NOTES

Software development:

Start vscode: open windows terminal, devel\setup.bat; code .

Build:

catkin\_make ; catkin\_make install

Run:

cd bin; gztaskboard.bat (handles env var setup)

process attach: rosrun test\_gazebo\_package test\_gazebo\_package rosbreak:=1

run: process attach: rosrun test\_gazebo\_package test\_gazebo\_package rosbreak:=0

## Install catkin tools

c:\opt\ros>pip3 install -U catkin\_tools

from <https://catkin-tools.readthedocs.io/en/latest/installing.html>

BUT!!!!!!!!!!

ModuleNotFoundError: No module named 'fcntl'

The fcntl module is not available on Windows. The functionality it exposes does not exist on that platform.

## Release build with debug symbols (I hope)

catkin\_make -DCMAKE\_BUILD\_TYPE=Debug -D\_ITERATOR\_DEBUG\_LEVEL=2

no instead

<https://docs.microsoft.com/en-us/cpp/build/reference/debug-generate-debug-info?view=msvc-160>

set\_property(TARGET ${target} APPEND\_STRING PROPERTY LINK\_FLAGS " /DEBUG")

## error LNK2038: mismatch detected for '\_ITERATOR\_DEBUG\_LEVEL': value '0' doesn't match value '2'

mixed dbug and release builds/links. Sigh.

<https://stackoverflow.com/questions/7668200/error-lnk2038-mismatch-detected-for-iterator-debug-level-value-0-doesnt>

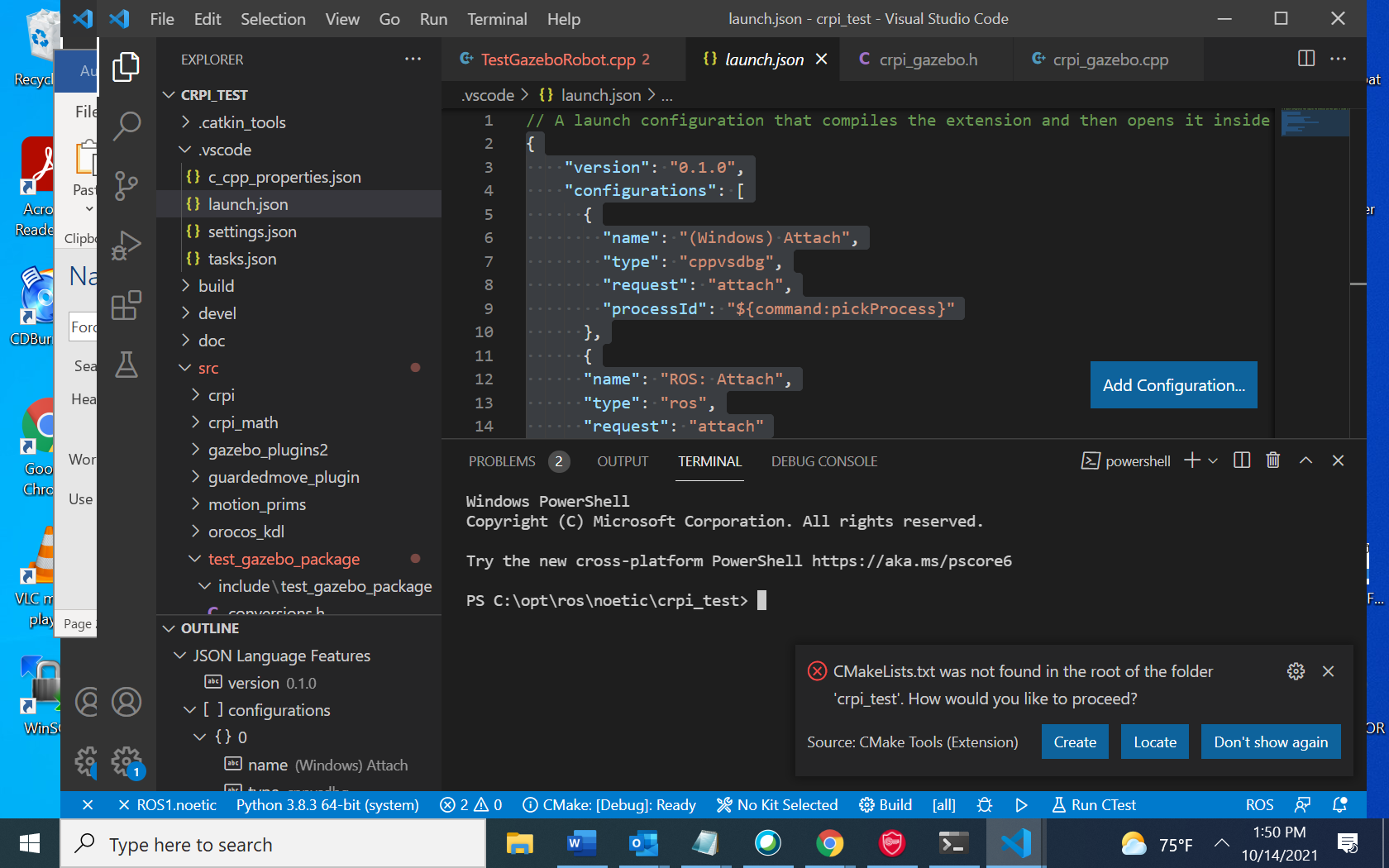
To get to compile in vscode had to get catkin\_make work in shell.

Cntl-Alt-B somehow builds.

Slower build in VSCODE

Tasks.json:

How was this created:



{

    "version": "2.0.0",

    "tasks": [

        {

            "label": "catkin\_make",

            "type": "shell",

            "command": "catkin\_make",

            "args": [

                 "-DCMAKE\_BUILD\_TYPE=Release",

                 "-DCMAKE\_EXPORT\_COMPILE\_COMMANDS=ON"

             ],

            "problemMatcher": ["$msCompile"],

            "group": {

                "kind": "build",

                "isDefault": true

            }

        },

        {

            "label": "crpi\_simulation",

            "type": "shell",

            "command": "roslaunch panda\_simulation simulation.launch",

            "problemMatcher": [],

            "group": {

                "kind": "test",

                "isDefault": true

            }

        }

    ]

}

c\_cpp\_properties.json

{

    "configurations": [

        {

            "browse": {

                "databaseFilename": "",

                "limitSymbolsToIncludedHeaders": true

            },

            "includePath": [

                "c:\\opt\\ros\\noetic\\x64\\include\\\*\*",

                "C:\\opt\\ros\\noetic\\x64\\tools\\vcpkg\\installed\\x64-windows\\include\\\*\*",

                "C:\\opt\\ros\\noetic\\x64\\tools\\vcpkg\\installed\\x64-windows\\include\\\*\*",

                "C:\\opt\\ros\\noetic\\crpi\_test\\src\\TestGazeboRobot\\include\\\*\*",

                "C:\\opt\\ros\\noetic\\crpi\_test\\src\\crpi\_math\\include\\\*\*",

                "C:\\opt\\ros\\noetic\\crpi\_test\\src\\orocos\_kdl\\include\\\*\*"

            ],

            "name": "ROS",

            "configurationProvider": "b2.catkin\_tools"

        }

    ],

    "version": 4

}

settings.json

{

    "ros.distro": "noetic",

    "python.autoComplete.extraPaths": [

        "c:\\opt\\ros\\noetic\\x64\\lib/site-packages",

        "C:\\opt\\ros\\noetic\\x64\\Lib\\site-packages"

    ],

    "python.analysis.extraPaths": [

        "c:\\opt\\ros\\noetic\\x64\\lib/site-packages",

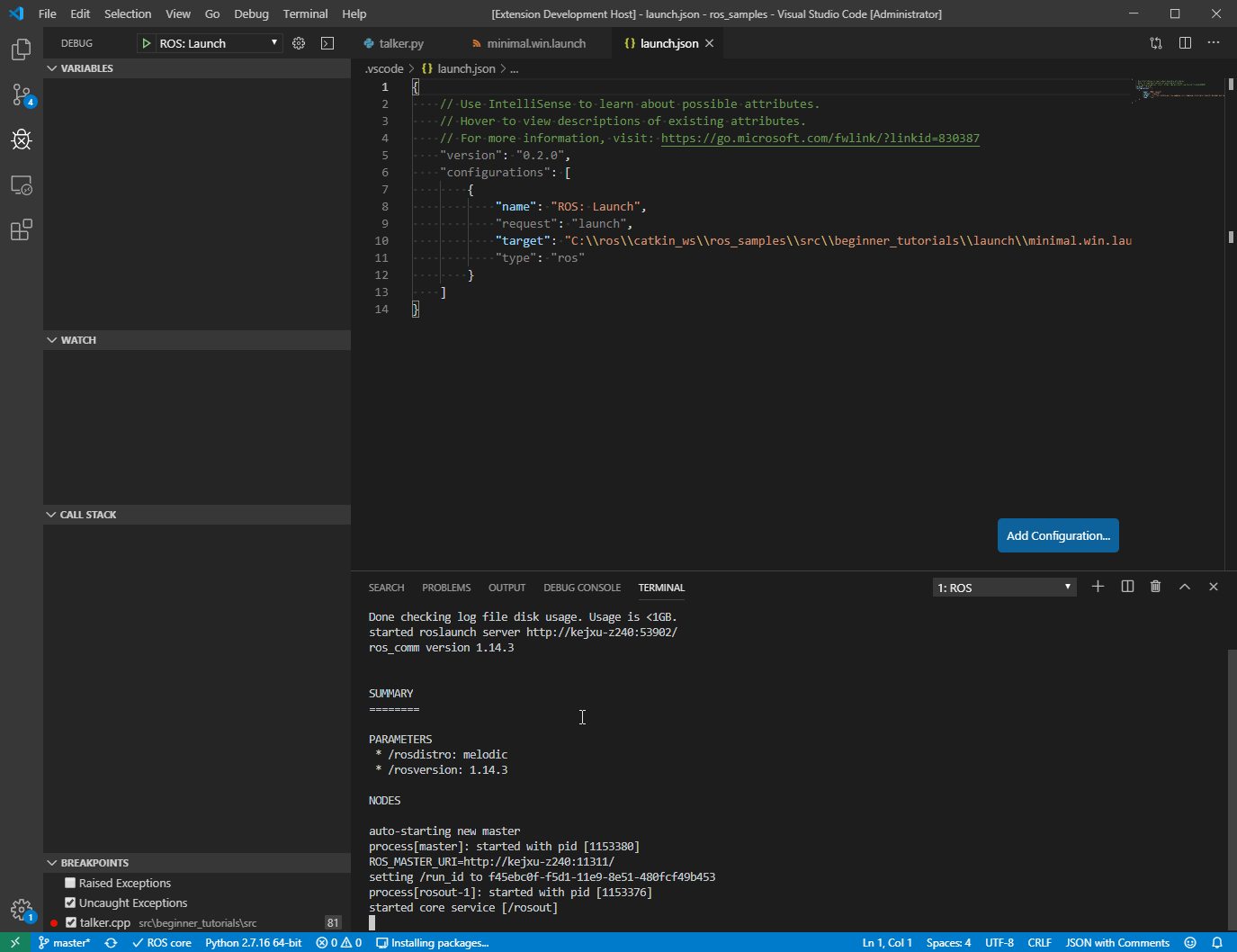
        "C:\\opt\\ros\\noetic\\x64\\Lib\\site-packages"

    ]

}

ctrl + shift + p

|  |  |
| --- | --- |
| ROS: Start | Start ROS1 core or ROS2 Daemon. |
| ROS: Update C++ Properties | Update the C++ IntelliSense configuration to include ROS and your ROS components. |



GzParallelGripperPlugin: Compiled Jun 22 2021 17:17:00

GzParallelGripperPlugin: Model plugin: lrmate

Joints:

fanuc\_base\_link-base

fanuc\_joint\_1

fanuc\_joint\_2

fanuc\_joint\_3

fanuc\_joint\_4

fanuc\_joint\_5

fanuc\_joint\_6

fanuc\_prism1

fanuc\_prism2

gzParallelGripperPlugin Model = lrmate

gzParallelGripperPlugin Namespace = fanuc\_lrmate200id

gzParallelGripperPlugin Joint1 = lrmate::fanuc\_prism1

gzParallelGripperPlugin Joint2 = lrmate::fanuc\_prism2

gzParallelGripperPlugin Link1 = fanuc\_finger\_1

gzParallelGripperPlugin Link2 = fanuc\_finger\_2

gzParallelGripperPlugin Service Control Topic = /fanuc\_lrmate200id/control

gzParallelGripperPlugin Debug = 1

gzParallelGripperPlugin Synchronous = 0

Subscribing contact manager to topic ~/lrmate/contacts

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4068")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4100")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4101")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4127")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4251")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4267")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4273")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4275")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4305")

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} /wd4458")

namespace conversion {

inline int RotMatrix2Rpy(tf::Matrix3x3 m, double& r, double& p, double& y)

{

p = atan2(-m[0][2], sqrt(SQ(m[0][0]) + SQ(m[0][1])));

if (fabs(p - M\_PI\_2) < RPY\_P\_FUZZ) {

r = atan2(m[1][0], m[1][1]);

p = M\_PI\_2;

/\* force it \*/

y = 0.0;

}

else if (fabs(p + M\_PI\_2) < RPY\_P\_FUZZ) {

r = -atan2(m[1][2], m[1][1]);

p = -M\_PI\_2;

/\* force it \*/

y = 0.0;

}

else {

r = atan2(m[1][2], m[2][2]);

y = atan2(m[0][1], m[0][0]);

}

return 0;

}

/\*!

\* \brief Empty conversion of type from into type to. If called, asserts.

\* \param f is defined in the template corresponding to the "From" typename.

\* \return to is defined in the template corresponding "To" typename

\*/

template<typename From, typename To>

inline To Convert(From f) {

To to;

//FIXME:?

//BOOST\_STATIC\_ASSERT(sizeof(To) == 0);

assert(0);

return to;

}

/\*\*

\* \brief Convert std::vector<double> into tf::Pose.

\* \param ds are an array of 6 or 7 doubles to define tf Pose.

\* If 6 double use rpy if 7 use quaternion.

\* \return tf::Pose

\*/

template<>

inline tf::Pose Convert<std::vector<double>, tf::Pose>(std::vector<double> ds)

{

tf::Pose pose;

if (ds.size() == 6)

{

std::array<double, 6> arr;

std::copy\_n(ds.begin(), 6, arr.begin());

pose = Convert<std::array<double, 6>, tf::Pose>(arr);

}

else if (ds.size() == 7)

{

std::array<double, 7> arr;

std::copy\_n(ds.begin(), 7, arr.begin());

pose = Convert<std::array<double, 7>, tf::Pose>(arr);

// message should now contain object - can't really detect if exists

}

else

{

assert(0);

}

return pose;

}

#if 0

/\*!

\* \brief Convert geometry\_msgs::Pose into tf::Pose.

\* \param pose is copy constructor of geometry\_msgs::Pose.

\* \return tf::Pose

\*/

template<>

inline tf::Pose Convert<geometry\_msgs::Pose, tf::Pose>(geometry\_msgs::Pose m) {

return tf::Pose(tf::Quaternion(m.orientation.x, m.orientation.y, m.orientation.z, m.orientation.w),

tf::Vector3(m.position.x, m.position.y, m.position.z));

}

/\*!

\* \brief Convert tf::Pose pose into an geometry\_msgs::Pose pose.

\* \param m is a tf::Pose transform matrix.

