# Introduction to Interactive Markers

Wyatt Newman
September, 2014

A useful ROS tool that interacts with Rviz is “interactive markers.” (See: http://wiki.ros.org/rviz/Tutorials/Interactive%20Markers%3A%20Getting%20Started ). These notes provide some additional description and a simple example.

Within the class code, there is a package: example_interactive_marker, which contains source code IM_example.cpp. This package depends on the package “interactive_markers.” The code itself is contains detailed comments, but the following is an overview.

An instance of an “interactiveMarkerServer” is created with the name “example_marker.” The node name is “simple_marker.” When this node is launched, it will appear under “rosnode list” as “simple_marker.” However, messages to/from this node will be under the name “example_marker”. (The same name could be used for both; using different names was merely for illustration).

The interactive marker server responds to “controls” that appear in Rviz (when the interactive marker is “added” to the visualization. Interacting with these controls causes the marker to translate or rotate. When the marker is moved, the updated pose is published on the topic “/example_marker/feedback”, which uses messages of type: visualization_messages/InteractiveMarkerFeedback. This message can be examined with the command:

```
rosmsg show visualization_messages/InteractiveMarkerFeedback
```

This message type includes a pose field, detailing the position of the origin and the orientation of the frame. It should be noted that the interactive-marker feedback messages are only published when the marker is moved; the marker pose is not continuously refreshed.

The example source code defines three markers of type ARROW, colored red, green and blue, respectively, for displaying x, y and z axes of the (conglomerate) marker pose. The arrows are scaled down to a more convenient size for indicating axes. The length of each arrow is set by specifying endpoints.

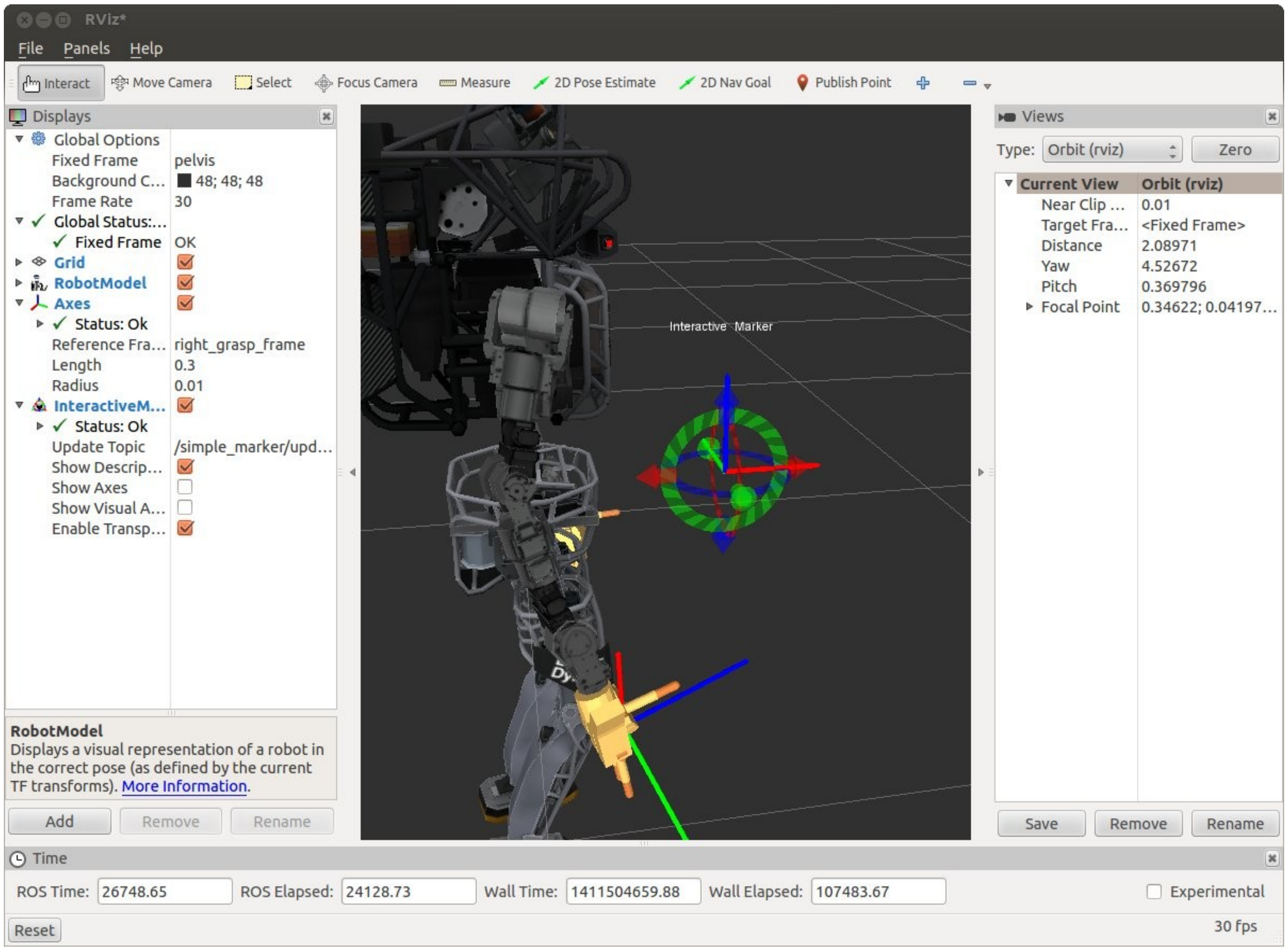
Six “controls” are created, of type visualization_msgs::InteractiveMarkerControl. These controls separately describe arrows and rings that can be manipulated via rviz. They are specified as three translation (x,y and z) controls and 3 rotation controls.

