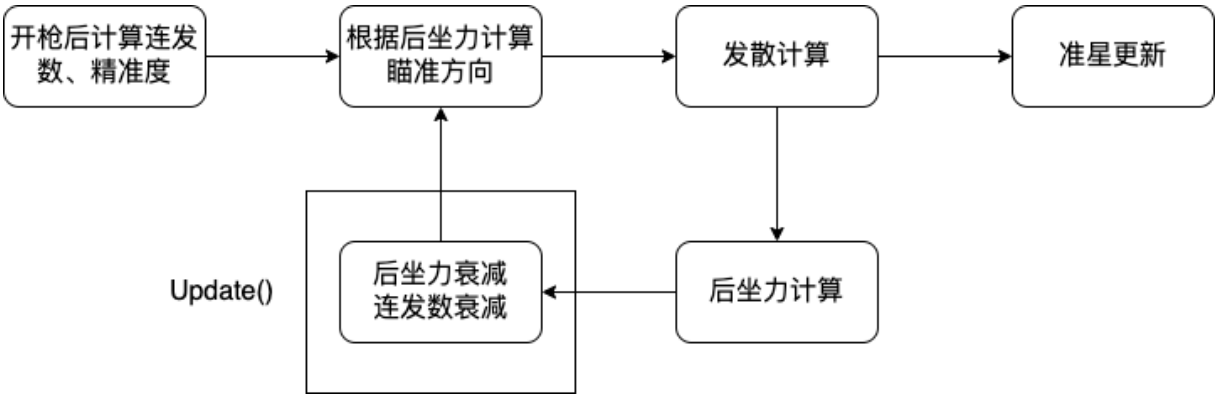
武器弹道模块

一、 CS 弹道模型



连发数（Brust）是指连续射击的次数，也是 CS 计算弹道的核心参数，主要有 3 个参数控制：

（1） 最大连发数

（2） 首次递减时间 t_1

（3） 每发递减时间 t_2

松开扳机后，先扣除 t_1 并减去一个 burst，再逐个扣除 t_2 直到 burst 变量为 0

二、 后坐力

1、 瞄准方向

后坐力决定了瞄准的方向。以 x 轴的后座方向计算为例：

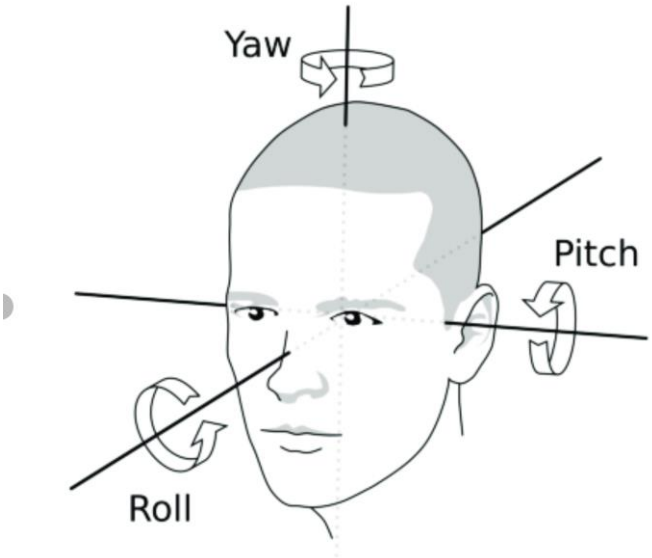
$$\begin{cases} recoil.x = baseX, & bursts = 1 \\ recoil.x = \min(baseX + burstsΔX, XMax) & bursts > 1 \end{cases}$$

瞄准方向的计算如下：

```
dRecoil = recoil - previousRecoil
aimRotation = cameraRotation - dRecoil
```

2、侧向和向上方向约束

有了方向以后，我们还需要给后坐力方向添加约束，防止瞄准方向偏移过度。在 Unity 坐标系中，Pitch 是围绕围绕 x 轴旋转的俯仰角，Yaw 是围绕 y 轴旋转的偏航角，Roll 是围绕 z 轴旋转是翻滚角。



因此我们主要限制 x 轴（侧向），y 上半轴（向上）方向上的移动，因为开火的枪口一般只会上跳。添加侧向偏移的约束：其中 recoil 表示后坐力的欧拉角

```
if lateralDir.Left:
    recoil.y -= deltaEulerLateral
    if recoil.y < -1 * LateralMax:
        recoil.y = -1 * LateralMax
else
    recoil.y += deltaEulerLateral
    if recoil.y > LateralMax:
        recoil.y = LateralMax
```

添加向上偏移约束

```
recoil.x -= deltaEulerUp
if recoil.x < -1 * UpMax:
    recoil.x = -1 * UpMax
```

3、后坐力的弹簧恢复

枪口向上抬起，左右晃动恢复到初始位置，可以用一个简化的弹簧模型

```
public float CalculateAccuracy(int bursts){
    float a = bursts * bursts * bursts;
    float b = Mathf.Max(1f/100, Divisor / constDivisor);
    float c = AccuracyOffset / constDivisor;
    float d = MaxInAccuracy / constDivisor;
    float accuracy = a / b + c;
    if (accuracy > d){
        accuracy = d;
    }

    return accuracy * (1 + AccuracyFactor);
}
```

```
    Scale = tan(BaseFOV / 2) / tan(CurrentFOV / 2);
```

五、 射速补偿

游戏帧是离散的，帧时间越长，累计误差越大，解决方法是做射速补偿，上一个子弹多出来的时间，会减少下一发子弹的 CD