Finding the Best Ender Pearl Angle

Discord: Julia_bwah

October 1, 2023

Contents

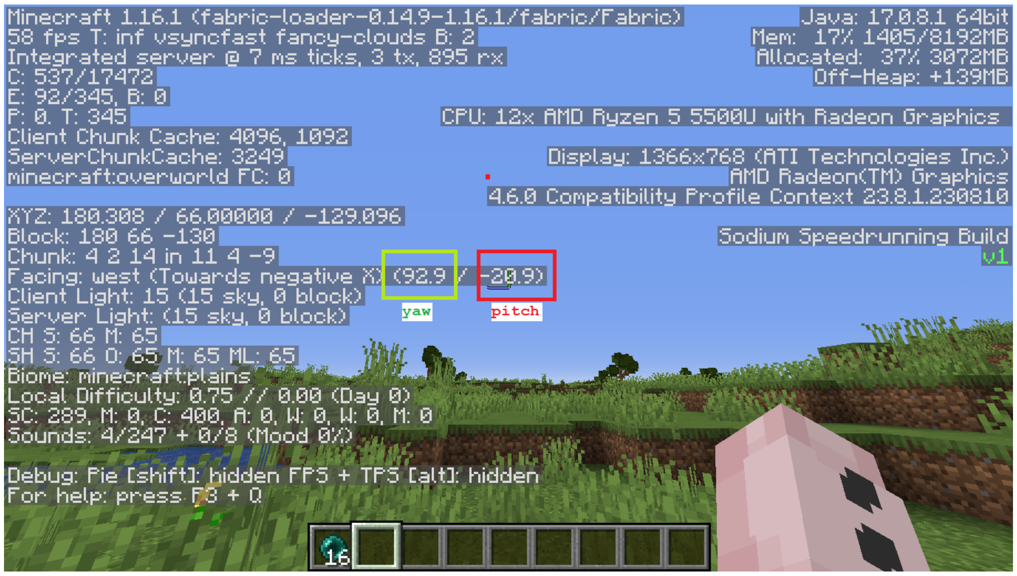
1 Introduction

Doing a Google search of the best angle to throw an ender pearl yields a wide range of results ranging from 35° to 40°. This uncertainty is due to the inherent randomness in a pearl's travel, which yields different distances even when the pearl is thrown at exactly the same angle.

It appears this uncertainty has remained since the pearl release in 2011, so we will calculate the exact optimal angle by analyzing the source code defining ender pearl travel.

2 Problem Setup

2.1 Pitch & Yaw



```
Minecraft 1.16.1 (fabric-loader-0.14.9-1.16.1/fabric/Fabric)
58 fps T: inf vsyncfast fancy-clouds B: 2
Integrated server @ 7 ms ticks, 3 tx, 895 rx
C: 537/17472
E: 92/345, B: 0
P: 0. T: 345
Client Chunk Cache: 4096, 1092
ServerChunkCache: 3249
minecraft:overworld FC: 0

XYZ: 180.308 / 66.00000 / -129.096
Block: 180 66 -130
Chunk: 4 2 14 in 11 4 -9
Facing: west (Towards negative X) (92.9 / -20.9)
Client Light: 15 (15 sky, 0 block)
Server Light: (15 sky, 0 block)
CH S: 66 M: 65
SH S: 66 O: 65 M: 65 ML: 65
Biome: minecraft:plains
Local Difficulty: 0.75 // 0.00 (Day 0)
SC: 289, M: 0, C: 400, A: 0, W: 0, W: 0, M: 0
Sounds: 4/247 + 0/8 (Mood 0%)

Debug: Pie [shift]: hidden FPS + TPS [alt]: hidden
For help: press F3 + Q
```

```
                              Java: 17.0.8.1 64bit
                           Mem:  17% 1405/8192MB
                          Allocated:  37% 3072MB
                                Off-Heap: +139MB

CPU: 12x AMD Ryzen 5 5500U with Radeon Graphics

      Display: 1366x768 (ATI Technologies Inc.)
                      AMD Radeon(TM) Graphics
4.6.0 Compatibility Profile Context 23.8.1.230810

                     Sodium Speedrunning Build
                                            v1
```

Figure 1: The F3 debug menu with the player's pitch and yaw

The *pitch* of a player is their line of sight with respect to the horizontal plane; it is the throwing angle of the pearl. The *yaw* of a player is their rotation about the vertical axis.