The default size of these controls in rviz is too large, and thus these are scaled down to 20% with the setting:     int_marker.scale = 0.2;

By default, the interactive marker will come up at the origin of the pelvis frame, which is inconvenient. Therefore, a starting position is defined, using the lines: int_marker.pose.position.x = 0.4;, etc.

Once the code is compiled and run, the terminal from which it is run will say “going into spin...”, and then will be silent, until the marker is moved via rviz. When the marker is moved, its updated coordinates are printed out by the callback function “processFeedback()”.

To operate the interactive marker, it must be added to the rviz visualization. To do this, click “add” and select InteractiveMarkers. A new menu item will appear in the rviz "displays" panel on the left side. The item "Update Topic" needs to be informed of the topic to which the interactive-marker node publishes ("example_marker/update"). If the interactive-marker node is running, this will show up as an option on the drop-down menu next to "Update Topic." Selecting this topic will result in a display such as that below:



| | |
|---|---|
| ▾ Global Options | |
| Fixed Frame | pelvis |
| Background C... | ■ 48; 48; 48 |
| Frame Rate | 30 |
| ▾ ✓ Global Status:... | |
| ✓ Fixed Frame | OK |
| ▸ Grid | ✓ |
| ▸ RobotModel | ✓ |
| ▾ Axes | ✓ |
| ▸ ✓ Status: Ok | |
| Reference Fra... | right_grasp_frame |
| Length | 0.3 |
| Radius | 0.01 |
| ▾ InteractiveM... | ✓ |
| ▸ ✓ Status: Ok | |
| Update Topic | /simple_marker/upd... |
| Show Descrip... | ✓ |
| Show Axes | ☐ |
| Show Visual A... | ☐ |
| Enable Transp... | ✓ |

**RobotModel**
Displays a visual representation of a robot in the correct pose (as defined by the current TF transforms). More Information.

Type: Orbit (rviz)

| ▾ Current View | Orbit (rviz) |
|---|---|
| Near Clip ... | 0.01 |
| Target Fra... | <Fixed Frame> |
| Distance | 2.08971 |
| Yaw | 4.52672 |
| Pitch | 0.369796 |
| ▸ Focal Point | 0.34622; 0.04197... |

ROS Time: 26748.65 ROS Elapsed: 24128.73 Wall Time: 1411504659.88 Wall Elapsed: 107483.67 ☐ Experimental

30 fps

The marker can then be moved interactively with a mouse, separately along x,y,z axis, or separately as rotations about the x, y or z axes.

Additional nodes can be composed that subscribe to the topic /example_marker/feedback, e.g. by emulating the callback function "processFeedback()" in the example code IM_example.cpp. In this way, interactive markers can be used to help define motion commands or help with perceptual interpretation.

**Additional Marker Examples:**
In the same package, example_interactive_marker, the program "marker_example.cpp" shows how to create a "list" of markers that can be displayed together. The type "SPHERE_LIST" is used. An object of type visualization_msgs::Marker is populated with a set of spheres in a regular grid on a single plane. Size, color and locations of the spheres are specified within the program. Sphere coordinates are interpreted with respect to the "pelvis" frame, as specified by the assignment:

```
marker.header.frame_id = "/pelvis";
```

The resulting sphere list is published to the topic "sphere_list_marker." To see the result, start up gazebo/drcsim, as well as rviz (rosrun rviz rviz). Then run the example program:

```
 rosrun example_interactive_marker marker_example
```

In Rviz, in the "displays" panel, click "Add", and choose "Marker". A new Marker item will appear in the Displays panel. Expand this entry. Next to "Marker Topic", type in "sphere_list_topic." You will see small, red spheres get added to the display, 0.5 m in front of the pelvis frame. The markers will persist in the display. However, once the marker_example node stops publishing (and concludes), the markers can be cleared by unchecking the box next to the "Marker" item that was added to rviz. If this Marker is re-checked, the display will be empty (until you re-run marker_example).

The marker list can be populated and published quite rapidly. The delay between addition of sphere has been inserted deliberately for illustration.

A variation on the marker_example is "marker_listener_v2.cpp." This node is a type of "broker." It receives messages on the topic "marker_listener" (of type geometry_msgs::PointStamped) and republishes the incoming points to the topic "marker_publisher" (of type visualization_msgs::Marker). A complementary node, example_marker_user_app_v2.cpp", creates the wall of points as "marker_example", but it publishes the points on topic "marker_listener" (of type geometry_msgs::PointStamped). The broker, "marker_listener_v2" can receive this data and translate it into the format required for visualization in rviz. To see the result (again, with gazebo/drcsim running, and with rviz running), do:

```
rosrun example_interactive_marker example_marker_user_app_v2
rosrun example_interactive_marker marker_listener_v2
```

In rviz, add (if necessary) a "Marker" item, and set its topic to: "marker_publisher." The net result of these two nodes will be the same as marker_example.

A valuable capability of the PointStamped data type is that it carries a reference to its own reference frame. In example_marker_user_app_v2, one may change the line:

```
  stmpdPoint.header.frame_id = "utorso";
```

to some other named frame (e.g. pelvis, right_grasp_frame, etc). The resulting visualization in rviz will display the points with respect to the declared reference frame.