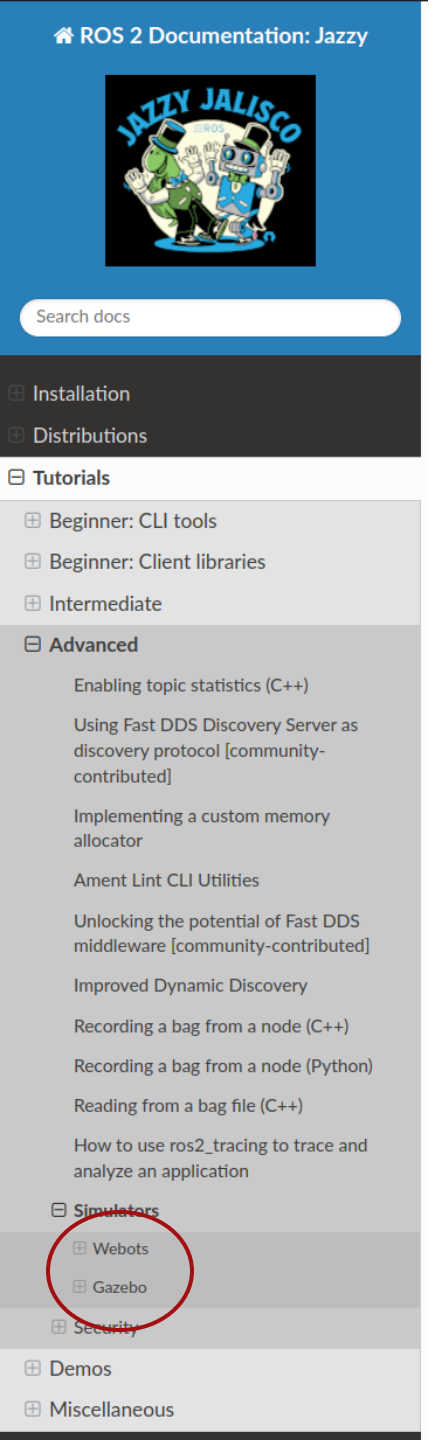
ROBOTEC AI

HUMAN CENTERED ROBOTICS

O3DE: Creating realistic simulations with an open-source game engine

Jan Hańca

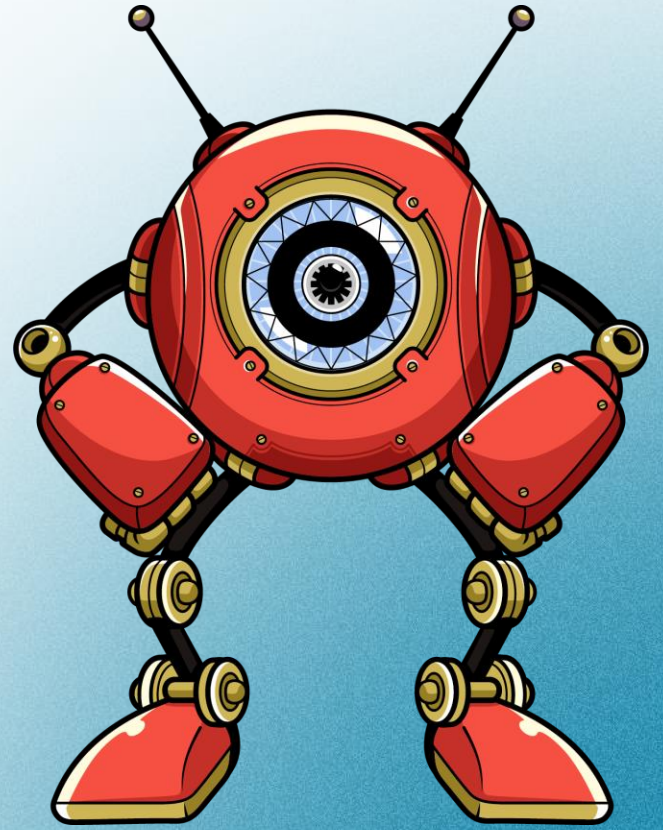
Introduction to O3DE: about



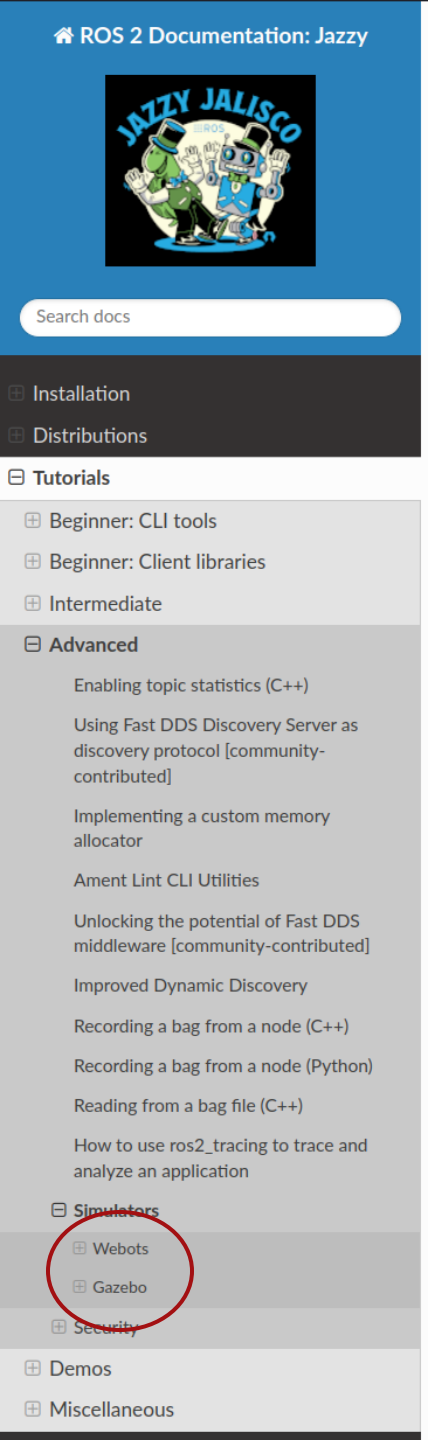
Introduction to O3DE: about

Can I use it for simulation?

- Yes!
- Open-source (Apache 2.0; MIT)
- Governed by O3DF (Linux Foundation)



O3DE



Introduction to O3DE: about

ROBOTEC AI

Can I use it for simulation?

- Yes!
- Open–source (Apache 2.0; MIT)
- Governed by O3DF (Linux Foundation)

Why O3DE?

- High–Quality Graphics (photorealistic rendering, physically–based rendering, real–time global illumination, support for ray tracing)
- Extensive Toolset (ScriptCanvas, animation editor, terrain editor, ...)
- Interoperable (C++, LUA, Python*, ScriptCanvas)
- Cross Platform (Windows, Linux, macOS, Android, iOS, support for AR/VR/XR)
- Modularity (simulation gems!)