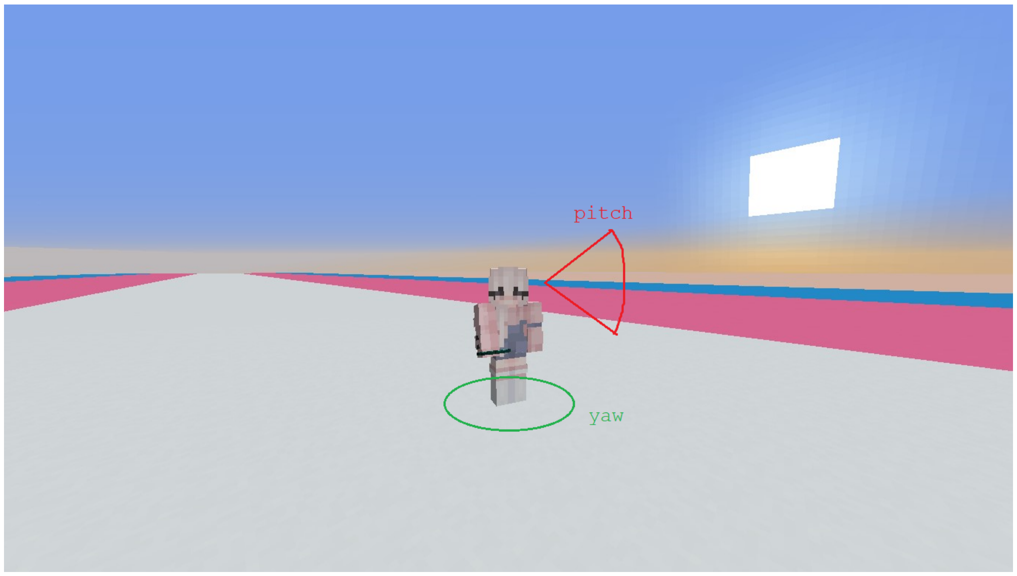
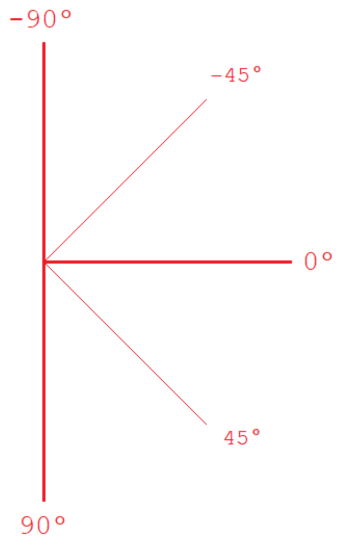
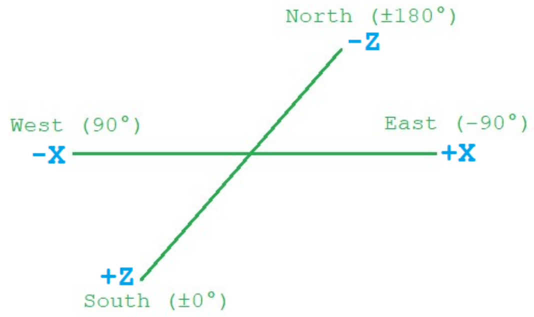


Figure 2: Pitch and Yaw Visualized





Pictured are common pitch and yaw values. Note that an upwards pitch corresponds to negative-valued angles.

Our calculations will assume a constant yaw of 0° (travelling completely south), so a pearl thrown at $(0, 0)$ would have its distance measured by simply reading the z-coordinate at landing. **Our goal** is to determine the pitch that achieves the furthest distance.

