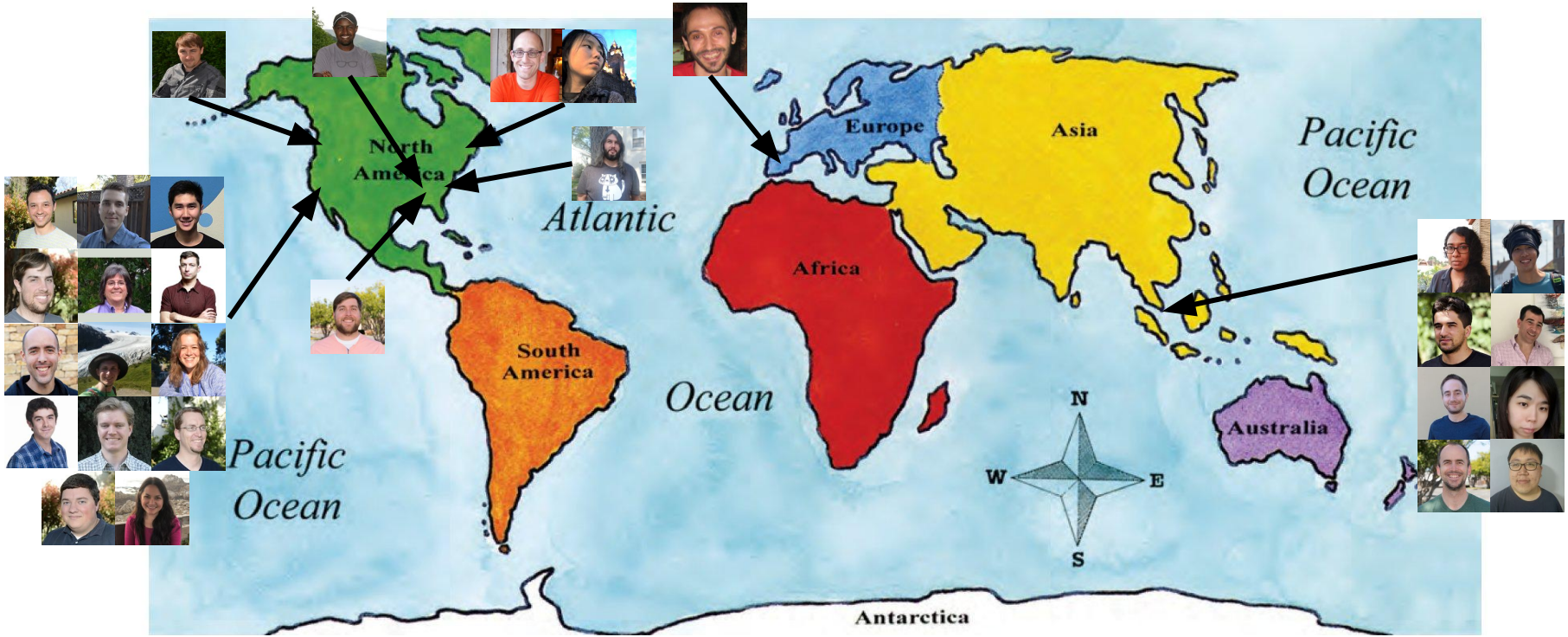
SITL improvements and ROS integration

Tully Foote
Open Robotics

Who we are

We create open software and hardware platforms for robotics. We use those platforms to solve important problems and we help others to do the same.

Where we are


North America
South America
Europe
Asia
Africa
Australia
Atlantic
Ocean
Pacific
Ocean
Pacific
Ocean
N
W
E
S
Antarctica

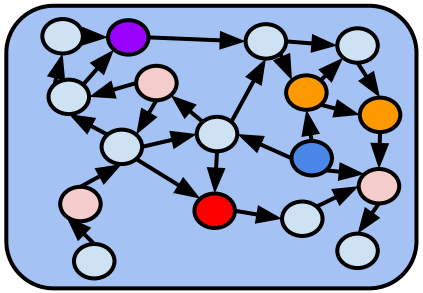
Our products

Open source robotics developer tools

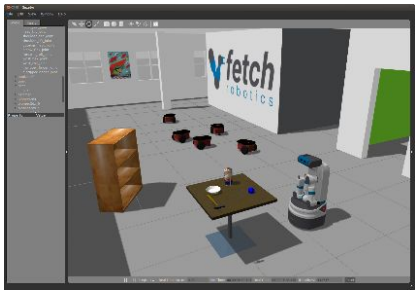
ROS: Robot application SDK

Ignition: Robot simulator

- Develop and test in simulation
- Deploy same software to robots
- Focus on differentiating capabilities



