Displays “Press E to pick up...” when near a weapon.
- **Recoil**: Camera kickback when firing.
- **Hitscan or Projectile**: WeaponDefinition decides if bullets are instant or physics-based.

4 System Installation and Usage

4.1 Project Setup

To run or modify the system:

1. Open the Unity project in Unity Editor.
2. Open the sample scene provided.
3. Ensure the Player object includes:
    - `SimpleFPSController`
    - `WeaponController`
    - `LegacyInputProvider`
    - `HitScanBehaviour`
    - `ProjectileBehaviour`
    - `SimpleRecoilModel`

4. Ensure the Canvas contains:
    - Weapon name text
    - Pickup message text
    - Crosshair

4.2 How To Use the System (Controls)

- **Left Click** – Fire the current weapon
- **R** – Reload
- **Mouse Scroll** – Swap between all weapons you own
- **G** – Drop the current weapon on the ground
- **E** – Pick up a weapon if standing near one

The player begins with one weapon equipped. A pistol pickup (black cube) is placed in the scene so the user can test weapon pickup and swapping.

5 System Components

This section briefly explains the core parts of the system without including full code dumps.

5.1 WeaponDefinition (ScriptableObject)

Defines data for each gun:

- weapon name
- damage
- fire rate
- magazine size and max reserve ammo
- hitscan or projectile setting
- recoil values

New guns are created by right-clicking in the Project window:

```
Create → Weapon System → Weapon Definition
```

5.2 WeaponController

Controls the entire weapon logic:

- equipping and swapping weapons
- firing logic
- reloading
- dropping weapons
- picking up new weapons item displaying weapon UI info

5.3 HitScanBehaviour

Instant shooting using a raycast.

5.4 ProjectileBehaviour

Spawns a projectile prefab and applies forward velocity.

5.5 SimpleRecoilModel

Moves the camera slightly to simulate recoil.

5.6 LegacyInputProvider

Provides all input (fire, reload, scroll, drop, pickup).

5.7 WeaponPickup Object

A world object that can be picked up using E. It stores a WeaponDefinition and gives it to the player.

6 System Algorithms and Models

6.1 Hitscan Shooting

Hitscan simply uses a Unity raycast:

- From the camera forward direction
- With optional spread added using random angles
- If it hits an object that implements `IDamageable`, damage is applied

6.2 Projectile Shooting

Projectiles:

- Are instantiated at the weapon muzzle
- Use a Rigidbody to move forward
- Destroy themselves or disable after collision

6.3 Weapon Swapping

The system reads:

$$\text{scrollDelta} = \begin{cases} +1 & \text{scroll up} \\ -1 & \text{scroll down} \end{cases}$$

This moves the weapon index in a circular list.

6.4 Pickup Logic

When the player enters a trigger:

1. Store the weapon as “nearbyPickup”
2. Show UI message
3. On pressing E:
    - Add weapon to list
    - Equip it automatically

- Destroy the pickup object

6.5 Dropping Weapons

When G is pressed:

1. Instantiate drop prefab at muzzle
2. Assign its WeaponDefinition
3. Remove current weapon from list
4. Equip the next weapon or none if empty

7 Demo Scene Description

The demo scene contains:

- A Player with movement and weapon scripts
- Two weapon definitions: Rifle and Pistol
- A pistol pickup in the world (black cube)
- Several cubes as targets implementing `IDamageable`
- A UI canvas with weapon name, crosshair, and pickup message

Demonstrated features:

- Shooting cubes reduces their health
- Player can walk to pistol pickup and press E
- After picking up, scroll wheel swaps between rifle and pistol
- Player can drop any weapon and re-pick it

8 How to Add New Guns

To create a new weapon:

1. Right-click in the Project window `Create → Weapon System → Weapon Definition`

2. Set stats (damage, fire rate, recoil, hitscan/projectile)
3. Add the new WeaponDefinition to the WeaponController list
4. If projectile-based, assign a projectile prefab
5. If you want a pickup:
    - Create a cube or model
    - Add the `WeaponPickup` script
    - Assign the new WeaponDefinition
    - Add a BoxCollider (IsTrigger = true)

9 Conclusion

This system provides a clean and expandable foundation for FPS weapon mechanics. It handles shooting, ammo, reloading, weapon swapping, pickups, and drops while using ScriptableObjects to keep weapon data modular.

The system is easy to extend with new guns, visual models, UI elements, or advanced mechanics.