Day 1: Overview

Assignment 1: Prep

Goal: Download, setup, and update the course Virtual Machine.

Task: I have provided a virtual machine (VM) based on Ubuntu 20.04 LTS for you to use for the rest of this course. If you prefer to use your own computer, you can provision it using the Course VM/computer setup instructions.

I've put together a video that walks through this process. You may want to watch it on Youtube or download it to watch locally.

The VM can be downloaded on Google Drive. The VM can be used with any virtual machine software that can open OVA files. If you don't already have a preference, VirtualBox (https://www.virtualbox.org/) is a decent cross-platform choice. VMWare Workstation Player 16 also seems to work fine (though you'll need to click "Retry" when it gives you a warning about the OVA file).

Once you have downloaded and installed your VM (File->Import Appliance on VirtualBox), boot it, and log in with:

- username: course
- password: dragon

Obviously, if you intend to put anything personal on this VM, please change the password.

The VM has a copy of Goby cloned to `/home/course/goby3`, with the branch `3.0-course` (https://github.com/GobySoft/goby3/tree/3.0-course) checked out. This is similar to the `3.0` main branch, but will allow for any course-related fixes to be quickly rolled out over the week.

Please pull any changes since the VM was generated and build them:

```
cd ~/goby3
git pull
./build.sh
```

The `goby3-course` repo contains all the code, launch files, and documentation specific to this course. Similarly, please update this:

```
cd ~/goby3-course
git pull
./build.sh
```

A side note on building

`goby` and `goby3-course` both use CMake to build. The `build.sh` script is a thin wrapper around running `cmake` to configure, followed by `cmake --build` to build (using Make or Ninja). This script uses the `build` directory to do an out-of-source build which avoids mixing the generated code with the version source code.

If you prefer to invoke CMake by hand, or use an IDE such as VSCode instead of using `./build.sh`, please do so.

Getting help

If you run into trouble with the homework, your help options include:

- posting to the appropriate course Slack channel (e.g. `day1-homework`).
- joining one of the Zoom office hours (please post to the Slack channel first so that we can minimize duplicate questions).
- if neither is possible (e.g. you are doing this homework after the course week has completed), please submit your question as an issue on `goby3-course` to https://github.com/GobySoft/goby3-course/issues. In this case, your question need not be a problem with the code, but can simply be a question that you would like clarified.

Assignment 2: Run the Alpha mission

2.a.

Goal: Run the Alpha mission and visualize the vehicle position as a topside operator in OpenCPN and Google Earth.

Task: To get familiar with running Goby code, please try running the Alpha mission as we did today in class:

(order of topside versus usv launch doesn't matter):

```
cd ~/goby3-course/launch/alpha
./usv.launch
```

and in a new terminal window:

```
cd ~/goby3-course/launch/alpha
./topside.launch
```

Then you can open Google Earth using `google-earth-pro` from the command line, and OpenCPN with `opencpn`.

Ensure that you can see the vehicle on both viewers, and then click "Deploy" from pMarineViewer.

Note: pMarineViewer is showing the on-board position of the vehicle, whereas OpenCPN and Google Earth are showing the operator's view of the vehicle position after the (simulated) satellite comms. The pMarineViewer view can be thought of as simulator's "cheater mode" since you have access to a visualization you would not have out on the real water. While this can be useful while simulating and understanding a system, it is critical to separate this from the real viewer by the time you move to in-water tests.

When you are done with the mission, you can type `<CTRL>+C` (SIGINT) in the terminal windows where you ran `./usv.launch` and `./topside.launch`. `goby_launch` will intercept the SIGINT and use `goby_terminate` to gracefully stop the goby processes, and SIGTERM to kill everything else (MOOS).

A side note on the Google Earth interface for Ocean Vehicles (GEOV)

GEOV can be configured using the website (within the VM) at http://127.0.0.1/geov/. Click on "profile manager" and you will be brought to the active profile. From here you can change many settings on how your vehicles are displayed.

You may have trouble running Google Earth in the VM. In that case:

- Install Google Earth Pro on your host computer instead from https://www.google.com/earth/versions/#earth-pro.
- Connect using a web browser on your host computer to the GEOV server using http://xxx.xxx.xxx.xxx/geov where xxx.xxx.xxx.xxx is the IP address of your VM.
- Click "download geov kml" and install that file in Google Earth.
- Finally, check the box next to "bind to this machine's ip" on the "profile manager" page and click "apply".

