Developing Software in Carmen

6.189/2.994/16.401

October 5th, 2005

3 Topics

- Writing New Modules
- Writing New Message Classes
- Writing Test Harnesses

State Machines

- You are used to thinking of programs like this:
  ```
  ....
  start_manipulator();
  brick_num = next_closest_brick();
  set_planner_goal(brick_num);
  start_planner();
  ....
  ```
- Instead, think of this:
  ![State Machine Diagram]

The Anatomy of a Module

- Initialize IPC Connection
- Create handlers
- Set subscriptions
- Dispatch
  - Handle a message
  - Do some work
  - Update the internal state
  - Go back to dispatching
package RSS;
import Carmen.*;
public class BrickFinder implements CameraHandler {
  public void handleCamera (CameraMessage message) {
    System.out.println("Please tell me the brick colour: ");
    String colourName = System.in.readline();
    int bricks[] = processImage(message, colourName);
    BrickMessage msg = new BrickMessage(bricks);
    msg.publish();
  }
  public static void main (String args[])
  {
    Robot.initialize("BrickFinder", argv[0]);
    BrickFinder finder = new BrickFinder();
    Robot.subscribeCamera(finder);
    Robot.dispatch();
  }
}
The Anatomy of a Message

Providing a constructor ensures that the module using your message does not have to remember to do things like fit in field lengths, or the timestamp and hostname.

Remember you have to define a separate interface class that handles your message.

The internal handler ensures that the handler that is called when a BrickMessage is received matches the handler type.

We IPC define'd the message in the subscribe. Why do we need to define it here as well? Is this a good idea? Could we do it better?
Issues

- Concurrency
  - What if a handler needs to run for a long time?
  - I warn you now: threads will **not** help you.
- Graphical Displays
  - How to display the internal state of a program for debugging?
  - Use a timer, and call Robot.listen() for a few ms.
- Initialization
  - How to make sure every module agrees on the size of the robot?
  - Use the param_daemon to store/get parameters.

Some useful IPC methods

```
public static int disconnect () ;
public static boolean isConnected () ;
public static boolean isModuleConnected (String moduleName);
public static int defineMsg (String msgName, String formatString);
public static boolean isMsgDefined (String msgName);
public static int listen (long timeoutMSecs);
public static int listenClear (long timeoutMSecs);
public static Object queryResponseData
(String msgName, Object data, Class responseClass, long timeoutMSecs);
public static int respondData (MSG_INSTANCE msgInstance,
String msgName, Object data)
```

Unit Testing

- How can things go wrong between the design review and implementation?
  - We forget what we promised messages would look like.
  - We forget what messages we promised to send.
  - We forget what messages we promised to subscribe to.
  - We forget what we said was reasonable for a message to contain.
  - We forget what we said was the right order of things (i.e., what our state machine is supposed to look like).
  - We forget some important edges cases.
  - We forget what happens to us if we use degrees.

Unit Testing for Message Sending

- “If a program feature lacks an automated test, we assume it doesn’t work. This seems much safer than the prevailing assumption, that if a developer assures us a program feature works, then it works now and forever.” [www.junit.org](http://www.junit.org)
- We can build unit tests to automate some simple coordination tests
- Can we automate everything?
- Many projects dictate that unit testing is part of the build process. Can we do this? Is it a good idea?
The Anatomy of a Unit Test

• Note that messages do not implement standard interfaces.
  • By convention, you should, however, implement a constructor, a message subscription method and a publication method.
  • You could also support query/response.
• Messages do, however, require a separate interface file to ensure type-safe message handling.

A Unit Test for the BrickFinder

package RSS;
import junit.*;
import Carmen.*;
public class BrickFinderTest implements BrickHandler {
  public void testFIRSTTESTNAME() {
    // Test code...
  }
  public void testSECONDTESTNAME() {
    // Test code...
  }
  public static void main(String args[]) {
    TestSuite suite = new TestSuite(BrickFinderTest.class);
    junit.textui.TestRunner.run(suite);
  }
}

A Unit Test for the BrickFinder

• Unit tests are not an exact science.
• What can go wrong with our test during these three statements?
• What about during just this statement alone?

Additional Carmen Modules

• Laser
• Laser simulator (no vision, manipulator simulator)
• Localize (laser-based, map-based localization)
• Navigator (laser-based, map-based numerical potential field motion planner) and navigator_panel (map gui)
• Map builder (vasco)
• Documentation exists at http://www.cs.cmu.edu/~carmen

Some gotchas in using these modules
• Many of them do not have Java libraries yet. (That’s ok – you’re replacing these modules anyhow. But they may help in bootstrapping your particular section.)
• You must have a map. (There is one checked into rssII/data, called “longwood.cmf”.)
• The simulator subsumes orc, laser and robot_central. You do not need to run any of these with the simulator.
• You do need to run localize, navigator and navigator_panel to position the robot graphically.

Additional Carmen Modules

• Laser
• Laser simulator (no vision, manipulator simulator)
• Localize (laser-based, map-based localization)
• Navigator (laser-based, map-based numerical potential field motion planner) and navigator_panel (map gui)
• Map builder (vasco)
• Documentation exists at http://www.cs.cmu.edu/~carmen

Some gotchas in using these modules
• Many of them do not have Java libraries yet. (That’s ok – you’re replacing these modules anyhow. But they may help in bootstrapping your particular section.)
• You must have a map. (There is one checked into rssII/data, called “longwood.cmf”.)
• The simulator subsumes orc, laser and robot_central. You do not need to run any of these with the simulator.
• You do need to run localize, navigator and navigator_panel to position the robot graphically.