Running commands during the autonomous period

Once commands are defined they can run in either the teleop or autonomous part of the program. In fact, the power of the command based programming approach is that you can reuse the same commands in either place. If the robot has a command that can shoot Frisbees during autonomous with camera aiming and accurate shooting, there is no reason not to use it to help the drivers during the teleop period of the game.

Creating a command to use for Autonomous

C++

```
        button1->WhenPressed(new PrepareToGrab());
        button2->WhenPressed(new Grab());
        button3->WhenPressed(new DriveToDistance(0.11));
        button4->WhenPressed(new PlaceSoda());
        button6->WhenPressed(new DriveToDistance(0.2));
        button8->WhenPressed(new Stow());

        button7->WhenPressed(new SodaDelivery());
```

Java

```
        public OI() {
                button1.whenPressed(new PrepareToGrab());
                button2.whenPressed(new Grab());
                button3.whenPressed(new DriveToDistance(0.11));
                button4.whenPressed(new PlaceSoda());
                button6.whenPressed(new DriveToDistance(0.2));
                button8.whenPressed(new Stow());

                button7.whenPressed(new SodaDelivery());
        }
```

Our robot must do the following tasks during the autonomous period: pick up a soda can off the floor then drive a set distance from a table and deliver the can there. The process consists of:

1. Prepare to grab (move elevator, wrist, and gripper into position)
2. Grab the soda can
3. Drive to a distance from the table indicated by an ultrasonic rangefinder
4. Place the soda
5. Back off to a distance from the rangefinder
6. Re-stow the gripper

To do these tasks there are 6 command groups that are executed sequentially as shown in this example.

Setting that command to run as the autonomous behavior

C++

```
Command* autonomousCommand;

class Robot: public IterativeRobot {

    /**
     * This function is run when the robot is first started up and should be
     * used for any initialization code.
     */
        void RobotInit()
        {
        // instantiate the command used for the autonomous period
                autonomousCommand = new SodaDelivery();
                oi = new OI();

        }
```

Running commands during the autonomous period

```
        void AutonomousInit()
        {
        // schedule the autonomous command (example)
                if(autonomousCommand != NULL) autonomousCommand->Start();
        }
        /*
         * This function is called periodically during autonomous
         */
        void AutonomousPeriodic()
        {
                Scheduler::GetInstance()->Run();
        }
```

Java

```
public class Robot extends IterativeRobot {



    Command autonomousCommand;

    /**
     * This function is run when the robot is first started up and should be
     * used for any initialization code.
     */
    public void robotInit() {
                oi = new OI();
        // instantiate the command used for the autonomous period
        autonomousCommand = new SodaDelivery();
    }

    public void autonomousInit() {
        // schedule the autonomous command (example)
        if (autonomousCommand != null) autonomousCommand.start();
    }

    /**
```

Running commands during the autonomous period

```
     * This function is called periodically during autonomous
     */
    public void autonomousPeriodic() {
        Scheduler.getInstance().run();
    }
```

To get the SodaDelivery command to run as the Autonomous program,

1. Instantiate it in the robotInit() method
2. Start it during the autonomousInit() method
3. Be sure the scheduler is called repeatedly during the autonomousPeriodic() method.

RobotInit() is called only once when the robot starts so it is a good time to create the command instance. AutonomousPeriodic() is called once at the start of the autonomous period so we schedule the command there. AutonomousPeriodic() is called every 20ms so that is a good time to run the scheduler which makes a pass through all the currently scheduled commands.