ROS



The next generation

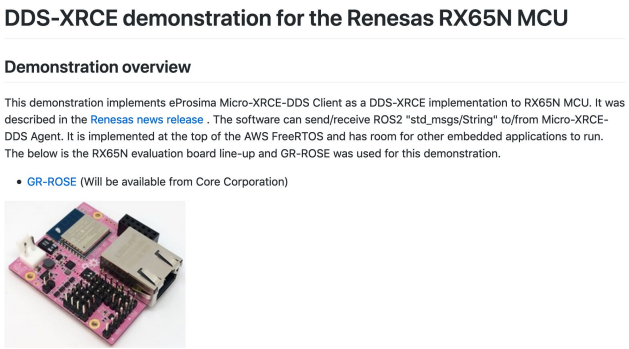
ROS 2: Goals

1. **Quality of design & implementation**
2. **System reliability**
3. **Real-time control & deterministic execution**
4. **Validation, verification, and certification**
5. **Flexibility in communication**
6. **Support for small embedded systems**

ROS 2 Dashing: First LTS release (May 31 2019)

1. **Actions: Python support & command line tool**
2. **IDL support**
3. **MoveIt 2 Alpha**
4. **Testing: QA, performance, security**
5. **Diagnostics**
6. **armhf support at Tier 2**

micro-ROS



DDS-XRCE demonstration for the Renesas RX65N MCU

Demonstration overview

This demonstration implements eProsima Micro-XRCE-DDS Client as a DDS-XRCE implementation to RX65N MCU. It was described in the Renesas news release . The software can send/receive ROS2 "std_msgs/String" to/from Micro-XRCE-DDS Agent. It is implemented at the top of the AWS FreeRTOS and has room for other embedded applications to run. The below is the RX65N evaluation board line-up and GR-ROSE was used for this demonstration.

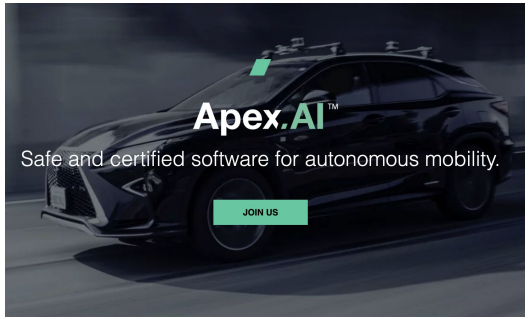
- GR-ROSE (Will be available from Core Corporation)