\* \return geometry\_msgs::Pose pose

\*/

template<>

inline geometry\_msgs::Pose Convert< tf::Pose, geometry\_msgs::Pose>(tf::Pose m) {

geometry\_msgs::Pose p;

p.position.x = m.getOrigin().x();

p.position.y = m.getOrigin().y();

p.position.z = m.getOrigin().z();

p.orientation.x = m.getRotation().x();

p.orientation.y = m.getRotation().y();

p.orientation.z = m.getRotation().z();

p.orientation.w = m.getRotation().w();

return p;

}

#endif

template<>

inline tf::Vector3 Convert<urdf::Vector3, tf::Vector3>(urdf::Vector3 v) {

return tf::Vector3(v.x, v.y, v.z);

}

inline std::vector<double> getTranslation(tf::Pose p) {

std::vector<double> trans = { p.getOrigin().getX(), p.getOrigin().getY(), p.getOrigin().getZ() };

return trans;

}

inline tf::Pose Convert(matrix\* forward, double lengthConversion = 1.0) {

tf::Pose p;

tf::Vector3 trans((\*forward).at(0, 3), (\*forward).at(1, 3), (\*forward).at(2, 3));

p.setOrigin(trans \* lengthConversion);

std::vector<double> q;

(\*forward).rotQuaternionMatrixConvert(q);

p.setRotation(tf::Quaternion(q[0], q[1], q[2], q[3]));

return p;

}

template<>

inline robotPose Convert<tf::Pose, robotPose>(tf::Pose p) {

robotPose pose;

pose.x = p.getOrigin().getX();

pose.y = p.getOrigin().getY();

pose.z = p.getOrigin().getZ();

RotMatrix2Rpy(p.getBasis(), pose.xrot, pose.yrot, pose.zrot);

return pose;

}

inline matrix getRotation(tf::Pose p) {

matrix r(3, 3);

tf::Matrix3x3 m = p.getBasis();

r.at(0, 0) = m[0][0];

r.at(0, 1) = m[0][1];

r.at(0, 2) = m[0][2];

r.at(1, 0) = m[1][0];

r.at(1, 1) = m[1][1];

r.at(1, 2) = m[1][2];

r.at(2, 0) = m[2][0];

r.at(2, 1) = m[2][1];

r.at(2, 2) = m[2][2];

return r;

}

template<>

inline tf::Pose Convert<robotPose&, tf::Pose >(robotPose& pose) {

tf::Pose p;

p.setOrigin(tf::Vector3(pose.x, pose.y, pose.z));

matrix r(3, 3);

std::vector<double> trans = { pose.x, pose.y, pose.z };

r.rotEulerMatrixConvert(trans);

std::vector<double> q;

r.rotQuaternionMatrixConvert(q);

p.setRotation(tf::Quaternion(q[0], q[1], q[2], q[3]));

return p;

}

/\*\*

\* @brief DumpPoseSimple generatseriallinkrobotes string of xyz origin and rpy rotation from a tf pose.

\* @param pose tf pose

\* @return std::string

\*/

inline std::string dumpPoseSimple(tf::Pose pose) {

std::stringstream s;

s << std::setprecision(3) << boost::format("%7.2f") % (pose.getOrigin().x()) << "," <<

boost::format("%7.2f") % (pose.getOrigin().y()) << "," <<

boost::format("%7.2f") % (pose.getOrigin().z()) << ",";

double roll = 0, pitch = 0, yaw = 0;

tf::Matrix3x3 m;

m.setRotation(pose.getRotation());

RotMatrix2Rpy(m, roll, pitch, yaw);

s << std::setprecision(3) << boost::format("%5.3f") % pose.getRotation().x() << ","

<< boost::format("%5.3f") % pose.getRotation().y() << ","

<< boost::format("%5.3f") % pose.getRotation().z() << ","

<< boost::format("%5.3f") % pose.getRotation().w();

return s.str();

}

template<>

inline tf::Pose Convert<KDL::Frame, tf::Pose >(KDL::Frame k)

{

tf::Pose m;

m.getOrigin().setX(k.p[0]);

m.getOrigin().setY(k.p[1]);

m.getOrigin().setZ(k.p[2]);

double x, y, z, w;

k.M.GetQuaternion(x, y, z, w);

m.setRotation(tf::Quaternion(x, y, z, w));

return m;

}

template<>

inline KDL::Frame Convert<tf::Pose, KDL::Frame >(tf::Pose m)

{

KDL::Frame k;

k.p[0] = m.getOrigin().x();

k.p[1] = m.getOrigin().y();

k.p[2] = m.getOrigin().z();

k.M = KDL::Rotation::Quaternion(m.getRotation().x(), m.getRotation().y(), m.getRotation().z(), m.getRotation().w());

return k;

}

template<>

inline KDL::JntArray Convert<std::vector<double>, KDL::JntArray >(std::vector<double> joints)

{

KDL::JntArray joint\_list(joints.size());

// Fill in KDL joint list

for (size\_t i = 0; i < joints.size(); i++)

joint\_list(i) = joints[i];

return joint\_list;

}

template<>

inline std::vector<double> Convert<KDL::JntArray, std::vector<double> >(KDL::JntArray joint\_list)

{

std::vector<double> joints;

// Fill in KDL joint list

for (size\_t i = 0; i < joint\_list.data.size(); i++)

joints.push\_back(joint\_list(i));

return joints;

}

}

## VSCODE to attach to running ROS node:

1. First, add looping code to break into main: Note bBreak MUST BE A GLOBAL. (not sure how to access local variables).

bool bBreak = 1;

int main()

{

// if no break then delay for Gazebo visual to finish loading.

while (bBreak)

{

// sleep 1 second

Sleep(1000);

}

1. Next, in all CMakelist.txt add /DEBUG to build pdf file while doing Release build (had too many problems mixing Debug and Release builds linking properly).

set\_property(TARGET ${target} APPEND\_STRING PROPERTY LINK\_FLAGS " /DEBUG")

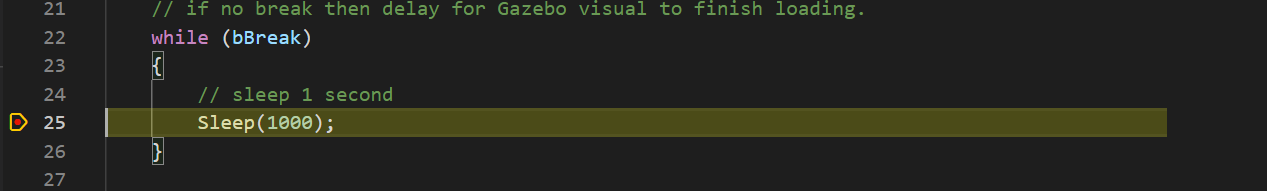
1. Build with catkin\_make in Windows Terminal configured for ROS noetic.
2. In Windows terminal after successful build:

devel\setup.bat

rosrun test\_gazebo\_package test\_gazebo\_package

In VSCODE:

1. Put a breakpoint in the main.cpp at the Sleep(1000); statement (which will stop the attach).



1. Code up (Windows) attach in Launch.json, which works because catkin\_make uses vs2019 to build the cpp code and we enabled .pdb (program db) file using the /DEBUG linker flag.

{

"version": "0.1.0",

"configurations": [

{

"name": "(Windows) Attach",

"type": "cppvsdbg",

"request": "attach",

"processId": "${command:pickProcess}"

},

{

"name": "ROS: Attach",

"type": "ros",

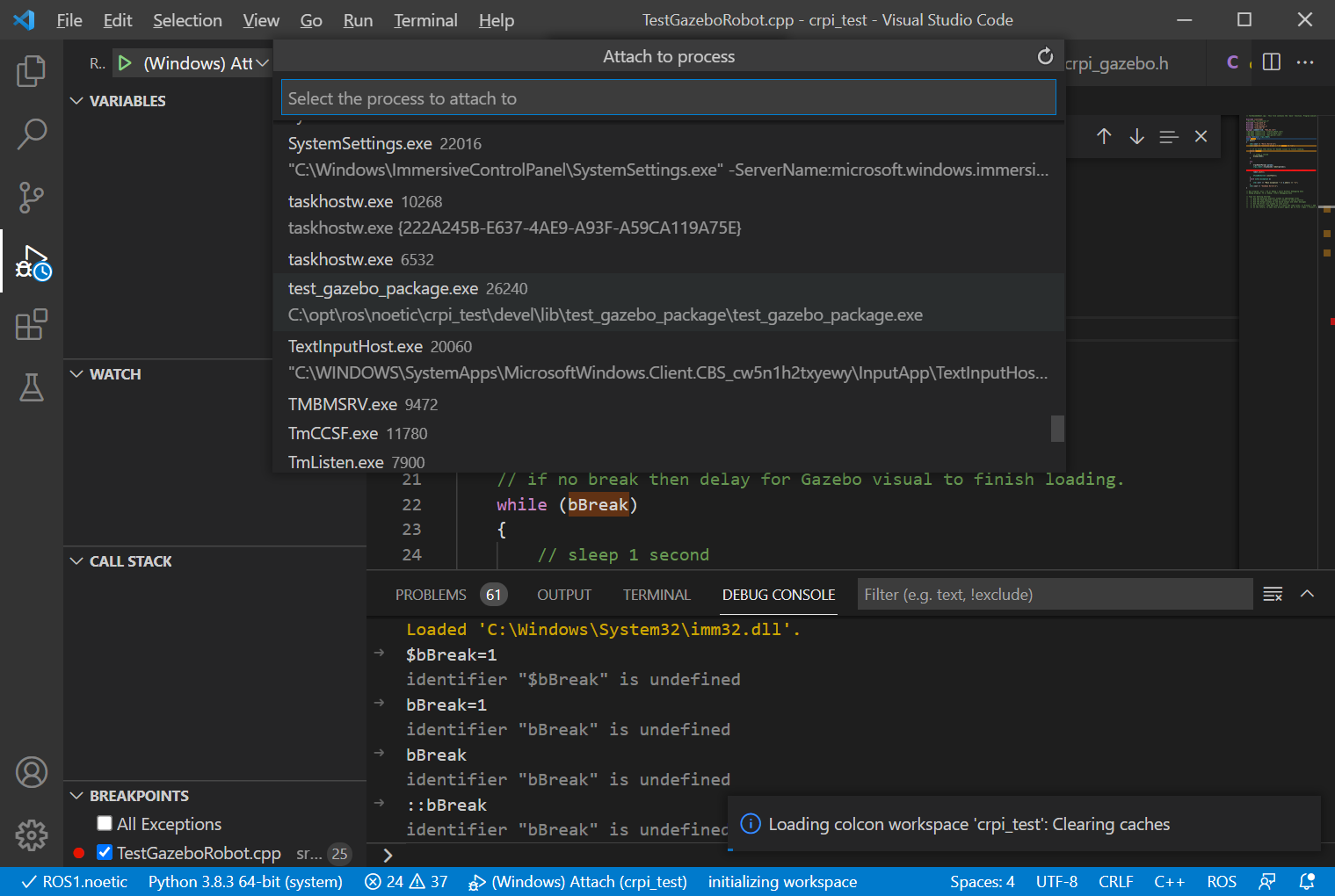
"request": "attach"

}

]

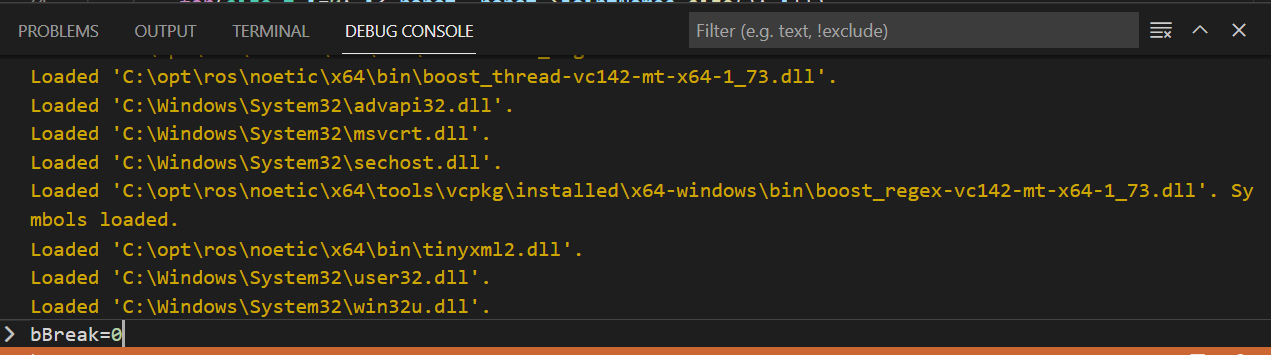
}

1. Select  which is the Debug Panel. Then, hit the green Arrow  for start debug attach command.
2. Select the process to attach to (the exe in rosrun i.e., in our case test\_gazebo\_package).



1. Switch to the DEBUG CONSOLE, and put the variable and the value:

bBreak=0



1. It should attach and then stop Use the debugger symbols to step through the code!

Doesn’t work:

std::vector<std::string> keys;

\_nh->getParamNames(keys);

for (auto key : keys)

{

std::string value;

if (!\_nh->param<std::string>(key, value, "missing"))

std::cout << key << "=" << value << "\n";

}

Added Visual Studio 2019 removal of excess warnings to CMakefile

CMake Warning (dev) at C:\opt\ros\noetic\crpi\_test\build\devel\share\crpi\_math\cmake\crpi\_mathConfig.cmake:77:

Syntax Warning in cmake code at column 54

# set variables for source/devel/install prefixes

if("TRUE" STREQUAL "TRUE")

set(crpi\_math\_SOURCE\_PREFIX C:/opt/ros/noetic/crpi\_test/src/crpi\_math)

set(crpi\_math\_DEVEL\_PREFIX C:/opt/ros/noetic/crpi\_test/build/devel)

set(crpi\_math\_INSTALL\_PREFIX "")

set(crpi\_math\_PREFIX ${crpi\_math\_DEVEL\_PREFIX})

else()

set(crpi\_math\_SOURCE\_PREFIX "")

set(crpi\_math\_DEVEL\_PREFIX "")

**set(crpi\_math\_INSTALL\_PREFIX C:/Program Files (x86)/Project)**

set(crpi\_math\_PREFIX ${crpi\_math\_INSTALL\_PREFIX})

endif()

Argument not separated from preceding token by whitespace.

[DEBUG] [1631640641.159719500, 4715.355000000]: 0,0,0,0,0,0,0

[[ 1, 0, 0;

0, 1, 0;

0, 0, 1]

[ 0.465, 0, 0.695]]

Home = 0.47, 0.00, 0.70,0.000,0.000,0.000,1.000

Goodbye World!

Extract

# TIPS:

Comment block in C++: select section ^/ comments, repeat ^/ to uncomment.

#if 0

chain.addSegment(Segment(Joint(Joint::RotZ), Frame(Vector(0.0, 0.0, 0.330))));

chain.addSegment(Segment(Joint(Joint::RotY), Frame(Vector(0.05, 0.0, 0.0))));

chain.addSegment(Segment(Joint(Joint::RotY,-1.), Frame(Vector(0.0, 0.0, 0.330))));

chain.addSegment(Segment(Joint(Joint::RotX,-1.0), Frame(Vector(0.0,0.0, 0.035))));

chain.addSegment(Segment(Joint(Joint::RotY,-1.0), Frame(Vector(0.335, 0.0, 0.0))));

chain.addSegment(Segment(Joint(Joint::RotX,-1.0), Frame(Vector(0.08, 0.00, 0.0))));

#endif

\* fanuc\_joint\_1 RotAxis [ 0, 0, 1] [ 0, 0, 0.33]

\* fanuc\_joint\_2 RotAxis [ 0, 1, 0] [ 0.05, 0, 0]

\* fanuc\_joint\_3 RotAxis [ 0, -1, 0] [ 0, 0, 0.33]

\* fanuc\_joint\_4 RotAxis [ -1, 0, 0] [ 0, 0, 0.035]

\* fanuc\_joint\_5 RotAxis [ 0, -1, 0] [ 0.335, 0, 0]

\* fanuc\_joint\_6 RotAxis [ -1, 0, 0] [ 0.08, 0, 0]

std::vector<double> Jacobian(std::vector<double> jnts,std::vector<double> vel)

    {

        KDL::JntArray q = this->Convert<std::vector<double>, KDL::JntArray>(jnts);

        KDL::JntArray qdot = this->Convert<std::vector<double>, KDL::JntArray>(vel);

        KDL::JntArray  tau;    // Joint position, velocity, torques

        jnt\_to\_jac\_solver\_->KDL::JntToJac(q, Jtmp);

        kin\_.KDLtoEigen(Jtmp, J);

    }

<http://wiki.ros.org/pr2_mechanism/Tutorials/Coding%20a%20realtime%20Cartesian%20controller%20with%20KDL>

Trajectory

https://github.com/snrkiwi/orocos-kdl/blob/master/orocos\_kdl/examples/trajectory\_example.cpp

A Path is the 6D geometric description of the positions for the motion.  
A VelocityProfile is the time-dependent change of a value.  
A Trajectory combines the two previous two primitives into a  
time-dependent 6D geometric motion.

// Path\_RoundedComposite defines the geometric path along which the robot will move.

Path\_RoundedComposite\* path = new Path\_RoundedComposite(0.2,0.01,new RotationalInterpolation\_SingleAxis());

path->Add(Frame(Rotation::RPY(M\_PI,0,0), Vector(-1,0,0)));

path->Add(Frame(Rotation::RPY(M\_PI/2,0,0), Vector(-0.5,0,0)));

path->Add(Frame(Rotation::RPY(0,0,0), Vector(0,0,0)));

path->Add(Frame(Rotation::RPY(0.7,0.7,0.7), Vector(1,1,1)));

path->Add(Frame(Rotation::RPY(0,0.7,0), Vector(1.5,0.3,0)));

path->Add(Frame(Rotation::RPY(0.7,0.7,0), Vector(1,1,0)));

// always call Finish() at the end, otherwise the last segment will not be added.

path->Finish();

// This defines a trapezoidal velocity profile.

VelocityProfile\* velpref = new VelocityProfile\_Trap(0.5,0.1);

velpref->SetProfile(0,path->PathLength());

// Trajectory defines a motion of the robot along a path.

