Marine Robotics

Unmanned Autonomous Vehicles in Air Land and Sea

Politecnico Milano – June 2016

Alfredo Martins
INESC TEC / ISEP
Portugal
alfredo.martins@inesctec.pt

Tools

isep Instituto Superior de Engenharia do Porto

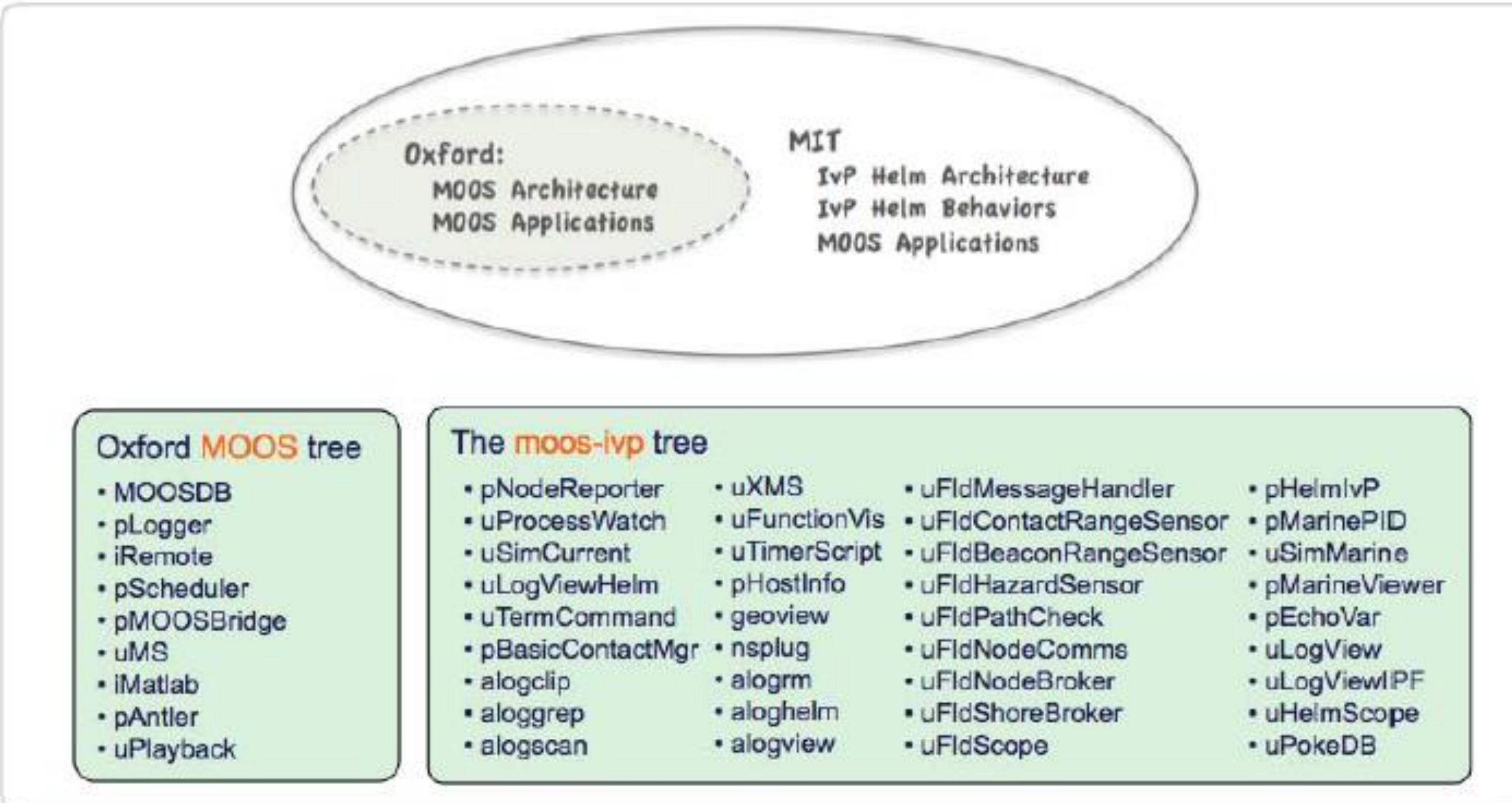
MOOS

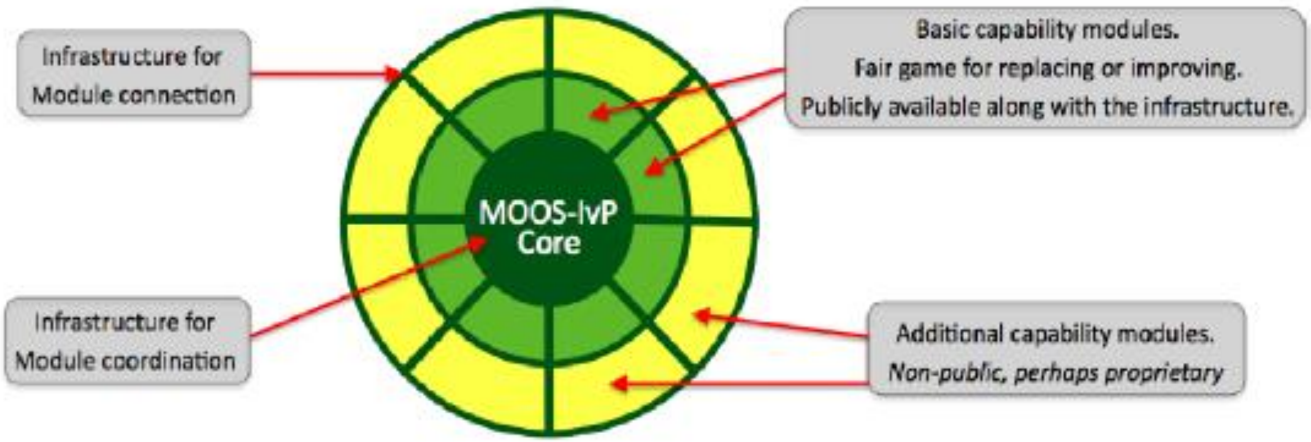
Mission Oriented Operating Suite

MOOS

- Developed by P. Newman at MIT for marine applications
- Centralized communications (MOOSDB)
- 2 repositories (Oxford and MIT)
- Several applications for marine missions (viewers, simulation, mission control, vehicle control, navigation)
- Ready to use in marine applications
- With relative large userbase in the marine robotics community
- Object oriented (C++) classes

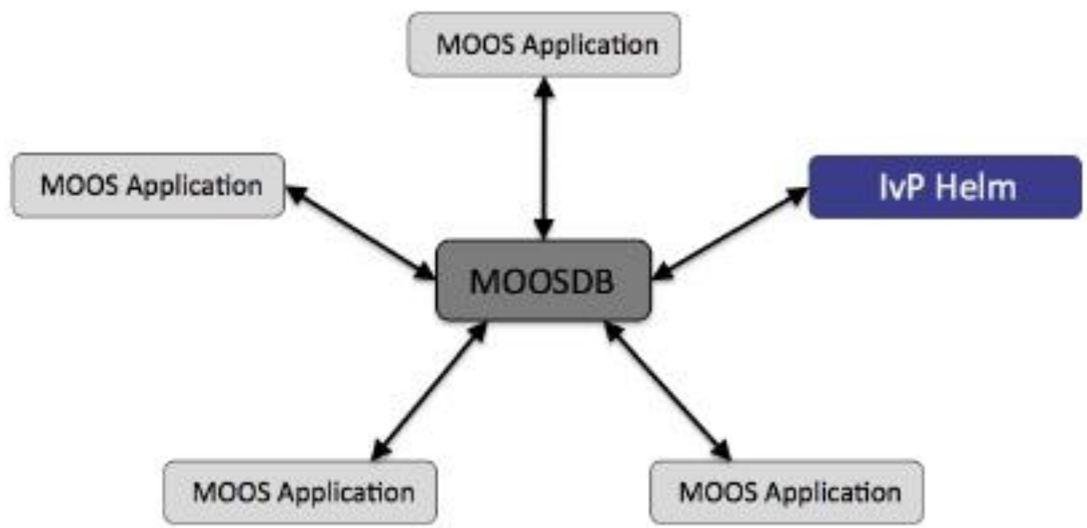
MOOS and MOOS IvP



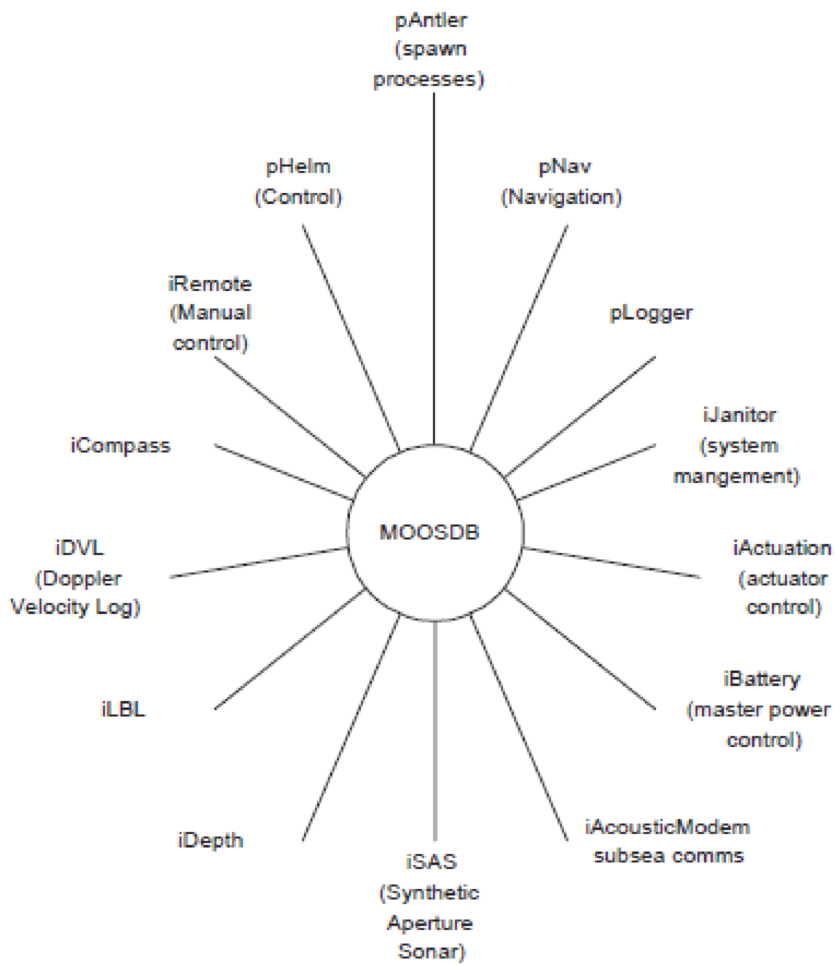

Infrastructure for
Module connection
Basic capability modules.
Fair game for replacing or improving.
Publicly available along with the infrastructure.
MOOS-IvP
Core
Infrastructure for
Module coordination
Additional capability modules.
Non-public, perhaps proprietary