2.b.

Goal: Understand how to configure a Goby application to provide debugging information to the terminal (via `screen`) and to a log file.

Task: To get familiar with debugging a Goby application, relaunch `usv.launch` and `topside.launch` and try attaching a screen session to the `goby3_course_topside_manager`:

```
screen -r topside.goby3_course_topside_manager
```

You will most likely just see a blank terminal window and a blinking cursor. This is due to the fact that glog is configured in the default `QUIET` mode. Stop the topside (`<CTRL>+C`) and edit `topside.launch` to add a `-v` to the command line flags, i.e.

```
goby3_course_topside_manager topside_config/manager.pb.cfg -v
```

Now if you relaunch `topside` and attach the screen session you will see the messages that are tagged with `glog.is_verbose()` in the code (in this case the navigation sentences coming in from the USV):

```
goby3_course_topside_manager [2481-Apr-30 17:49:10.141710]: Received DCCL nav: vehicle: 1 time:
1613604175 x: 99.9 y: -262.3 z: 0 speed_over_ground: 0 heading: 0 type: USV
goby3_course_topside_manager [2481-Apr-30 17:49:10.160100]: ^^ Converts to frontseat NodeStatus:
time: 16136041750 name: "USV_1" type: USV global_fix { lat: 41.59000026842395 lon:
-70.709999548495745 depth: -0 } local_fix { x: 99.9 y: -262.3 z: 0 } pose { heading: 0 } speed {
over_ground: 0 }
goby3_course_topside_manager [2481-Apr-30 17:49:30.134870]: Received DCCL nav: vehicle: 1 time:
1613604177 x: 99.9 y: -262.3 z: 0 speed_over_ground: 0 heading: 0 type: USV
goby3_course_topside_manager [2481-Apr-30 17:49:30.139730]: ^^ Converts to frontseat NodeStatus:
time: 16136041770 name: "USV_1" type: USV global_fix { lat: 41.59000026842395 lon:
-70.709999548495745 depth: -0 } local_fix { x: 99.9 y: -262.3 z: 0 } pose { heading: 0 } speed {
over_ground: 0 }
```

Similarly, you can have any process write a log file with this debug information. Open `topside_config/manager.pb.cfg` in your favorite editor (`gedit`, `emacs`, `vi`), and add the following `glog_config` anywhere within the existing `app {}` block:

```
app {
...
    glog_config {
        file_log {
            file_dir: "../../logs/topside"
            verbosity: VERBOSE
        }
    }
...
}
```

Save the file, and relaunch the `topside`. Now if you inspect the log file, you should see similar output to that which you saw when attached to the `screen` session:

```
tail -f ~/goby3-course/logs/topside/goby3_course_topside_manager_latest.txt
```

If this file doesn't exist for you, ensure the process launched successfully. If not, see "A side note on failed launches", below.

A side note on glog output

Generating the glog output can be fairly CPU intensive, especially on higher debug settings (DEBUG2, DEBUG3) and high time warp values. If you find your CPU is getting taxed by a particular process, try lowering the debug log output (e.g. to WARN or QUIET) and see if that helps.

If your log directory gets cluttered, there's a script, `goby3-course/scripts/clean_logs.sh`, which will delete all the logs in `goby3-course/logs`.

A side note on failed launches

If you make a syntax error in the configuration, or your code doesn't launch for some other reason, `goby_launch` will abort the entire launch, e.g.:

```
[toby@aubergine ~/opensource/goby3-course/launch/alpha] ./topside.launch
Launched (screen: 102792, child:  102794) gobyd topside_config/gobyd.pb.cfg -vvv
Launched (screen: 102808, child:  102810) goby3_course_topside_manager topside_config/manager.pb.cfg
Launched (screen: 102830, child:  102832) goby_opencpn_interface
topside_config/goby_opencpn_interface.pb.cfg
Launched (screen: 102852, child:  102854) goby_liaison topside_config/liaison.pb.cfg
Launched (screen: 102868, child:  102876) goby_geov_interface
topside_config/goby_geov_interface.pb.cfg -vvv
Failed to launch gobyd topside_config/gobyd.pb.cfg -vvv

Cleaning up...
[20210217T233329]: All processes exited
```