Trajectory\* traject = new Trajectory\_Segment(path, velpref);

Trajectory\_Composite\* ctraject = new Trajectory\_Composite();

ctraject->Add(traject);

ctraject->Add(new Trajectory\_Stationary(1.0,Frame(Rotation::RPY(0.7,0.7,0), Vector(1,1,0))));

void Trajectory(double t, TrajectoryPoint pointNew, TrajectoryPoint pointOld )

{

JointSplineTrajectory trj(chain.getNrOfJoints()); // create trajectory for 6DOF

trajectory\_port.read(point);

if (point.accelerations)

{

trj.generateCofficients(pointOld.positions, pointNew.positions,

pointOld.velocities, pointNew.velocities, pointOld.accelerations,

pointNew.accelerations, pointNew.duration);

}

else if (point.velocites)

{

trj.generateCofficients(pointOld.positions, pointNew.positions,

pointOld.velocities, pointNew.velocities, pointNew.duration);

}

else

{

trj.generateCofficients(pointOld.positions, pointNew.positions,

pointNew.duration);

}

time += dt;

Setpoints set;

trj.sampleSpline(set.positions, set.velocities, set.accelerations, time);

}

## Test KDL IK/FK:

fanuc\_joint\_1,fanuc\_joint\_2,fanuc\_joint\_3,fanuc\_joint\_4,fanuc\_joint\_5,fanuc\_joint\_6,

KDL chain =

\* fanuc\_joint\_1 RotAxis [ 0, 0, 1] [ 0, 0, 0]

\* fanuc\_joint\_2 RotAxis [ 0, 1, 0] [ 0, 0, 0]

\* fanuc\_joint\_3 RotAxis [ 0, -1, 0] [ 0, 0, 0]

\* fanuc\_joint\_4 RotAxis [ -1, 0, 0] [ 0, 0, 0]

\* fanuc\_joint\_5 RotAxis [ 0, -1, 0] [ 0, 0, 0]

\* fanuc\_joint\_6 RotAxis [ -1, 0, 0] [ 0, 0, 0]

[DEBUG] [1631726133.219423300]: tf::Pose CTinyKdlSolver::FK

[DEBUG] [1631726133.220390000]: 0,0,0,0,0,0

Home = 0.47, 0.00, 0.70,0.000,0.000,0.000,1.000

Fanuc 200iD Home = 0.47,0.00,0.70,0.000,0.000,0.000,1.000

IK=-1.67845e-05,0.00104596,0.00104588,0.000222525,0.000126024,0.000301468

## Testing: [DEBUG] [1631726133.908563300, 8559.772000000]: gzModelStatesCallback

[DEBUG] [1631726133.908563300, 8559.772000000]: gzModelStatesCallback

Problem with mutex on update – exception thrown when same mutex in same thread called twoice

## Testing: curentJoints() currentPose() – both use gazebo.

Home = 0.47, 0.00, 0.70,0.000,0.000,0.000,1.000

Fanuc 200iD Home = 0.47,0.00,0.70,0.000,0.000,0.000,1.000

IK=-1.67845e-05,0.00104596,0.00104588,0.000222525,0.000126024,0.000301468

[DEBUG] [1631726133.908563300, 8559.772000000]: gzModelStatesCallback

Current Joints=0,0,0,0,0,0

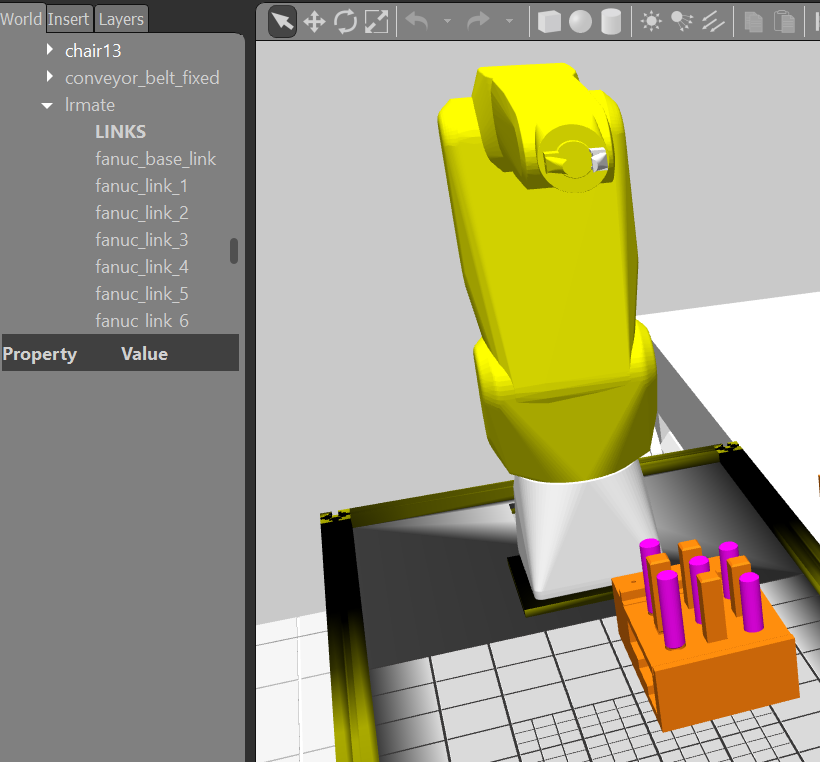
With base -0.17, -1.14, 0.93,0.000,0.000,0.000,1.000

Wout base 0.00, 0.00, 0.00,0.000,0.000,0.000,1.000

Current Pose = 0.00, 0.00, 0.00,0.000,0.000,0.000,1.000

This means that the robot facing forward is the 0,0,0 position. So its shifted 90o in x axis.

Determining name of robot model joints decorated joint names:



void CRobotImpl::gzMovePt(trajectory\_msgs::JointTrajectoryPoint pt)

{

gazebo\_msgs::SetModelConfiguration traj;

traj.request.model\_name = this->\_gzrobot\_name; //lrmate

traj.request.urdf\_param\_name = "";

traj.request.joint\_names = this->jointNames;

for (size\_t k = 0; k < jointNames.size(); k++)

traj.request.joint\_names[k] = "lrmate::" + traj.request.joint\_names[k];

for (size\_t k = 0; k < pt.positions.size(); ++k)

traj.request.joint\_positions.push\_back(pt.positions[k]);

if(bDebug)

std::cout << "gzMovePt Joints=" << vectorDump<double>(pt.positions) << "\n";

if (!\_gzSetJointTraj.call(traj))

{

ROS\_ERROR("Failed to call service /gazebo/set\_model\_configuration\n");

}

}

|  |  |  |
| --- | --- | --- |
| Peg1 | round | 0.2697 -1.1046 0.9307 |
| Peg2 | square | 0.2697 -1.0538 0.9307 |
| Peg3 | round | 0.2697 -1.003 0.9307 |
| Peg4 | square | 0.2189 -1.1046 0.941723 |
| Peg5 | round | 0.2189 -1.0538 0.941723 |
| Peg6 | square | 0.21897 -1.003 0.941723 |
| Peg7 | round | 0.1681 -1.1046 0.9544 |
| Peg8 | square | 0.1681 -1.0538 0.9544 |
| Peg9 | round | 0.1681 -1.003 0.9544 |

OPEN add -.25 in Y

|  |  |  |
| --- | --- | --- |
| Open | round | 0.2697 -1.3546 0.9307 |
| Open | square | 0.2697 -1.3038 0.9307 |
| Open | round | 0.2697 -1.253 0.9307 |
| Open | square | 0.2189 -1.3546 0.941723 |
| Open | round | 0.2189 -1.3038 0.941723 |
| Open | square | 0.21897 -1.253 0.941723 |
| Open | round | 0.1681 -1.3546 0.9544 |
| Open | square | 0.1681 -1.3038 0.9544 |
| Open | round | 0.1681 -1.253 0.9544 |

CRPI conversion issues – rpy ,-> quaternion issues

        // Math::matrix r(3, 3);

        // std::vector<double> rpy = { pose.xrot, pose.yrot, pose.zrot};

        // r.rotEulerMatrixConvert(rpy);

        // std::vector<double> q;

        // r.rotMatrixQuaternionConvert(q);

        // p.setRotation(tf::Quaternion( q[1], q[2], q[3],q[0]));

Test Pose = 0.27, -1.35, 0.93,0.000,0.707,0.000,0.707

Test robotPose = (0.269700, -1.354600, 0.930700) (-0.000000, -1.570796, 0.000000), -1, -1)

Converted robotPose2tf = 0.27, -1.35, 0.93,0.707,0.000,-0.707,0.000

tf::Pose p;  
 p.setOrigin(tf::Vector3(pose.x, pose.y, pose.z) );  
 p.setRotation( tf::Quaternion (pose.yrot, pose.zrot, pose.xrot));

Test Pose = 0.27, -1.35, 0.93,0.000,0.707,0.000,0.707

Test robotPose = (0.269700, -1.354600, 0.930700) (-0.000000, -1.570796, 0.000000), -1, -1)

Converted robotPose2tf = 0.27, -1.35, 0.93,0.000,-0.707,0.000,0.707

# Running CRPI Taskboard Assembly

The debugging platform is to launch a ROS/Gazebo using a Windows command shell (bat).

To run the Gazebo APRS lab setup with the taskboard, we will assume Windows ROS noetic has been installed and that you have installed the Windows terminal.

:: If you have windows terminal setup then this is automatic

cd C:\opt\ros\noetic

cd devel\setup.bat

:: No need to devel\setup.bat as Gazebo APRS setup uses gazebo\_ros\_api

cd noetic\gztaskboard

cd bin

gztaskboard.bat

:: ^C will kill the Gazebo/ROS program

:: to rerun the Gazebo/ROS program

cd ..

gztaskboard.bat

:: To save the history

doskey.exe /history > command.txt

## Attaching to running crpi test program, or let crpi test program run

For debugging, the rosrun command is used to run a ROS program. In order to break or pass through the attach program infinte loop code, the "rosbreak" parameter is passed as part the command line arguments while running the "rosrun" shell command given as:

rosrun <package> <node> <parameter>

If we want to allow Visual code to attach and then continue, we use the following rosrun command line:

c:\opt\ros\noetic\crpi\_test>rosrun test\_gazebo\_package test\_gazebo\_package rosbreak:=1

so that rosbreak:=1 is a command line argument (described next) that is read and can be used as part of a roslaunch command line argument also.

Below, is a command line to ignore the break command loop, and the program will continue without the opportunity to break.

c:\opt\ros\noetic\crpi\_test>rosrun test\_gazebo\_package test\_gazebo\_package rosbreak:=0

The code to enable attaching to program before it reaches the running part, an infinite waiting loop is appended to the start of the program.

bool bBreak = 1;

int main(int argc, char\*\* argv)

{

std::vector<std::string> args;

for(size\_t i=0; i< argc; ++i)

args.push\_back(argv[i]);

bBreak = (bool) atoi( getCmdOption(args, "rosbreak:=", "1").c\_str());

// if no break then delay for Gazebo visual to finish loading.

while (bBreak)

{

// sleep 1 second

Sleep(1000);

}

Issues that were resolved

1. Conversion to/from tf::Pose and crpi::robotPose (never really use crpi for motion control, so used tf to handle the problematic roll/pitch/yaw converions).
2. Top of peg: hole centroid(z) + z length of peg (zpos-zneg)
3. Gripper facing down to pick up peg: for URDF, not (1,0,0,0) but (0,0.707107,0,0.707107)
4. Gripper open size is close to get width – can vary size of pegs with scale – now its all xyz dimensions
5. Test standalone KDL and parse URDF into KDL.

## VSCODE to attach to running ROS node:

1. First, add looping code to break into main: Note bBreak MUST BE A GLOBAL. (not sure how to access local variables).

bool bBreak = 1;

int main()

{

// if no break then delay for Gazebo visual to finish loading.

while (bBreak)

{

// sleep 1 second

Sleep(1000);

}

1. Next, in all CMakelist.txt add /DEBUG to build pdf file while doing Release build (had too many problems mixing Debug and Release builds linking properly).

set\_property(TARGET ${target} APPEND\_STRING PROPERTY LINK\_FLAGS " /DEBUG")

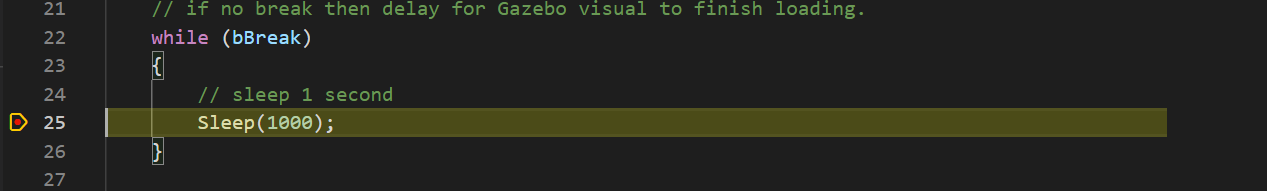
1. Build with catkin\_make in Windows Terminal configured for ROS noetic.
2. In Windows terminal after successful build:

devel\setup.bat

rosrun test\_gazebo\_package test\_gazebo\_package

In VSCODE:

1. Put a breakpoint in the main.cpp at the Sleep(1000); statement (which will stop the attach).



1. Code up (Windows) attach in Launch.json, which works because catkin\_make uses vs2019 to build the cpp code and we enabled .pdb (program db) file using the /DEBUG linker flag.

{

"version": "0.1.0",

"configurations": [

{

"name": "(Windows) Attach",

"type": "cppvsdbg",

"request": "attach",

"processId": "${command:pickProcess}"

},

{

"name": "ROS: Attach",

"type": "ros",

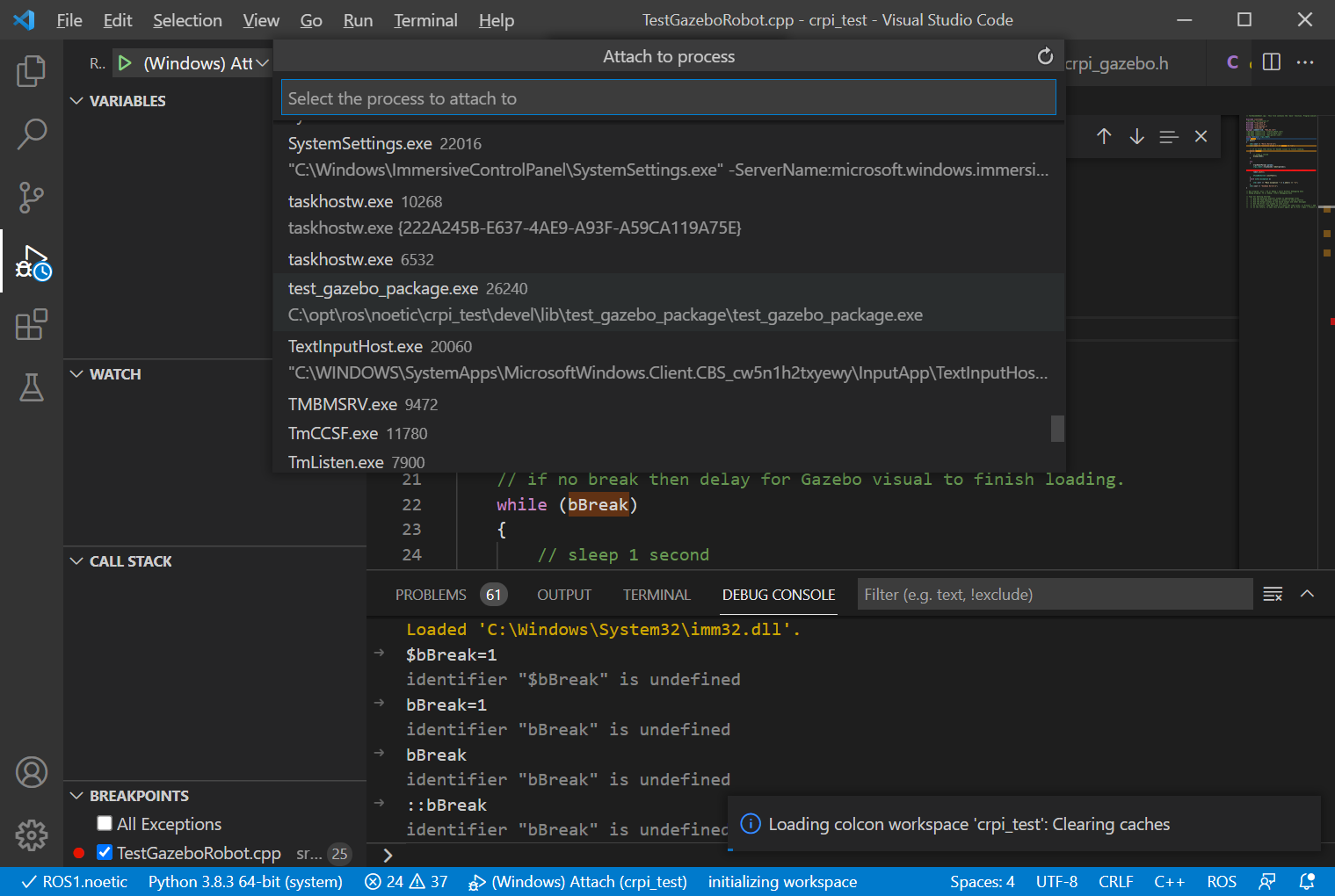
"request": "attach"

}

]

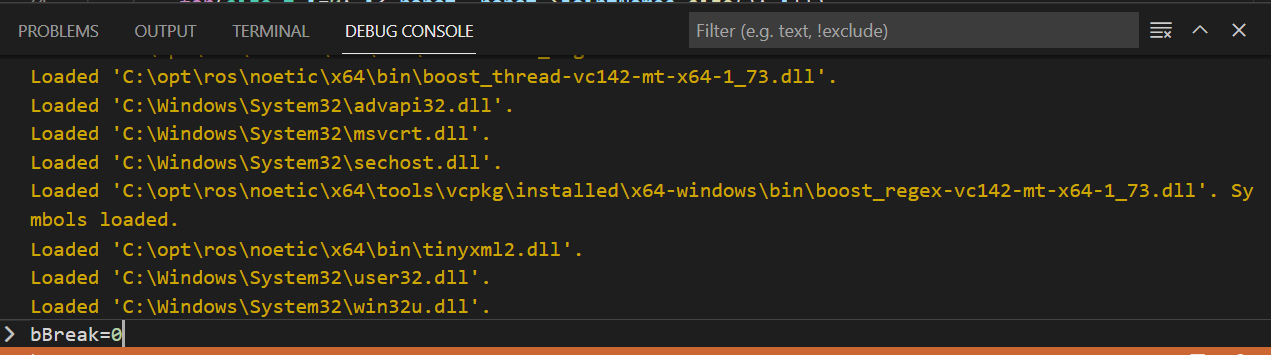
}

1. Select  which is the Debug Panel. Then, hit the green Arrow  for start debug attach command.
2. Select the process to attach to (the exe in rosrun i.e., in our case test\_gazebo\_package).