MOOS

- MOOS community
  - collection of MOOS applications running on a single machine with a separate process ID
  - Independent processes, possibly running at different frequencies
  - Communications through a MOOSDB (publish-subscriber)



MOOS example

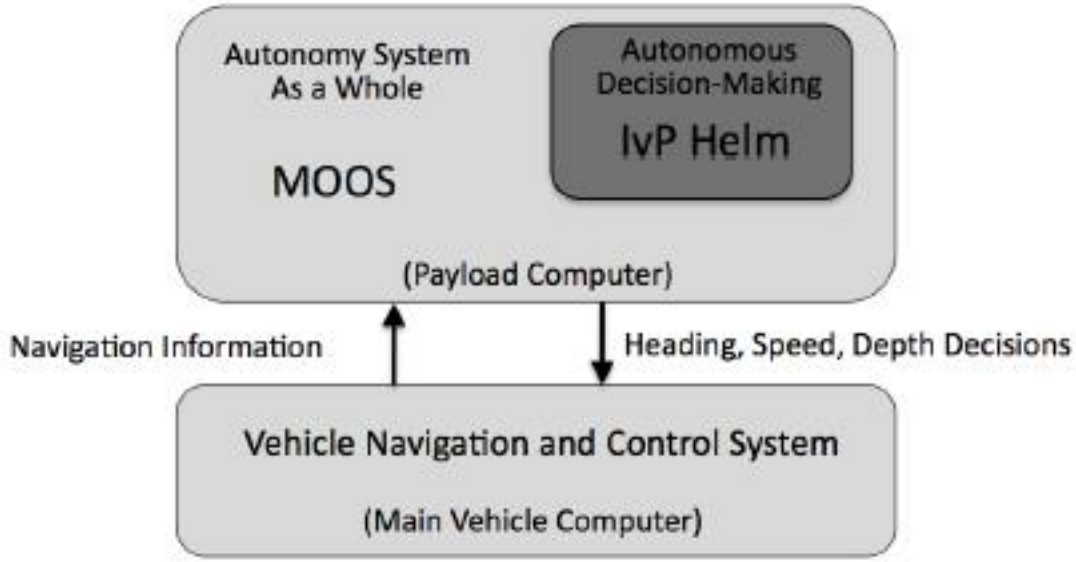


MOOSDB

- Star topology
- each MOOS community application with a connection with a single MOOS Database
- No peer to peer communications
- All communication instigated by client
- Each client (MOOSApp) has a unique name
- A client does not need to know of the existence of others
- Network can be distributed over multiple machines

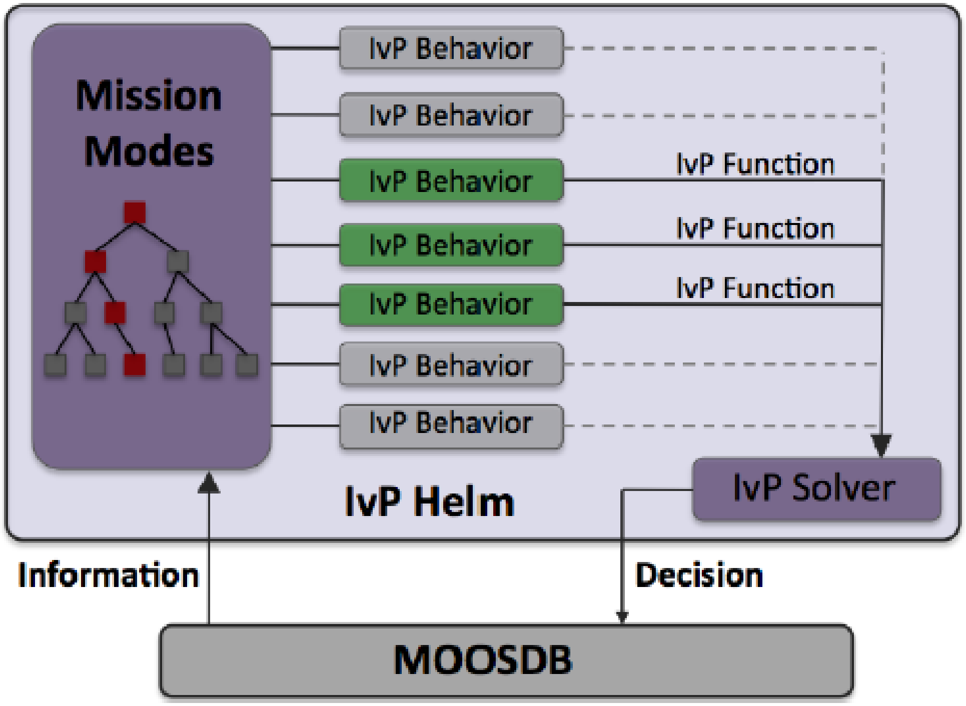
MOOS design

- Backseat design philosophy
- Separation from vehicle autonomy (mission oriented) and vehicle control