Introduction to O3DE: demo

Introduction to O3DE: demo

Introduction to O3DE: demo

Getting Started with O3DE: resources

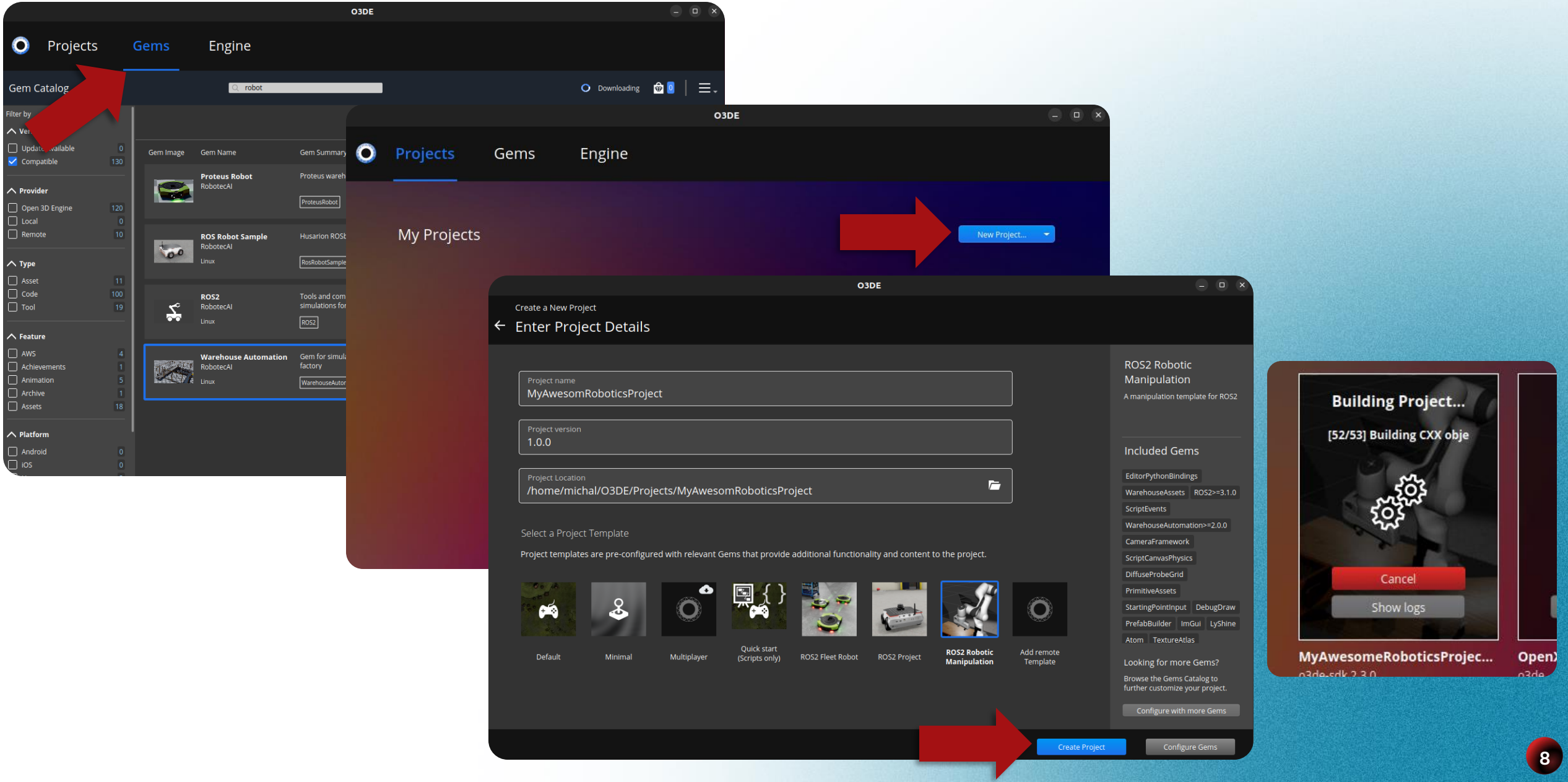
- O3DE: https://o3de.org or https://github.com/o3de/o3de
- Extras (simulation Gems, templates): https://github.com/o3de/o3de-extras

- Demo ROSCon2021: https://github.com/o3de/RobotVacuumSample
- Demo ROSCon2022: https://github.com/o3de/ROSConDemo
- Demo ROSCon2023: https://github.com/RobotecAI/ROSCon2023Demo
- Embodied AI things (with or without O3DE): https://github.com/RobotecAI/rai

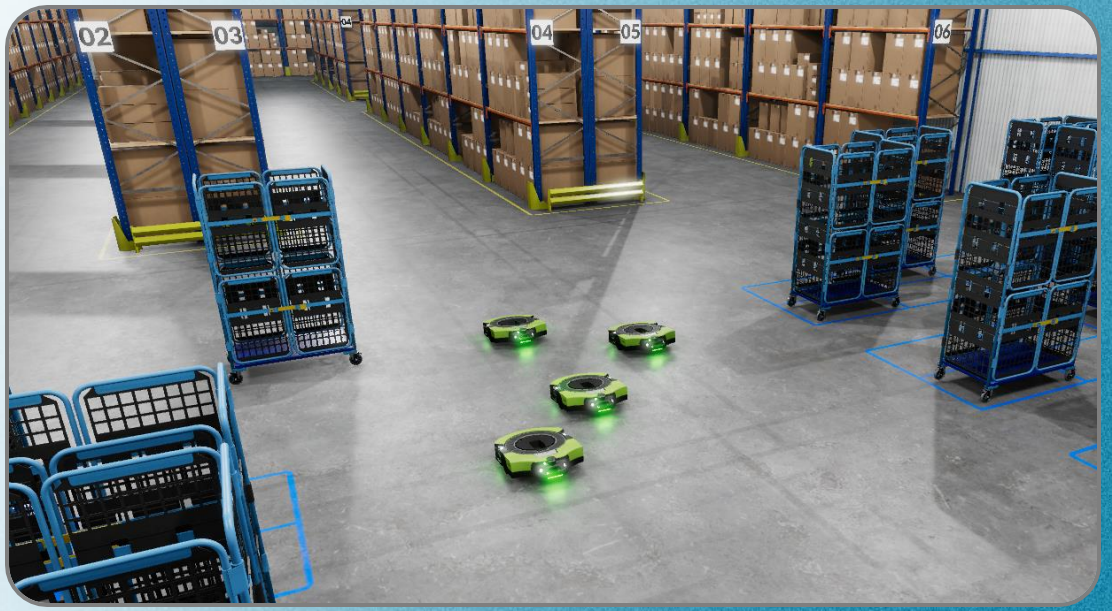
- Robotec GPU Lidar Gem: https://github.com/RobotecAI/o3de-rgl-gem