Below is the code that dictates the behavior of ender pearls from which we will base our analysis.

`https://imgur.com/a/FCSpqAH`

3 Velocity

3.1 Initial Velocity

Minecraft's smallest increment of time is a *tick* with 20 ticks in 1 real-time second. In-game events occur only on every tick and none in between.

Code A calls Code D to begin setting the initial velocity vector of the pearl. Code D receives the player's pitch θ_p and yaw θ_y and sets three constants, which are passed to Code E:

$$\begin{aligned} f &= -\sin(\theta_y)\cos(\theta_p) \\ f_1 &= -\sin(\theta_p) \\ f_2 &= \cos(\theta_y)\cos(\theta_p) \end{aligned}$$

Code D and E together compute the initial velocity vector: (where v_p is the player's velocity and $\mathcal{N}$ the Gaussian distribution)

$$v_{x0} = v_{px} + 1.5\left(\frac{f}{\sqrt{f^2 + f_1^2 + f_2^2}} + 0.0075 \cdot \mathcal{N}(0,1)\right)$$

$$v_{y0} = v_{py} + 1.5\left(\frac{f_1}{\sqrt{f^2 + f_1^2 + f_2^2}} + 0.0075 \cdot \mathcal{N}(0,1)\right)$$

$$v_{z0} = v_{pz} + 1.5\left(\frac{f_2}{\sqrt{f^2 + f_1^2 + f_2^2}} + 0.0075 \cdot \mathcal{N}(0,1)\right)$$

Note that the pearl's randomness originates here and only here, so this perturbation in initial velocity carries through the whole pearl's airtime, leading to different ending positions. These velocities could be simplified under our testing conditions:

1. The randomness $0.0075 \cdot \mathcal{N}(0,1)$ is independent of pitch and yaw and averages to 0, so it is safe to ignore.
2. The player velocity v_p is a constant, so it is safe to ignore.
3. Our assumption of $\theta_y = 0°$ implies $\sin(\theta_y) = 0$ and $\cos(\theta_y) = 1$, leading to:

$$\begin{aligned} f &= 0 \\ f_1 &= -\sin(\theta_p) \qquad \text{(Unchanged)} \\ f_2 &= \cos(\theta_p) \end{aligned}$$

Applying 1-3, we obtain a simplified velocity vector:

$$\begin{aligned} v_{x0} &= 0 \\ v_{y0} &= -1.5\sin(\theta_p) \\ v_{z0} &= 1.5\cos(\theta_p) \end{aligned} \tag{1}$$

3.2 Velocity at Tick t

The velocity for subsequent ticks after tick 0 is determined recursively by Code F. In line 91, the next tick's velocity is computed by multiplying the current velocity vector by 0.99 (provided the pearl isn't in water) and subtracting 0.03 from the y-coordinate. Letting $\theta = \theta_p$ for brevity, we have:

$$\begin{aligned} v_x(t,\theta) &= 0.99 \cdot v_x(t-1) \\ v_y(t,\theta) &= 0.99 \cdot v_y(t-1) - 0.03 \\ v_z(t,\theta) &= 0.99 \cdot v_z(t-1) \end{aligned}$$

To enable further computation, we will compute closed forms for these recursions. At a given tick t, there will have been t multiplications by 0.99 over ticks 1 to t. And down t levels in the recursion has the base case of the initial velocity. Hence, $v_x(t) = 0.99^t v_{x0} = 0$ and $v_z(t) = 0.99^t v_{z0}$.

The same applies to v_y, but more care needs to be given to the -0.03 term. In the ith tick before t, the -0.03 term is multiplied i times by 0.99 (e.g. consider $i = 0$ and $i = 1$). So from $i = 0$ to $i = t-1$, an additive value of $\sum_{i=0}^{t-1}(-0.03)0.99^i$ is accumulated. Using the formula for the sum of a finite geometric sequence, [1] this sum is $-0.03\left(\frac{1-0.99^t}{0.01}\right) = -3 + 3 \cdot 0.99^t$. Therefore, we have $v_y(t,\theta) = 0.99^t v_{y0} + (-3 + 3 \cdot 0.99^t) = (v_{y0} + 3) \cdot 0.99^t - 3$.