1. Switch to the DEBUG CONSOLE, and put the variable and the value:

bBreak=0



1. It should attach and then stop Use the debugger symbols to step through the code!

Sleep(5000);

srand((unsigned int) time (NULL)); //activates the generator

for(size\_t i=0; i< robot.\_arm->jointNames.size(); i++)

{

double f = (double)rand() / RAND\_MAX;

robaxes.axis[i]=robot.\_arm->jointMin[i] + f \* (robot.\_arm->jointMax[i] - robot.\_arm->jointMin[i]);

}

robot.MoveToAxisTarget(robaxes,true);

Add force/torque sensor to joint6 fanuc robot

In fanuc\_lrmate200id.urdf

<gazebo reference="fanuc\_joint\_6">

<provideFeedback>true</provideFeedback>

</gazebo>

<gazebo>

<plugin name="ft\_sensor" filename="gazebo\_ros\_ft\_sensor.dll">

<updateRate>50.0</updateRate>

<topicName>ft\_sensor\_topic</topicName>

<jointName>fanuc\_joint\_6</jointName>

</plugin>

</gazebo>

[ INFO] [1632339022.195453400, 0.065000000]: ft\_sensor plugin reporting wrench values to the frame [fanuc\_link\_6]

[ INFO] [1632339022.195702600, 0.065000000]: imu plugin missing <gaussianNoise>, defaults to 0.0

Then see what the rostopic echo ft\_sensor\_topic has on it:

---

header:

seq: 85

stamp:

secs: 46

nsecs: 344000000

frame\_id: "fanuc\_link\_6"

wrench:

force:

x: 0.0

y: 0.0

z: 0.0

torque:

x: 0.0

y: 0.0

z: 0.0

not sure why no force seems to be touching.

c:\opt\ros\noetic\gztaskboard>rostopic echo ft\_sensor\_topic

header:

seq: 0

stamp:

secs: 43

nsecs: 165000000

frame\_id: "fanuc\_link\_6"

wrench:

force:

x: 0.0

y: 0.0

z: 0.0

torque:

x: 0.0

y: 0.0

z: 0.0

rosservice call /gazebo/set\_model\_configuration is far better than the solutions posted here so far, no plugin needed. Here is a complete example of how to achieve this from terminal:

rosservice call /gazebo/set\_model\_configuration "model\_name: 'acrobat'

urdf\_param\_name: 'robot\_description'

joint\_names:

- 'joint1'

joint\_positions:

- 3.3"

Tested on Gazebo 7, ROS kinetic and Ubuntu 16.04 with the [acrobat robot](https://github.com/socrob/acrobat_robot)

Removed from fanuc\_lrmate200id.urdf – joint state publishers way to slow

<gazebo>

<plugin name="joint\_state\_publisher" filename="libgazebo\_ros\_joint\_state\_publisher.dll">

<robotNamespace>/lrmate</robotNamespace>

<jointName>fanuc\_joint\_1,fanuc\_joint\_2,fanuc\_joint\_3,fanuc\_joint\_4,fanuc\_joint\_5,

fanuc\_joint\_6,fanuc\_prism1,fanuc\_prism2</jointName>

<updateRate>100.0</updateRate>

<alwaysOn>true</alwaysOn>

</plugin>

</gazebo>

ADD FT/SENSOR

Never giving anything but 0,0,0 for f/t

<https://stackoverflow.com/questions/22729445/gazebosim-2-2-2-getting-joint-forces-getforcetorque-warning-odejoint-cc1105>

So did this in fanuc urdf:

<gazebo reference="joint\_name">

<provideFeedback>true</provideFeedback>

</gazebo>

Get this error when ROS gazebo APRS is loading:

[ INFO] [1632947564.170624600, 0.001000000]: ft\_sensor plugin reporting wrench values to the frame [fanuc\_link\_6]

[ INFO] [1632947564.170846900, 0.001000000]: imu plugin missing <gaussianNoise>, defaults to 0.0

<http://gazebosim.org/tutorials?tut=ros_gzplugins>

Explains IMU for link.

I wonder how kinematic link responds to f/t sensing.

Interesting summary of all the Gazebo plugins:

<http://gazebosim.org/tutorials?tut=ros_gzplugins>

Why does FT sensor always show zero:

<https://answers.gazebosim.org/question/15465/ft-sensor-shows-zero-values/>

ft\_sensor plugin - does it work? zero force/torque published

<https://answers.gazebosim.org//question/7218/ft_sensor-plugin-does-it-work-zero-forcetorque-published/>

Explains:

It seems that, as for Gazebo 4.0.0, the <provide\_feedback> tag must be put outside of ode, inside the physics tag.

Here the fixed gist:

<https://gist.github.com/clynamen/441b41efcc643039ff89>

with **--verbose** a warning is issued when the <provide\_feedback> tag is missing.

Explanation of the force/torque sensor use IN Gazebo

<https://github.com/osrf/gazebo_tutorials/blob/master/force_torque_sensor/tutorial.md>

c:\opt\ros\noetic\gztaskboard>rosservice call /lrmate/gazebo\_ros\_joint\_state\_publisher\_service lrmate

name:

- fanuc\_base\_link-base

- fanuc\_joint\_1

- fanuc\_joint\_2

- fanuc\_joint\_3

- fanuc\_joint\_4

- fanuc\_joint\_5

- fanuc\_joint\_6

- fanuc\_prism1

- fanuc\_prism2

position: [**nan**, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.011124996423616643, -0.011124996423616643]

velocity: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

effort: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

success: True

status\_message: ''

ERROR: transport error completing service call: receive\_once

c:\opt\ros\noetic\gztaskboard>rosservice call /lrmate/gazebo\_ros\_joint\_state\_publisher\_service lrmate

ERROR: transport error completing service call: receive\_once[/lrmate/gazebo\_ros\_joint\_state\_publisher\_service]: unexpected error [WinError 10054] An existing connection was forcibly closed by the remote host

<gazebo>

<plugin name="gazebo\_ros\_joint\_state\_publisher" filename="gazebo\_ros\_joint\_state\_publisher.dll">

<service\_topic>/lrmate/gazebo\_ros\_joint\_state\_publisher\_service</service\_topic>

<robotNamespace>lrmate</robotNamespace>

<debug> 1 </debug>

</plugin>

</gazebo>

To get force/torque readings from joint 6 needed to enable gravity and not make kinematic.

<gazebo reference="fanuc\_link\_6">

<static>0</static>

**<gravity>1</gravity>**

**<kinematic>0</kinematic>**

<material>Gazebo/Yellow</material>

</gazebo>

BIBLIOGRAPHY

pr2\_controllersTutorialsMoving the arm through a Cartesian pose trajectory <http://wiki.ros.org/pr2_controllers/Tutorials/Moving%20the%20arm%20through%20a%20Cartesian%20pose%20trajectory>

Force/Torque Sensor  
<http://gazebosim.org/tutorials?tut=force_torque_sensor&cat=sensors>

gazebo\_ros\_joint\_state\_publisher.h   
<https://github.com/ros-simulation/gazebo_ros_pkgs/blob/37726f6b3c51f6c20dad9834d7741c64811b7b10/.ros1_unported/gazebo_plugins/include/gazebo_plugins/gazebo_ros_joint_state_publisher.h>

ERDAL'S BLOG Improving Panda Gazebo integration with Position based Control  
<https://erdalpekel.de/?p=285>

Gazebo plugins in ROS  
<http://gazebosim.org/tutorials?tut=ros_gzplugins>

Interpretation of Force/Torque Sensor using gazebo\_ros\_ft\_sensor plugin  
<https://answers.ros.org/question/243173/interpretation-of-forcetorque-sensor-using-gazebo_ros_ft_sensor-plugin/>

The inertia matrix explained  
<https://answers.gazebosim.org/question/4372/the-inertia-matrix-explained/>

If you use geometric primitives (box, cylinder, sphere) to model a link, and you assume equal distribution of mass, you can use wikipedia's list of moment of inertia tensors to compute the correct matrix. I've created a couple of xacro macros that do this automatically: common.xacro.

If you use CAD models to model a link, you can use Meshlab to compute the inertia tensor (again, assuming equal distribution of mass):

View -> Show Layer Dialog

Filters -> Quality Measures and Computations -> Compute Geometric Measures

Unfortunately, this usually gives a matrix that is rounded to all zeroes, so you might have to scale up your model, compute the inertia tensor, and divide the result by an appropriate factor.

Using a C++ Dependency in a ROS Node using vcpkg  
<https://ms-iot.github.io/ROSOnWindows/GettingStarted/UsingVCPKG.html>

gazebo\_msgs/GetJointProperties Service  
<http://docs.ros.org/en/jade/api/gazebo_msgs/html/srv/GetJointProperties.html>

Gazebosim 2.2.2. Getting joint forces; getforcetorque "Warning [ODEJoint.cc:1105] GetForceTorque: forgot to set <provide\_feedback>?"  
<https://stackoverflow.com/questions/22729445/gazebosim-2-2-2-getting-joint-forces-getforcetorque-warning-odejoint-cc1105>

FT Sensor shows zero values  
<https://answers.gazebosim.org/question/15465/ft-sensor-shows-zero-values/>

tutorial describes how to use a force/torque sensor on a joint. This sensor publishes force and torque readings to a topic.  
<https://github.com/osrf/gazebo_tutorials/blob/master/force_torque_sensor/tutorial.md>

        <Guard>

            <Name>ContactTable</Name>

            <SensorID>Ignored</SensorID>

            <SubField>Fz</SubField>

            <LimitType>DECREASE\_BEYOND\_LIMIT</LimitType>

            <LimitValue>.05</LimitValue>

            <RecheckTimeMicroSeconds>20</RecheckTimeMicroSeconds>

            <ForceTorqueJoint>fanuc\_joint\_6</ForceTorqueJoint>

        </Guard>

#include <termios.h>

inline int getch()

{

static struct termios oldt, newt;

tcgetattr( STDIN\_FILENO, &oldt); // save old settings

newt = oldt;

newt.c\_lflag &= ~(ICANON); // disable buffering

tcsetattr( STDIN\_FILENO, TCSANOW, &newt); // apply new settings

int c = getchar(); // read character (non-blocking)

tcsetattr( STDIN\_FILENO, TCSANOW, &oldt); // restore old settings

return c;

}

.h:

// Single step command (no console input)

bool bSingleStep;

std::string singlestep\_topic;

bool OnSingleStepCommand(std\_srvs::SetBool::Request &\_req,

std\_srvs::SetBool::Response &\_res);

ros::ServiceServer singlestep\_service;

Cpp:

void GazeboRosGuardedCmd::Load(physics::ModelPtr \_parent, sdf::ElementPtr \_sdf)

{

. . .

singlestep\_service=rosnode->advertiseService(singlestep\_topic,   
 &GazeboRosGuardedCmd::OnSingleStepCommand, this);

. . .

}

bool GazeboRosGuardedCmd::OnSingleStepCommand(std\_srvs::SetBool::Request &\_req,

std\_srvs::SetBool::Response &\_res)

{

// single to single step a service

bSingleStep=true;

return true;

}

USE:

c:\opt\ros\noetic\gztaskboard>rosservice call /lrmate/GazeboRosGuardedCmdDebug 1

success: False

message: ''

Not implemented but the getch worked?!?!?!?

Error appears to be related to numbering of joints. Not by names!

TRACE OF HIGH LEVEL GUARDED MOVE TO "TOUCH" table

Force Z

|  |
| --- |
| 7.54972e-05 |
| 0.00020815 |
| 0.000370869 |
| 0.000642896 |
| 0.00120387 |
| 0.00218225 |
| 0.00360353 |
| 0.0054333 |
| 0.00762992 |
| 0.0101537 |
| 0.0129765 |
| 0.016079 |
| 0.0194296 |
| 0.023009 |
| 0.0267921 |
| 0.0307605 |
| 0.0348941 |
| 0.0391712 |
| 0.0435767 |
| 0.0480943 |
| 0.0527021 |
| 0.0527021 |

To run the deadreckoning peg in the hole insertion exxperiements:

rosrun test\_gazebo\_package test\_gazebo\_package rosbreak:=0 deadReckoning:=1

THIS IS HOW CRPI NOW PERFORMS RASTER SCAN

peg\_in\_hole = asbly.RunAssemblyStep(counter++, curPose, poseMe, io); //initialization of new search

while (peg\_in\_hole == CANON\_RUNNING)

{

poseMe.z = peg\_z - 7.0f; //attempt to move peg down by 12 mm

ulapi\_mutex\_take(pm.grabmutex);

pm.robArm->MoveAttractor(poseMe);

pm.robArm->GetRobotPose(&curPose);

pm.robArm->GetRobotIO(&io);

pm.robArm->GetRobotAxes(robot\_axes);

ulapi\_mutex\_give(pm.grabmutex);

peg\_in\_hole = asbly.RunAssemblyStep(counter++, curPose, poseMe, io);

//if (peg\_in\_hole != CANON\_SUCCESS) {

// if (fabs(robot\_axes->axis[2]) > 2.0f) { //search taking a long time. Reset e1 and j4, otherwise elbow will drift towards table and fail

// ulapi\_mutex\_take(pm.grabmutex);

// robot\_axes->axis[2] = 0.0f;

// robot\_axes->axis[4] = 0.0f;

// pm.robArm->MoveToAxisTarget(\*robot\_axes);

// ulapi\_mutex\_give(pm.grabmutex);

// }

//}

}

if (peg\_in\_hole == CANON\_SUCCESS)

{

//poseMe.z = peg\_z - 15.0f; // insert peg 20 mm into hole

//ulapi\_mutex\_take(pm.grabmutex);

//if (pm.robArm->MoveAttractor(poseMe) == CANON\_SUCCESS) { curPose = poseMe; }

//ulapi\_mutex\_give(pm.grabmutex);

// Open Gripper

gripper\_pose(pm, eef\_open);

tInter = timer\_inter.timeElapsed();

results << tInter << ",";

}

else {

cout << "Insertion unsuccessful. Press enter to exit...";

cin.get();

return;

}

LIBRARY\_API CanonReturn Assembly::RunAssemblyStep (int counter, robotPose &robPose, robotPose &newPose, robotIO &ios)

{

bool state;

CanonReturn returnMe = CANON\_RUNNING;

vector<terminatorParams>::iterator tpi;

vector<assemblyParams>::iterator api;

int index;

//! known\_freq = get\_freq(10, new\_search)

state = true;

curPose\_ = robPose;

curIO\_ = ios;

if (termParams\_.size() < 1)