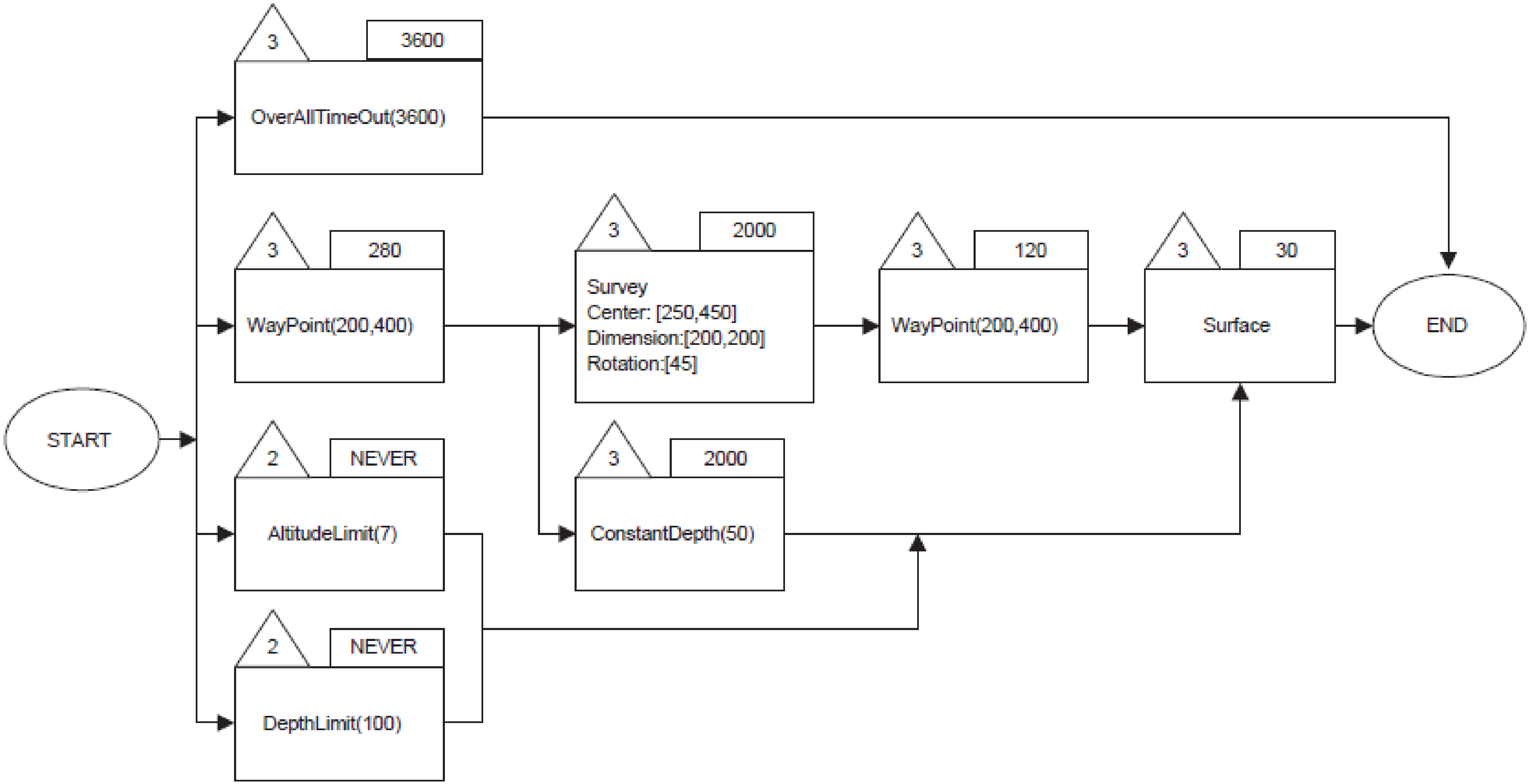


MOOS IvP Helm

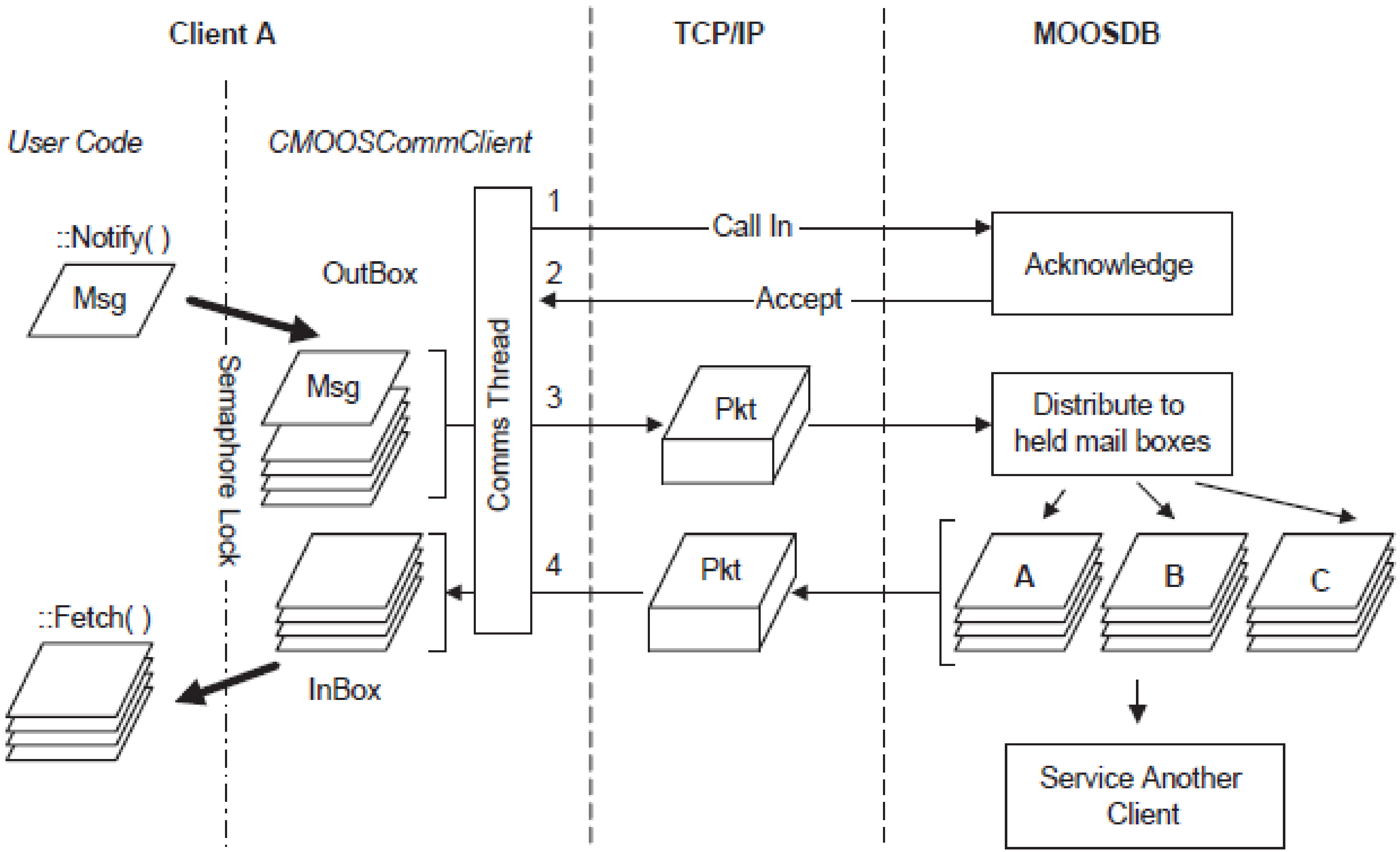
- single MOOS app running process pHelmIvP
- IvP – Interval Programming
- Behavior based architecture
- IvP solver multi-objective optimization to find the best action in each iteration of helm
- One to four times per second
- Only one subset of behaviors active in each time



MOOS Mission



MOOS communication architecture



MOOS comms

- Each application
  - Publishes data – issue notification on named data
  - Register for notifications on named data (subscription)
  - Collect notifications on named data
- User code calls Notify() to transmit data
- User code can retrieve the list of messages at any time with Fetch()

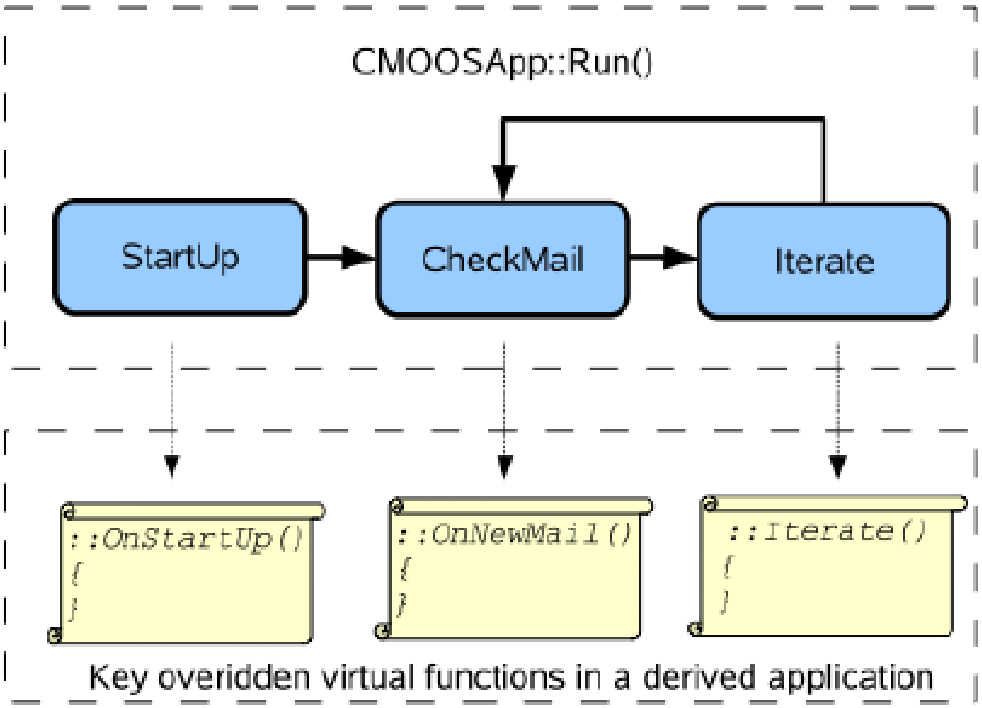
MOOS Message content

- Data sent in string or doubles
- Packed in messages – CMOOSMsg class
- String data comma separated “name = value pairs” - Human readable

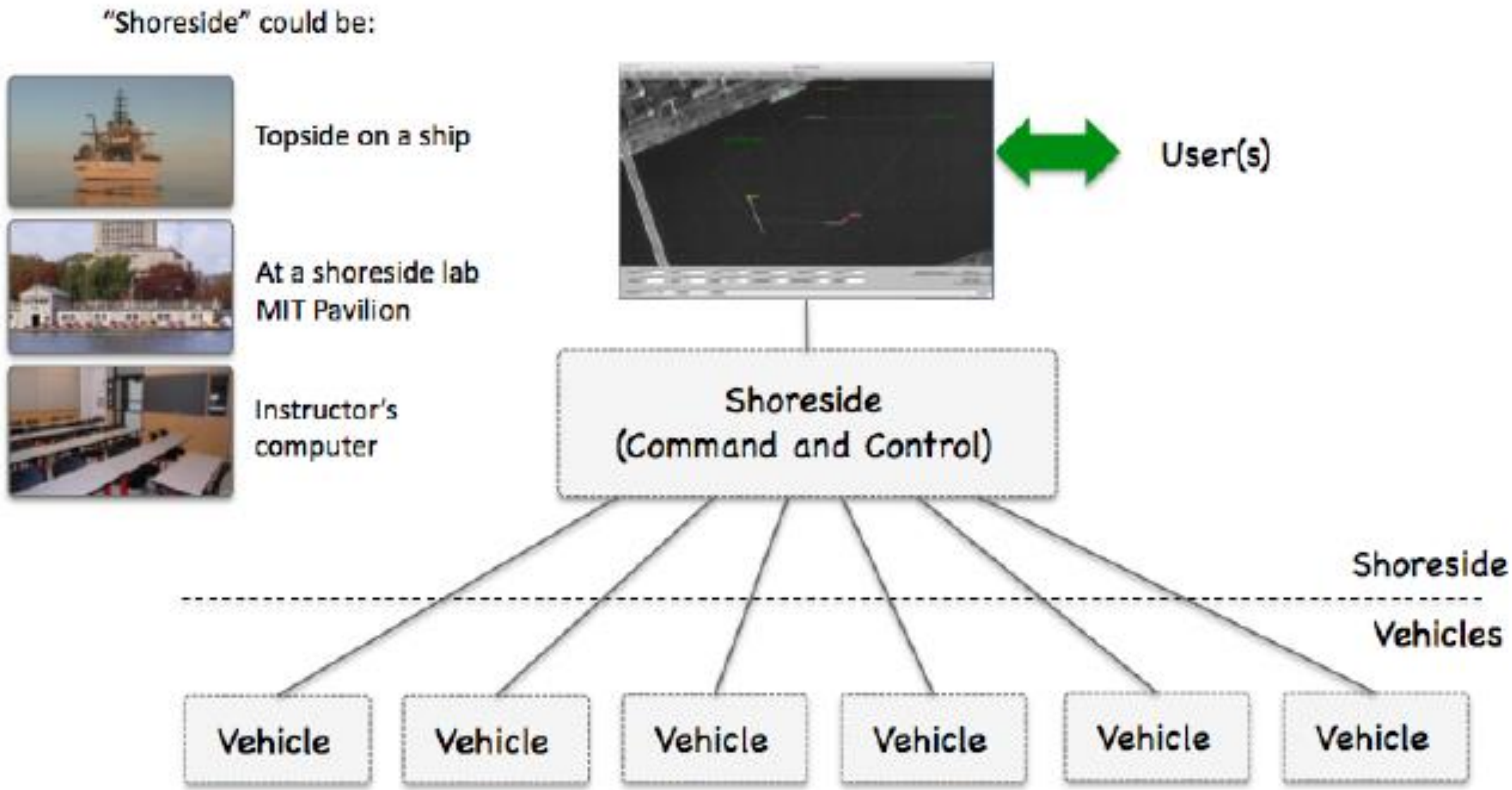
Variable	Meaning
Name	The name of the data
String Value	Data in string format
Double Value	Numeric double float data
Source	Name of client that sent this data to the `MOOSDB`
Time	Time at which the data was written
Data Type	Type of data (`STRING` or `DOUBLE`)
Message Type	Type of Message (usually `NOTIFICATION`)
Source Community	The community to which the source process belongs

MOOS CMOOSApp

- CMOOSApp base class for writing new applications
- It calls Iterate() repetitively (user should provide content
- OnNewMail() newly received data