- Robotec.ai Github: https://github.com/RobotecAI

Getting Started with O3DE

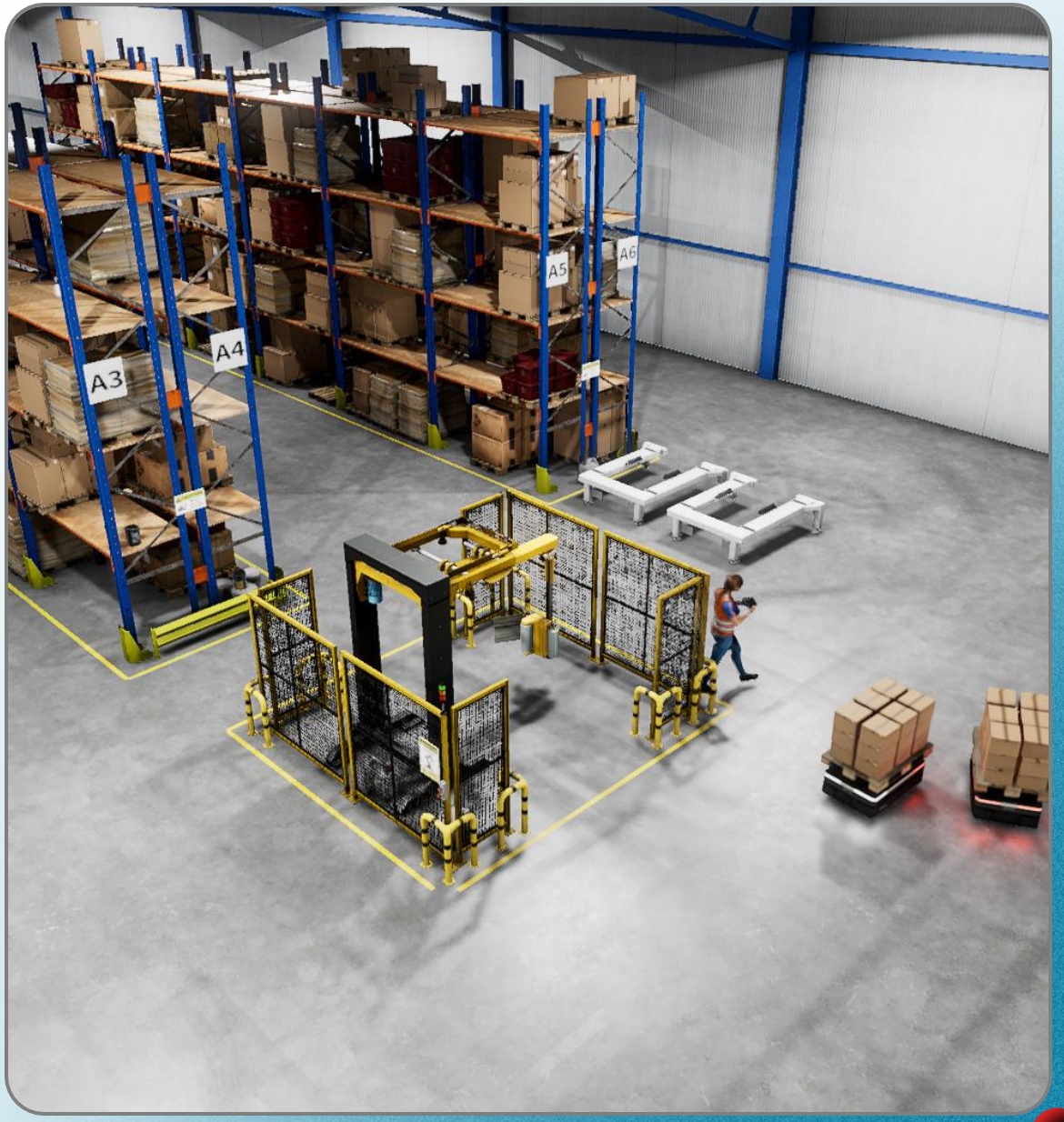


ROS 2 Ecosystem in O3DE: templates



ROS 2 Ecosystem in O3DE: intro

- The Gem creates a ROS 2 *node* (your simulation will not use any bridges) – a singleton