Renesas MCU demo

Embedded Systems

- Lighter weight communication system
- More modular design

Autoware

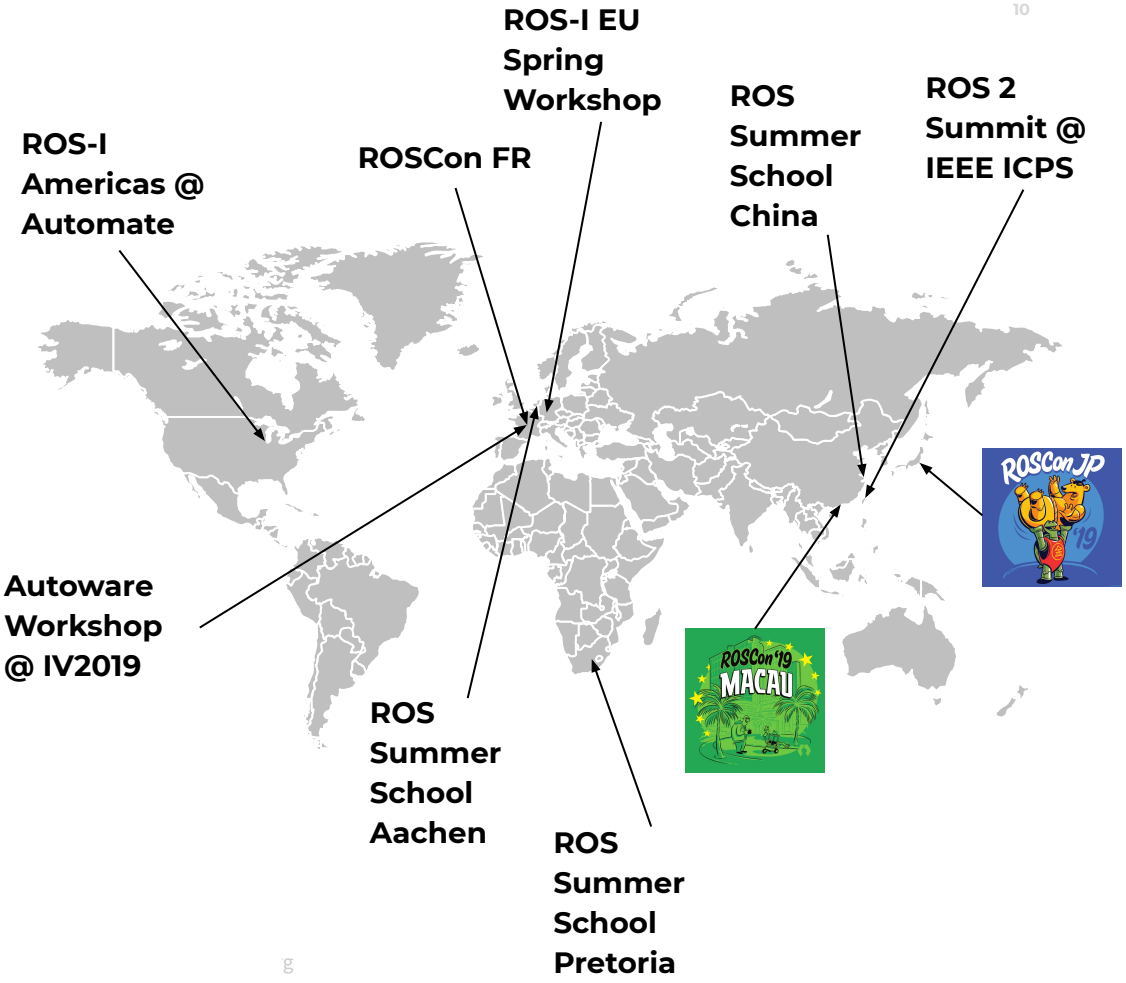


Apex.AI

Automotive

- **Deterministic execution**
- **Safety & certification**

(some) Community Events in 2019



Ignition Gazebo: Organizing principles

Provide the best software stand-in for a physical robot

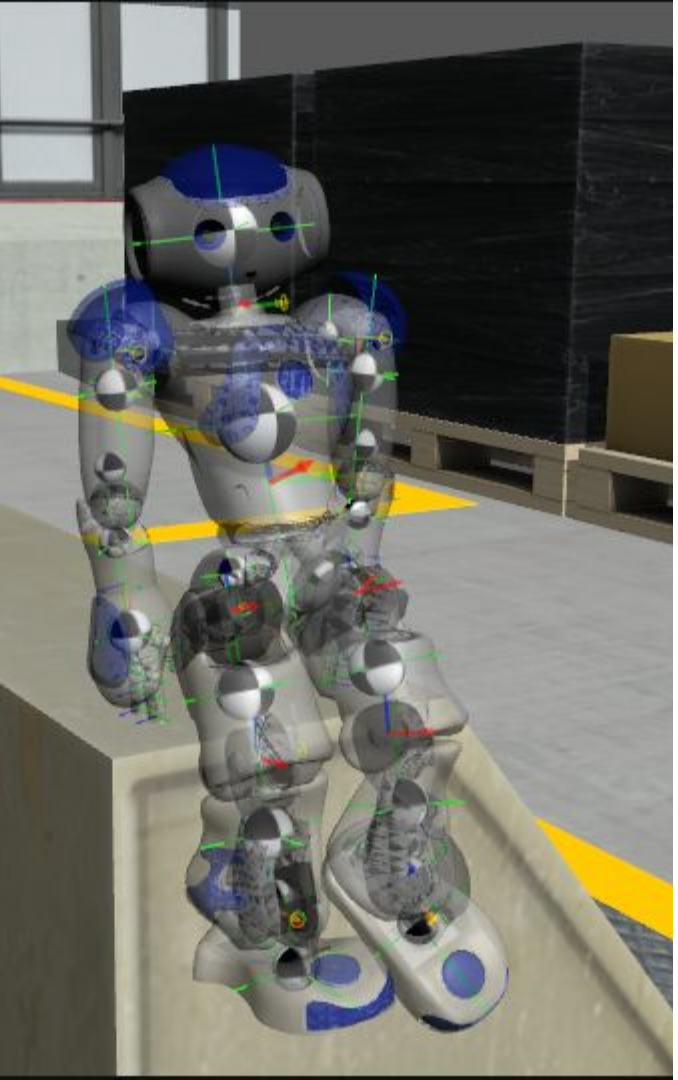
1 **Physics**

2 **Sensing**

3 **Extension**

4 **Modularity**

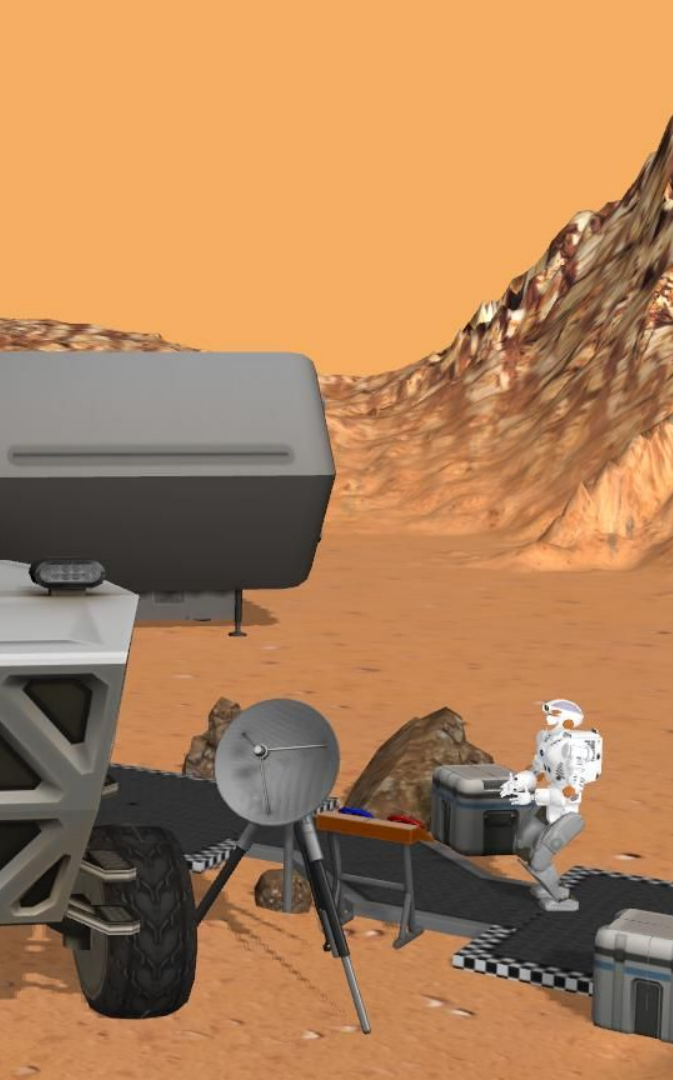
Physics



- No single engine is best for all situations
- Common API atop multiple physics engines
- Choose engine at runtime
- Maximal and reduced coordinate approaches
- Allows simple engines, e.g., kinematics-only