In this case, you can try running the errant process manually, e.g.:

```
gobyd topside_config/gobyd.pb.cfg -vvv
```

given the error above. In many cases, the output will illuminate the problem (typo in `interprocess`):

```
gobyd [2021-Feb-17 23:34:46.911984]: (Warning): [line  16]nterprocess {
...
gobyd [2021-Feb-17 23:34:46.912662]: (Warning): line: 16 col: 12 Message type
"goby.apps.zeromq.protobuf.GobyDaemonConfig" has no field named "nterprocess".
```

Sometimes, however, the problem only exhibits after all the rest of the code is running. In this case, comment out (`#`) the offending process in the launch file, run it, and then try running the problematic process in a separate command window. You may also find the log output to be useful, if you enable `file_log` in `glog_config` and inspect the contents of `goby3-course/logs`.

You can also add `-L` to the `goby_launch` which will log the screen output to `/tmp/goby_launch_screen*` and this will persist after the `goby_launch` session ends.

Assignment 3: Run the Trail mission

If you have not already done so, terminate (`<CTRL>+C`) the launch scripts from the previous mission. You can leave Google Earth and OpenCPN running.

Goal: Run the Trail mission and visualize the AUV and USV positions as a topside operator in OpenCPN and Google Earth.

First, take a look at the all.launch file:

```
cd ~/goby3-course/launch/trail
cat all.launch
```

You will note the `[env=]` syntax before some of the launch lines. This sets one or more Environmental Variables before that line is executed.

You will also notice that this launch file consists solely of `goby_launch` directives, each of which launches the code required for a single vehicle. By creating a hierarchy of `goby_launch` files, we can start multiple vehicles with a single command, but easily fall back to individually launching each vehicle as needed for debugging.

Now, run the mission:

```
cd ~/goby3-course/launch/trail
./all.launch
```

Again, pMarineViewer shows the "cheater" view of all the vehicles in realtime, and OpenCPN and Google Earth show the realistic operator's view.

Debug logging directories

The trail mission is configured to write debug logs to `goby3-course/logs` in the appropriate directories. The log file verbosity (normally `QUIET`, so no output), can be set in the `launch/trail/config/usv.pb.cfg.py`, `launch/trail/config/auv.pb.cfg.py`, and `launch/trail/config/topside.pb.cfg.py`:

Change `log_file_verbosity` to the desired value (WARN, VERBOSE, DEBUG1, DEBUG2, or DEBUG3):

```
app_common = config.template_substitute(templates_dir+'/_app.pb.cfg.in',
                                 app=common.app,
                                 tty_verbosity = 'QUIET',
                                 log_file_dir = debug_log_file_dir,
                                 log_file_verbosity = 'QUIET',
                                 warp=common.sim.warp,
                                 lat_origin=common.origin.lat(),
                    lon_origin=common.origin.lon())
```

A side note on CPU loading.

Take a look at `top` and see what your load average looks like. If it's more than the number of cores you've given your VM, you're overloading your CPU, which can lead to unexpected effects in the simulation.

If that's the case, try these fixes:

- If your host has more CPU cores available than the VM (Machine->Settings->System in VirtualBox), try shutting down the VM and increase the CPU core count).
- Try running Google Earth in the host, rather than the VM (see "A side note on the Google Earth interface for Ocean Vehicles (GEOV)", above).
- Reduce the number of AUVs in your simulation and/or reduce the simulation "warp" speed (the factor faster than realtime that the simulation runs at). This can be done automatically using the `generate_all_launch.sh` script in `~/goby3-course/launch/trail` (you can `cat` this script if you want to see how to change these values manually in `all.launch` and `~/goby3-course/launch/trail/config/common/sim.py`). Usage: `./generate_all_launch.sh [number_of_AUVs] [warp_factor]`. The defaults for the Trail mission are set to 4 AUVs and warp 5, so try reducing the number of AUVs first, e.g., `./generate_all_launch.sh 2 5`.

Wrap up

Well done! You've successfully set up your VM and got the code running that we'll be digging into in more depth in the next few days. Again, if you ran into any trouble getting your setup going, please reach out on Slack and we'll give you a hand.