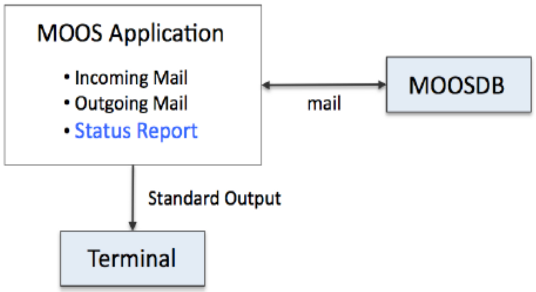


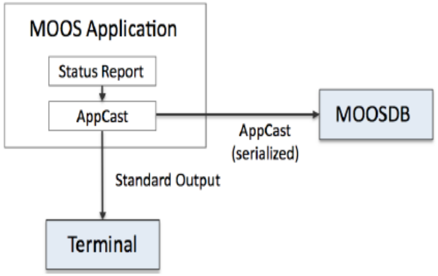
Multiple vehicles (uField Toolbox)

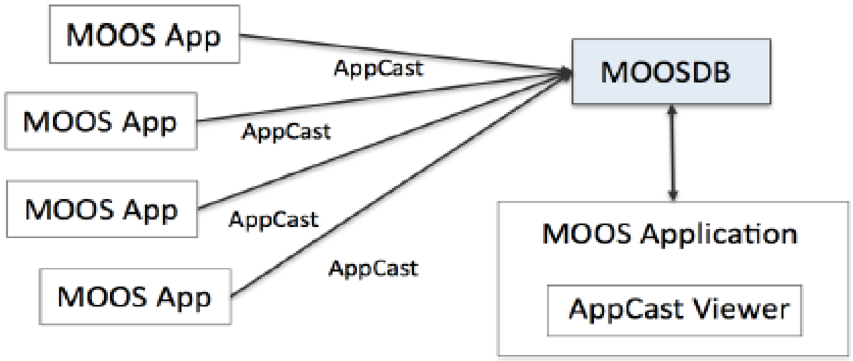


MOOS apps and terminal output

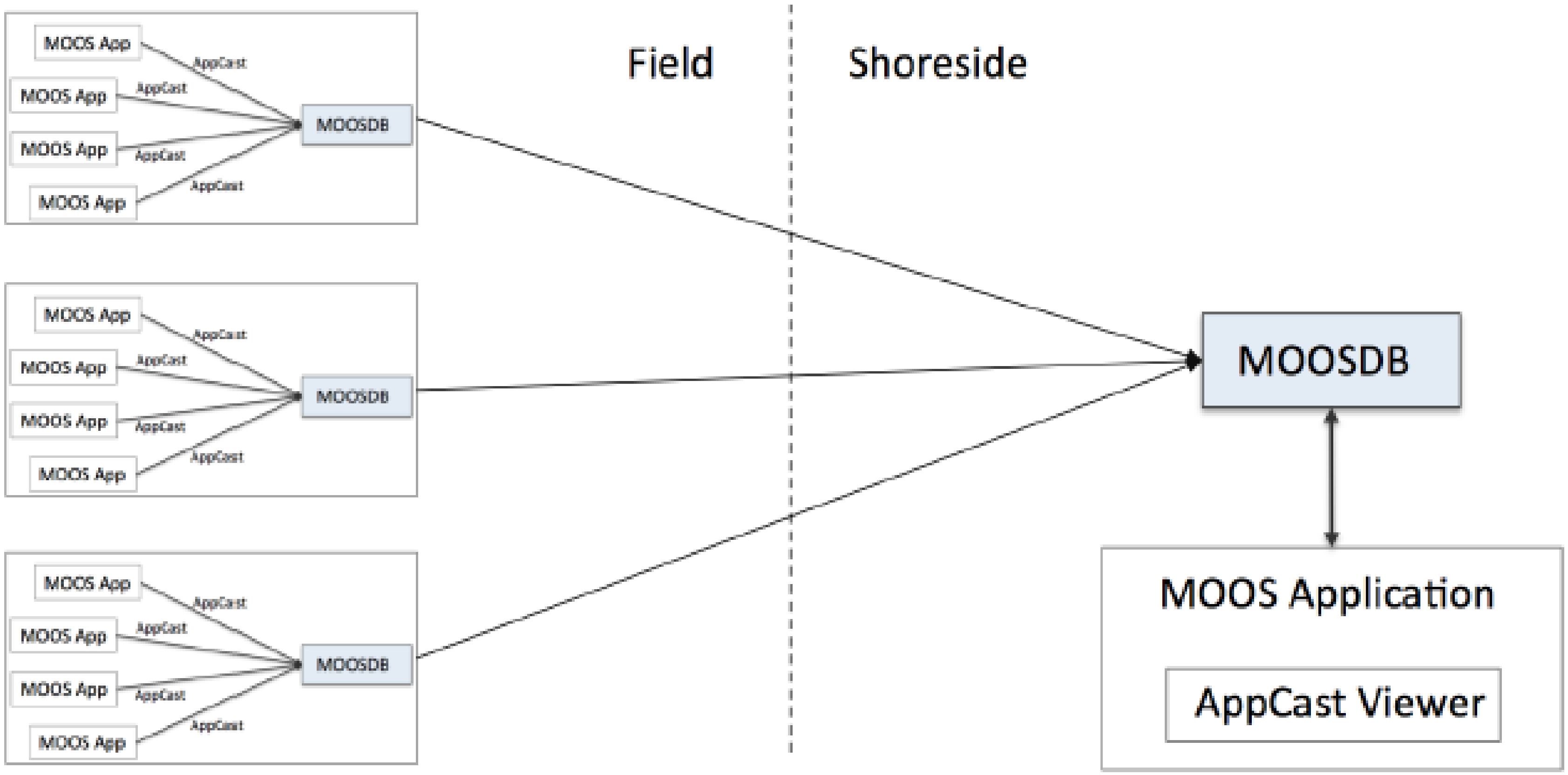
- AppCasting
  - Providing easer to see aplication terminal output
  - Optional feature
  - Prodies an aditional report published to MOOSDB
  - Ease of viewing multiple applications with one veiewer (AppCast Viewer)

