Choosing a ROS 2 and Gazebo Version

Structure your project

Creating Gazebo system plugins to control custom simulator behavior

Map ROS 2 and Gazebo topics/messages using a bridge

Creating or importing assets to be simulated

Running a simulation
Choosing a version

● Choose a version that matches any hard project constraints
  ○ e.g. OS version, ROS version, Gazebo version

● Levels of support
  ○ Official binary supported combinations
  ○ Official source supported combinations
  ○ Unsupported combinations
*Binary packages indicate recommended combinations*
binaries available
only from source

*Binary packages indicate recommended combinations
Noetic
Foxy
Galactic
Humble
Rolling

Citadel
Edifice
Fortress
Garden

*Binary packages indicate recommended combinations
Though Gazebo Classic is widely used, half of surveyed ROS2 users have already worked with Gazebo Fortress.
Structure your Project

To get started quickly, use the ros_gz_project_template!

- **ros_gz_example_application**: ROS 2 application libraries and nodes
- **ros_gz_exampleBringup**: ROS 2 launch files
- **ros_gz_example_description**: SDFFormat description of simulation assets
- **ros_gz_example_gazebo**: Gazebo specific system implementations

https://github.com/gazebosim/ros_gz_project_template
Writing Gazebo Systems

- Gazebo systems encapsulate all simulation-specific logic
- In contrast to Gazebo-classic, systems aren't specifically attached to models or worlds, but instead act upon entities and components
- Systems implement various interfaces to dictate behavior

**TIP:** Check to see what systems have been implemented before choosing to create one: https://github.com/gazebosim/gz-sim
The Entity Component Manager

- Every piece of simulation is an entity
- Each entity has one more more components attached to it
Writing Gazebo Systems

APIs system developers can implement:

- **Configure**
  - Called when plugin loaded, provided ECM and SDF attributes
- **PreUpdate**
  - Can mutate entities and components to set forces, torques, velocities
- **Update**
  - Physics update, generally should not be implemented by any other systems
- **PostUpdate**
  - Cannot mutate, but can read components and publish/send events
- **Reset**
  - Can be used to add reset-specific behavior
The Simulation Loop

Server
- Load SDF

Update Entities
Propagate Time
PreUpdate
Update
PostUpdate

System 1
- Configure
- Update

System 2
- Configure
- PreUpdate
- PostUpdate

System 3
- Configure
- PostUpdate
Building a Gazebo system in ROS 2

- Gazebo systems are shared libraries located via environment variables
- With ROS 2, use ament_hooks to install and locate Gazebo systems

# CMakeLists.txt

add_library(RosGzExampleSystem SHARED src/RosGzExampleSystem.cc)
install(TARGETS RosGzExampleSystem DESTINATION lib/${PROJECT_NAME})
ament_environment_hooks("${CMAKE_CURRENT_SOURCE_DIR}/hooks/${PROJECT_NAME}.dsv.in")

# Hooks file (.dsv format)

prepend-non-duplicate;GZ_SIM_RESOURCE_PATH;@CMAKE_INSTALL_PREFIX@/share/@PROJECT_NAME@/worlds
prepend-non-duplicate;GZ_SIM_SYSTEM_PLUGIN_PATH;lib/@PROJECT_NAME@/
Connecting Gazebo and ROS 2

Two primary mechanisms depending on your application:

Use `ros_gz_bridge` to dynamically connect topics between ROS 2 and Gazebo

Embed ROS 2 directly in a Gazebo system plugin
The bridge isolates Gazebo transport topics and ROS 2 topics. Each topic can be connected in one direction or bidirectionally.
Configuring ros_gz_bridge

---

- ros_topic_name: "/diff_drive/cmd_vel"
  gz_topic_name: "/model/diff_drive/cmd_vel"
  ros_type_name: "geometry_msgs/msg/Twist"
  gz_type_name: "gz.msgs.Twist"
  direction: ROS_TO_GZ

- ros_topic_name: "/diff_drive/odometry"
  gz_topic_name: "/model/diff_drive/odometry"
  ros_type_name: "nav_msgs/msg/Odometry"
  gz_type_name: "gz.msgs.Odometry"
  direction: GZ_TO_ROS

- ros_topic_name: "/diff_drive/scan"
  gz_topic_name: "/scan"
  ros_type_name: "sensor_msgs/msg/LaserScan"
  gz_type_name: "gz.msgs.LaserScan"
  direction: GZ_TO_ROS

