TruPhysics ROSConnector documentation (May 29, 2017)

Subscribing to topics

- Needs the TruRosConnectionManager-plugin (TruRosConnectionManager.dll) and additionally TruRosConnector.dll
- Add a TruRosConnectionManager-Element to the scene graph. This element has the following two parameters:
  - rosMasterURI: A ROS master URI
  - rosTopics: (optional) A list of subscribed ROS-topics. A list entry has the following format: <topic name>:::<message type>. List entries are separated by ';'
- Example:
  <TruRosConnectionManager name="truPhysicsRos"
  rosMasterURI="http://10.1.1.3:11311"
  rosTopics="/joint_states:::sensor_msgs::JointState;/gripper_state:::std_msgs::Bool"/>

Currently, the TruRosConnectionManager supports subscribing to messages of type 'sensor_msgs::JointState' and 'std_msgs::Bool'. To implement subscribing to a new message type, a few lines of code that instantiate the subscriber class when needed must be added to the function TruRosConnectionManager::bwdInit(). For std_msgs::Bool messages this is:

```
if (messageType.compare("std_msgs::Bool") == 0)
{
  supported = true;

  boost::shared_ptr< TruRosConnectorTopicSubscriber<std_msgs::Bool> > tmp(
    new TruRosConnectorTopicSubscriber<std_msgs::Bool>(
      m_ros_connector->getROSNode(),
      rosTopic
    )
  );
  //add a message queue length (default: 50) and 'true' to the constructor
  //parameters of the TruRosConnectorTopicSubscriber, to get a subscriber
  //that writes every incoming message to the command line (for testing)

  m_ros_connector->addTopicListener(
    boost::dynamic_pointer_cast<TruRosListener>(tmp)
  );
  topicListeners.push_back(
    boost::dynamic_pointer_cast<TruRosListener>(tmp)
  );
}
```

For other message types, this can be adapted easily.

Processing messages to subscribed topics

When a message is published to a subscribed topic, the subscriber class emits a boost::signal2. Connecting to this signal is currently work in progress. The file connectToSubscriberSignalExample.txt contains a code example of how it works currently.

Publishing topics

- Needs the TruRosConnectionManager-plugin (TruRosConnectionManager.dll) and also TruRosConnector.dll and TruRosPublishingHandler.dll
- Add a TruRosConnectionManager-Element to the scene graph (same as for subscribing, see above). Note that published rostopics do not need to be given as parameters of the TruRosConnectionManager-Element.
- The project in which messages are to be published needs to link against TruRosConnector and TruRosPublishingHandler and include the directories with their headers.
- The class that is supposed to publish messages needs to include 'TruPhysicsROSConnector.h', 'TruRosPublishingHandler.h'. The publisher then needs to be initialized once, before messages can be published.
- to initialize the publisher:
  - first instantiate an object of class TruRosPublishingHandler and call the 'setROSConnectionManagerByContext(BaseContext*)' method with a scene-graph context as an argument.
    The method looks for a TruRosConnectionManager in the scene graph and returns true if successful.
    (CAREFUL! When this is called, the TruRosConnectionManager in the scene graph must already be instantiated, so a constructor or an init() method are bad places to call setROSConnectionManagerByContext - I usually use a bwdInit() method)
  - instantiate the TruRosConnectorTopicPublisher object, using the message-type as template-argument. The constructor-arguments are the 'getROSNodeHandle()' method of the TruRosPublishingHandler and a string with the name of the topic to publish.
  - call the 'registerPublisher(ROSConnector::TruRosPublisher*)' method of the TruRosPublishingHandler, using a pointer to the publisher object as argument
- to publish a message:
  - create an object of the message type that is supposed to be published and fill it with data
  - call the 'publishMessage' method of the publisher

Example, initialization:

```
TruPhysics::ROSPublishing::TruRosPublishingHandler publishingHandler;
TruPhysics::ROSConnector::TruRosConnectorTopicPublisher<
  std_msgs::Float32
>* timePublisher;

connectionManagerFound =
  publishingHandler.setROSConnectionManagerByContext(
    getContext()
  );

if (connectionManagerFound)
{
  timePublisher =
    new TruPhysics::ROSConnector::TruRosConnectorTopicPublisher<
      std_msgs::Float32
    >
    (
      publishingHandler.getROSNodeHandle(),
      "sofaSimTime"
    );
  publishingHandler.registerPublisher(timePublisher);
}
```

Example, publish message:

```
if (connectionManagerFound)
{
  // publish simulation time
  std_msgs::Float32 msgFloat32;
  msgFloat32.data = getContext()->getTime();
  if (timePublisher)
  {
    timePublisher->publishMessage(msgFloat32);
  }
}
```