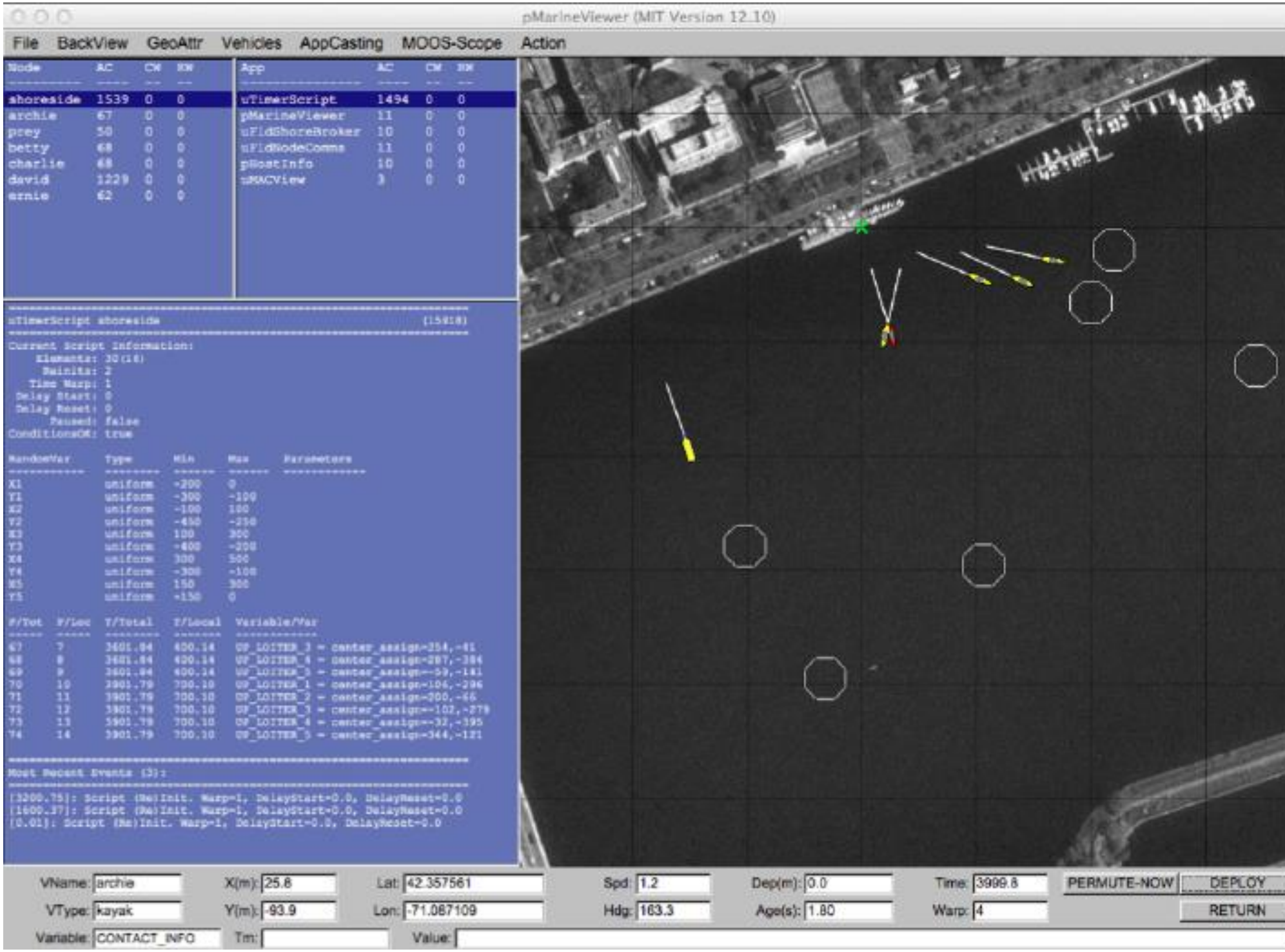




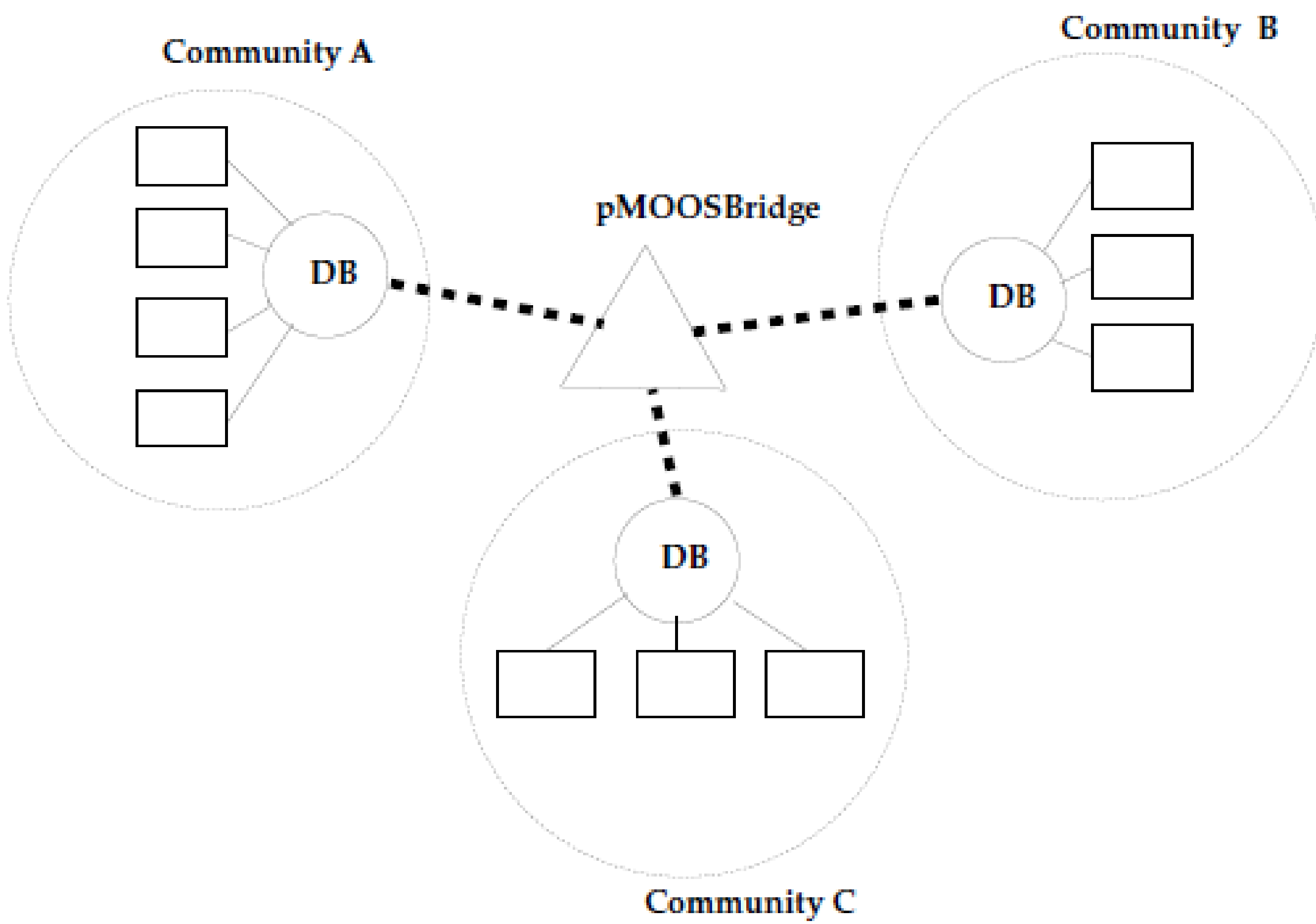


Multiple vehicles

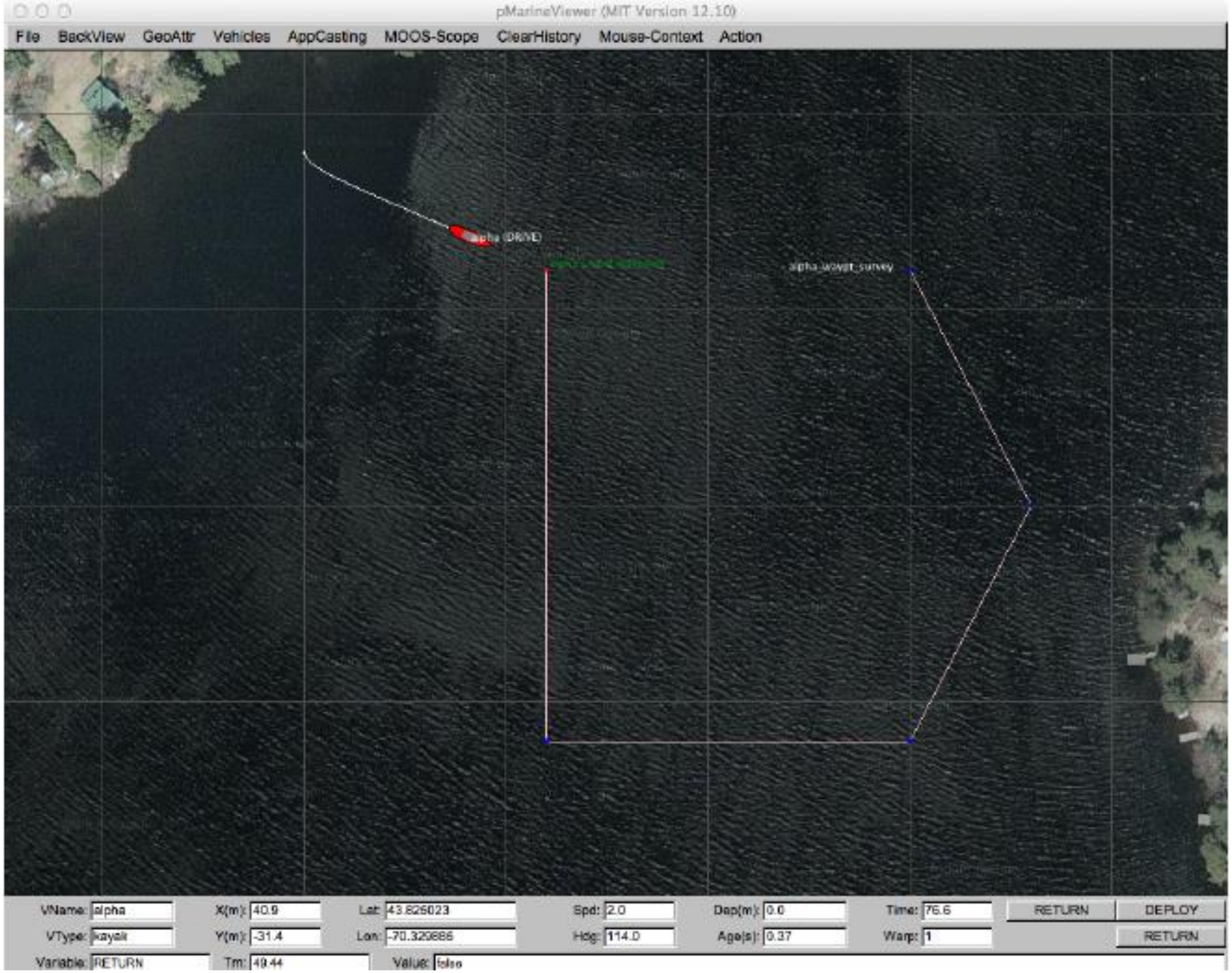



pMarineViewer (MIT Version 12.10)
File BackView GeoAttr Vehicles AppCasting MOOS-Scope Action
Node AC CW RW
shoreside 1539 0 0
archie 67 0 0
prey 50 0 0
betty 68 0 0
charlie 68 0 0
david 1229 0 0
ernie 62 0 0
App AC CW RW
uTimerScript 1494 0 0
pMarineViewer 11 0 0
uFldShoreBroker 10 0 0
uFldNodeComms 11 0 0
pHostInfo 10 0 0
uMACView 3 0 0
VName: archie
VType: kayak
Variable: CONTACT_INFO
X(m): 25.8
Y(m): -93.9
Tm:
Lat: 42.357561
Lon: -71.087109
Value:
Spd: 1.2
Hdg: 163.3
Dep(m): 0.0
Age(s): 1.80
Time: 3999.8
Warp: 4
PERMUTE-NOW
DEPLOY
RETURN

