JacisNonsense / **Pathfinder**

Branch: master ▾ **Pathfinder** / **Pathfinder-Java** / Create new file | Find file | History

JacisNonsense Build for 2018 Latest commit `093ee88` 2 days ago

..		
examples	Include Documentation, READMEs and Examples	2 years ago
src	Build for 2018	2 days ago
README.md	Include Documentation, READMEs and Examples	2 years ago
build.gradle	Build for 2018	2 days ago
jni.gradle	Build for 2018	2 days ago

README.md

Pathfinder Java Wrapper

This is the Java Wrapper for the 'Pathfinder' motion profiling library.

Windows, Mac and RoboRIO native binaries are included with the jar. To compile for your own platform, see the master README.

Using the Library

Full examples are provided under `examples/`

Generating a Trajectory

```
// 3 Waypoints
Waypoint[] points = new Waypoint[] {
    new Waypoint(-4, -1, Pathfinder.d2r(-45)),      // Waypoint @ x=-4, y=-1, exit angle=-45 degrees
    new Waypoint(-2, -2, 0),                        // Waypoint @ x=-2, y=-2, exit angle=0 radians
    new Waypoint(0, 0, 0)                           // Waypoint @ x=0, y=0,   exit angle=0 radians
};

// Create the Trajectory Configuration
//
// Arguments:
// Fit Method:          HERMITE_CUBIC or HERMITE_QUINTIC
// Sample Count:        SAMPLES_HIGH (100 000)
//                      SAMPLES_LOW  (10 000)
//                      SAMPLES_FAST (1 000)
// Time Step:           0.05 Seconds
// Max Velocity:        1.7 m/s
// Max Acceleration:    2.0 m/s/s
// Max Jerk:            60.0 m/s/s/s
Trajectory.Config config = new Trajectory.Config(Trajectory.FitMethod.HERMITE_CUBIC, Trajectory.Config.SAMPLES_HIGH,

// Generate the trajectory
Trajectory trajectory = Pathfinder.generate(points, config);
```

Using the Trajectory

```
for (int i = 0; i < trajectory.length(); i++) {
    Trajectory.Segment seg = trajectory.get(i);

    System.out.printf("%f,%f,%f,%f,%f,%f,%f,%f\n",
        seg.dt, seg.x, seg.y, seg.position, seg.velocity,
            seg.acceleration, seg.jerk, seg.heading);
}
```

Modifying your Trajectory

Tank Drive

```
// The distance between the left and right sides of the wheelbase is 0.6m
double wheelbase_width = 0.6;

// Create the Modifier Object
TankModifier modifier = new TankModifier(trajectory);

// Generate the Left and Right trajectories using the original trajectory
// as the centre
modifier.modify(wheelbase_width);

Trajectory left  = modifier.getLeftTrajectory();       // Get the Left Side
Trajectory right = modifier.getRightTrajectory();      // Get the Right Side
```

Swerve Drive

```
// The distance between the left and right sides of the wheelbase is 0.6m
double wheelbase_width = 0.6;

// The distance between the front and back sides of the wheelbase is 0.5m
double wheelbase_depth = 0.5;

// The swerve mode to generate will be the 'default' mode, where the
// robot will constantly be facing forward and 'sliding' sideways to
// follow a curved path.
SwerveModifier.Mode mode = SwerveModifier.Mode.SWERVE_DEFAULT;

// Create the Modifier Object
SwerveModifier modifier = new SwerveModifier(trajectory);

// Generate the individual wheel trajectories using the original trajectory
// as the centre
modifier.modify(wheelbase_width, wheelbase_depth, mode);

Trajectory fl = modifier.getFrontLeftTrajectory();       // Get the Front Left wheel
Trajectory fr = modifier.getFrontRightTrajectory();      // Get the Front Right wheel
Trajectory bl = modifier.getBackLeftTrajectory();        // Get the Back Left wheel
Trajectory br = modifier.getBackRightTrajectory();       // Get the Back Right wheel
```