- The Gem is a subject to configuration through settings such as Environment Variables
- Base component: *ROS2FrameComponent*



ROS 2 Ecosystem in O3DE: sensors; control

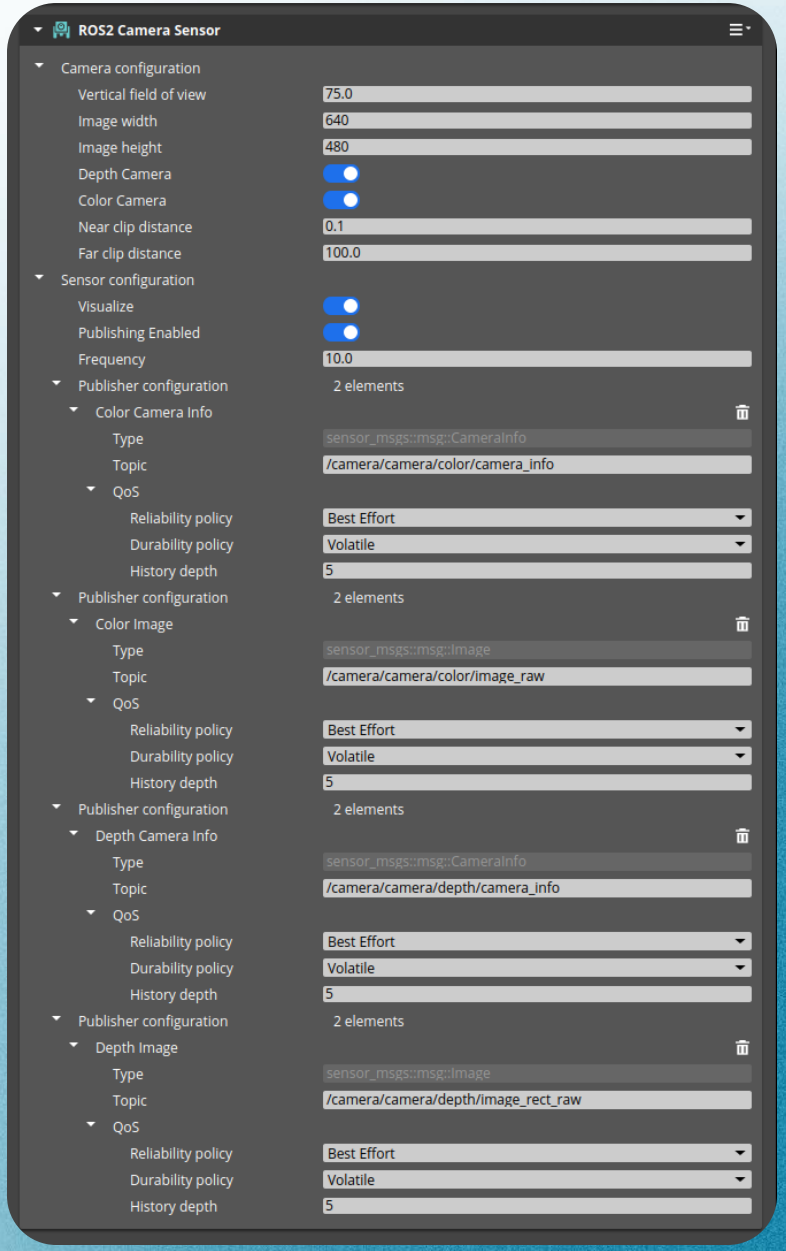
Sensors within ROS 2 Gem:

