**ROS – Roboterlabor – Tic Tac Toe Roboter**

**Bisherige Arbeit zum Einrichten des Baxter Roboters in RViz:**

Etapen Konfiguration Roboter:

- MoveIt Tutorial zum Integrieren vom Baxter Roboter:

<https://github.com/RethinkRobotics/sdk-docs/wiki/MoveIt-Tutorial>

- Planning Groups:

* **baxter\_both\_arms**
  + **baxter\_left\_arm**
  + **baxter\_right\_arm**
* **baxter\_left\_arm**
  + left\_s0
  + left\_s1
  + left\_e0
  + left\_e1
  + left\_w0
  + left\_w1
  + left\_w2
  + left\_hand
  + left\_endpoint
* **baxter\_right\_arm**
  + right\_s0
  + right\_s1
  + right\_e0
  + right\_e1
  + right\_w0
  + right\_w1
  + right\_w2
  + right\_hand
  + right\_endpoint
* **baxter\_left\_hand**
  + left\_gripper
  + l\_gripper\_l\_finger
  + l\_gripper\_l\_finger\_tip
  + l\_gripper\_r\_finger
  + l\_gripper\_r\_finger\_tip
* **baxter\_right\_hand**
  + right\_gripper
  + r\_gripper\_l\_finger
  + r\_gripper\_l\_finger\_tip
  + r\_gripper\_r\_finger
  + r\_gripper\_r\_finger\_tip

- End Effectors:

* right\_hand
* left\_hand

- ROS Control: Add Controller

* both\_arms\_position\_controller
* right\_arm\_position\_controller
* left\_arm\_position\_controller
* right\_hand\_position\_controller
* left\_hand\_position\_controller

- MoveIt nicht erneut installiert

- command: ./baxter.sh à Datei nicht gefunden

- catkin\_make ok

- Anderes Tutorial:

<https://github.com/RethinkRobotics/sdk-docs/wiki/Installing-the-Research-SDK>

à vor allem Punkt 3 durchgeführt:

$ wstool init .

$ wstool merge https://raw.github.com/RethinkRobotics/baxter/master/baxter\_sdk.rosinstall

$ wstool update

à hier ist die urdf-Datei unter folgendem Pfad mitenthalten:

urdf Datei: src/baxter\_common/baxter\_descprition/urdf

- über den MoveIt Setup Assistenten: eigenes „package“ für den Baxter Roboter erstellt

à wird alles Wichtige wie planning groups und Endeffektoren definiert

- dieser wird dann in eigenem neuem Workspace integriert:

$ mkdir -p tictactoe\_ws/src

$ cd tictactoe\_ws

$ catkin\_make

$ source devel/setup.bash

$ cd src/

$ catkin\_create\_pkg baxter\_robot std\_msgs rospy roscpp

$ cd ..

$ catkin\_make

Dann kann das “package”, das mit dem MoveIt Setup Assistenten erstellt worden ist, im Catkin Package reinkopiert werden: in den Ordner src/baxter\_robot reinkopieren, dabei kann der gesamte Inhalt davor rausgelöscht werden.

Anschließend lässt sich z.B. das MoveIt RViz Plugin aufrufen über:

roslaunch baxter\_robot demo.launch

à Der Roboter startet in RViz und kann auf eine Position angefahren werden. Entweder direkt in RViz mit dem „MotionPlanning“ Tool oder mit dem Commander Scripting (siehe unten)

- MoveIt Bewegungstest mit Commander Scripting:

<http://docs.ros.org/melodic/api/moveit_tutorials/html/doc/moveit_commander_scripting/moveit_commander_scripting_tutorial.html>

rosrun moveit\_commander moveit\_commander\_cmdline.py

**Aktueller Stand 20.05.2020**

- Der Baxter Roboter ist in RViz integriert und kann von dort aus oder mit dem Commander Scripting Terminal auf eine Position gebracht werden. Hierbei scheinen alle Gelenke zu funktionieren (auch Endeffektoren bei den Händen des Roboters). Mit dem Commander Scripting Terminal lässt sich das gut testen.

- Bis jetzt aber keine genaue Vorstellung wie die Bewegungen über einen C++-Skript gesteuert werden sollen. Als Idee: siehe Link unten

- zum Testen Ordner „tictactoe\_ws“ herunterladen

à cd tictactoe\_ws

à catkin\_make

à source devel/setup.bash

à roslaunch baxter\_robot demo.launch

**Wie geht es weiter?**

- Entscheidung treffen, ob es sinnvoll ist mit dem Baxter Roboter weiterzumachen, oder wir doch nicht auf den Panda erstmal wechseln.

