Programming Exercises 3.1, 3.2 and 3.4

Guðmundur Geir Gunnarsson

University of Southern Denmark

gunu@mmmi.sdu.dk

September 19, 2017

Overview

Programming Exercise 3.3

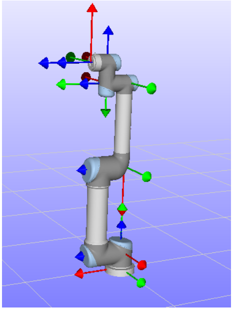
RobWork math and workcells

Programming Exercises 3.1 and 3.2

Programming Exercises 3.4

Programming Exercise 3.3

- Solution is on blackboard



RobWork Math

- RobWork includes types for all the transformations used in this course
- Take a look at the HelloRobwork program to see usage of the various transformations
- There is also a `rw::math::Rotation3D<>` class
- Take a look at `http://www.robwork.dk/apidoc/nightly/rw/namespacerw_1_1math.html` to see what other types there are

Loading a workcell with RobWork

- Workcells can be loaded into C++ using RobWork
- See the pages for `rw::models::WorkCell` and `rw::models::Device` on `www.robwork.dk`

```cpp
const string workcell_path = "/path/to/workcell/Scene.wc.xml";
const string device_name = "device_name";
WorkCell::Ptr wc = WorkCellLoader::Factory::load(workcell_path);
Device::Ptr device = wc->findDevice(device_name);
```

Programming Exercises 3.1 and 3.2

- **Programming Exercise 3.1**
  - Create rotation matricies
  - Ignore the part about fixed axes rotations. We are only looking at Euler rotations
- **Programming Exercise 3.2**
  - Pay special attention to the singularities in the RPY representation
- Compare your functions to the RobWork builtins
- Strongly recommended to use C++ for these exercises

Programming Exercise 3.4

- Program a function to calculate the forward dynamics
- `rw::math::Transform3D<>` can be used to represent the transformations $\mathbf{T}$
- `rw::math::Q` can be used for the state vector $\mathbf{q}$
- Compare your solution to the workcell from Programming Exercise 3.3