- Camera
- Contact
- GNSS
- IMU
- Lidar (3D/2D)
- Odometry

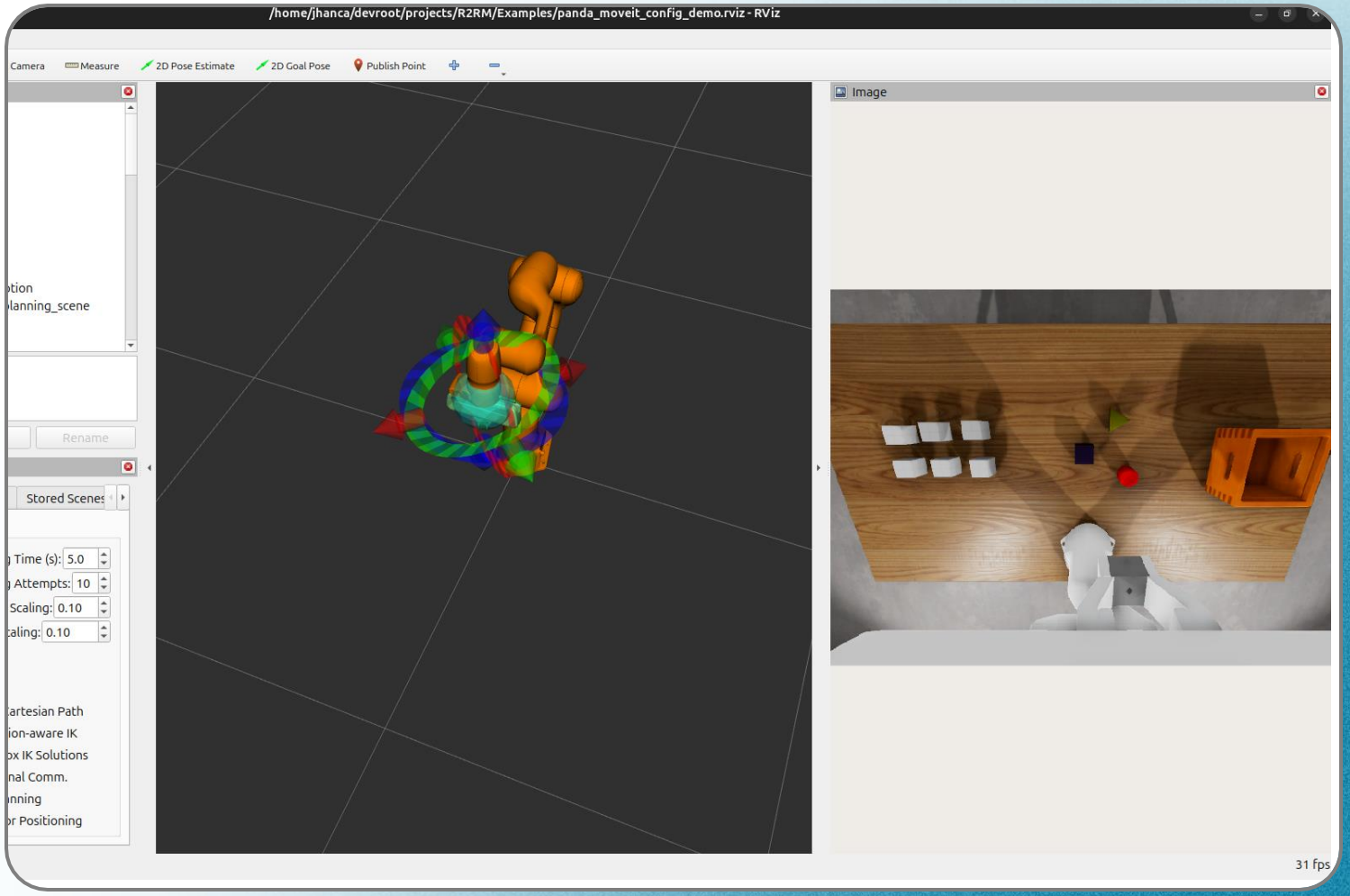
Robot Control

- Ackermann
- Rigid body
- Twist

Joints (control and state), Spawner, Georeference, ...