{

//! At least one termination condition must be defined

return CANON\_FAILURE;

}

if (newSearch)

{

initPose\_ = robPose;

for (tpi = termParams\_.begin(); tpi != termParams\_.end(); ++tpi)

{

if (tpi->tType == TERMINATOR\_TIMER)

{

timer\_.stopTimer();

timer\_.startTimer();

}

} // for (tpi ...)

newSearch = false;

} // if (newSearch)

newPose\_ = initPose\_;

for (api = assemblyParams\_.begin(); api != assemblyParams\_.end(); ++api)

{

state = state && applyOffset(counter, \*api);

}

if (state)

{

state &= testTerm (index);

}

else

{

newSearch = true;

return CANON\_FAILURE;

}

if (state)

{

newPose = newPose\_;

//! Termination condition not met, still searching

// state &= (robot\_->MoveTo(newPose\_) == CANON\_SUCCESS);

//counter += 1;

}

else

{

//! Search terminated (success, error, fail)

//! Type of terminator condition found at termParms\_.at(index).tType

returnMe = termParams\_.at(index).rType;

newSearch = true;

}

return returnMe;

} //RunAssembly

LIBRARY\_API bool Assembly::applyOffset (int counter, assemblyParams &ap)

{

robotPose deltas;

//! Initialize variables to 0

deltas.x = deltas.y = deltas.z = deltas.xrot = deltas.yrot = deltas.zrot = 0.0f;

int localCount;

double radius;

double degOffset;

double position;

double ratio;

double temp;

int wholeval;

switch (ap.sType)

{

case ASSEMBLY\_RANDOM:

if (ap.randWalk)

{

deltas.x = (curPose\_.x - initPose\_.x);

deltas.y = (curPose\_.y - initPose\_.y);

}

if (counter > 0) {

deltas.x += (ap.radius \* ((2500.0f - (double)(rand() % 5000)) / 2500.0f));

deltas.y += (ap.radius \* ((2500.0f - (double)(rand() % 5000)) / 2500.0f));

//cout << endl << "Random Offset " << counter << ": (" << deltas.x << ", " << deltas.y << ")";

}

break;

case ASSEMBLY\_SOBOL:

if (ap.randWalk)

{

deltas.x = (curPose\_.x - initPose\_.x);

deltas.y = (curPose\_.y - initPose\_.y);

}

if (counter == 0) { sobseq(&deltas.x, &deltas.y, 0); }

else {

sobseq(&deltas.x, &deltas.y, 1);

deltas.x = (ap.radius \* ((0.5f - deltas.x) / 0.5f));

deltas.y = (ap.radius \* ((0.5f - deltas.y) / 0.5f));

//cout << endl << "Sobol Offset " << counter << ": (" << deltas.x << ", " << deltas.y << ")";

}

break;

case ASSEMBLY\_SPIRAL:

case ASSEMBLY\_SPIRAL\_OPTIMAL:

localCount = counter % (int)(ap.thetaMax / ap.degOffsetDelta);

degOffset = localCount \* ap.degOffsetDelta;

radius = degOffset \* ap.radiusOffset;

//cout << endl << "Helix Iteration " << counter << " Degrees: " << degOffset << " Radius: " << radius;

deltas.x = radius \* cos(degOffset\*(3.141592654/180.0f));

deltas.y = radius \* sin(degOffset\*(3.141592654/180.0f));

break;

case ASSEMBLY\_SQ\_SPIRAL:

if (counter < ap.totalPoints)

{

deltas.x = sqs\_x[counter] \* ap.lengthStep;

deltas.y = sqs\_y[counter] \* ap.lengthStep;

//std::cout << "Square Spiral: total = " << ap.totalPoints << ", count = " << counter << ", x = " << deltas.x << ", y = " << deltas.y << std::endl;

}

break;

case ASSEMBLY\_RASTER:

temp = ap.totalLength / ap.lengthDelta;

localCount = counter % (int)(2.0f \* temp);

if (localCount > temp)

{

localCount = (int)(2.0f \* temp) - localCount;

}

position = localCount \* ap.lengthDelta;

ratio = position / (ap.width + ap.lengthStep);

temp = decimal(ratio);

wholeval = whole(ratio);

if (temp > ap.rasterRatio)

{

//! In the raster step

if (even(wholeval))

{

deltas.y = ap.width;

}

else

{

deltas.y = 0.0f;

}

deltas.x = (wholeval + ((temp - ap.rasterRatio) / (1.0 - ap.rasterRatio))) \* ap.lengthStep;

}

else

{

//! On the raster

if (even(wholeval))

{

deltas.y = (temp / ap.rasterRatio) \* ap.width;

}

else

{

deltas.y = (1.0f - (temp / ap.rasterRatio)) \* ap.width;

}

deltas.x = wholeval \* ap.lengthStep;

}

break;

case ASSEMBLY\_TILT:

//! TODO

break;

case ASSEMBLY\_ROTATION:

deltas.zrot = ap.magnitude \* sin((counter \* ap.degOffsetDelta)\*(3.141592654/180.0f));

break;

case ASSEMBLY\_CIRCLE:

localCount = counter % (int)(360.0f / ap.degOffsetDelta);

temp = localCount \* ap.degOffsetDelta;

deltas.x = ap.radius - (ap.radius \* cos(temp));

deltas.y = ap.radius \* sin(temp\*(3.141592654/180.0f));

break;

case ASSEMBLY\_HOP:

deltas.z = ap.magnitude \* sin((counter \* ap.degOffsetDelta)\*(3.141592654/180.0f));

break;

case ASSEMBLY\_LINEAR:

temp = ap.totalLength / ap.lengthStep;

localCount = counter % (int)(2.0f \* temp);

if (localCount > temp)

{

localCount = (int)(2.0f \* temp) - localCount;

}

deltas.x = localCount \* ap.xOffset;

deltas.y = localCount \* ap.yOffset;

deltas.z = localCount \* ap.zOffset;

break;

case ASSEMBLY\_CONST\_OFFSET:

//temp = ap.totalLength / ap.lengthStep;

if (counter < 2)

{

deltas.x = counter \* ap.xOffset;

deltas.y = counter \* ap.yOffset;

deltas.z = counter \* ap.zOffset;

}

else

{

deltas.x += ap.xOffset;

deltas.y += ap.yOffset;

deltas.z += ap.zOffset;

}

break;

default:

//! ERROR

return false;

break;

}

newPose\_ = newPose\_ + deltas;

//newPose\_.x += deltas.x;

//newPose\_.y += deltas.y;

return true;

} // applyOffset

LIBRARY\_API CanonReturn Assembly::AddTerminatorTimer (CanonReturn rType, double timeout)

{

terminatorParams tP;

tP.tType = TERMINATOR\_TIMER;

tP.rType = rType;

tP.endTime = fabs(timeout);

return AddTerminator(tP);

}

# roslaunch gazebo.launch

Successful launch of gazebo. Not sure why some processes are killed later on in trace….

c:\opt\ros\noetic\gzpegboard\bin\launch>roslaunch gazebo.launch

... logging to C:\Users\michalos\.ros\log\e9c4a4bd-b026-11ec-858f-c03ebaa186b4\roslaunch-COSMIC-20428.log

Checking log directory for disk usage. This may take a while.

Press Ctrl-C to interrupt

Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://127.0.0.1:60018/

SUMMARY

========

PARAMETERS

\* /gazebo/enable\_ros\_network: True

\* /joint\_state\_publisher2/source\_list: ['/lrmate/joint\_s...

\* /joint\_state\_publisher2/use\_gui: False

\* /lrmate/baselink: fanuc\_base\_link

\* /lrmate/robot\_description: <?xml version="1....

\* /lrmate/tiplink: fanuc\_link\_6

\* /robot\_description1: <?xml version="1....

\* /robot\_description2: <?xml version="1....

\* /robot\_description: <?xml version="1....

\* /rosdistro: noetic

\* /rosversion: 1.15.9

\* /use\_sim\_time: True

NODES

/

gazebo (gazebo\_ros/gzserver)

gazebo\_gui (gazebo\_ros/gzclient)

joint\_state\_publisher2 (joint\_state\_publisher/joint\_state\_publisher)

robot\_state\_publisher2 (robot\_state\_publisher/robot\_state\_publisher)

spawn\_gazebo\_model1 (gazebo\_ros/spawn\_model)

spawn\_gazebo\_model2 (gazebo\_ros/spawn\_model)

auto-starting new master

process[master]: started with pid [21312]

ROS\_MASTER\_URI=http://127.0.0.1:11311

setting /run\_id to e9c4a4bd-b026-11ec-858f-c03ebaa186b4

process[rosout-1]: started with pid [11928]

started core service [/rosout]

process[gazebo-2]: started with pid [26752]

process[gazebo\_gui-3]: started with pid [22264]

process[spawn\_gazebo\_model1-4]: started with pid [25480]

process[spawn\_gazebo\_model2-5]: started with pid [23624]

process[joint\_state\_publisher2-6]: started with pid [22508]

process[robot\_state\_publisher2-7]: started with pid [26968]

[INFO] [1648644181.700883, 0.000000]: Loading model XML from ros parameter robot\_description1

[INFO] [1648644181.701768, 0.000000]: Loading model XML from ros parameter robot\_description2

[INFO] [1648644181.711782, 0.000000]: Waiting for service /gazebo/spawn\_urdf\_model

[INFO] [1648644181.711782, 0.000000]: Waiting for service /gazebo/spawn\_urdf\_model

[rospack] Error: failed to rename cache file C:\Users\michalos\.ros\\.rospack\_cache.a10644 to C:\Users\michalos\.ros\rospack\_cache\_00000000003714325853: File exists

[ INFO] [1648644182.573944800]: Finished loading Gazebo ROS API Plugin.

[ INFO] [1648644182.578074900]: waitForService: Service [/gazebo\_gui/set\_physics\_properties] has not been advertised, waiting...

[ INFO] [1648644182.594963300]: Finished loading Gazebo ROS API Plugin.

[ INFO] [1648644182.597929400]: waitForService: Service [/gazebo/set\_physics\_properties] has not been advertised, waiting...

[ INFO] [1648644187.650310900]: waitForService: Service [/gazebo/set\_physics\_properties] is now available.

[INFO] [1648644187.672584, 0.000000]: Calling service /gazebo/spawn\_urdf\_model

[INFO] [1648644187.672711, 0.000000]: Calling service /gazebo/spawn\_urdf\_model

[INFO] [1648644209.437743, 0.001000]: Spawn status: SpawnModel: Successfully spawned entity

[ INFO] [1648644209.671764100, 0.001000000]: ft\_sensor plugin reporting wrench values to the frame [fanuc\_link\_6]

[ INFO] [1648644209.672012900, 0.001000000]: imu plugin missing <gaussianNoise>, defaults to 0.0

gzParallelGripperPlugin: Compiled Feb 15 2022 17:32:57

GzParallelGripperPlugin: Compiled Feb 15 2022 17:32:57

GzParallelGripperPlugin: Model plugin: lrmate

Joints:

fanuc\_base\_link-base

fanuc\_joint\_1

fanuc\_joint\_2

fanuc\_joint\_3

fanuc\_joint\_4

fanuc\_joint\_5

fanuc\_joint\_6

fanuc\_prism1

fanuc\_prism2

gzParallelGripperPlugin Model = lrmate

gzParallelGripperPlugin Namespace = fanuc\_lrmate200id

gzParallelGripperPlugin Joint1 = lrmate::fanuc\_prism1

gzParallelGripperPlugin Joint2 = lrmate::fanuc\_prism2

gzParallelGripperPlugin Link1 = fanuc\_finger\_1

gzParallelGripperPlugin Link2 = fanuc\_finger\_2

gzParallelGripperPlugin Service Control Topic = /fanuc\_lrmate200id/control

gzParallelGripperPlugin Debug = 1

gzParallelGripperPlugin Synchronous = 0

Subscribing contact manager to topic ~/lrmate/contacts

GazeboRosJointStatePublisher: Compiled Feb 15 2022 17:32:57

[ INFO] [1648644209.756665400, 0.001000000]: GazeboRosJointStatePublisher number joints "9"

[ INFO] [1648644209.756855600, 0.001000000]: GazeboRosJointStatePublisher is going to publish joint: fanuc\_joint\_1

[ INFO] [1648644209.757030900, 0.001000000]: GazeboRosJointStatePublisher is going to publish joint: fanuc\_joint\_2

[ INFO] [1648644209.757192600, 0.001000000]: GazeboRosJointStatePublisher is going to publish joint: fanuc\_joint\_3

[ INFO] [1648644209.757377000, 0.001000000]: GazeboRosJointStatePublisher is going to publish joint: fanuc\_joint\_4

[ INFO] [1648644209.758173200, 0.001000000]: GazeboRosJointStatePublisher is going to publish joint: fanuc\_joint\_5

[ INFO] [1648644209.758903100, 0.001000000]: GazeboRosJointStatePublisher is going to publish joint: fanuc\_joint\_6

[ INFO] [1648644209.759543000, 0.001000000]: Starting GazeboRosJointStatePublisher Plugin (ns = lrmate/)!, parent name: lrmate

GazeboRosGuardedCmd: Compiled Feb 15 2022 17:32:57

[ INFO] [1648644209.794705400, 0.001000000]: GazeboRosGuardedCmd number joints "9"

[ INFO] [1648644209.794862800, 0.001000000]: GazeboRosGuardedCmd is going to publish joint: fanuc\_joint\_1

[ INFO] [1648644209.794968800, 0.001000000]: GazeboRosGuardedCmd is going to publish joint: fanuc\_joint\_2

[ INFO] [1648644209.795085100, 0.001000000]: GazeboRosGuardedCmd is going to publish joint: fanuc\_joint\_3

[ INFO] [1648644209.795178400, 0.001000000]: GazeboRosGuardedCmd is going to publish joint: fanuc\_joint\_4

[ INFO] [1648644209.795535300, 0.001000000]: GazeboRosGuardedCmd is going to publish joint: fanuc\_joint\_5

[ INFO] [1648644209.795720300, 0.001000000]: GazeboRosGuardedCmd is going to publish joint: fanuc\_joint\_6

[ INFO] [1648644209.796921300, 0.001000000]: Starting GazeboRosGuardedCmd Plugin (ns = lrmate/)!, parent name: lrmate

[INFO] [1648644210.160566, 0.065000]: Spawn status: SpawnModel: Successfully spawned entity

[ INFO] [1648644210.173359400, 0.065000000]: Physics dynamic reconfigure ready.

[spawn\_gazebo\_model2-5] process has finished cleanly

log file: C:\Users\michalos\.ros\log\e9c4a4bd-b026-11ec-858f-c03ebaa186b4\spawn\_gazebo\_model2-5\*.log

[spawn\_gazebo\_model1-4] process has finished cleanly

log file: C:\Users\michalos\.ros\log\e9c4a4bd-b026-11ec-858f-c03ebaa186b4\spawn\_gazebo\_model1-4\*.log

^CTerminate batch job (Y/N)? [robot\_state\_publisher2-7] killing on exit

[joint\_state\_publisher2-6] killing on exit

[gazebo\_gui-3] killing on exit

[gazebo-2] killing on exit

SUCCESS: The process with PID 22508 (child process of PID 20428) has been terminated.

SUCCESS: The process with PID 26968 (child process of PID 20428) has been terminated.

ERROR: The process with PID 19364 (child process of PID 22264) could not be terminated.

Reason: There is no running instance of the task.

SUCCESS: The process with PID 22264 (child process of PID 20428) has been terminated.

ERROR: The process with PID 13740 (child process of PID 26752) could not be terminated.

Reason: There is no running instance of the task.

SUCCESS: The process with PID 26752 (child process of PID 20428) has been terminated.

[rosout-1] killing on exit

SUCCESS: The process with PID 11928 (child process of PID 20428) has been terminated.