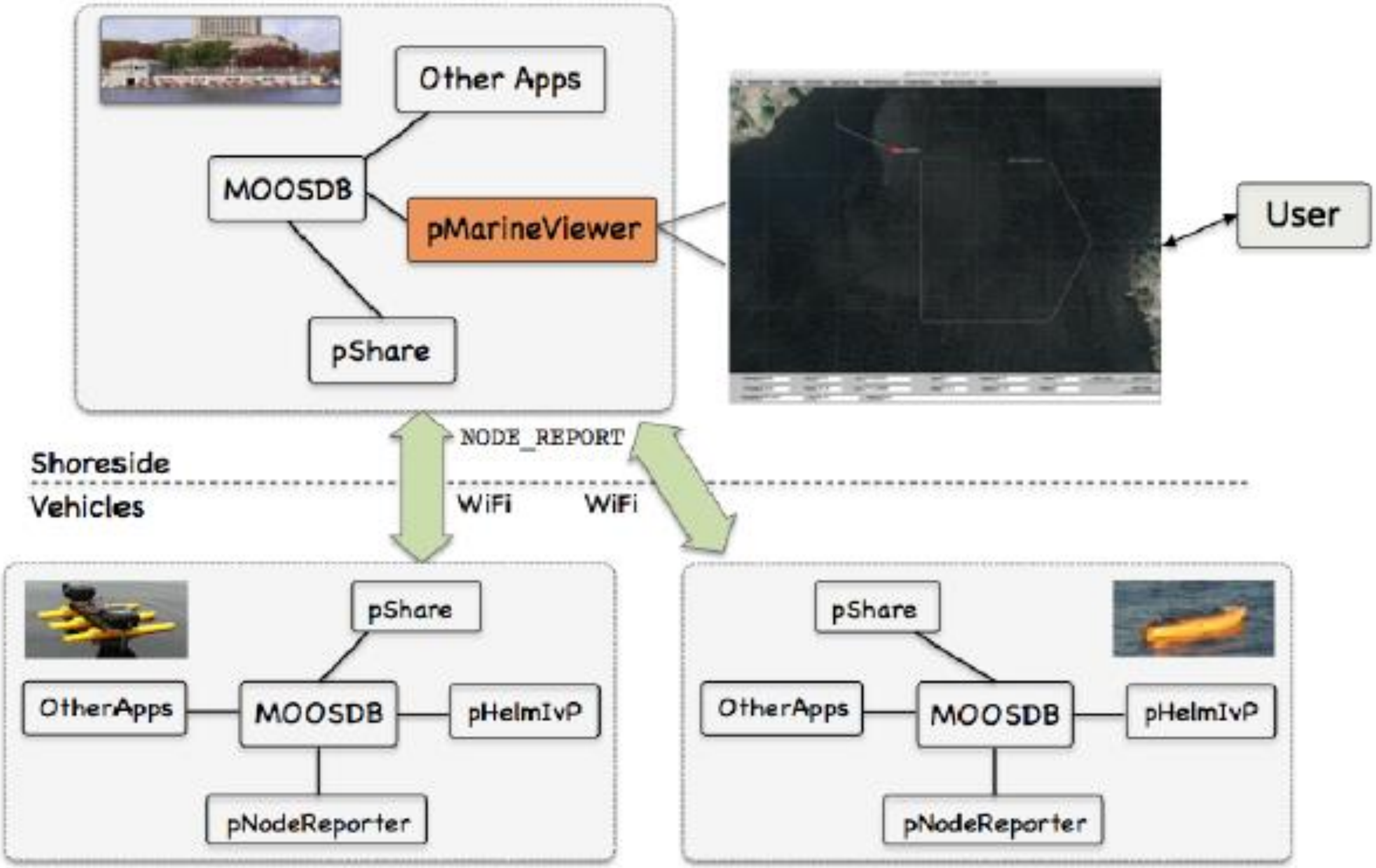
MOOS multiple communities



Mission control and monitoring: pMarineViewer



pMarineViewer typical use



MOOS source tree

- MOOS
  - Core
    - MOOSLib
    - MOOSgenLib
    - MOOSBD
    - pSCheduler
    - pMOOSBridge
    - pAntler
    - pLogger
  - BasicApps
    - pHelm
    - pNav
    - iRemote
    - MOOSNavLib
    - MOOSTaskLib
  - Instruments
    - Common
      - iGPS
    - Ocean
      - iDepth
      - iDVL
      - etc....
    - Land
      - iAGV
      - etc....
    - Virtual
      - iMatlab
  - Utils
    - uMVS
    - uMS
  - Thirdparty
    - newmat
    - FLTKVW
  - ***MOOSUSER***
    - AppX
    - AppY
    - AppZ

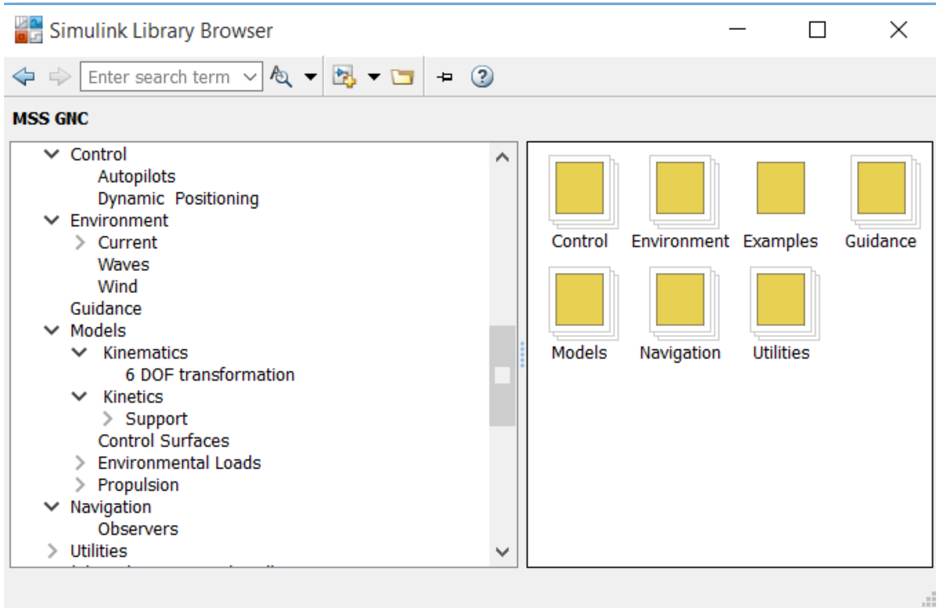
Marine Robotics Simulation

Marine robotic systems simulation

- Detailed dynamic simulation
  - – CFD, (panel methods) – WAMIT, Ansys, Fluent
- Generic system simulation
  - MATLAB/Simulink (MSS toolbox, other matlab tooboxes)
- Robotics Simulators
  - UWSim
  - MORSE / Blender
  - Gazebo

Marine Systems Simulator (MSS)

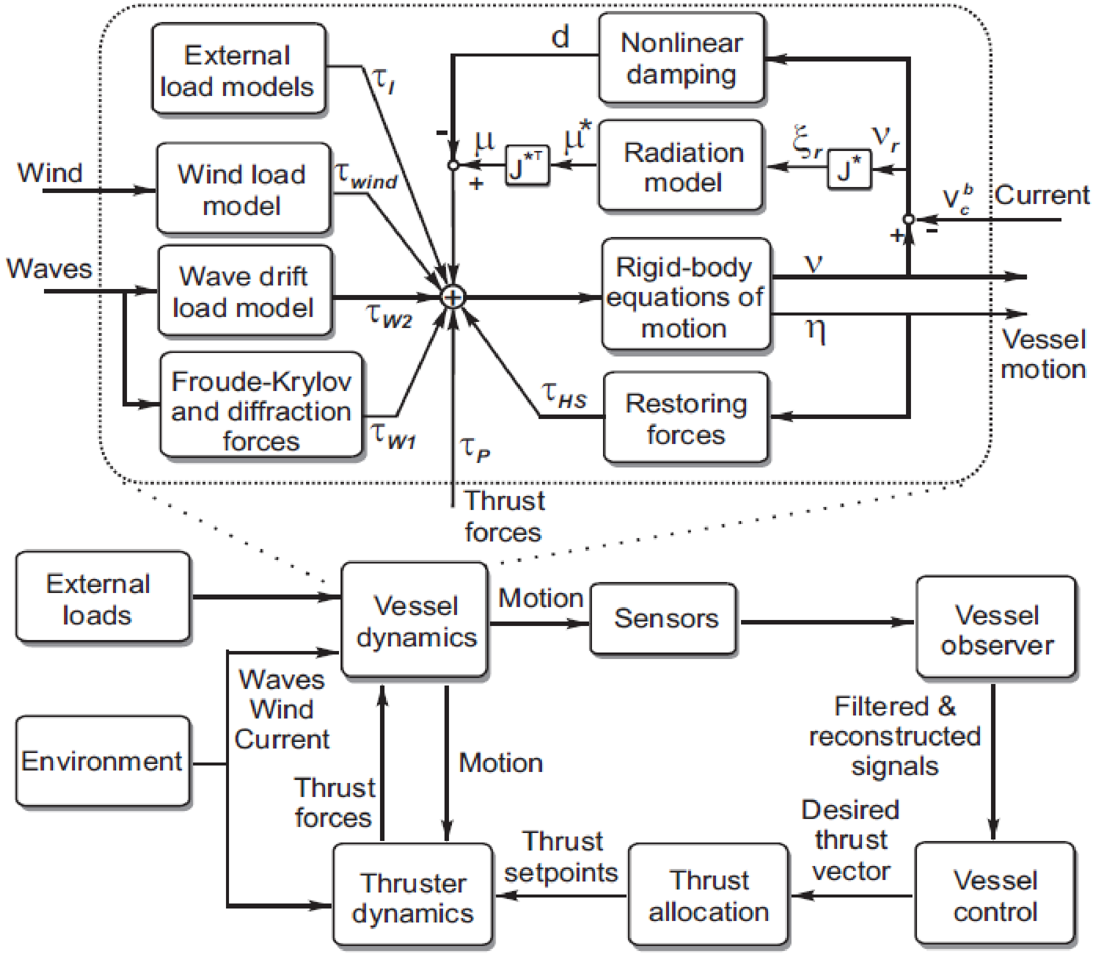
- Matlab/Simulink toolbox developed at NTNU
- Components
  - **MSS GNC** guidance, navigation and control library (*the most useful component*)
  - MSS HYDRO reads info from potential theory programs generating data for Matlab simulation (requires ShipX or WAMIT license)
  - MSS FDI standalone toolbox for identification of radiation-force models and fluid memory effects (seakeeping theory)



www.marinecontrol.org

MSS

- MSS GNC Simulink Library
  - Controllers
  - Wave and wind generation
  - Guidance blocks
  - Vehicle models and utilities
  - Observers and navigation filters

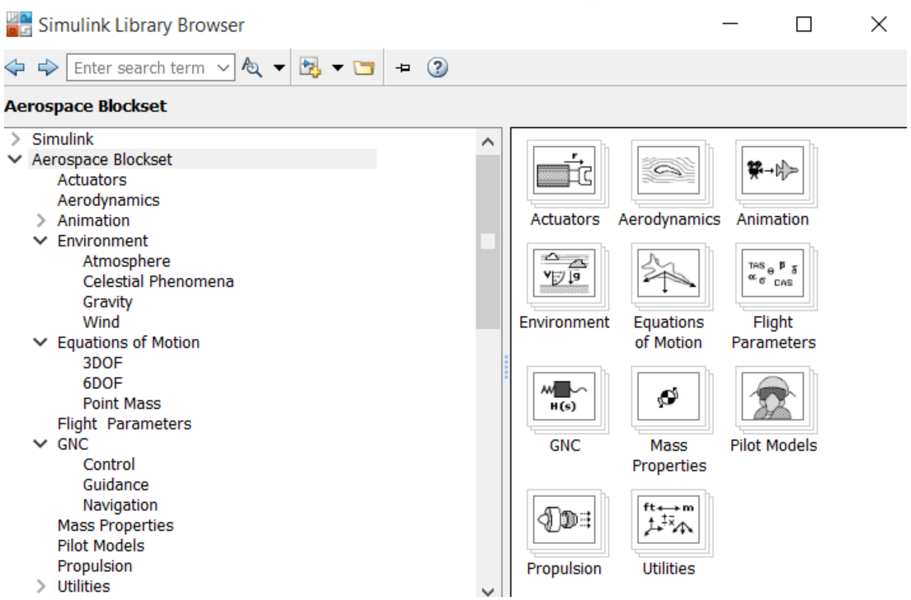


[1] T. Perez et al, “An Overview of Marine Systems Simulator (MSS): A Simulink Toolbox for Marine Control Systems”, Modeling, identification and Control 27.4, pp 259-275, 2006

Other toolboxes: Aerospace Blockset

- Simulink blockset for aerospace applications (Mathworks)
- Provides useful models and block utilities for simulation of vehicle motion also usable in marine environment