Putting it all together, we obtain the velocity equations:

$$\begin{aligned} v_x(t,\theta) &= 0 \\ v_y(t,\theta) &= (v_{y0} + 3) \cdot 0.99^t - 3 \\ v_z(t,\theta) &= v_{z0} \cdot 0.99^t \end{aligned} \tag{2}$$

[1]Sum of finite geometric sequence: $\sum_{k=0}^{n-1} ar^k = a\frac{1-r^n}{1-r}$ for $-1 < r < 1$.

4 Position Equations

The velocity functions enable us to find the position functions of a pearl. Since velocity is by definition meters travelled per tick, the distance travelled up to a tick t is the sum of velocities over all prior ticks. [2] Note that distance travelled and position here are equivalent as we assume a starting position of $(0,0)$. We write this algebraically for each axis below. We then again use the sum of finite geometric sequences.

$$\begin{aligned} s_y(t,\theta) &= \sum_{i=0}^{t-1}\big((v_{y0}+3)\cdot 0.99^i - 3\big) \\ &= (v_{y0}+3)(100-100\cdot 0.99^t) - 3t \end{aligned} \qquad \begin{aligned} s_z(t,\theta) &= \sum_{i=0}^{t-1} v_{z0}\cdot 0.99^i \\ &= v_{z0}(100-100\cdot 0.99^t) \end{aligned}$$

Substituting in formulas (1), we have our position equations:

$$\begin{aligned} s_y(t,\theta) &= (-1.5\sin(\theta)+3)(100-100\cdot 0.99^t) - 3t \\ s_z(t,\theta) &= 1.5\cos(\theta)(100-100\cdot 0.99^t) \end{aligned} \tag{3}$$

Rephrasing the question of best angle – our goal is to find the pitch that achieves the highest s_z.

5 Best Angle

To find the ending distance for an angle θ, we need to know the landing time T_θ. This is because $s_z(T_\theta,\theta)$ is precisely the ending distance of a pearl. Letting the ground level be g, then T_θ can be computed by solving $s_y(t,\theta) = g$ and rounding down to the previous tick. [3] We devise the following procedure: for each angle θ in $[-90, 90]$, solve $s_y(t,\theta) = g$ and plug its answer into s_z; take the angle that produced the maximum s_z as the best throwing angle.

However, doing this procedure for all angles in $[-90, 90]$ would be computationally infeasible if we want float-level precision. Fortunately, due to the way Minecraft does math, the range of angles is much more discrete, only taking on multiples of $\dfrac{360}{65536}^\circ \approx 0.0055^\circ$. We will call these *significant angles* as they are by the technical parkour community. [4]

Fact 5.1. The significant angles of our working pitch range $[-90, 90]$ are given by $\dfrac{360}{65536}k$ for $-16384 \leq k \leq 16384$ because $\dfrac{360}{65536}(\pm 16384) = \pm 90$.

Fact 5.2. The player's angle θ is stored as a float, and at time of throwing, the pearl adopts the highest significant angle lower than θ.

Below is our revised algorithm, iterating over all significant angles and computing the distance for each.

[2] Those familiar with calculus may know this as taking the definite integral of velocity up to $t-1$. This approach is allowed provided that time is floored to preserve the discreteness of ticks.

[3] More technically, the tick at which the pearl lands is the latest tick before which $s_y(t,\theta) < g$.

[4] `https://www.mcpk.wiki/wiki/Parkour_Nomenclature#Technical_parkour_terms`

Algorithm 1 Determine Best Pitch

```
for $k = -16384$ to $16384$:
    $\theta = \frac{360}{65536}k$
    Solve $s_y(t, \theta) = g$, and round down the result; call this result $T_\theta$
    Compute $s_z(T_\theta, \theta)$
Take the pitch with the highest $s_z$ as the best angle
```

This procedure was implemented in Python (`https://pastebin.com/33whTy1J`), yielding the following top 10 entries, ranked by distance. They were also tested against the actual distance obtained in-game (we will discuss the small discrepancies later).