[master] killing on exit

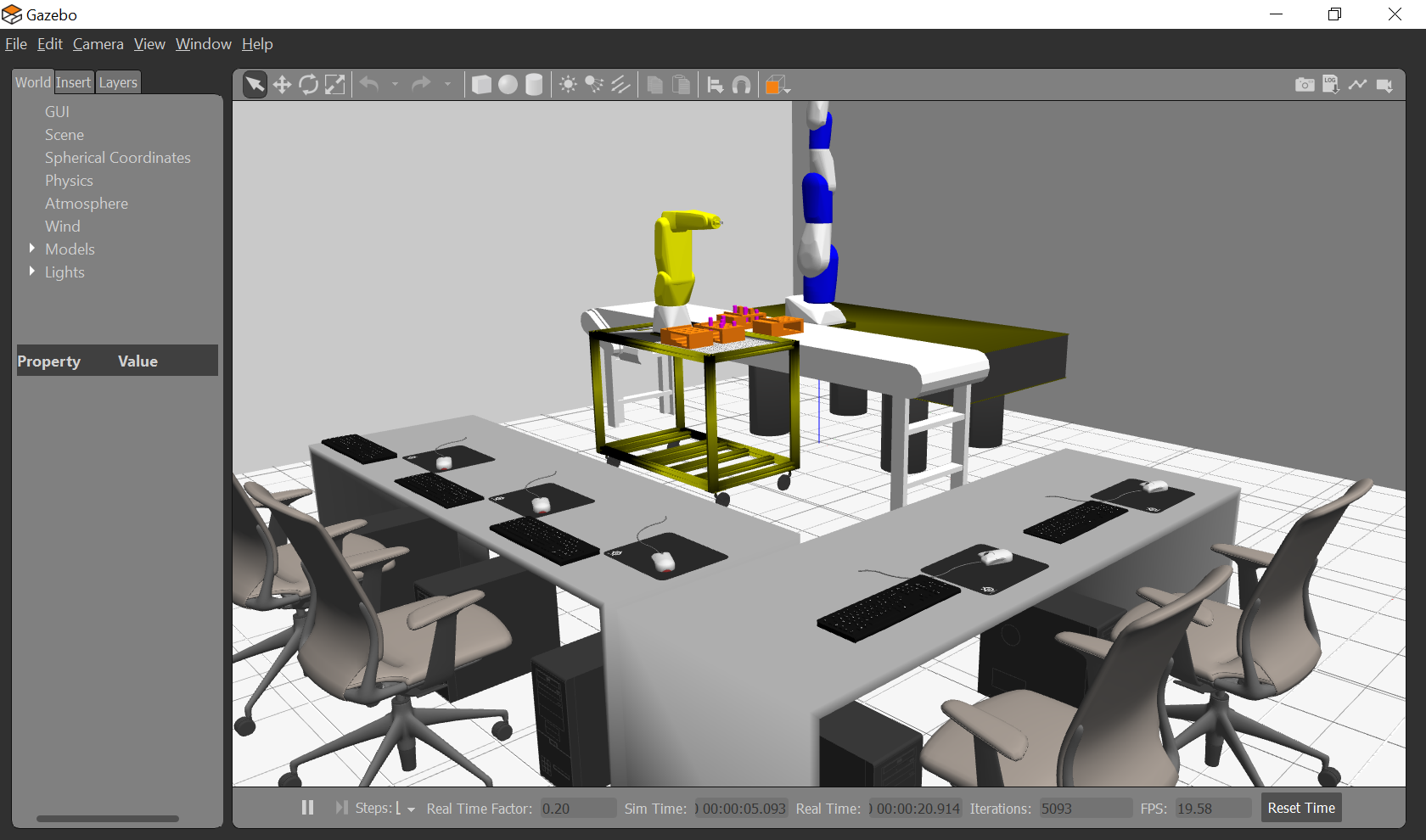
shutting down processing monitor...

... shutting down processing monitor complete

done

^CThe system cannot open the device or file specified.

c:\opt\ros\noetic\gzpegboard\bin\launch>



c:\opt\ros\noetic\crpi\_test>rosservice list

/fanuc\_lrmate200id/control

/gazebo/apply\_body\_wrench

/gazebo/apply\_joint\_effort

/gazebo/clear\_body\_wrenches

/gazebo/clear\_joint\_forces

/gazebo/delete\_light

/gazebo/delete\_model

/gazebo/get\_joint\_properties

/gazebo/get\_light\_properties

/gazebo/get\_link\_properties

/gazebo/get\_link\_state

/gazebo/get\_loggers

/gazebo/get\_model\_properties

/gazebo/get\_model\_state

/gazebo/get\_physics\_properties

/gazebo/get\_world\_properties

/gazebo/pause\_physics

/gazebo/reset\_simulation

/gazebo/reset\_world

/gazebo/set\_joint\_properties

/gazebo/set\_light\_properties

/gazebo/set\_link\_properties

/gazebo/set\_link\_state

/gazebo/set\_logger\_level

/gazebo/set\_model\_configuration

/gazebo/set\_model\_state

/gazebo/set\_parameters

/gazebo/set\_physics\_properties

/gazebo/spawn\_sdf\_model

/gazebo/spawn\_urdf\_model

/gazebo/unpause\_physics

/gazebo\_gui/get\_loggers

/gazebo\_gui/set\_logger\_level

/joint\_state\_publisher2/get\_loggers

/joint\_state\_publisher2/set\_logger\_level

/lrmate/GazeboRosGuardedCmdDebug

/lrmate/gazebo\_ros\_guarded\_move

/lrmate/gazebo\_ros\_joint\_state\_publisher\_service

/robot\_state\_publisher2/get\_loggers

/robot\_state\_publisher2/set\_logger\_level

/rosout/get\_loggers

/rosout/set\_logger\_level

c:\opt\ros\noetic\crpi\_test>

Peg1 Bottom World Coord 0.27, -1.10, 0.93,

Peg1 Top World coord = 0.27, -1.10, 1.08,

Peg1 actual Top in World Coord = 0.27, -1.10, 1.03,

Peg1 Robot Coord Pose = 0.439, 0.035, 0.146,0.000,0.707,0.000,0.707

Peg1 Robot Coord Retract Pose 0.439, 0.035, 0.166,0.000,0.707,0.000,0.707

TopHole1 World Coord Pose = 0.290, -1.355, 1.080,0.000,0.707,0.000,0.707

TopHole1 Robot Coord Pose = 0.459, -0.215, 0.146,0.000,0.707,0.000,0.707

TopHole1 Robot Coord Approch = 0.459, -0.215, 0.307,0.000,0.707,0.000,0.707

Holetouch Robot Coord Near = 0.459, -0.215, 0.247,0.000,0.707,0.000,0.707

[DEBUG] [1648647832.573743300, 60.016000000]: gzModelStatesCallback

TopHole1 World Coord Pose = 0.290, -1.355, 1.080,0.000,0.707,0.000,0.707

TopHole1 Robot Coord Pose = 0.459, -0.215, 0.146,0.000,0.707,0.000,0.707

TopHole1 Robot Coord Approch = 0.459, -0.215, 0.307,0.000,0.707,0.000,0.707

Holetouch Robot Coord Near = 0.459, -0.215, 0.247,0.000,0.707,0.000,0.707

[DEBUG] [1648648140.192300900, 120.714000000]: gzModelStatesCallback

TopHole1 Robot Coord Approch = 0.459, -0.215, 0.307,0.000,0.707,0.000,0.707

Holetouch Robot Coord Near = 0.459, -0.215, 0.247,0.000,0.707,0.000,0.707

GuardedFinal Robot Coord = 0.459, -0.215, 0.044,0.000,0.707,0.000,0.707

GuardedFinal Robot Coord Approach = 0.459, -0.215, **0.255**,0.000,0.707,0.000,0.707

Current Robot Pos = 0.459, -0.215, **0.229**,0.000,0.707,0.000,0.707

KDL chain =

\* fanuc\_joint\_1 RotAxis [ 0, 0, 1] [ 0, 0, 0.33]

\* fanuc\_joint\_2 RotAxis [ 0, 1, 0] [ 0.05, 0, 0]

\* fanuc\_joint\_3 RotAxis [ 0,-1, 0] [ 0, 0, 0.33]

\* fanuc\_joint\_4 RotAxis [-1, 0, 0] [ 0, 0, 0.035]

\* fanuc\_joint\_5 RotAxis [ 0,-1, 0] [ 0.335, 0, 0]

\* fanuc\_joint\_6 RotAxis [-1, 0, 0] [0.08, 0, 0]

[DEBUG] [1648684229.922821000]: Finish CrpiGazebo::init()

"C:\\opt\\ros\\noetic\\crpi\_test\\src\\test\_gazebo\_package\\logs\\-19-50-34--30.3.2022.dat"

Bend = 0.000, 0.000, 0.000,-0.000,-1.571,0.000

Peg1Bottom World Coord 0.27, -1.10, 0.93,

Peg1Top World coord = 0.27, -1.10, 1.08,

Peg1actual Top in World Coord = 0.27, -1.10, 1.03,

Peg1Robot Coord Pose = 0.439, 0.035, 0.146,0.000,0.707,0.000,0.707

Peg1Robot Coord Retract Pose 0.439, 0.035, 0.166,0.000,0.707,0.000,0.707

TopHole1 World Coord Pose = 0.290, -1.355, 1.080,0.000,0.707,0.000,0.707

TopHole1 Robot Coord Pose = 0.459, -0.215, 0.146,0.000,0.707,0.000,0.707

TopHole1 Robot Coord Approch = 0.459, -0.215, 0.307,0.000,0.707,0.000,0.707

Holetouch Robot Coord Near = 0.459, -0.215, 0.247,0.000,0.707,0.000,0.707

Length of Peg = 0.1016

TopHole1 Robot Coord Pose = 0.459, -0.215, 0.146,0.000,0.707,0.000,0.707

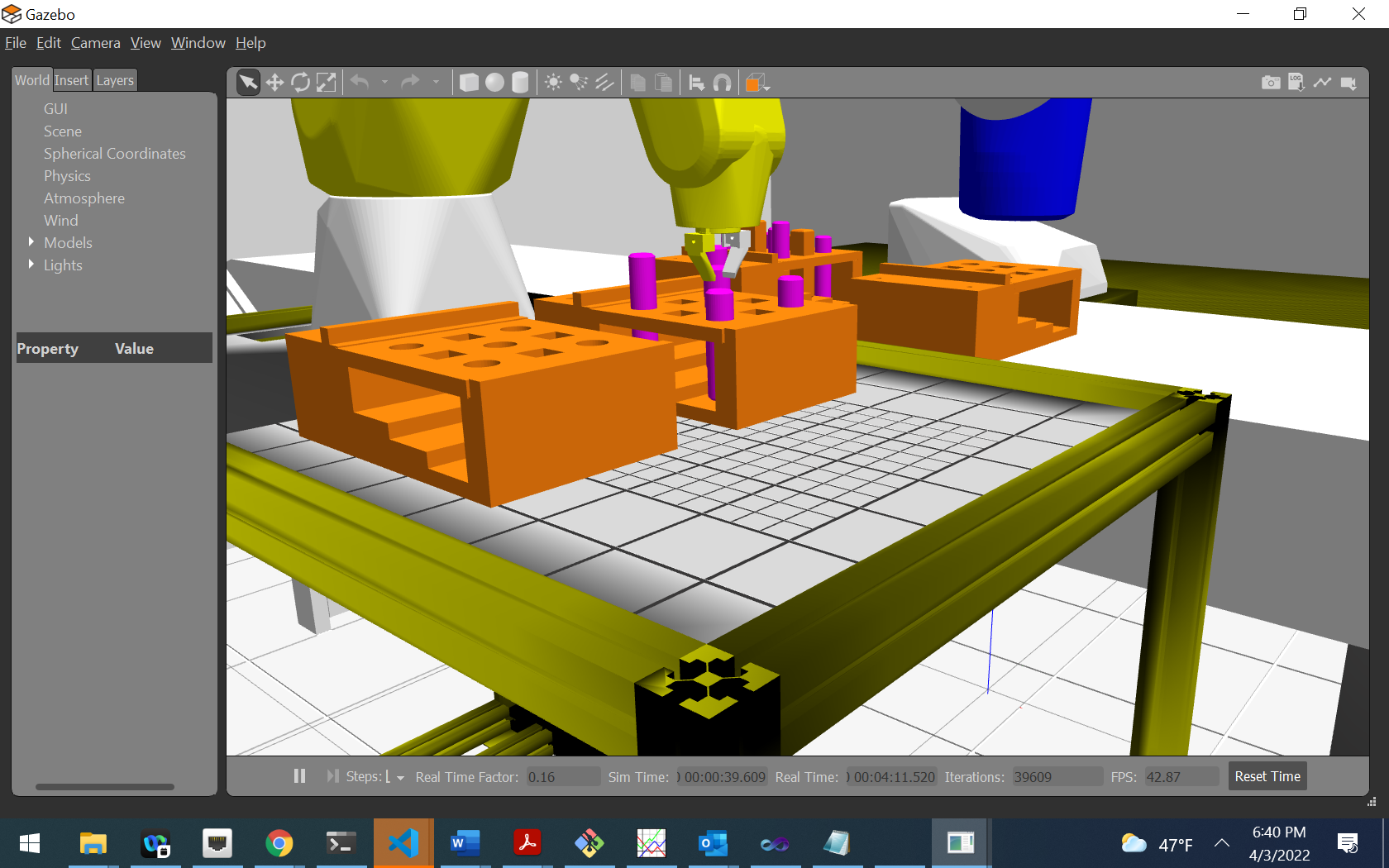
TopHole1 Robot Coord Approch = 0.459, -0.215, 0.307,0.000,0.707,0.000,0.707

Holetouch Robot Coord Near = 0.459, -0.215, 0.247,0.000,0.707,0.000,0.707

GuardedFinal Robot Coord = 0.459, -0.215, 0.044,0.000,0.707,0.000,0.707

GuardedFinal Robot Coord Approach = 0.459, -0.215, 0.207,0.000,0.707,0.000,0.707

Current Robot Pos = 0.459, -0.215, 0.229,0.000,0.707,0.000,0.707



Insert Peg into empty hole

Empty hole1 at 0.27, -1.10, 0.93,

Hole1 Pose = 0.270, -1.105, 1.080,0.000,0.707,0.000,0.707

Hole Robot Coord Pose = 0.439, 0.035, 0.146,0.000,0.707,0.000,0.707

Hole Robot Coord Retract Pose 0.439, 0.035, 0.287,0.000,0.707,0.000,0.707

Peg1 Bottom World Coord 0.27, -1.10, 0.93,

Peg1 Top World coord = 0.27, -1.10, 1.08,

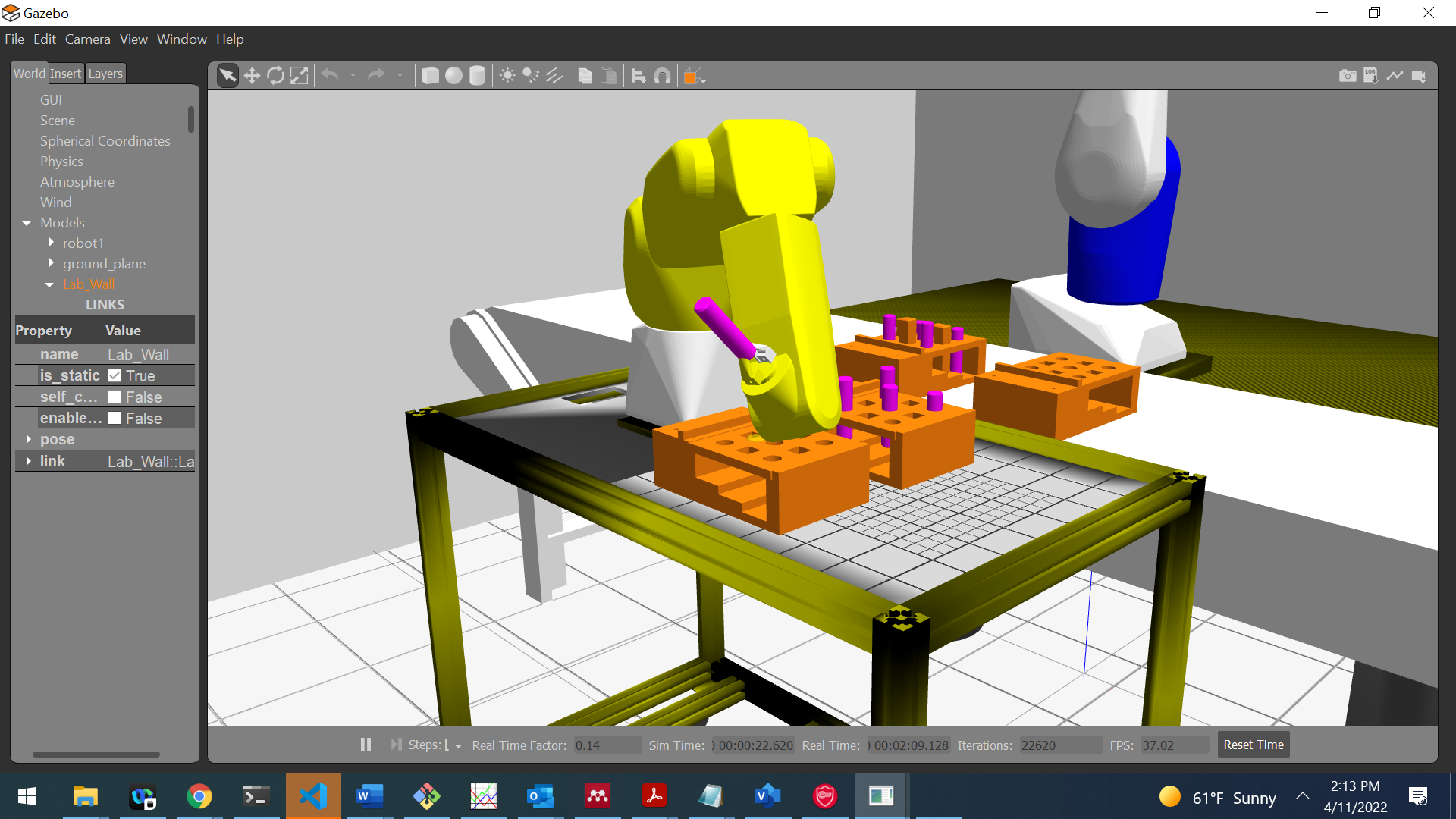
**Peg1 actual Top in World Coord = 0.27, -1.10, 1.03,**

Peg1 Robot Coord Pose = 0.439, 0.035, 0.146,0.000,0.707,0.000,0.707

Peg1 Robot Coord Retract Pose 0.439, 0.035, 0.166,0.000,0.707,0.000,0.707

True? Missed grasping peg by a bunch.