MathWorks



http://www.mathworks.com/products/aeroblks/

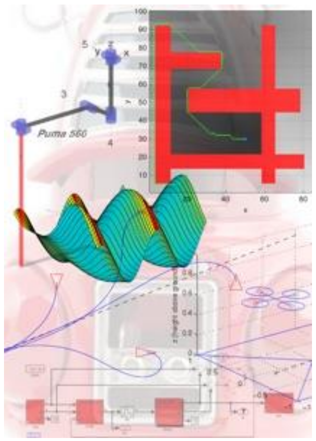
Other toolboxes:Peter Corke's Robotics and Machine Vision Toolboxes

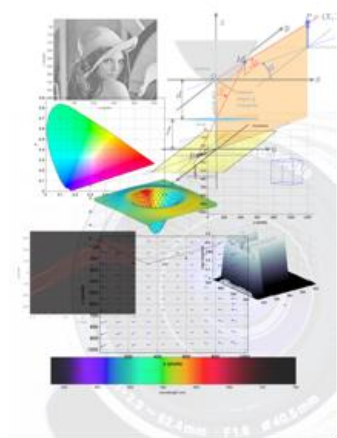
- Robotics toolboxes developed by Prof. Peter Corke from QUT

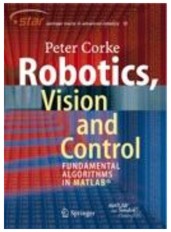
- Robotics toolbox
  - Coordinate conversion utilities
  - Planning and localization tools for mainly for manipulator and *ground* vehicles

- Machine vision toolbox
  - Useful toolbox in computer vision
  - Implements common tools with the standard Matlab Image Processing Toolbox and also additional functions

- Useful companion to the book: Robotics, Vision and Control







http://www.petercorke.com/Toolboxes.html

Robotic Simulators

- Generic simulators for robotics development
- Sensors and environment simulation
- From high detail in physical dynamic simulations to basic simple kinematic models
- Some allow hardware in the loop simulation
- Interfaces to common robotics middleware
- Examples
  - USARSIM
  - Player/Stage
  - Webots
  - Gazebo
  - MORSE

Simulators for underwater robotics

- Few attention has been dedicated to the underwater or marine environment

- Marine robotic sensor models such as sonars are in general not developed

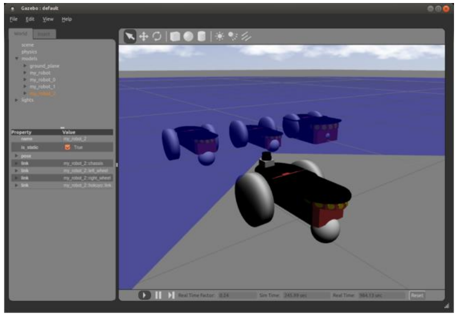
- Some simulators
  - Gazebo
  - V-REP
  - MORSE/Blender
  - **UWSim**

Gazebo

- 3D simulator
- Recent developments from DARPA robotics challenge
- Linux environment
- Modular architecture Google protobufs for communications between modules)
- 3D simulation with OGRE3D
- Simulator independent from visualization
- Simulation description in a definition file: SDF (Simulation Definition File)
  - robots
  - environment
- Multiple interfaces
  - plugins (C++ code making the interface and shared linking)
  - ROS (plugins ROS)

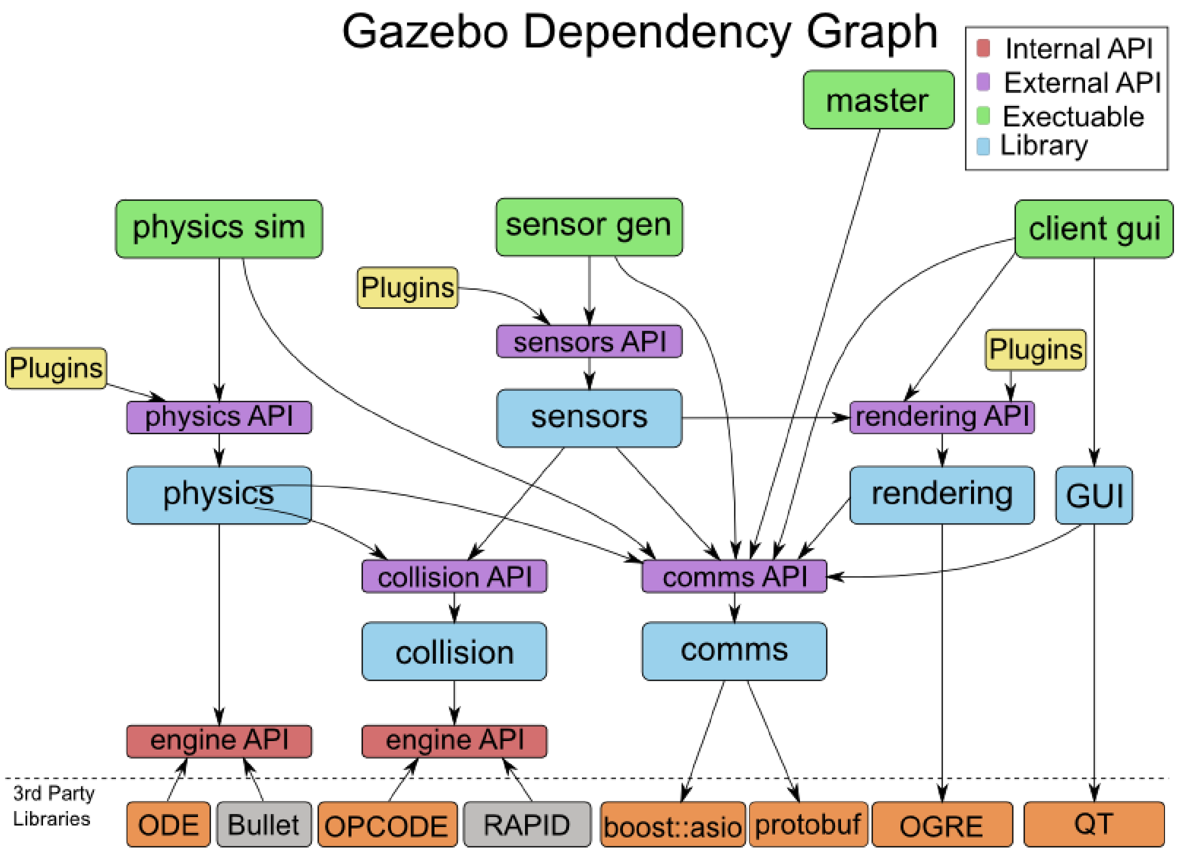
GAZEBO

gazebosim.org



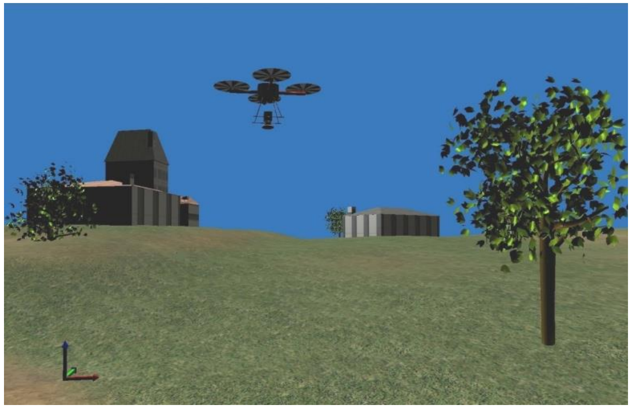
Gazebo

- gazebo server
  - communication management
  - controls physics simulation loop
  - generates sensor readings
- gazebo client
  - user graphic interface



Morse

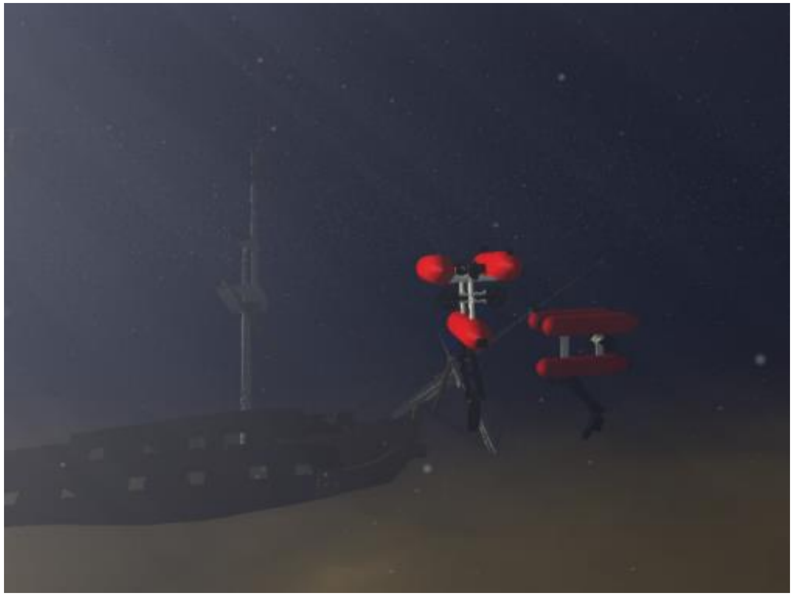
- 3D Simulation of indoor and outdoor environments;
- Provides several robots, sensors and actuators;
- Support different middlewares used in robotics (ROS, Sockets, MOOS and YARP);
- The rendering is based on Blender Game Engine;
- Realistic gathered data is possible through addition of gaussian noise to sensors outputs
- Licensed under a permissive BSD license;
- Considerable community
- Good documentation - tutorials, code examples, reusable snippets, etc.