SENSING

- **Parameterizable models of common sensor types**
- **Parameterizable models of common noise types**
- **Common API atop multiple rendering engines**
- **Export sensor data via middleware (e.g., ROS)**

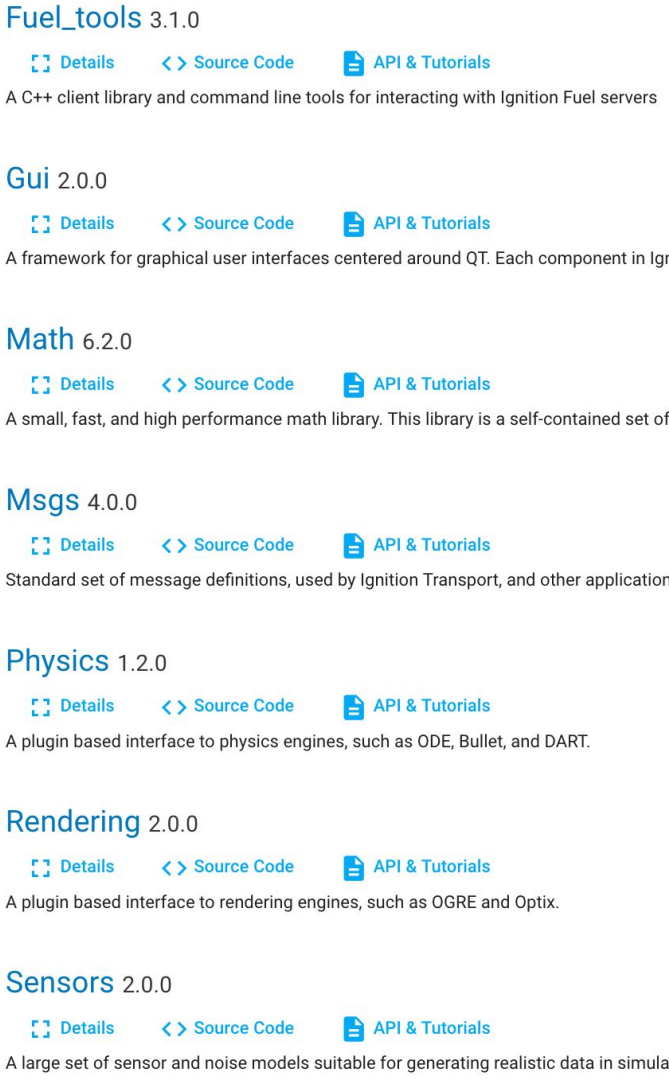


EXTENSION

- C++ plugin API allows any kind of extension
- Get and/or set the world between physics steps
- Add or extend sensors
- Interface with hardware input devices
- Fake interactions that are impractical to simulate
- Delegate interactions to other systems

MODULARITY

- **Monolithic Gazebo decomposed into Ignition libraries**
- **Libraries can be reused in other applications**
- **Ignition Gazebo is just one particular composition**