bridge = Node(
    package='ros_gz_bridge',
    executable='parameter_bridge',
    arguments=[
      '/diff_drive/odometry@nav_msgs/msg/Odometry.gz.msgs.Odometry',
      '/diff_drive/cmd_vel@geometry_msgs/msg/Twist[gz.msgs.Twist',
      '/diff_drive/scan@sensor_msgs/msg/LaserScan.gz.msgs.LaserScan',
    ],
    output='screen'
)
Embedding ROS 2 in Gazebo

```cpp
void RosSystem::Configure(
    const gz::sim::Entity & entity,
    const std::shared_ptr<const sdf::Element> & element,
    gz::sim::EntityComponentManager & ecm,
    gz::sim::EventManager & eventManager)
{
    // Ensure that ROS is setup
    if (!rclcpp::ok()) {
        rclcpp::init(0, nullptr);
    }

    // Read configuration from SDF file
    auto node_name = element->Get<std::string>("node_name", "RosSystem").first;
    auto talker_topic = element->Get<std::string>("talker_topic", "talker").first;
    auto listener_topic = element->Get<std::string>("listener_topic", "listener").first;

    node_ = rclcpp::Node::make_shared(node_name);
    listener_sub_ = node_->create_subscription<std_msgs::msg::String>(listener_topic, 1, std::bind(&RosSystem::OnStringMessage, this, std::placeholders::_1));
    talker_pub_ = node_->create_publisher<std_msgs::msg::String>(talker_topic, 1);
}
```
Embedding ROS 2 in Gazebo

Gazebo

- Simulator
  - System Plugin
  - System Plugin

ROS

- ros_gz_bridge
  - gz-transport topic
  - ROS topic

ROS Nodes

Bridge

Direct Embedding

Gazebo

- Simulator
  - System Plugin
  - System Plugin

ROS 2 Nodes

- ROS 2 topic
  - ROS 2 topic
Bridge vs Embedding

**ros_gz_bridge**
- Limited to topics and services
+ Isolates Gazebo and ROS versions
+ Isolates Gazebo and ROS runtime
- Access to simulator state only through exposed transport topics

**Embedded Node**
+ More access to ROS primitives
- Couples Gazebo and ROS versions
- Couples Gazebo and ROS runtime
+ Direct access to simulator state

**Bonus:** In ROS 2, no roscore makes embedding easier than ever!
Simulation Assets

- Assets = models (URDF, SDF, etc), meshes and materials, world SDFs
- Can be installed as part of ROS 2 packages and exported as model:// or package: //
Simulated models and worlds are defined by SDF description files in Gazebo.

SDF files can be static (loaded from disk) or dynamically generated on the fly:
- Using template languages like ERB
- Using pySDF

Gazebo systems are attached via the sdf plugin tag.

```xml
<?xml version="1.0" ?>
<model name="diff_drive">
  <self_collide>true</self_collide>
  <link name="chassis">
    <pose>0.5 1.0 @(offset) 0.0 0.0 0.0</pose>
  </link>
</model>
```
Importing Existing Assets

- There is a library of SDF simulation assets on Fuel
- Support for other file formats:
  - USD
  - Mujoco
  - URDF
- Assimp loader for other mesh formats
  - Collada, Blender, glTF
Use assets in ROS

Use `sdformat_urdf` to share common assets:

```python
# Path to the robot model SDF format description
pkg_project_description = get_package_share_directory('ros_gz_example_description')
sdf_file = os.path.join(pkg_project_description, 'models', 'diff_drive', 'model.sdf')

# Read the description into a string
with open(sdf_file, 'r') as infp:
    robot_desc = infp.read()

# Get the parser plugin convert sdf to urdf using robot_description topic
robot_state_publisher = Node(
    package='robot_state_publisher',
    executable='robot_state_publisher',
    name='robot_state_publisher',
    parameters=[
        {'use_sim_time': True},
        {'robot_description': robot_desc},
    ]
)
```
Using Assets in ROS

Use sdformat Urdf to share common assets:
Running a simulation
Examples of successful integrations

DARPA SubT Challenge

MBZIRC UAV and USV Challenge

MBARI Wave Energy Converter

TurtleBot 4 Simulator
Please fill out the ROS and Gazebo User Survey!
Thank you!
Any questions?

https://github.com/gazebo simul/ros_gz
https://github.com/gazebo simu/ros_gz_project_template
Migration Notes for Gazebo - Classic to Gazebo Sim

// OLD
class GAZEBO_VISIBLE ArduPilotPlugin:
  public ModelPlugin

// NEW
class GZ_SIM_VISIBLE ArduPilotPlugin:
  public gz::sim::System,
  public gz::sim::ISystemConfigure,
  public gz::sim::ISystemPostUpdate,
  public gz::sim::ISystemPreUpdate
Migration Notes for Gazebo - Classic to Gazebo Sim

// OLD
virtual void Load(
    physics::ModelPtr _model,
    sdf::ElementPtr _sdf);

// NEW
void Configure(const gz::sim::Entity &_entity,
               const std::shared_ptr<const sdf::Element> &_sdf,
               gz::sim::EntityComponentManager &_ecm,
               gz::sim::EventManager &_eventMgr);
Migration Notes for Gazebo - Classic to Gazebo Sim

// OLD
void OnUpdate()

// NEW
void PreUpdate(const gz::sim::UpdateInfo &_info,
               gz::sim::EntityComponentManager &_ecm);

void PostUpdate(const gz::sim::UpdateInfo &_info,
                const gz::sim::EntityComponentManager &_ecm);