UWSim

- Underwater simulator developed by IRS Lab of Jaume-I University, Madrid
- Developed under the RAUVI and TRIDENT projects
- Uses OSG (Open Scene Graph)
- Simulates underwater environments
- Configurable environment
- Multiple robot
- Underwater manipulators

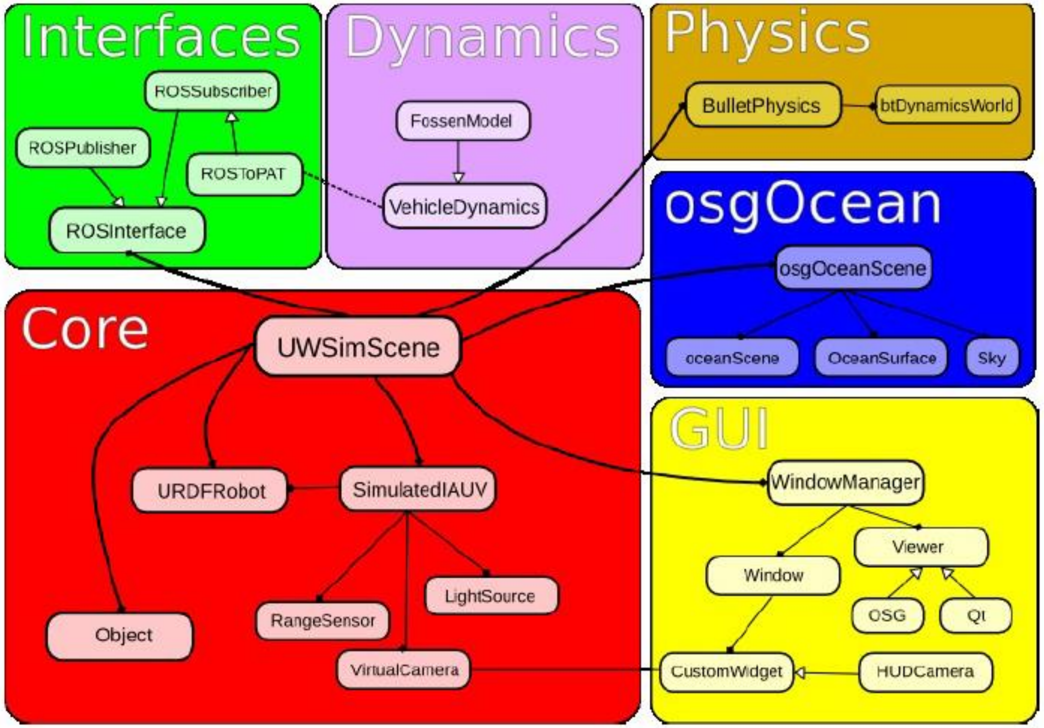
- Common underwater sensors
  - Camera
  - Pressure
  - DVL
  - GPS
  - Multibeam sonar
  - Structured light projector
  - …
- ROS interface



http://www.irs.uji.es/uwsim/
http://www.irs.uji.es/uwsim/wiki/

UWSim

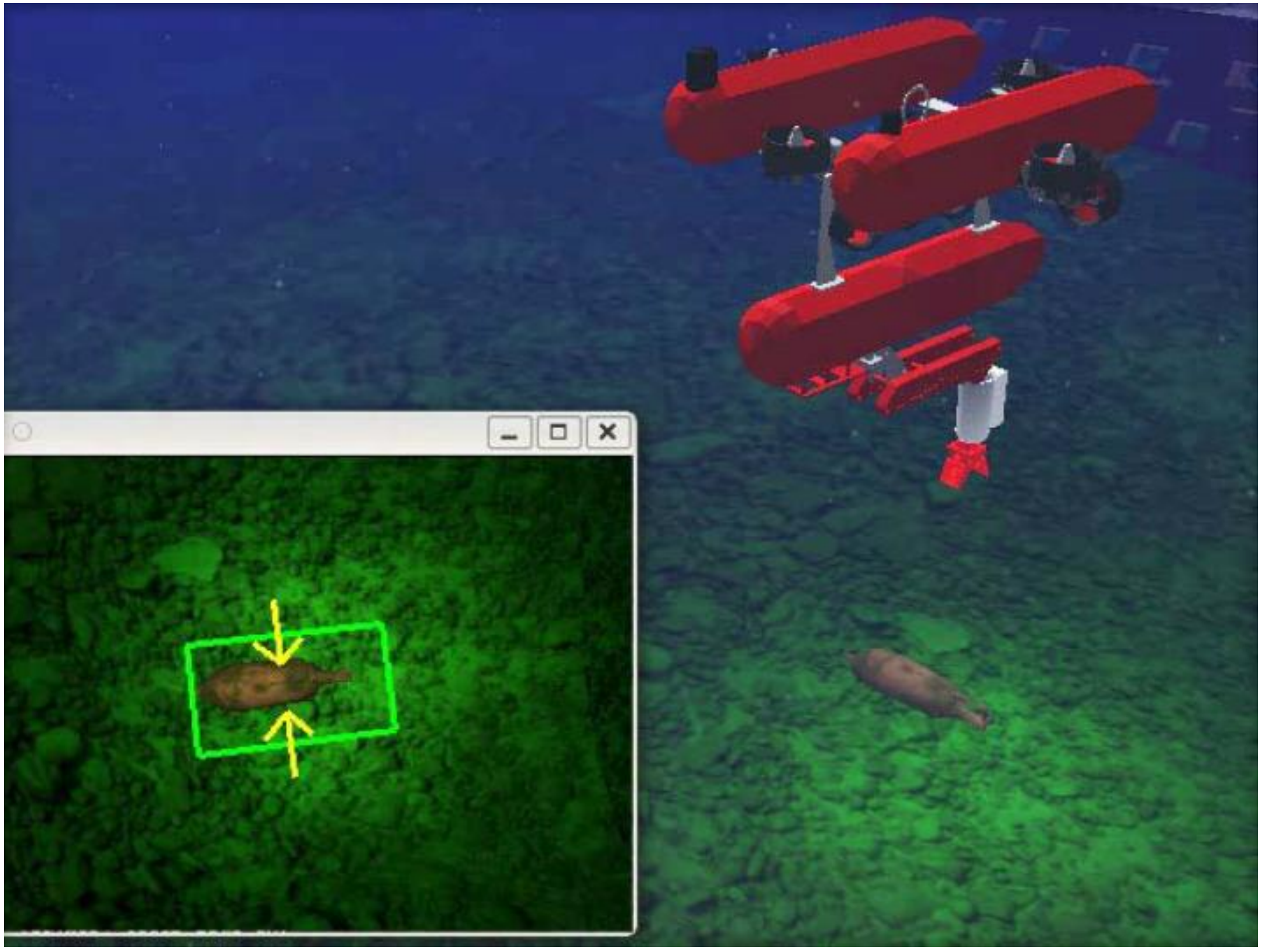
Architecture



UWSim architecture, from [1]

[1] M. Prats, et al, “An open source tool for simulation and supervision of underwater intervention missions”, IEEE IROS Conference, 2012

UWSim

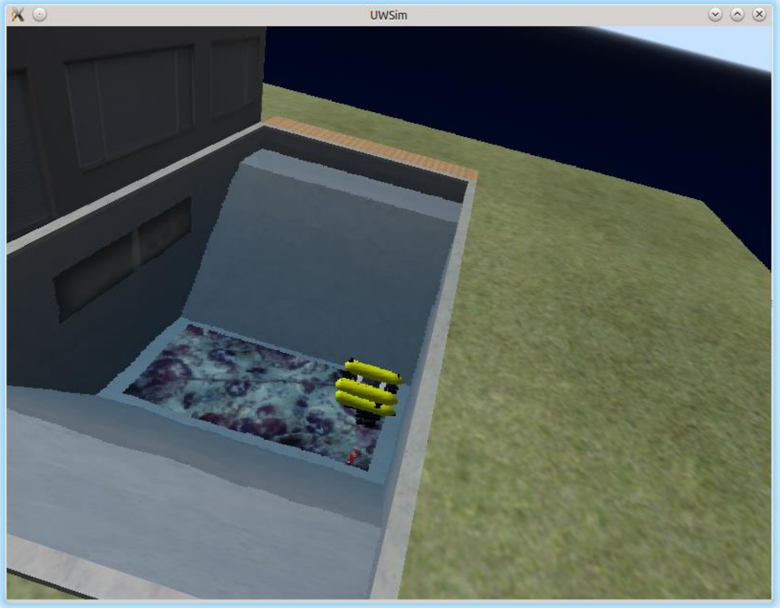


UWSim Practical Introduction

Interacting with UWSIM

- Start a *roscore*
  - `>> roscore`
- Start UWSim
  - `>> rosrun uwsim uwsim`

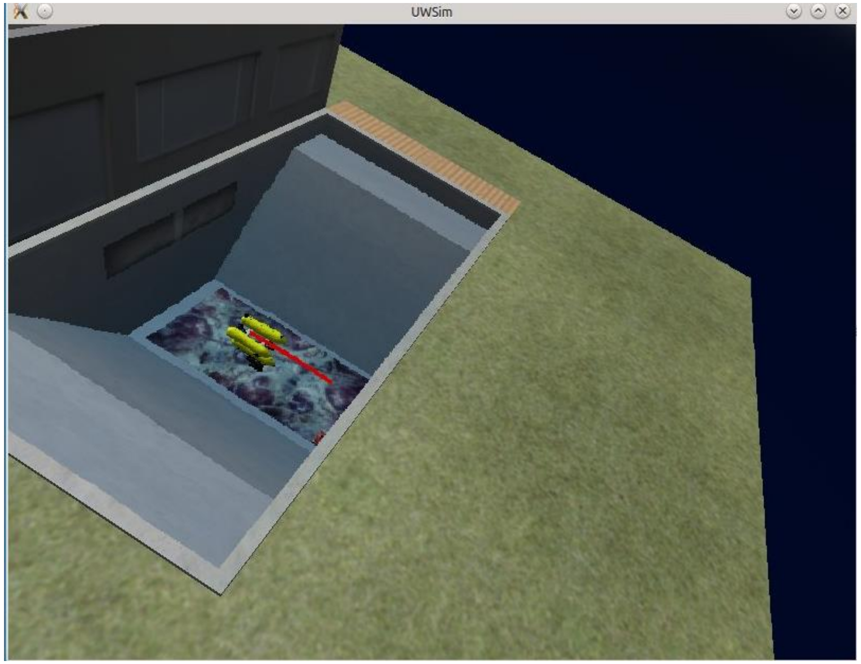
You should get something like this



Interacting with UWSIM

- Check the list of ROS topics available
  - `>> rostopic list`
- Set a new Vehicle Position (x=2, y=0, z=3, roll, pitch, yaw=0)
  - `>> rosrun uwsim setVehiclePosition /dataNavigator 2 0 3 0 0 0`

You should get something like this



Task I – Simulation Vehicle Dynamics

- Start a *roslaunch*
  - `>> cd /catkin_ws`
  - `>> source setup.bash`
  - `>> roslaunch underwater_vehicle_dynamics UWSim_g500_dynamics.launch`
- Use the control by keyboard

```
Use the keys to control:
```

Task II- ROS interface

- Step 1. Compile a ROS node to send reference commands to G500 thrusters

- Step 2. Implement guidance controllers for the vehicle (ex. Similar to the ones previously tested in Matlab/Simulink)