Odometry Simulation Noise + Camera Simulation

Reasoning

When we run robots on the field our wheel odometry will drift over time caused by a variety of sources (wheels slipping, impacts with other robots, model mismatch, sensor noise, ect). It is useful to model this imperfect system with noise in our simulation to test some of the systems that are designed to combat that like the vision system.

The vision simulation generates frame based on a pose given to it. Therefore we need to track both a "true" and "noisy" position of the robot. We can feed the "true" value into our vision system to generate frames and test our vision pipeline then combine it with the "noisy" pose from the wheel odometry.

Methodology Drive

We will be using a crude approximation of noise that is proportional to the square of wheel delta and normally distributed centered at zero (gaussian white noise). This will be added at every time step of the simulation. We will also keep track of SimNoNoise versions of the variables

```
double delta = (driveSim.getAngularVelocityRadPerSec() * Constants.LOOP_PERIODIC_SECS);
inputs.drivePositionSimNoNoise += delta;
inputs.drivePositionRad += delta + simNoise.nextGaussian(0.0, 4.0) * Math.pow(delta,2) * 0.1;
```

Methodology Turning

Instead of adding noise at every timestep we opted to instead pick a constant error representing a slightly incorrect calibration of the absolute position of the wheels (which makes the sim perform like real life). We choose to be uniformly distributed between +/- 4 degrees

```
private static final double moduleErrorBound = 8.0;
private double moduleOffsetError = Math.floor(simNoise.nextDouble() * moduleErrorBound) -
moduleErrorBound/2.0;
```

Drivetrain

Inside drivetrain we want to minimize the amount of code changing for simulation vs normal operation. Therefore we create duplicate odometry class that calls all the NoNoiseSim versions from the SwervePod. Then the SimNoNoiseOdom class is only instantiated if `Constants.getMode() == Mode.SIM`.

This means in simulation there are 3 logged Poses.

- Pose: pose estimator results using vision and wheels
- Odom: wheel only odometry pose **with** Noise
- OdomTrue: wheel only odometry without any noise

Add Vision Data

The vision simulation will use the OdomTrue value called from getPoseSimNoNoise() to generate a frame for the vision system.

Note: the vision system will also apply noise to the frame to mimic a real camera. Thus the pose coming out of the vision system will also have noise similar to if it was a real system

The drivetrain has a poseEstimator class that can accept vision data combining it with the wheel odometry. We must however set the covarience or our confidence in the vision measurments. We take a simple strategy saying our confidence is linked to how far away we are and how many tags we see.

Revision #1
Created 9 August 2023 03:37:33 by jheidegg
Updated 9 August 2023 03:56:19 by jheidegg