Von meiner Seite aus hätte ich doch nichts dagegen auf den Panda zu wechseln, weil man vielleicht leicht in die Thematik reinkommt, besonders was das Programmieren angeht. Da ich mir jetzt den Aufwand besser vorstellen kann, mit einem neuen/anderen Roboter anzufangen, wäre ich doch für den Panda. Der Baxter Roboter ist auch viel komplexer z.B. was die Anzahl an Gelenken und an integrierten Sachen im Modell angeht…

à Bis Ende der Woche kann das gerne noch jemand ausprobieren mit dem Baxter weiterzukommen, wenn wir aber bis dahin kein erstes C++ Programm haben, was z.B. nur ein Gelenk ansteuert oder den Roboter auf eine bestimmte Position fährt, würde ich definitiv damit aufhören. Schließlich haben wir dann doch auch nur ein bisschen weniger als zwei Wochen übrig…

- MoveIt Tutorials:

\* <http://docs.ros.org/melodic/api/moveit_tutorials/html/doc/move_group_interface/move_group_interface_tutorial.html>

\* Pick and Place: <https://ros-planning.github.io/moveit_tutorials/doc/pick_place/pick_place_tutorial.html>

**Aktueller Stand 24.05.2020**

- Erstellen eines neuen Workspaces „panda\_tictactoe\_ws“:

à panda\_moveit\_config

à moveit\_tutorials

à package „panda\_control“:

Datei „move\_group\_interface\_tutorial.cpp“ wird übernommen und verändert.

Datei „pick\_place\_tutorials.cpp“ wird auch übernommen

- Änderung in CMakeList:

\* Einbinden von packages und EIGEN3 damit Referenz zum Roboter-Model erstellt wird

\* Hinzufügen der zwei Zeilen, um die cpp-Dateien als ausführbare Dateien zu deklarieren

- Launch file erstellen für die neue cpp-Dateien

- Die Datei „pick\_place\_tutorials.cpp“ wurde angepasst als Datei „pick\_place\_panda.cpp“:

Darin werden 3 Würfel, die später die Spielsteine seien könnten, von einer Startposition bewegt und auf dem Spielfeld platziert. Hier wurden die Objekte mit C++-Code in RViz erstellt.

- Ausführen des C++-Programms:

* Runterladen vom Workspace „panda\_tictactoe\_ws”
* cd panda\_tictactoe\_ws
* source devel/setup.bash
* roslaunch panda\_moveit\_config demo.launch

In einem anderen Terminal im selben Workspace nach sourcen:

* roslaunch panda\_control pick\_place.launch

- zum Mitnehmen von diesem Zwischenstand/-code:

* wie bewegt man den Roboter von einem Punkt zum anderen
* wie greift der Roboter ein Objekt
* wie wird der Effektor des Roboters gedreht und richtig orientiert (siehe Benutzung der Funktion „.setRPY“ (roll: Y-Achse, pitch: X-Achse, yaw: Z-Achse))

- zu verbessern bzw. nächste Etappen:

* wie kann man eine Verbindung zu Elementen aus einer Gazebo World aufbauen?  
  à Ist es einfacher die Objekte (Spielsteine und Spielfeld) in Gazebo aufzubauen, ansonsten könnte es auch in RViz erstellt werden? (Ich denke, dass es einfacher in Gazebo ist…)
* danach kann das Programmieren des Spiels angefangen werden (Aufteilung der Funktionen, große Etappen / Schritten, die zu programmieren sind)
* man muss gucken, wie man mit dem Benutzer kommunizieren kann, um z.B. eine Abfrage zu machen, wo er seine nächste Spielfigur positionieren will.

**Aktueller Stand 30.05.2020**

- Node “move\_robot”, “compute\_pose” komplett funktionsfähig

- Node “eval\_game”:

* Überprüft, ob sich ein Spielstein bereits auf dem ausgewählten Spielfeld befindet oder nicht
* Speichert die Position mit Spielernummer aller platzierten Steine

--> macht aber noch keine Auswertung des Spielstands

- funktionierender Workspace unter “panda\_tictactoe\_ws” hochgeladen

- zum Testen:

* Herunterladen und Entpacken
* cd panda\_tictactoe\_ws
* source devel/setup.bash
* roslaunch panda\_moveit\_config demo.launch

In einem anderen Terminal im selben Workspace (nach sourcen):

* roslaunch panda\_control pick\_place.launch

In einem anderen Terminal im selben Workspace (nach sourcen):

* Folgender Befehl ausführen:

rostopic pub /next\_move std\_msgsnt8MultiArray "layout:

dim:

- label: 'test'

size: 2

stride: 1

data\_offset: 0

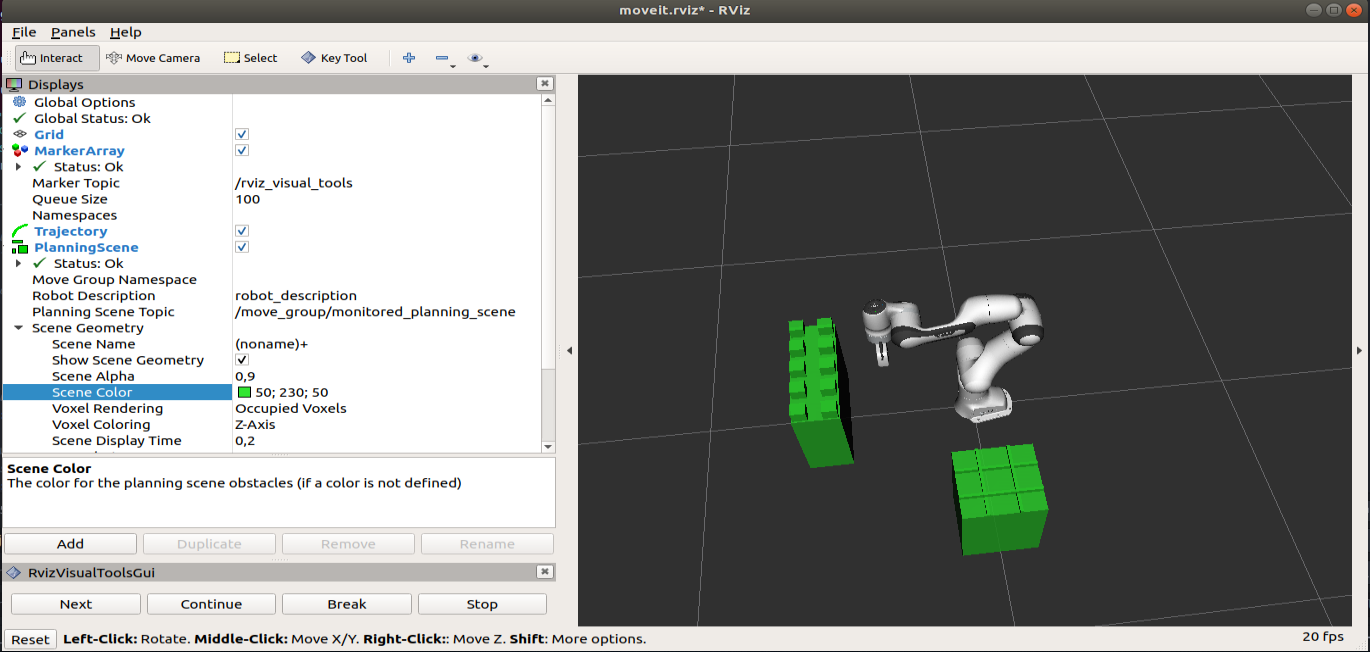
data: [8,1]"

Hier wird auf dem Topic “next\_move” eine Nachricht gepublisht, die einen Int8-Array enthält, in dem die Spielfeldnummer und die Spielernummer gespeichert sind.

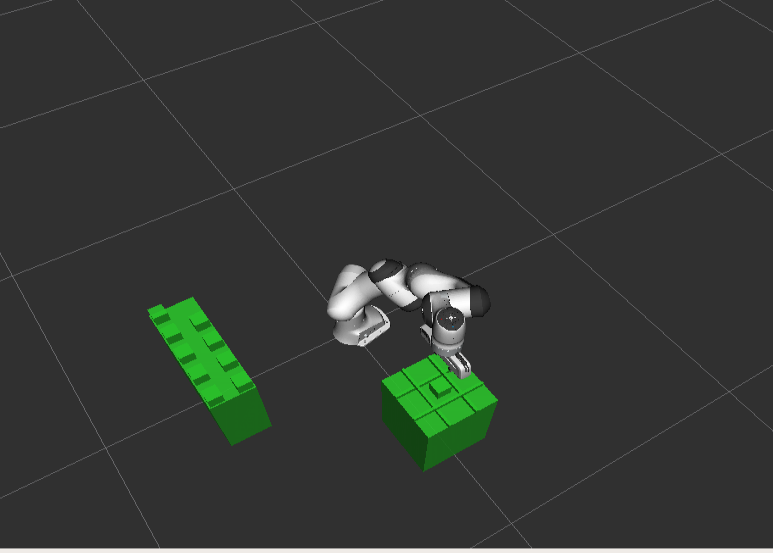
Diese Nachricht geht zur Node “eval\_game” und die Schleife fängt von dort an Richtung “move\_robot”, was die Spielsteine bewegt, wenn das ausgewählte Spielfeld frei ist.

**Aktueller Stand 01.06.2020**

Spielfeld auf Table\_2 ist implementiert worden und wird auch angezeigt.



Beispiel:



Zu bearbeiten sind die Spielsteine, da sie zurzeit noch für beide Spieler identisch sind.

Dabei können folgende Möglichkeiten umgesetzt werden:

1. Unterschiedliche Spielsteine
   * -> Neue Objekte erstellen -> planning\_scene und monitored\_planing\_scene Topic
2. Bild oder Spielnummer auf der Oberfläche der Würfel anzeigen. Dazu kann ein Topic : Spieler 1 oder 2 verwendet werden
   * -> Neue Topic Spieler