Add trial number; why did it work at all. Did not make sense but watched graviety pull it down to its actual position!!!



A great deal of research of the robotic peg-in hole assembly operation has  
evolved over the past decades. In the case of a chamfered peg-hole, it is normal to use a  
simple wrist such as the Remote Centre Compliance (RCC), which is fitted between the  
robot arm and the end effector to accommodate misalignments between the peg and the  
hole during the engagement stage. Otherwise jamming could occur and cause damage to  
the assembly robot and or the mating parts.

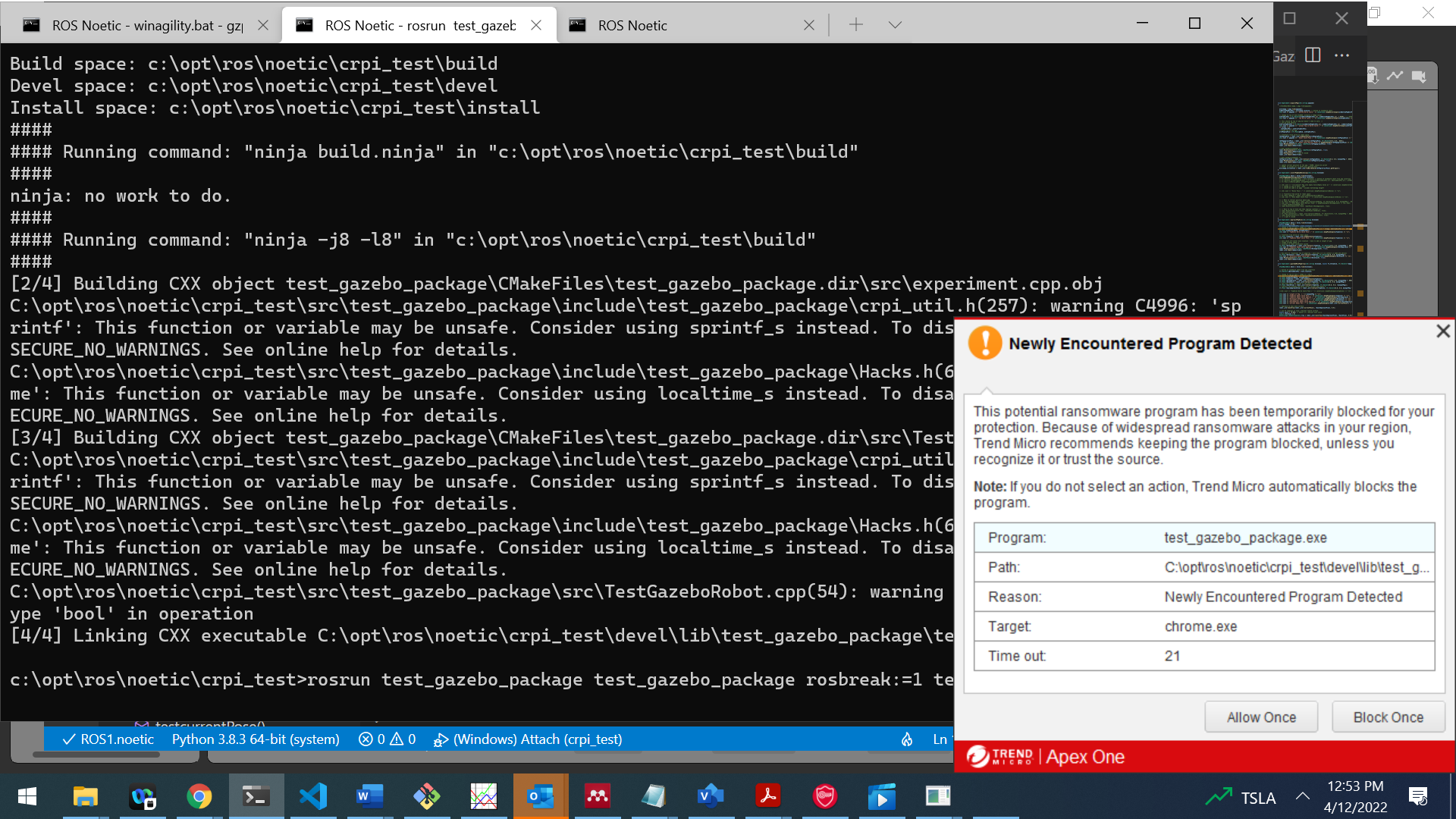
Given the relative ease to modify the pegboard CAD design, it is possible to enhance the shape and forms of the pegs and holes.

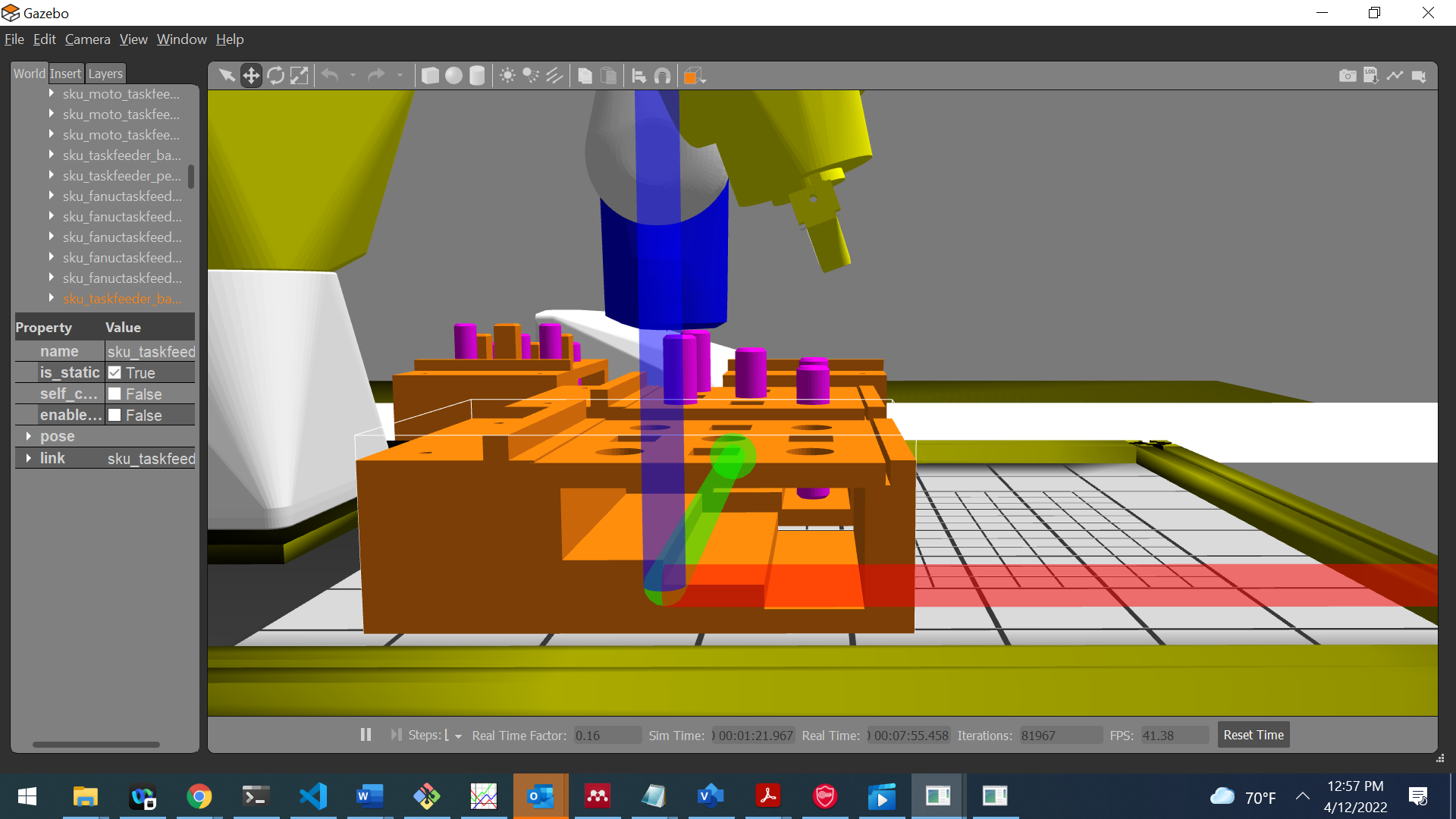
One critical problem that arises during component assembly is that small error in relative lateral position or angular position can produce large reaction forces. These forces can prevent successful completion of the assembly (jamming) and can cause damage to the parts and the robot.

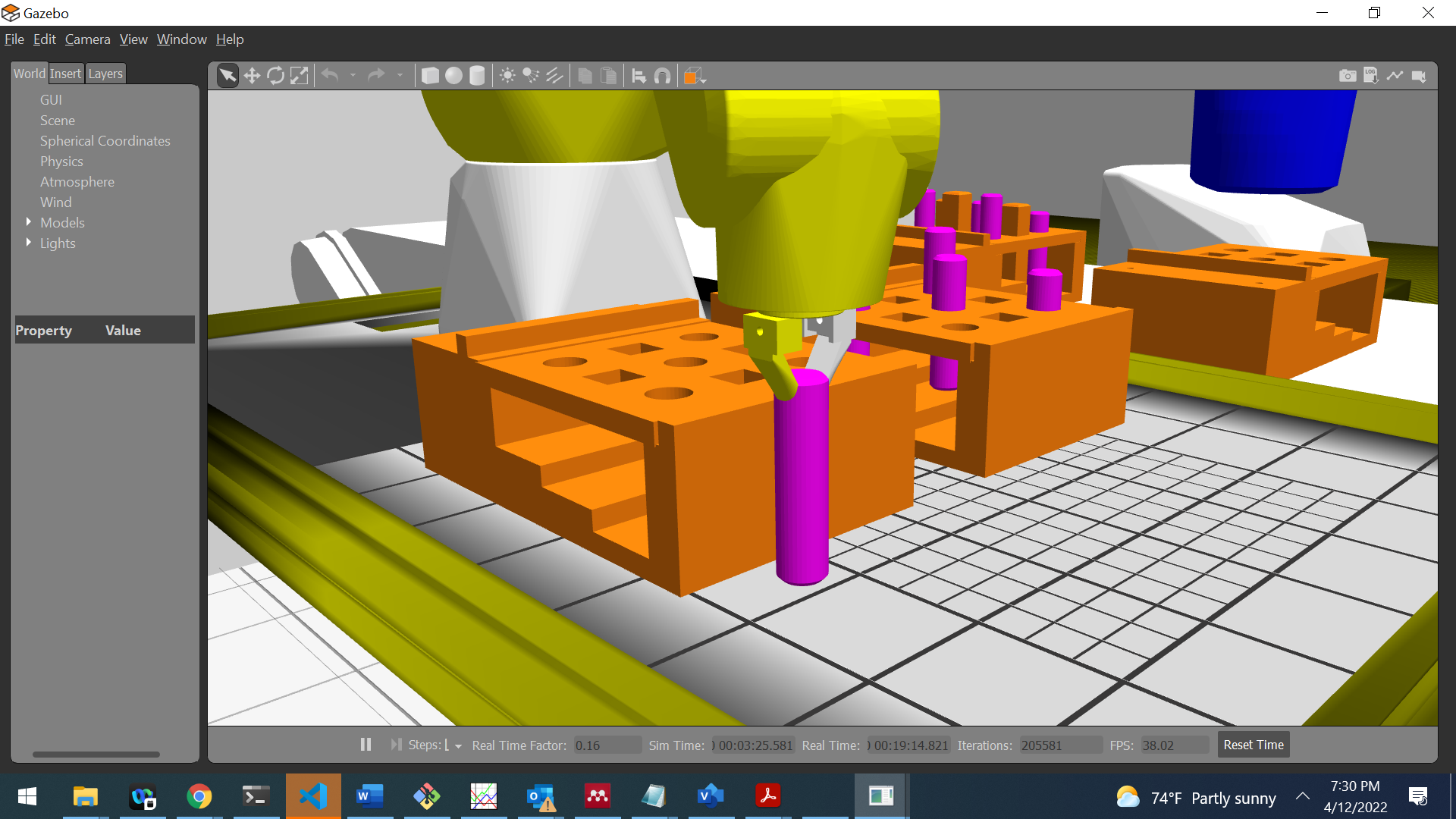
The objective is to list remedial robot simulated active compliant actions that use simulated F/T sensing to resolve angular/lateral misalignment between peg-in-hole insertion.

In order for a simulated peg-in-hole to be valuable, it must understand and replicate these real-world uncertainties. At worst, a simulation may not understand why errors occur, but should be able to inject a reasonable approximation.

For Gazebo, we used the Open Dynamics Engine (ODE) physics engine a now antiquated but groundbreaking to model the physics of the current simulated environment.







Move Peg to empty hole location

Empty hole1 at 0.27, -1.35, 0.93,

Top Hole Wrld Pose = 0.270, -1.355, 1.080,0.000,0.707,0.000,0.707

With base 0.270, -1.355, 1.080,0.000,0.707,0.000,0.707

Wout base 0.439, -0.215, 0.146,0.000,0.707,0.000,0.707

Top Hole Robot Coord Pose = 0.439, -0.215, 0.146,0.000,0.707,0.000,0.707

With grasped peg 0.439, -0.215, 0.146,0.000,0.707,0.000,0.707

Wout grasped peg 0.540, -0.215, 0.146,0.000,0.707,0.000,0.707

Top Hole Robot Coord Pose - Peg = 0.540, -0.215, 0.146,0.000,0.707,0.000,0.707

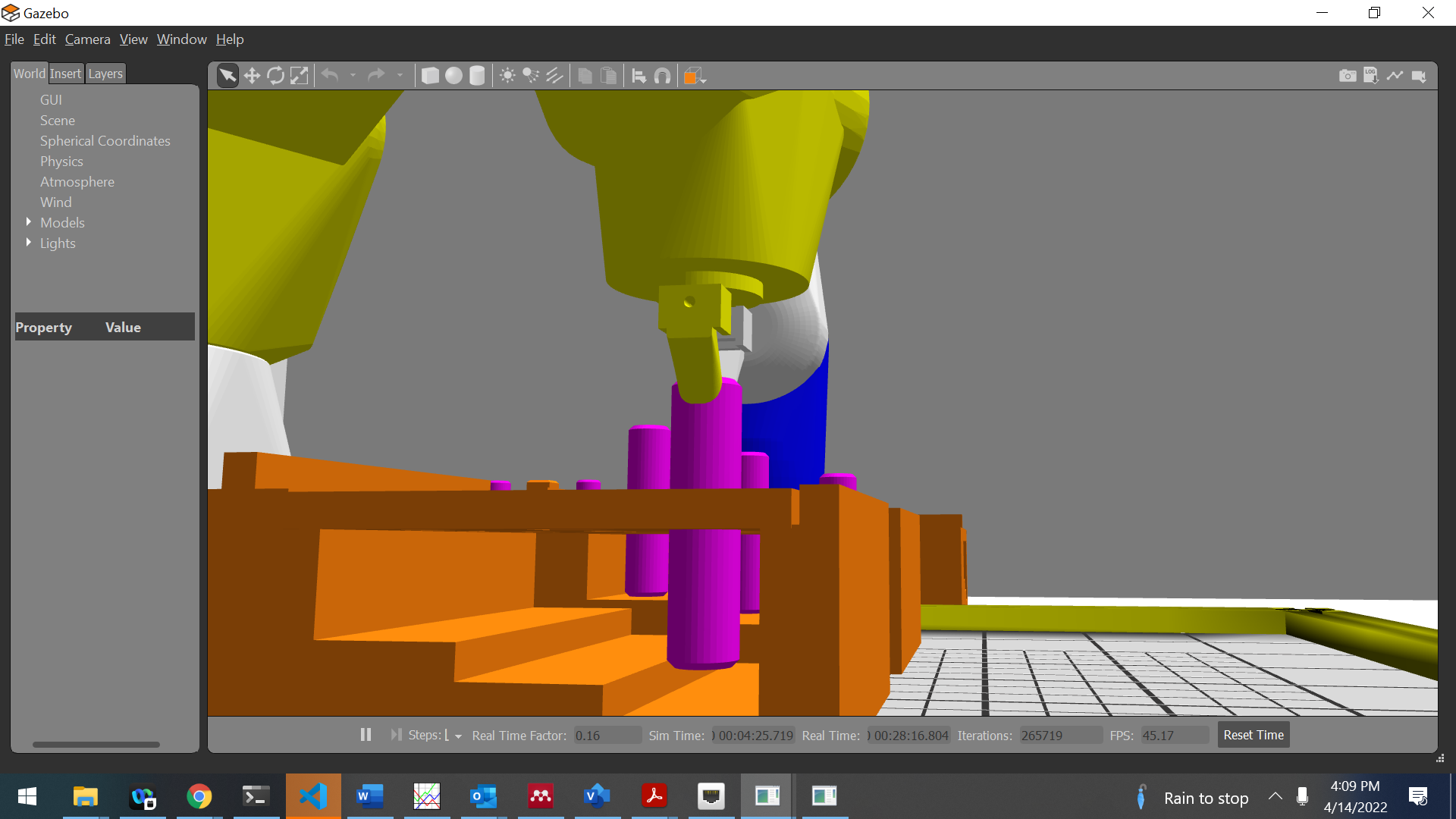
Top Hole Robot Coord Retract Pose 0.540, -0.215, 0.186,0.000,0.707,0.000,0.707

