Wraith FPS Unity Project Documentation

Main Top Features:

- Accurate Triple A Style Recoil
- Weapon System
- Expandable Weapon System, supporting two guns, and a knife
- Intermediate FPS controller
- FPS Player Controller
- RTS Manager

Details:

Shootable Script
Anything with the shootable script can be shot and registered. Other scripts can listen to the events of this script to register shots to their scripts.

Player Animations Script
During crouching, the mesh size is decreased. Also, the mesh is hidden in the first player view and enabled on switching to RTS mode.

Player Camera
Like a standard FPS, the camera rotates the player body, and rotates the camera. The heights are clamped, even when adding recoil. The camera has a dynamic FOV system, during events the FOV changes. Sprinting increases the FOV, shooting temporarily increases the FOV. The camera has multiple sensitivities for aiming and regular view.

IController

IController holds a contract that other scripts can use for changing behavior depending on movement state. Any controller can implement this interface. This is useful because multiple different controllers can be used and communicate with other scripts through this interface.

FPS Controller
The player controller supports movement, jumping, jump rising on holding space bar, sprinting, and crouching. There is an OnLand event for creating

Player Controller
The player controller uses the FPS controller and simply passes the input to it.

Weapon System
The weapon system implements the IController. Prefabs are used to load the weapons. The weapon system holds the audio sources and is responsible for playing the weapons audio. This is because on switching weapons, or disabling them, lingering sound effects are still played. Weapons are switched, optionally including the knife in the weapon cycle. The weapon system is not tied to the player, so any enemy or friendly can use the same controller with ease.

Weapon Transform
The weapon transform scripts enabled weapon bobbing, idle animations, weapon sway with the mouse, and fall effects. These are created using a sine wave.

Weapon Recoil System
The recoil works like standard FPS games. When fired, the camera is pushed up. If the player does nothing, the camera damps back

down the original position. If the player controls the recoil, the recoil is canceled out. Weapons can select from a range from Extreme to Low recoil, both on the X and Y axis. When looking at the sky or in a location where the camera can’t be moved, recoil won’t be added and will be subtracted and canceled correctly.

Weapon Bloom System
The bloom system creates a randomized recoil spread in the center of the screen in a circular pattern. The bloom is influenced by many factors such as the iController (more on that in a second) and weapon spread additions. During the shot process a Ray is calculated by the Bloom system with the direction in the circle.

Weapon SObj
Each weapon uses a scriptable object to set all of its properties. These can be edited during playmode, updated in real time, and saved. This holds the weapon types and their characteristics.

GunScript
Gunscript handled input, reloading, shooting, shooting effects, and gun behavior.

GunMag
Gunmag holds the current mag ammo, mag size, and total ammo. Reloads are subtracted from the total ammo count. If all rounds are shot, the last bullet isn't reloaded. This is because the last bullet isn't in the chamber, a clever and realistic detail. Reloads can be canceled by shooting or sprinting, or by switching weapons. If

you run out of ammo, the gun automatically reloads if possible. Reloads can't be canceled if there is 0 ammo in the mag, however the weapon can be switched. If there is no ammo, the weapon is automatically switched if the other one has ammo.

Decals
Decals are using the URP decal projector. These wrap around the textures in a realistic way.

UI Rundown
UIAmmo displays the current ammo and remaining ammo. Full mags will be green, and empty ammo will be red. UICrosshair adapts based on the gun bloom, or a preset position for non guns. All movement is animated. The crosshair fades out when aiming or when entering RTS mode. UIHitmarker handles the hitmarker when hitting *shootables* and changes color based on what is hit. The hit marker appears and fades out quickly. UIRedScreen text is used to alert the player, currently used to display "No ammo." There is a quick debug menu allowing you to disable recoil. Bloom, toggle ads, and some other things, which introduced me to the script structure.

Many features scripts are not dependent on other scripts being active. Meaning, disabling recoil is as simple as disabling the script. Bloom, weapon transforms, are examples of scripts that can just be disabled to disable certain behaviors. Disabling the PlayerWeaponController will no longer allow the player to use weapons.

Utils
The utils file contains some damp functions that are custom made to work consistently across multiple frame rates. Color alphas can also be damped, layers can be bitshifted for layer masks, or

destroying all children from a game object. Settingsmanager manages the settings. Refs is a singleton that carries references to the Player, RTSManager, EventManager, UICAnvas, and Decal Prefab. The GameManager manages the framerate and mouse capturing. The EventManager has events that can be globally subscribed to, such as playerShootingEnemy (ex. Hitmarker trigger) or RTS mode switches. Const carries input names, button indexes, smoothing values, layer masks and layers, and the camera limit.

RTSManager
The RTS to FPS mode can be activated by pressing F1. RTSManager handles letting any part of the game know which mode it is in, and also when it was just changed, a one time trigger.

RTSCamera
The RTSCamera smoothly transitions from RTS and FPS mode. The RTS Camera has a smooth controller, and its limits can be choosed. These, of course, are also smoothed.

RTSSelectedComponent
These are components added to selected objects. Only objects with a certain layer mask defined in Const can be selected.

RTSSelectedUnitsTable
This is a dictionary that holds the selected objects.

RTSSelection
RTSSelection handles creating the trigger mesh for selection, as well as the UI drag box. The drag box fades out after being made. Inputs must be very intentional, and there is code to prevent accidental boxes and shooting while switching code.

Organization

All methods, private fields, properties, public properties, const, and static fields are named according to Rider’s default system scheme. All private fields are marked private with an underscore. All files are named consistently and organized in folders to make it easy to find exactly what you want.