FPS Weapon Implementation & Animation Pipeline

Design Document

Lunar-Skie Hackett

2025

# Aims of the project

As outlined by the project definition, this is intended to be a multi-faceted project and will take place in two programs primarily: Blender for Rigging and 3D Animation and Unity 6000.0.32f1 as the game engine. This Framework would, in theory, provide an avenue for non-programmers to implement most kinds of weapons through heavy parameterisation. Animators will be able to easily work closely with/independently of the programmer to implement these weapons due to a streamlined pipeline and a convenient and efficient system for assigning animations to weapons.

This project also aims to further refine my animation skills and to enable me to develop a stable, clean, and consistent export pipeline from Blender to Unity, including the process of cleaning the exported files within Unity.

# Planned Features

## Weapon Implementation

### Firearms

Firearms are defined, for the purpose of this Framework, as anything that does not require the user to be within close range of their target. Firearms will have two main modes of “attack” – hereby referred to as a “type” – that they can use. Firearms will also have the option to “launch” the attack from either the centre of the player’s view, or from the weapon itself. Firearms will be influenced by a variety of parameters and states from the game, such as whether a player is aiming, moving, crouching, or sliding, how much the player has fired this weapon, the weapon’s own spread parameters and more.

#### Hitscan Firearms

Hitscan Firearms will fire an imaginary “ray” called a “raycast.” This ray checks along the path from the origin of the attack to an endpoint influenced by the parameters above that affect accuracy.

#### Projectile Firearms

Projectile Firearms will fire a rigidbody projectile (an object with physics). These projectiles will react to other objects in the game world, will be able to bounce off walls, and can be affected by anything that might affect any rigidbodies in the scene.

### Melee

Melee weapons will trace between two points each fixed timestep by storing their last position and performing a linecast between those two points, ensuring that no targets are missed between strikes. To ensure that targets are not missed due to being too close, the system will also cast outwards from the player to the end of the weapon along the path of the strike. Anything that can be damage will be cached and ignored by further traces until the attack has ended.

# Raycast Commands & Bullet Scheduling

An important part of the project’s codebase is multithreading. This ensures that gunfights remain performant in intense situations. In testing, Physics.Raycast could not be called from other threads. Thankfully, Unity provides another solution: RaycastCommand. These are, I believe, executed through the C# Jobs system themselves, but are called from the main thread and performed at the best available time. They are the easiest way to call a Raycast from the Jobs system. There is also an additional performance impact from the normalisation of the fire vector, since RaycastCommands will not accept a vector that is not normalised.

In its first iteration, the system used two lists of a BulletData struct, used to determine some information about the “bullets.” A Native Array of RaycastCommands and RaycastHits is currently used to store the information needed by the Raycast batch.

In its second iteration, the Raycast system uses an array with fixed length, set somewhere between 500 and 1000, depending on the expected workload. However, in tests, this system holds up *incredibly* well under stress tests.

### Test Conditions

In the scene, there is a camera for rendering, the Input Manager that the player uses, a Rigidbody with a hinge joint placed in front of the test weapon, and the test weapon itself.

The test weapon fires a variable number of raycasts every frame, with an optional divergence. The divergence is a maximum measure, divided by 2, of how perpendicular the fire vector can be to the forward vector of the weapon.

When a raycast hits the Rigidbody in the scene, it applies a force at the point the ray hits, pushing the object around in a circle.

The below image is a graph taken from the profiler, covering the span of 300 frames. For the duration of this, the test weapon was set to fire 5000 rays every tick. This does, however, also account for the slight impact of having the profiler open and recording frame statistics.

A screen shot of a screen

AI-generated content may be incorrect.