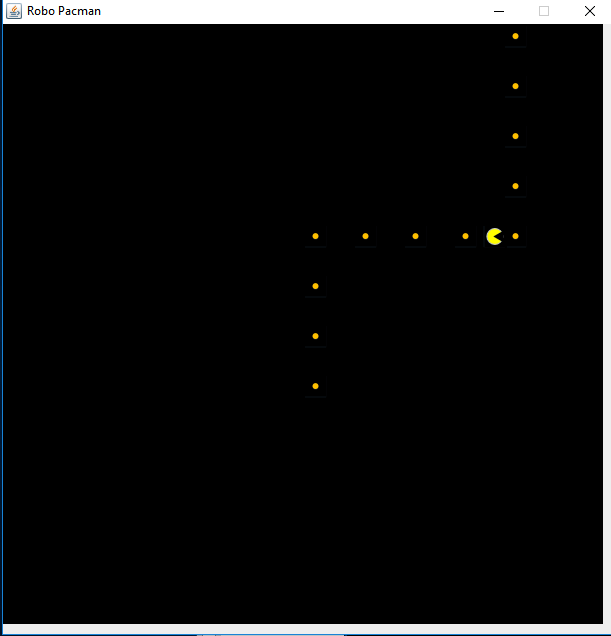
# Robo Pacman – DRAFT 1

Challenge: write an autonomous routine to guide Pacman around the field using a tank drive. Eat as many dots as possible within the time allowed.



Learn how a robot is broken down into many subsystems, starting with the drive train.

public class Robot extends RobotBase {

    // subsystems

    public static DriveTrain driveTrain;

    public void robotInit() {

        driveTrain = new DriveTrain();

    }

    public static void main(String[] args) {

        Robot robot = new Robot();

        RobotRunner.run(robot, new DriveFromAtoB());

    }

}

Learn how to break down a task into reusable commands.

public class DriveFromAtoB extends CommandGroupBase{

    public DriveFromAtoB() {

        addSequential(new DriveStraight(300));

        addSequential(new Turn(90));

        addSequential(new DriveStraight(200));

        addSequential(new Turn(0));

        addSequential(new DriveStraight(200));

    }

}

Learn how about the command pattern and the basic structure of a WPILib command and how it can be used to implement behavior.

Learn how to control movement using a tank drive.

public class DriveStraight extends CommandBase {

    private double targetDistance;

    private double startDistance;

    private boolean success = false;

    public DriveStraight(double distance) {

        // save the target distance for later

        this.targetDistance = distance;

        Robot.log("DriveStraight:targetDistance:"+targetDistance);

    }

        // Called just before this Command runs the first time

    protected void initialize() {

        // save the starting point for later

        startDistance = Robot.driveTrain.getDistance();

    }

    // Called repeatedly when this Command is scheduled to run

    protected void execute() {

        // calc distance traveled since this command started

        double currentDistance = Robot.driveTrain.getDistance() - startDistance;

        Robot.log("DriveStraight:currentDistance:"+currentDistance);

        // if we have gone far enough then stop

        if (currentDistance >= targetDistance) {

            Robot.driveTrain.tankDrive(0, 0);

            success = true;

        } else {

            Robot.driveTrain.tankDrive(1, 1);

        }

    }

    // Make this return true when this Command no longer needs to run execute()

    protected boolean isFinished() {

        return success;

    }

}

Learn how sensors can be used to improve the capabilities of a robot.

public class Turn extends CommandBase {

    private boolean success = false;

    private int targetAngle;

    public Turn(int angle) {

        // save the target angle for later

        this.targetAngle = angle;

        Robot.log("Turn:targetAngle:"+targetAngle);

    }

    // Called repeatedly when this Command is scheduled to run

    protected void execute() {

        int currentAngle = Robot.driveTrain.getAngle();

        Robot.log("Turn:currentAngle:"+currentAngle);

        // if we have turned far enough then stop

        if (currentAngle == targetAngle) {

            Robot.driveTrain.tankDrive(0, 0);

            success = true;

        } else {

            // compare the current to the target angle to see which way to turn

            if (currentAngle < targetAngle) {

                Robot.driveTrain.tankDrive(1, -1);

            } else {

                Robot.driveTrain.tankDrive(-1, 1);

            }

        }

    }

    protected boolean isFinished() {

        return success;

    }

}