API 생성 및 ros connection

참조

Create API: http://gazebosim.org/tutorials?cat=guided_i&tut=guided_i5

Ros connection: http://gazebosim.org/tutorials?cat=guided_i&tut=guided_i6

1. Create API

Sdf file에서 velocity tag를 이용해 velocity를 설정해주는 것은 편리하지만, world file이 실행되고 있는 동안 velocity의 값을 수정하지 못한다. Velocity를 동적으로 조절해주기 위해 다른 프로그램에서 velocity의 값을 수정하도록 api를 만든다

1.1 velodyne_plugin.cc file 수정

$ gedit gazebo_ws/velodyne_plugin.cc

1.1.1 VelodynePlugin class내부에 Setvelocity function추가

```
/// brief Set the velocity of the Velodyne
/// param[in] _vel New target velocity
public: void SetVelocity(const double &_vel)
{
  // Set the joint's target velocity.
  this->model->GetJointController()->SetVelocityTarget(
      this->joint->GetScopedName(), _vel);
}
```

1.1.2 Message passing infrastructure 만들기

Node와 subscriber 변수 추가

```
/// brief A node used for transport
private: transport::NodePtr node;

/// brief A subscriber to a named topic.
       private: transport::SubscriberPtr sub;
```

1.1.3 Load function의 끝에 node와 subscriber instantiate

```
// Create the node
this->node = transport::NodePtr(new transport::Node());
```

```
#if GAZEBO_MAJOR_VERSION < 8
this->node->Init(this->model->GetWorld()->GetName());
#else
this->node->Init(this->model->GetWorld()->Name());
#endif

// Create a topic name
std::string topicName = "~/" + this->model->GetName() + "/vel_cmd";

// Subscribe to the topic, and register a callback
this->sub = this->node->Subscribe(topicName,
   &VelodynePlugin::OnMsg, this);
```

1.1.4 Incoming message들을 취급하기 위한 callback 함수 작성

```
/// \param[in] _msg Repurpose a vector3 message. This function will
/// only use the x component.
private: void OnMsg(ConstVector3dPtr &_msg)
{
  this->SetVelocity(_msg->x());
}
```

1.1.5 Include 추가

```
#include <gazebo/transport/transport.hh>
#include <gazebo/msgs/msgs.hh>
```

2. Test the message passing API

```
$ gedit ~/gazebo_ws/vel.cc
```

```
#include <gazebo/gazebo_config.h>
#include <gazebo/transport/transport.hh>
#include <gazebo/msgs/msgs.hh>

// Gazebo's API has changed between major releases. These changes are
// accounted for with #if..#endif blocks in this file.
#if GAZEBO_MAJOR_VERSION < 6
#include <gazebo/gazebo.hh>
#else
#include <gazebo/gazebo_client.hh>
#endif

/////////////////////////////////////////////////
int main(int _argc, char **_argv)
{
  // Load gazebo as a client
#if GAZEBO_MAJOR_VERSION < 6
  gazebo::setupClient(_argc, _argv);
#else
  gazebo::client::setup(_argc, _argv);
#endif

  // Create our node for communication
```

```
  gazebo::transport::NodePtr node(new gazebo::transport::Node());
  node->Init();

  // Publish to the  velodyne topic
  gazebo::transport::PublisherPtr pub =
    node->Advertise<gazebo::msgs::Vector3d>("~/my_velodyne/vel_cmd");

  // Wait for a subscriber to connect to this publisher
  pub->WaitForConnection();

  // Create a a vector3 message
  gazebo::msgs::Vector3d msg;

  // Set the velocity in the x-component
#if GAZEBO_MAJOR_VERSION < 6
  gazebo::msgs::Set(&msg, gazebo::math::Vector3(std::atof(_argv[1]), 0,
0));
#else
  gazebo::msgs::Set(&msg, ignition::math::Vector3d(std::atof(_argv[1]), 0,
0));
#endif

  // Send the message
  pub->Publish(msg);

  // Make sure to shut everything down.
#if GAZEBO_MAJOR_VERSION < 6
  gazebo::shutdown();
#else
  gazebo::client::shutdown();
#endif
}
```

//cmakefile 열어서 추가

```
$ gedit ~/gazebo_ws/CMakeLists.txt

add_executable(vel vel.cc)

if (${gazebo_VERSION_MAJOR} LESS 6)
  # These two
  include(FindBoost)
  find_package(Boost ${MIN_BOOST_VERSION} REQUIRED system filesystem
regex)
  target_link_libraries(vel ${GAZEBO_LIBRARIES} ${Boost_LIBRARIES})
else()
  target_link_libraries(vel ${GAZEBO_LIBRARIES})
endif()
```