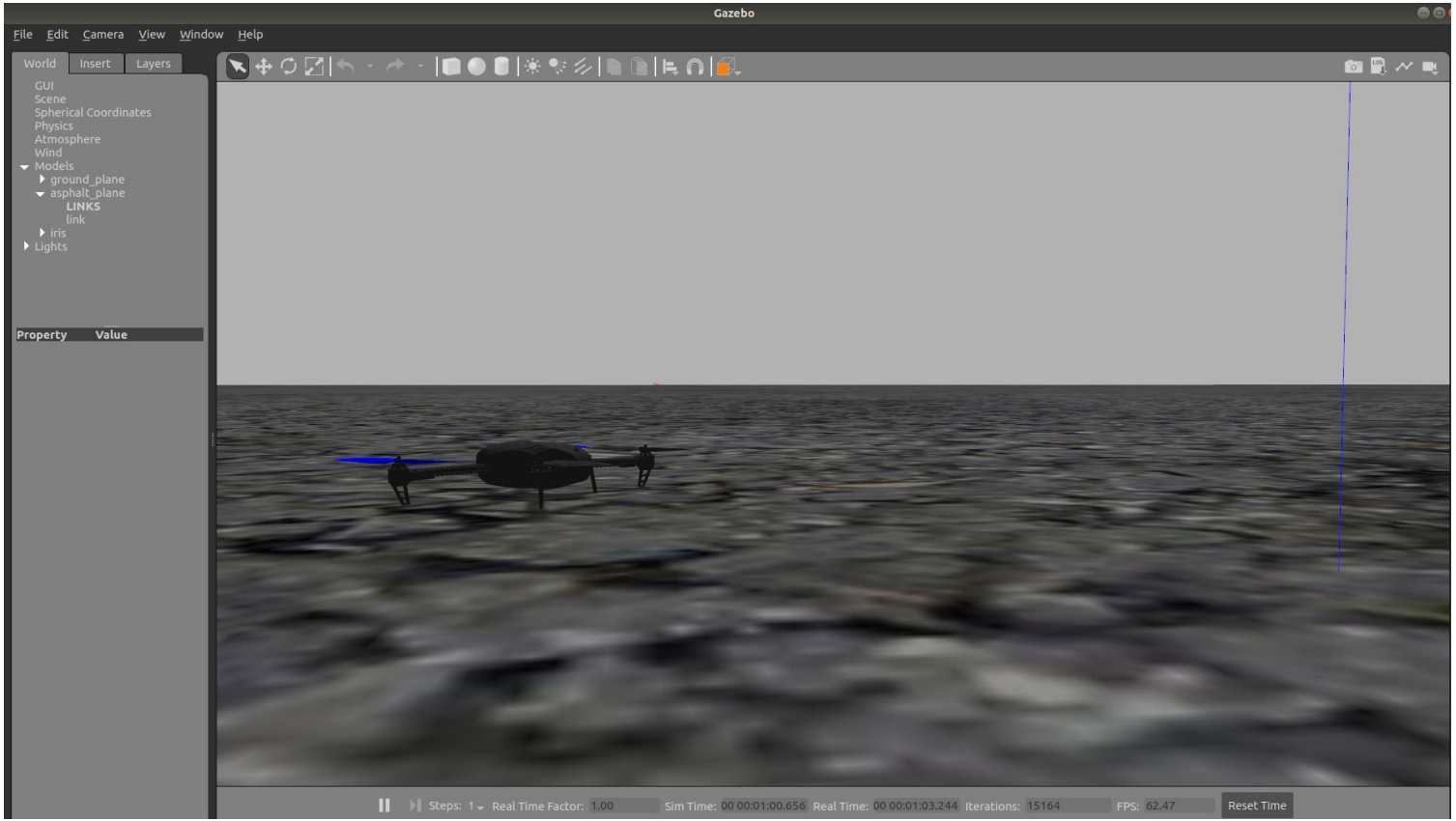


Ignition Blueprint (31 May 2019)

1. **Physically based rendering (PBR) materials**
2. **GUI tools for model placement**
3. **Payload-dependent battery model**
4. **New command line tools**
5. **Incremental level loading**
6. **Distributed simulation**