With gripper 0.540, -0.215, 0.186,0.000,0.707,0.000,0.707

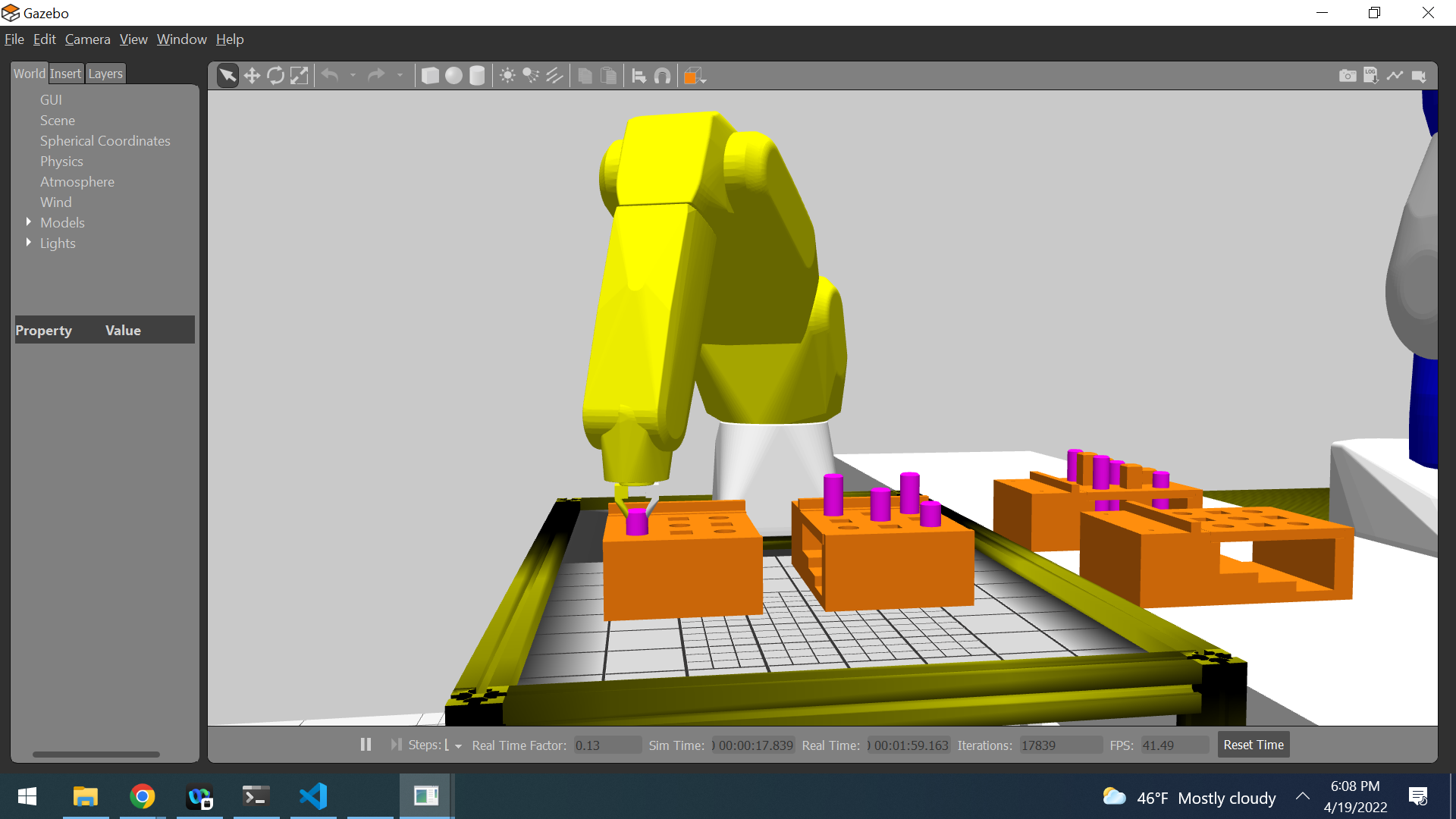
Wout gripper 0.540, -0.215, 0.168,0.000,0.707,0.000,0.707

CRobotImpl::calcMoveTo DestPose = 0.540, -0.215, 0.168,0.000,0.707,0.000,0.707

CRobotImpl::calcMoveTo CurPose = 0.439, 0.035, 0.269,-0.001,0.707,0.001,0.707



One experiment is to alter the orientation of the peg once inserted in the hole to understand the F/T reading simulated sensor reading under a known circumstance.



for (double t = 1.; t < 10.; t = t + 1.)

bend.setRPY(0., fromDegree(-.1 \* t), 0.);

0 0.439, -0.215, 0.146,0.000,1.571,0.000,-5.3745e-05,-0.00254024,0.00601487

1 0.439, -0.215, 0.146,0.000,1.569,0.000,-0.000115773,-0.00224053,0.00691573

2 0.439, -0.215, 0.146,0.000,1.567,0.000,-0.000165908,-0.00178461,0.00708796

3 0.439, -0.215, 0.146,0.000,1.566,0.000,-0.000201314,-0.000533305,0.00214078

4 0.439, -0.215, 0.146,0.000,1.564,0.000,-0.000239094,0.00170386,-0.0110352

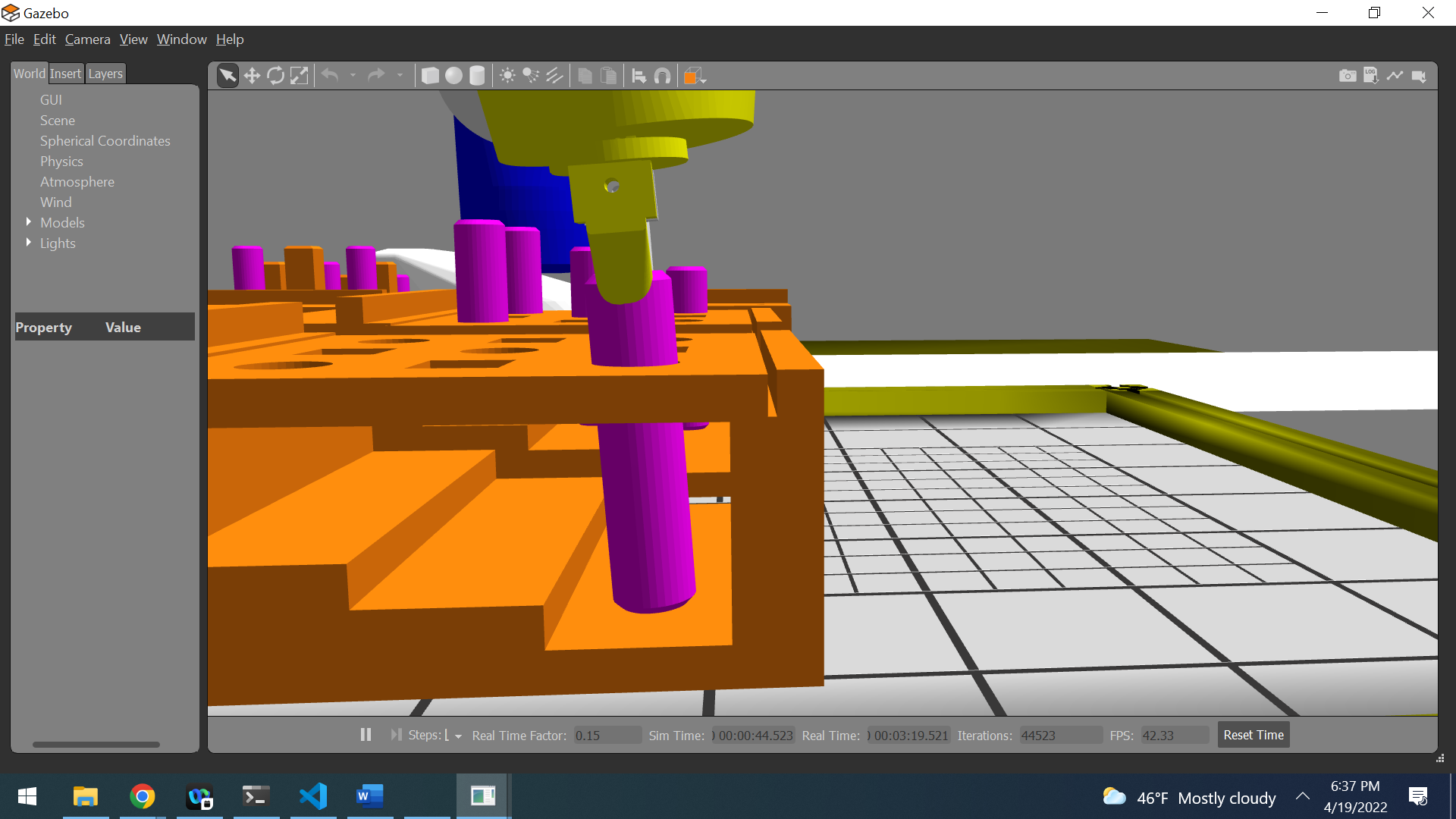
5 0.439, -0.215, 0.146,0.000,1.562,0.000,-0.000284053,0.0041277,-0.028085

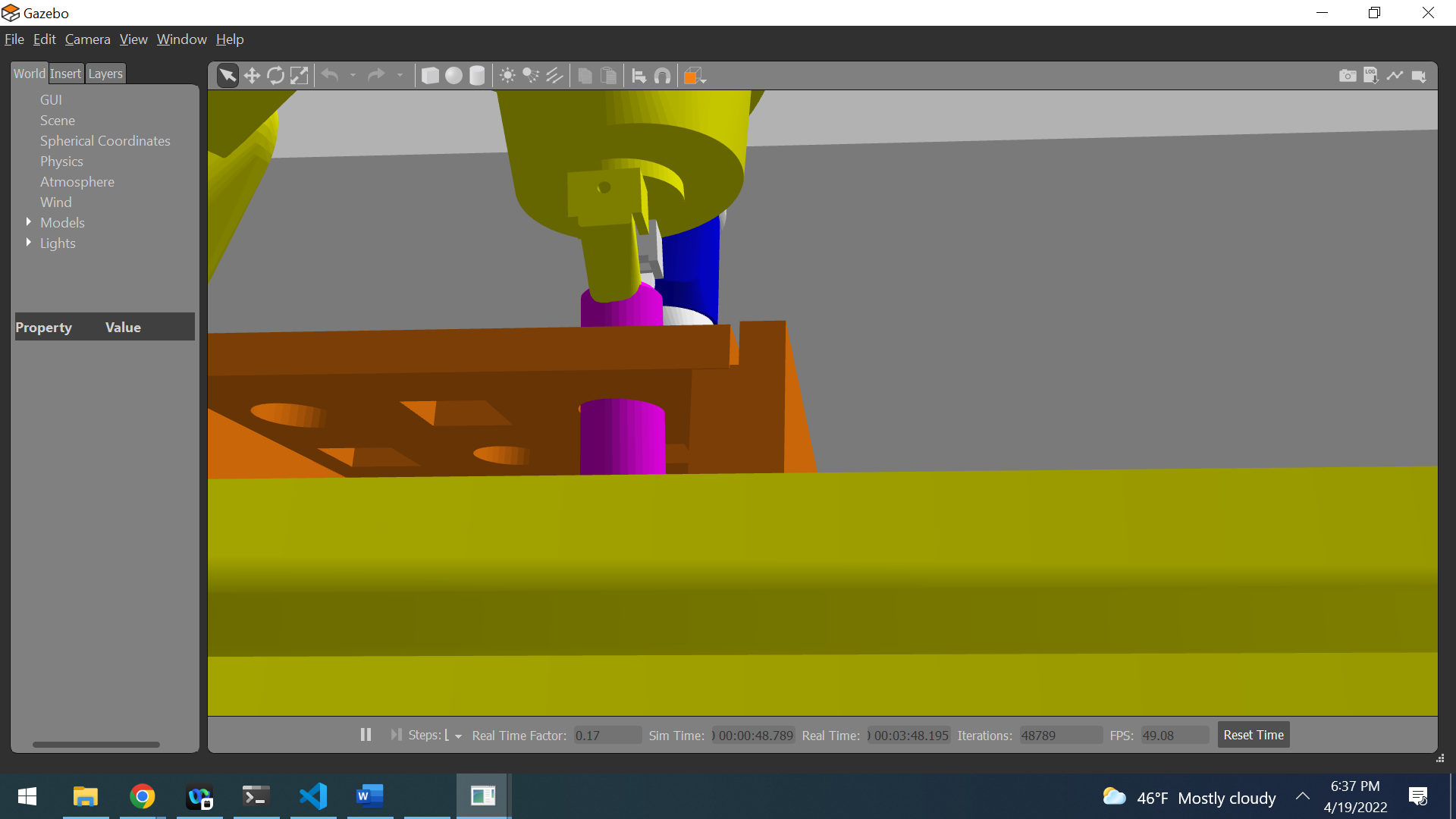
6 0.439, -0.215, 0.146,0.000,1.560,0.000,-0.000369726,0.00764045,-0.0555572

7 0.439, -0.215, 0.146,0.000,1.559,0.000,-0.000532538,0.0121364,-0.0962944

8 0.439, -0.215, 0.146,0.000,1.557,0.000,-0.000755106,0.0155667,-0.14063

9 0.439, -0.215, 0.146,0.000,1.555,0.000,-0.00103485,0.0177976,-0.184929





for (double t = 1.; t < 10.; t = t + 1.)

bend.setRPY(0., fromDegree(-1. \* t), 0.);

0 0.439, -0.215, 0.146,0.000,90.000,0.000,-0.000121897,-0.00333398,0.0075073

1 0.439, -0.215, 0.146,0.000,89.000,0.000,-0.000192223,-0.00171826,0.00297609

2 0.439, -0.215, 0.146,0.000,88.000,0.000,-0.00113776,0.00737135,-0.123064

3 0.439, -0.215, 0.146,0.000,87.000,0.000,-0.00238374,0.0174757,-0.349324

4 0.439, -0.215, 0.146,0.000,86.000,0.000,-0.00396434,0.0304635,-0.684774

5 0.439, -0.215, 0.146,0.000,85.000,0.000,0.0407051,0.0257664,-1.0269

6 0.439, -0.215, 0.146,0.000,84.000,0.000,0.173193,0.0580799,-1.38646

7 0.439, -0.215, 0.146,0.000,83.000,0.000,0.268039,0.113669,-1.74386

8 0.439, -0.215, 0.146,0.000,82.000,0.000,0.310501,0.17135,-2.12471

9 0.439, -0.215, 0.146,0.000,81.000,0.000,0.323334,0.250779,-2.49835

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Onyx | Ubuntu 18.04 | Dell Laptop E6530 |  | 855209 |

Since the clearance between the peg and the hole is very tight, straightening the end effector of the robot would not be enough to prevent jamming. The compliance of the robot can be used. Since there is a constant downward force, as described in section 5.2.3, tilting the peg in any axis could release the jamming for a brief time. Repeating this step could hopefully lead to a complete insertion. Rotating the tool tip for the purpose of prevent jamming is not done in the same way as moving the tool tip in a predefined trajectory. The x, y and z-coordinates needs to be static while the orientation has to be manipulated based on the forces recorded on the peg.