(5) Pluginlib

⇒ The pluginlib package provides tools for writting and dynamically loading plugins using the ROS build infrastructure.

⇒ To work, these tools require plugin providers to register their plugins in the package.xml of their package.

⇒ pluginlib is a C++ library for loading and unloading plugins from within a ROS package.

Plugin ⟶ dynamically loaddble Class (Shared object)

Pluginlib ⟶ With pluginlib, one does not have to explicitly link their application against the library containing the classes.

⟶ Instead pluginlib can open a library containing the exported classes at any point without the applications having any prior awareness of the library or the header file containing the class defination.

ting

⇒ Plugins are useful for extending/modifying application behavior without modifying the application source code.

★ Writing and Using a Simple Plugin

Create a Base Class

⇒ Now we'll Create a base class from which all our plugins will inherit.

abstract class

Create the Plugin

Registering the Plugin

```
#include <pluginlib/class_list_macros.h>
```

⇒ Here, we include the pluginlib macros that allows us to register class as plugins.

```
PLUGINLIB_EXPORT_CLASS(Polygon_plugin::Triangle,
                       Polygon_base::RegularPolygon)
```

Class to export

base class

Building the Plugin Library

```
include_directories (include)
add_libraries (polygon_plugins src/polygon_plugins
```

Making the plugin Available to the ROS Toolchain

Polygon_plugins.xml

```
<library path="lib/libpolygon_plugins">
  < class type ="polygon_plugins::Triangle"
          base_class_type ="polygon_base::RegularPolygon">
    <description> This is a triangle plugin </description>
</library>
```

Exporting plugin

Package.xml

```
<export>
   <Package_name Plugin="${Prefix}/polygon_plugin.xml"/>
```

⇒ To check if plugin is Setup:

```
rospack plugin --attrib=plugin packagename
```

→ This will return path to Polygon_plugin.xml

U

#

#

Ru

Ad

La

Using a plugin

```
#include <pluginlib/class_loader.h>
#include <Package_name/Polygon_base.h>
int main (int argc , char** argv)
{
 pluginlib::ClassLoader<polygon_base::RegularPolygon>
  Poly_loader("package_name","Polygon_base::RegularPolygon");

 try
 {
 boost:: shared_ptr <Polygon_base :: RegularPolygon>
  triangle  = Poly_loader. createInstance(
 "Polygon_plugins:: Triangle");
 }
 Catch (Pluginlib:: PluginException& ex)
  {
  --- Failed to load Plugin ---
  }
  return 0;
  }
```

Running the Code

```
add_executable (Polygon_loader src/Polygon_loader.cpp)
target_link_libraries (Polygon_loader ${catkin_LIBRARIES})
```