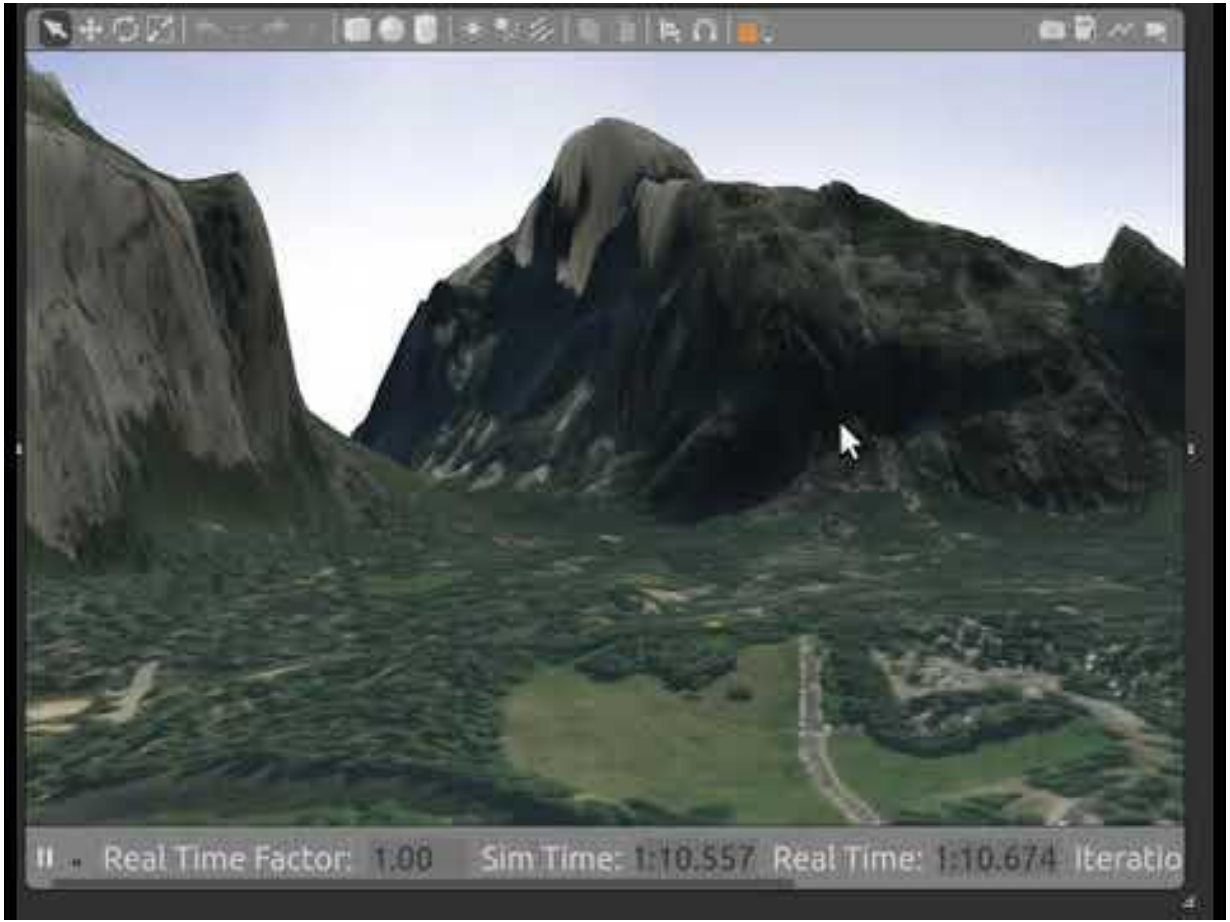
SITL

Default Gazebo SITL

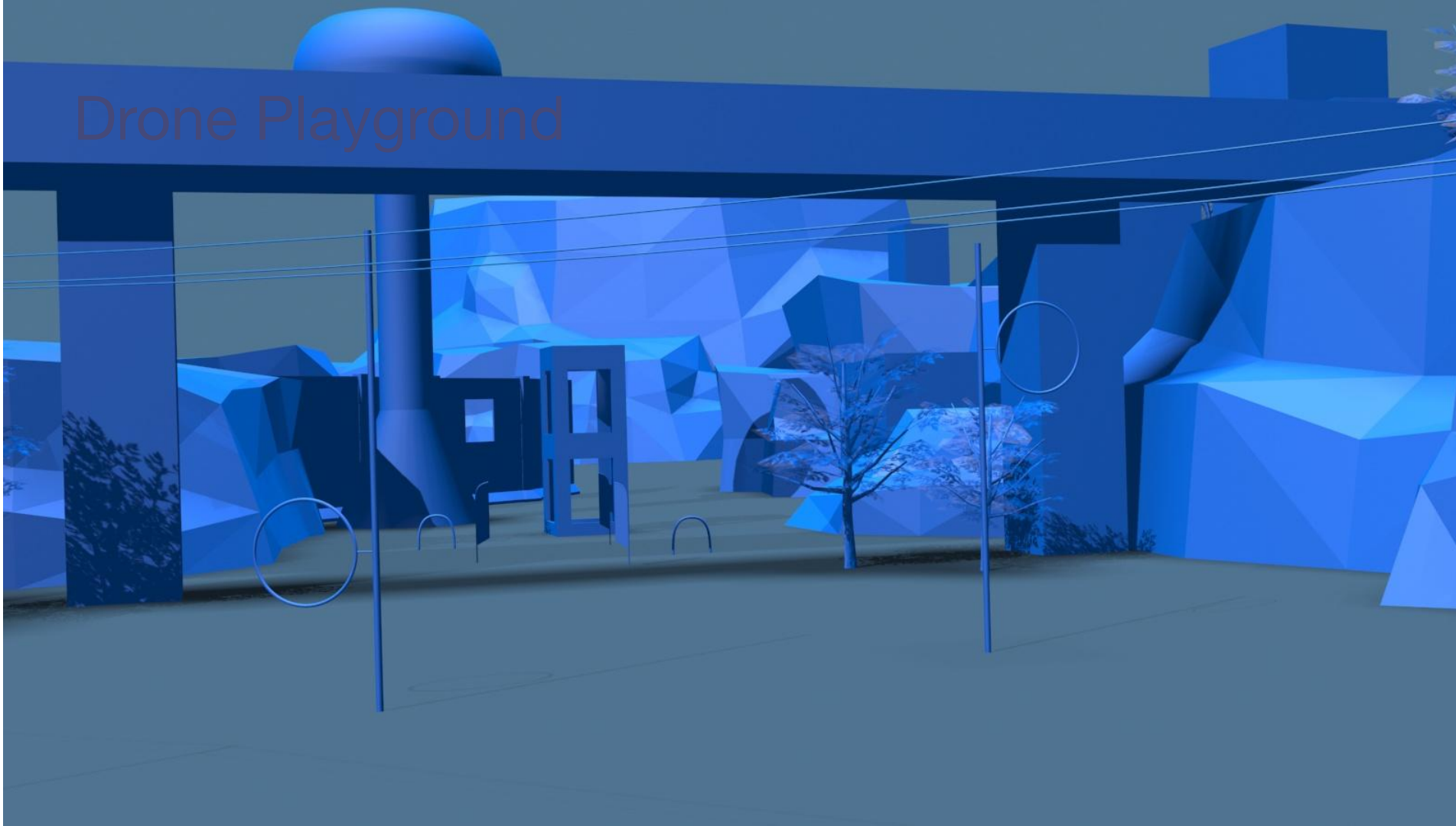


McMillan

Yosemite


Real Time Factor: 1.00
Sim Time: 1:10.557
Real Time: 1:10.674
Iteratio

San Carlos


Drone Playground

Ignition Rendering Capabilities

SITL Containers

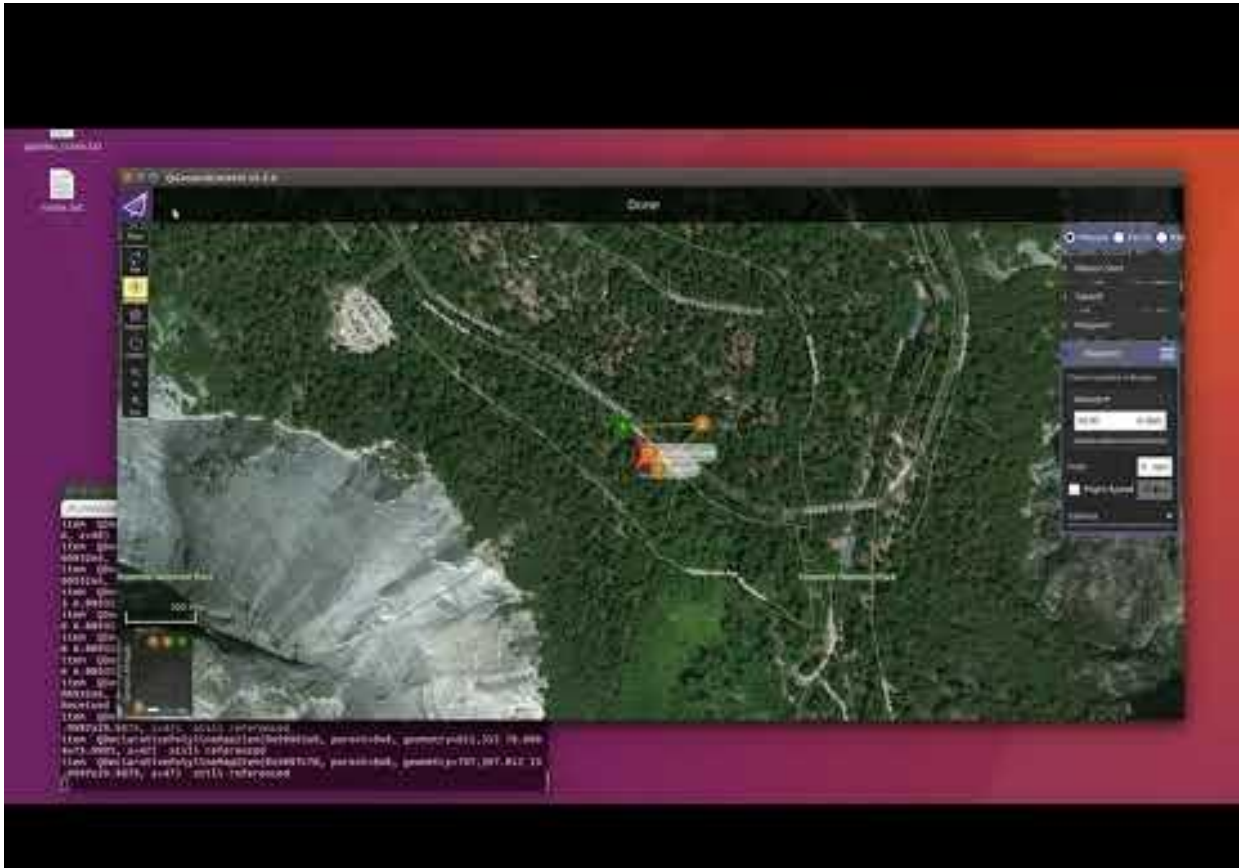
```
$ rocker --nvidia --x11 --home --user --pulse tfoote/drone_demo

==>

docker run -it   --rm    -v /home/tfoote:/home/tfoote   --runtime=nvidia
--security-opt seccomp=unconfined -v
/run/user/1000/pulse:/run/user/1000/pulse --device /dev/snd  -e
PULSE_SERVER=unix:/run/user/1000/pulse/native -v
/run/user/1000/pulse/native:/run/user/1000/pulse/native --group-add 29
-e DISPLAY -e TERM   -e QT_X11_NO_MITSHM=1   -e
XAUTHORITY=/tmp/.docker.xauth -v /tmp/.docker.xauth:/tmp/.docker.xauth
-v /tmp/.X11-unix:/tmp/.X11-unix   -v /etc/localtime:/etc/localtime:ro
rocker_tfoote/drone_demo_home_nvidia_pulse_x11_user
```