3. Compile하고 simulation하기

```
$ cd ~/gazebo_ws/build

$ cmake ../

$ make

$ gazebo velodyne.world
```

새로운 terminal 창 열고

```
$ cd ~/gazebo_ws/build

$ ./vel 2
```

4. Add ROS transport

4.1 velodyne_plugin.cc file을 수정

```
$ gedit ~/gazebo_ws/velodyne_plugin.cc
```

4.1.1 header file 추가

```
#include <thread>
#include "ros/ros.h"
#include "ros/callback_queue.h"
#include "ros/subscribe_options.h"
#include "std_msgs/Float32.h"
```

4.1.2 member variable들 추가

```
/// brief A node use for ROS transport
private: std::unique_ptr<ros::NodeHandle> rosNode;

/// brief A ROS subscriber
private: ros::Subscriber rosSub;

/// brief A ROS callbackqueue that helps process messages
private: ros::CallbackQueue rosQueue;

/// brief A thread the keeps running the rosQueue
private: std::thread rosQueueThread;
```

4.1.3 Load function의 끝에 추가

```
// Initialize ros, if it has not already bee initialized.
if (!ros::isInitialized())
{
  int argc = 0;
  char **argv = NULL;
  ros::init(argc, argv, "gazebo_client",
      ros::init_options::NoSigintHandler);
```

```
}

// Create our ROS node. This acts in a similar manner to
// the Gazebo node
this->rosNode.reset(new ros::NodeHandle("gazebo_client"));

// Create a named topic, and subscribe to it.
ros::SubscribeOptions so =
  ros::SubscribeOptions::create<std_msgs::Float32>(
      "/" + this->model->GetName() + "/vel_cmd",
      1,
      boost::bind(&VelodynePlugin::OnRosMsg, this, _1),
      ros::VoidPtr(), &this->rosQueue);
this->rosSub = this->rosNode->subscribe(so);

// Spin up the queue helper thread.
this->rosQueueThread =
  std::thread(std::bind(&VelodynePlugin::QueueThread, this));
```

4.1.4 Load function의 밖에 필요한 함수들 추가

```
/// \brief Handle an incoming message from ROS
/// \param[in] _msg A float value that is used to set the velocity
/// of the Velodyne.
public: void OnRosMsg(const std_msgs::Float32ConstPtr &_msg)
{
  this->SetVelocity(_msg->data);
}

/// \brief ROS helper function that processes messages
private: void QueueThread()
{
  static const double timeout = 0.01;
  while (this->rosNode->ok())
  {
    this->rosQueue.callAvailable(ros::WallDuration(timeout));
  }
}
```

4.2 CMakeLists.txt file 열고 수정

```
cmake_minimum_required(VERSION 2.8 FATAL_ERROR)

find_package(roscpp REQUIRED)
find_package(std_msgs REQUIRED)
include_directories(${roscpp_INCLUDE_DIRS})
include_directories(${std_msgs_INCLUDE_DIRS})

# Find Gazebo
find_package(gazebo REQUIRED)
include_directories(${GAZEBO_INCLUDE_DIRS})
link_directories(${GAZEBO_LIBRARY_DIRS})
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${GAZEBO_CXX_FLAGS}")
```

```
# Build our plugin
add_library(velodyne_plugin SHARED velodyne_plugin.cc)
target_link_libraries(velodyne_plugin ${GAZEBO_LIBRARIES}
${roscpp_LIBRARIES})

# Build the stand-alone test program
add_executable(vel vel.cc)

if (${gazebo_VERSION_MAJOR} LESS 6)
  include(FindBoost)
  find_package(Boost ${MIN_BOOST_VERSION} REQUIRED system filesystem
regex)
  target_link_libraries(vel ${GAZEBO_LIBRARIES} ${Boost_LIBRARIES})
else()
  target_link_libraries(vel ${GAZEBO_LIBRARIES})
endif()
```

새로운 터미널 열고

```
$ source /opt/ros/kinetic/setup.bahs

$ cd ~/gazebo_ws/build

$ cmake ../

$ make
```

5. Control velodyne from ROS

Start roscore

```
$ source /opt/ros/kinetic/setup.bash

$ roscore
```

In a new terminal, start Gazebo

```
$ ~/gazebo_ws/build

$ source /opt/ros/kinetic/setup.bash

$ gazebo velodyne.world
```

In a new terminal, use rostopic to send a velocity message

```
$ source /opt/ros/kinetic/setup.bash

$ rostopic pub /my_velodyne/vel_cmd std_msgs/Float32 100.0
```