**Understanding Roll, Pitch, & Yaw**

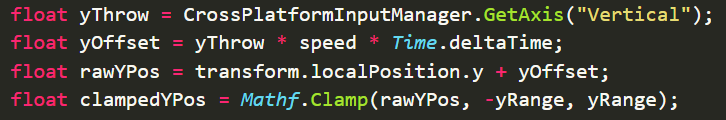
**Objective**: We're going to organize vertical movement of the ship. And we're going to start to think about where the ship is facing at any one time, ie the center of the screen, and what impact that has on our aiming. We'll also learn about the terms, yaw, pitch, and roll, and find out how we can use that in the future to fix this aiming issue.

**Challenge**: Move the ship vertically and clamp the vertical movement. The movement should feel natural.

1. Create a SerializeField for yRange find that sweet spot. For me it was 3.



1. Copy the **xThrow**, **xOfffset**, **rawXPos**, and **clampedXPos** variables and rework them for the y direction. While you’re at it rename **xSpeed** to **speed**.



1. Change the transform.localPosition to now use the clammpedYPos parameter since we don’t want the current Y position anymore.



1. Lets go to Unity and test it out!

**POSSIBLE BUG:** Now my ship kept going off screen. The reason was that my serial field Y range component in the inspector window as incorrectly set to 6.



1. Refactoring time! How do you think we should refactor our code since our Update method is now getting a little big? Let’s create a method called **ProcessTranslation**.

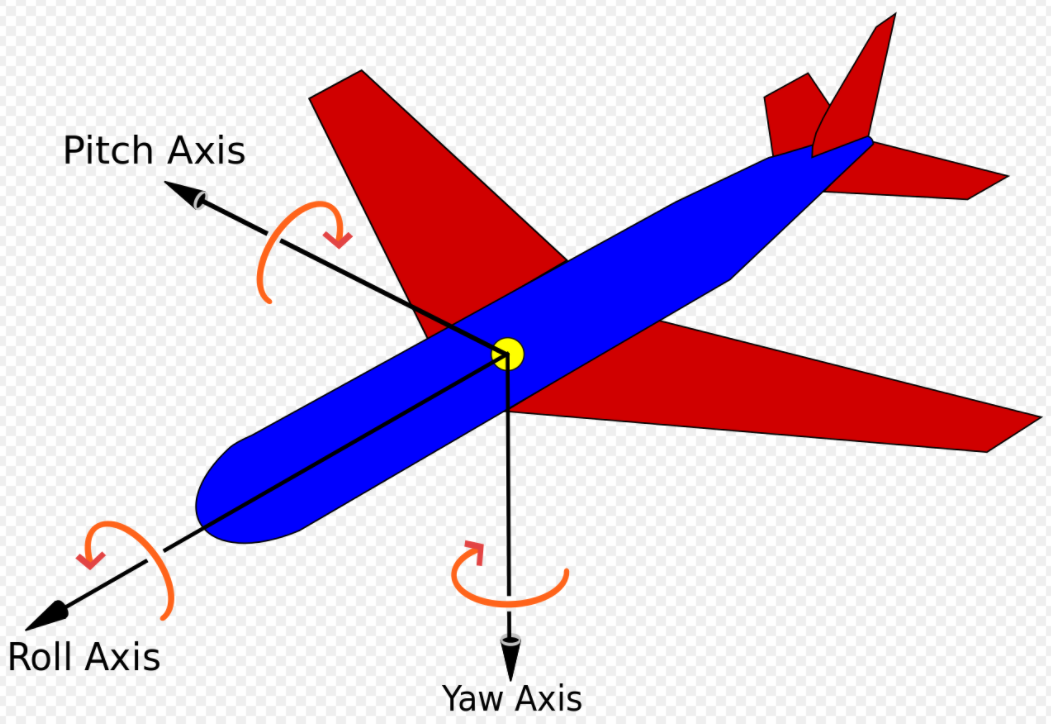


Now at the moment, the ship is constantly shooting, or aiming, towards the middle of the screen. Imagine we draw some lines, and think about now how the ship might aim, and how we might hit things on different parts of the screen.





In this paradigm, we're not going to be able to actually shoot at anything unless it's coming at us from the center of the screen, and that's not much fun. So what we're going to want to do, is to actually manipulate the rotation of the ship, both the pitch of the ship and the yaw of the ship.



In an airplane, the **roll** if you put your hand out in front of you with your middle finger facing forward, roll is where you tilt your hand left and right.

Now the **yaw** that's where your nose wiggles left or right.

And **pitching** is up and down, so when you take off a plane pitches up.

This is going to correspond to our Scene gizmo



So our **Z** is our **roll** axis, our **X** is our **pitch** axis, and our **Y** is our **yaw** axis.

But what we want to be able to do, is to programmatically say, when the ship's on the left-hand side of the screen, we want to do something with it. Now what are we going to do with it? Are we going to pitch it, yaw it, roll it? What we want to do is to **yaw it to the left slightly**, so that it's now aiming to the left of the screen.

And if the ship's really low, then we want to **pitch it actually slightly downwards** so that it's aiming more towards the bottom of the screen, you get the idea.