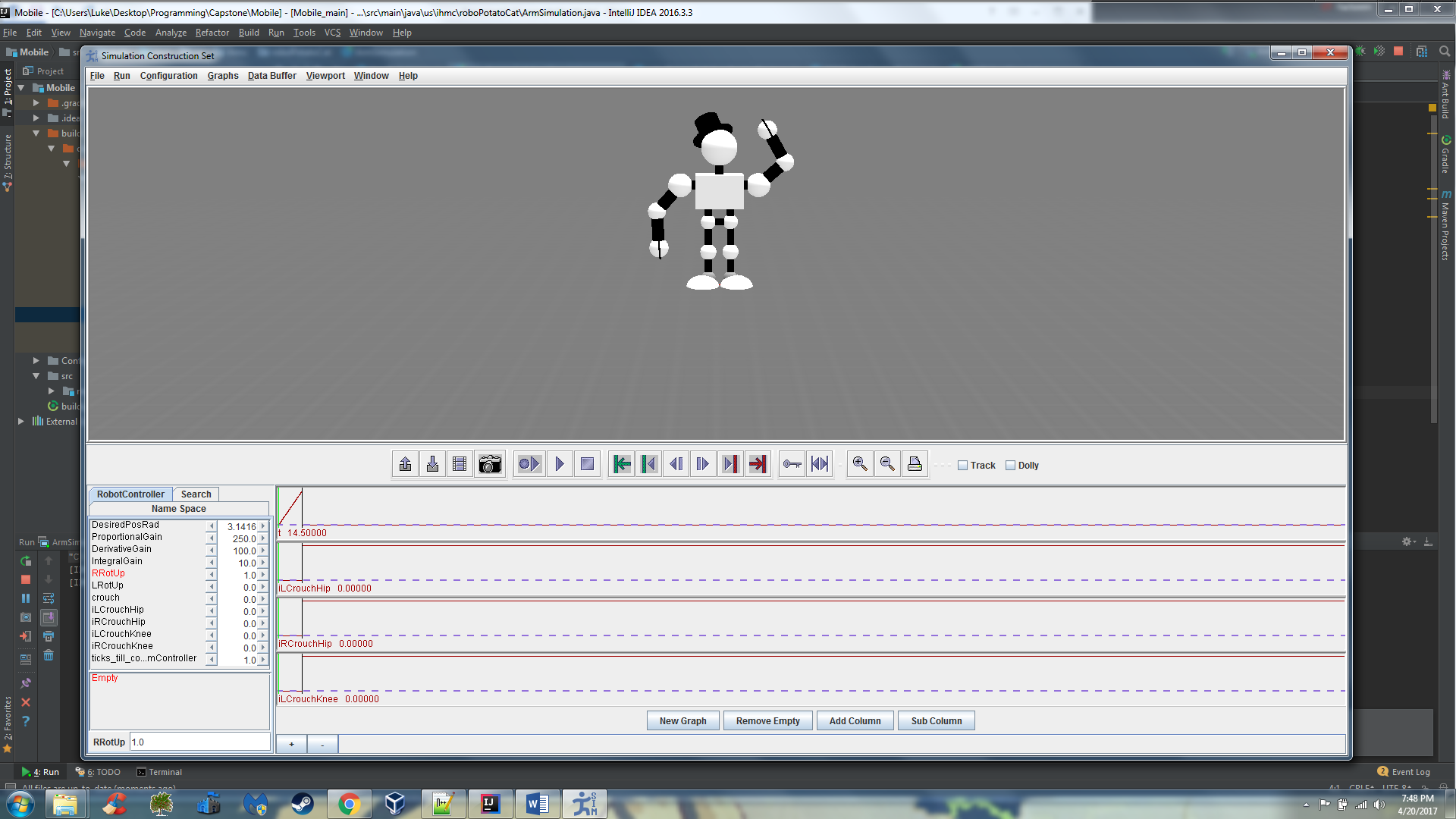
Super Amazing Awesome Tutorial For JDBot



-So you’ve done the other tutorials from IHMC, you think you know what you’re doing at this point, right?

No, of course you don’t. We sure didn’t. That’s why we’re here to help you understand what’s actually going on with this code. Hopefully by the end you will understand SCS well enough that you can create your very own robot... simulation!

Tips Before Getting Started

-This tutorial is going to assume that you have installed SCS and have at least the pendulum tutorial done. Those can be found here:

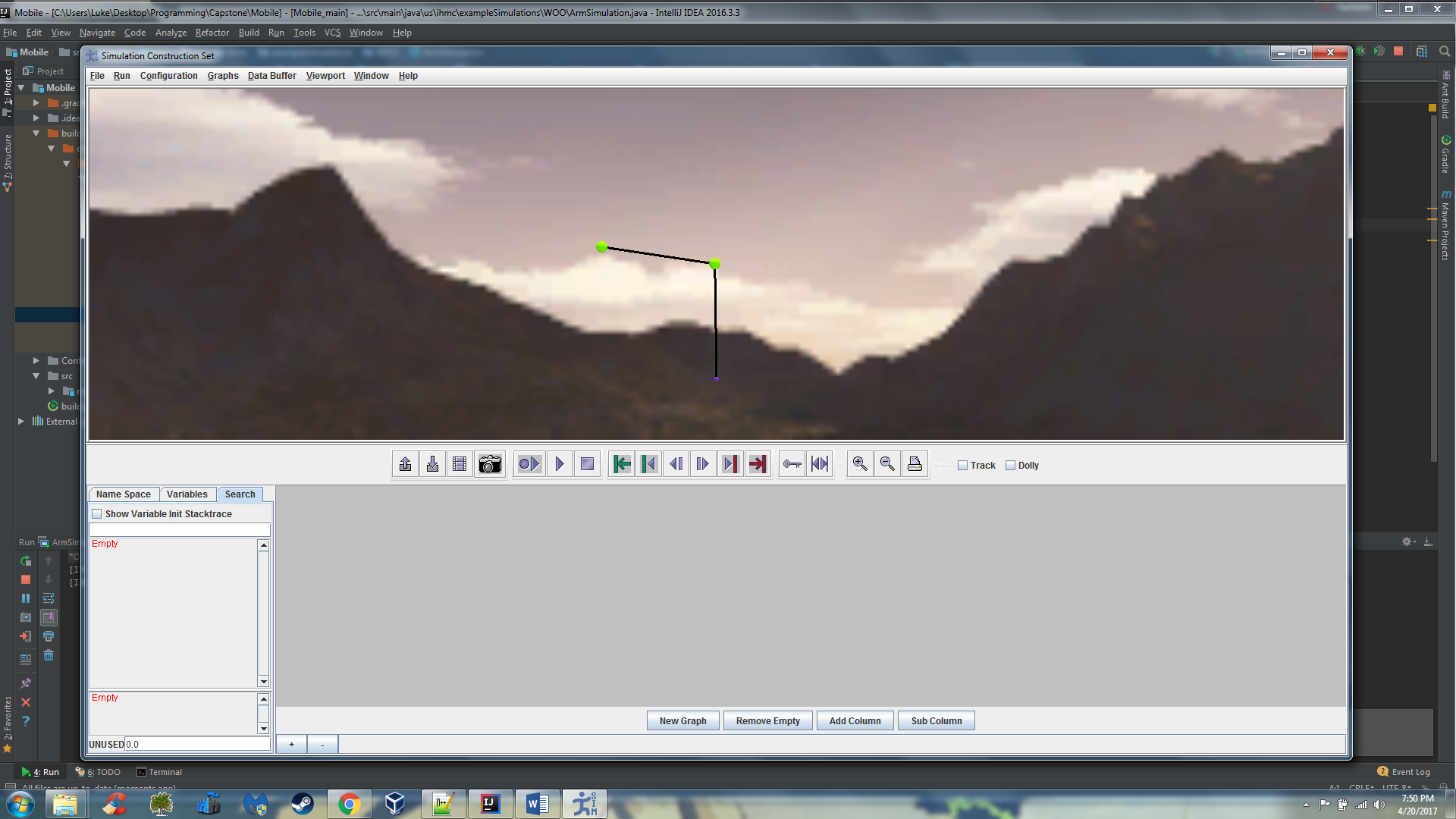
[Setting up SCS](https://ihmcrobotics.github.io/documentation/00-quickstart/00-quickstart/)

[Creating a pendulum](https://ihmcrobotics.github.io/documentation/20-scs/00-tutorials/02-creating-a-new-simulation/)

[Giving the pendulum a control](https://ihmcrobotics.github.io/documentation/20-scs/00-tutorials/03-adding-control-to-a-simulation/)

It’s also worth mentioning we created this project using IntelliJ. We like it better than Eclipse (we couldn’t figure out how to get SCS to work in Eclipse).

Creating WOOWOOBot



-WOOWOOBot will show you how to take a basic pendulum and add a second link to it, effectively creating a two-jointed pendulum. It will also show you some basic control over the individual joints.

If you have the basic pendulum already working with a control, you can use that as a base. First, right click exampleSimulations > New > Package. Name this package WOO. Now right click WOO > New > Java Class. Name this class ArmSimulation.

Now, the worst thing we did when we were learning SCS was simply copy/paste code from the tutorials without really understanding what they were doing in a desperate attempt to make things work. We didn’t fully understand how everything worked until the IHMC guys themselves explained it to us. With that in mind, we are giving you the full code to each of these classes. Please, take the time to learn about each piece of code as we go over it. You will still probably get stuck when making your own robots, but this will hopefully put you in a better place than just taking code, dropping it in, and expecting it to work (it won’t).

So here is the code for ArmSimulation:

package us.ihmc.exampleSimulations.WOO;  
  
import us.ihmc.simulationconstructionset.SimulationConstructionSet;  
import us.ihmc.simulationconstructionset.SimulationConstructionSetParameters;  
  
  
public class ArmSimulation  
{  
  
 public static final double *DT* = 0.001;  
 private SimulationConstructionSet sim;  
  
 public ArmSimulation()  
 {  
 ArmRobot robot = new ArmRobot();  
 robot.setController(new ArmController(robot));  
  
 SimulationConstructionSetParameters parameters = new SimulationConstructionSetParameters();  
 parameters.setDataBufferSize(32000);  
  
 sim = new SimulationConstructionSet(robot, parameters);  
 sim.setDT(*DT*, 20);  
 sim.setGroundVisible(false);  
 sim.setCameraPosition(0, -9.0, 0.6);  
 sim.setCameraFix(0.0, 0.0, 0.70);  
  
 sim.setSimulateDuration(60.0); // sets the simulation duration to 60 seconds  
  
 Thread myThread = new Thread(sim);  
 myThread.start();  
 }  
  
 public static void main(String[] args)  
 {  
 new ArmSimulation();  
 }  
}

This class sets up the environment for our robot. It determines things like the time between simulated frames

public static final double *DT* = 0.001;

which is currently set to 1 millisecond (0.001).

It will instantiate the other two classes we will be making.

ArmRobot robot = new ArmRobot();  
robot.setController(new ArmController(robot));

Don’t worry, the red lines are normal right now.