http://github.com/osrf/rocker

SITL Process

- Setup world
- Spawn a drone
- Spawn a 2nd drone
- Spawn a 3rd drone of another type
- Spawn a 4th drone
- Fly them all through qgc
- Introspect them via

Multiple Heterogeneous Drones



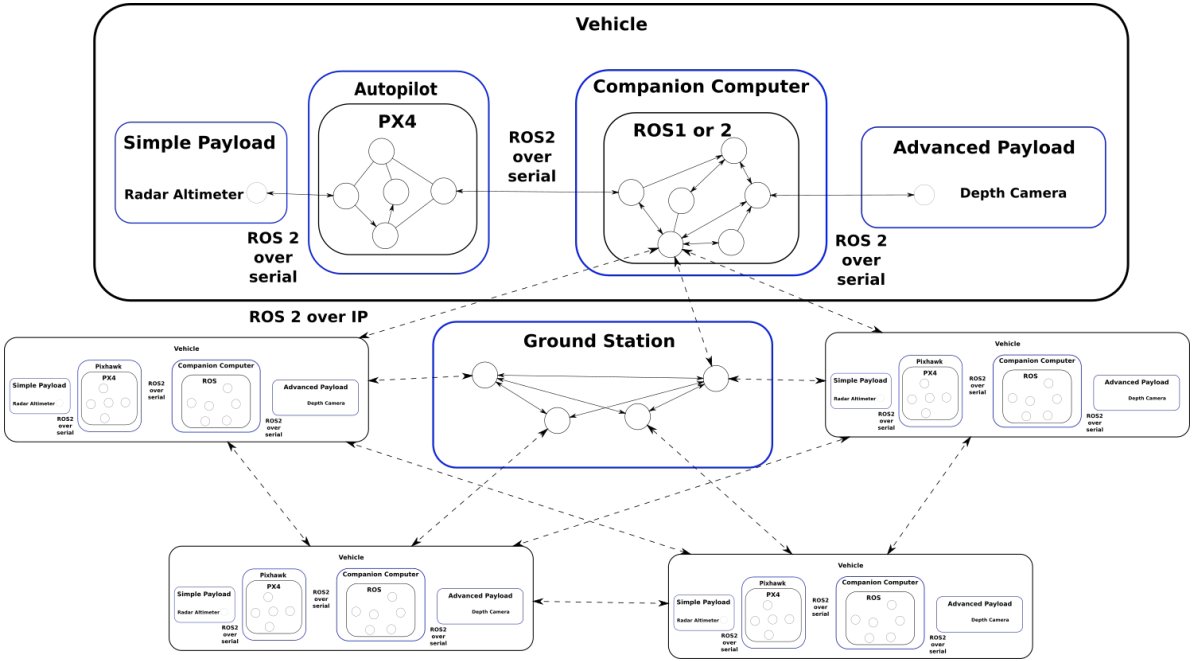
Formation Flying via mavros

Looking Ahead

Vision for future Drone communications

Integrated vision with ROS 2

- Introspection
- Modular
- Extendable
- Compatible



Working on standard messages

Request for feedback
REP 147: A Standard interface for Aerial Vehicles

http://www.ros.org/reps/rep-0147.html
https://github.com/ros-infrastructure/rep/pull/188

Thank You

Questions?