#	Pitch (°)	Predicted Distance	Actual Distance
1	-37.9742431640625	52.89209945925462	52.890613835325986
2	-37.979736328125	52.88814101272806	52.88665432209301
3	-38.86962890625	52.885939185946924	52.884344437028545
4	-37.9852294921875	52.884182080065074	52.88269480886004
5	-38.8751220703125	52.88185210098631	52.8802550219269
6	-37.99072265625	52.880222661302035	52.77145306831007
7	-38.880615234375	52.87776452994706	52.87616560682527
8	-37.9962158203125	52.876262756475334	52.874775782394124
9	-38.8861083984375	52.873676472866755	52.872080240649446
10	-38.001708984375	52.87230236562137	52.87081226965284

Figure 3: Top 10 Pitch-Distance Combinations

Pitch (°)	Predicted Distance	Actual Distance
-38	52.876262756475334	52.874775782394124
-40	52.66673513971652	52.66499709116808
-45	51.427474832292816	51.42502374586834
-35	52.17929959463825	52.1781583024189

Figure 4: -38 Compared to Commonly Believed Optimal Angles

Our predictions have a small margin of error, but on average, it is only 0.012 blocks of the actual distances. It is currently unknown what causes this, but at least the sequence of errors is monotonic with the pitch.

We obtain that the best pitch to maximize distance is $-37.9742431640625°$. Since the F3 menu displays one's pitch to only 1 decimal place, $-38.0°$ would be the best angle in practical settings. However, the precision obtained in this angle provides a definitive resolution (and in my opinion, a very satisfying one) to the longstanding debate of the best pearl angle.

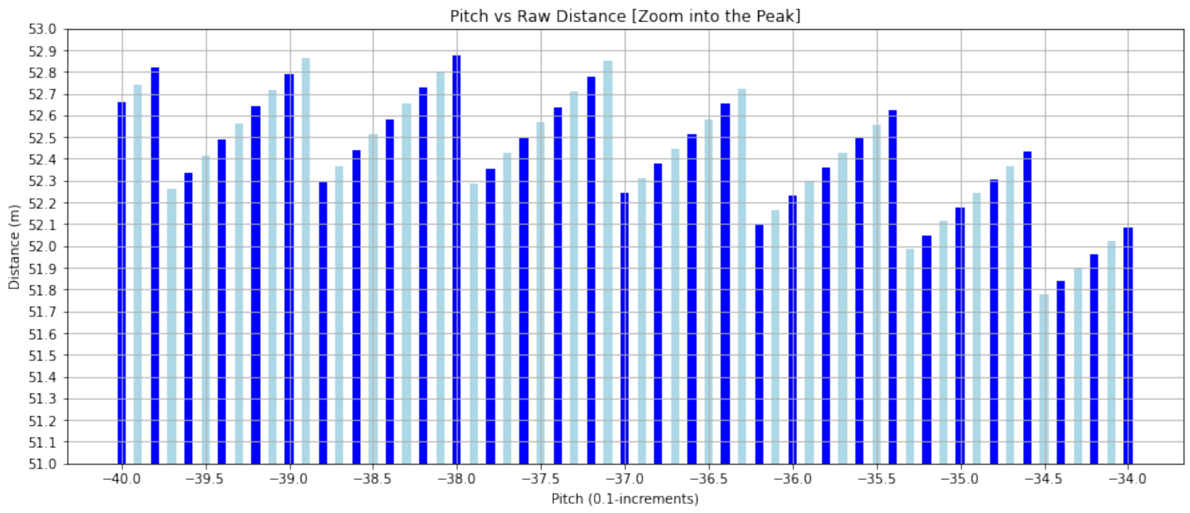


Figure 5: Graph of Distances for angles $[-40, -34]$. If curious about the oscillations, feel free to contact me.