ROS 2 Ecosystem in O3DE: sensors; control



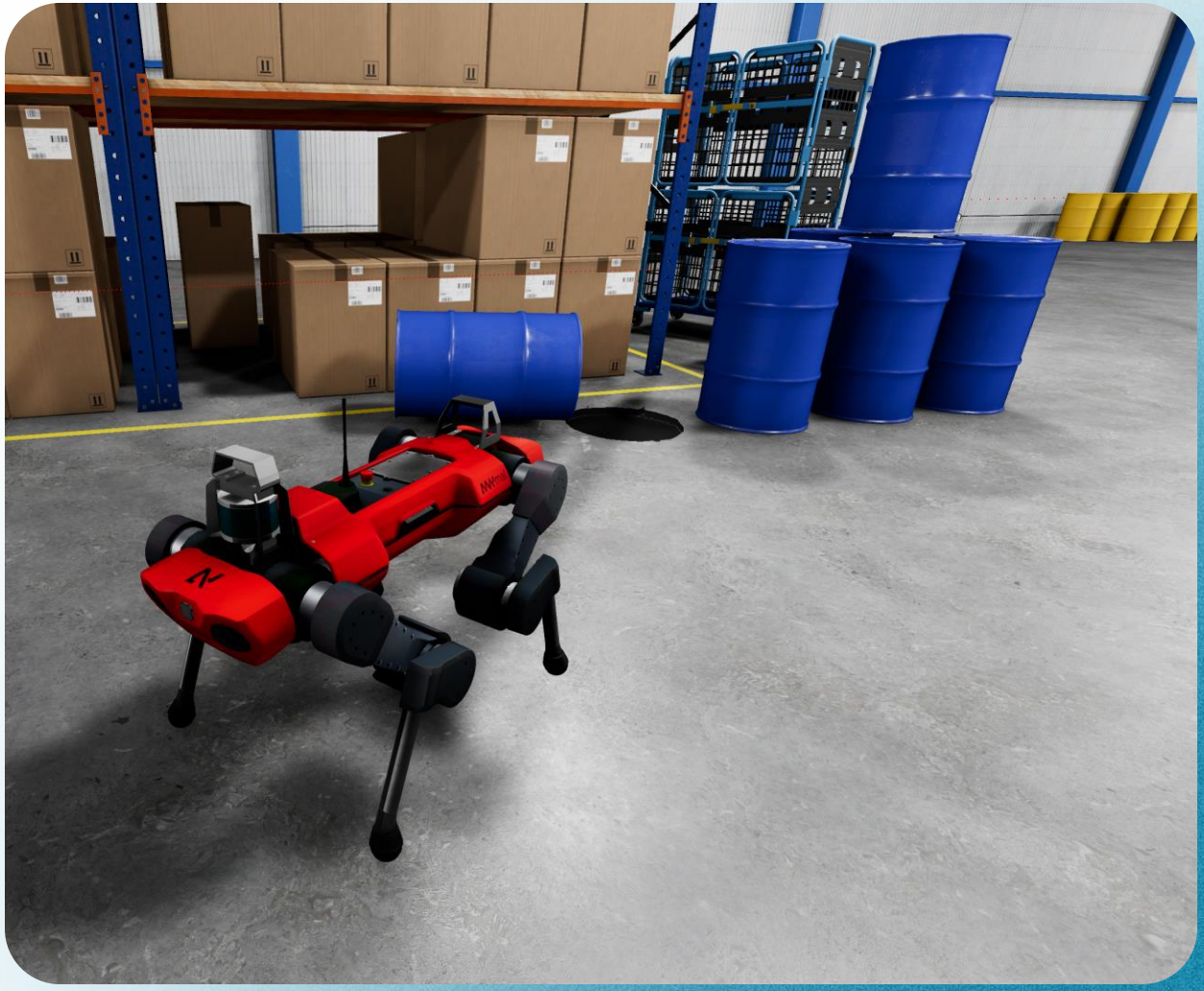
ROS 2 Ecosystem in O3DE: SDF/URDF import

SDF/URDF import:

- ROS 2 sourcing supported
- *xacro* supported
- Multiple mesh formats (*assimp*) supported
- Standard sensors (and control tools) are supported

Turtlebot4: full tutorial in the docs

USD format support: ongoing work

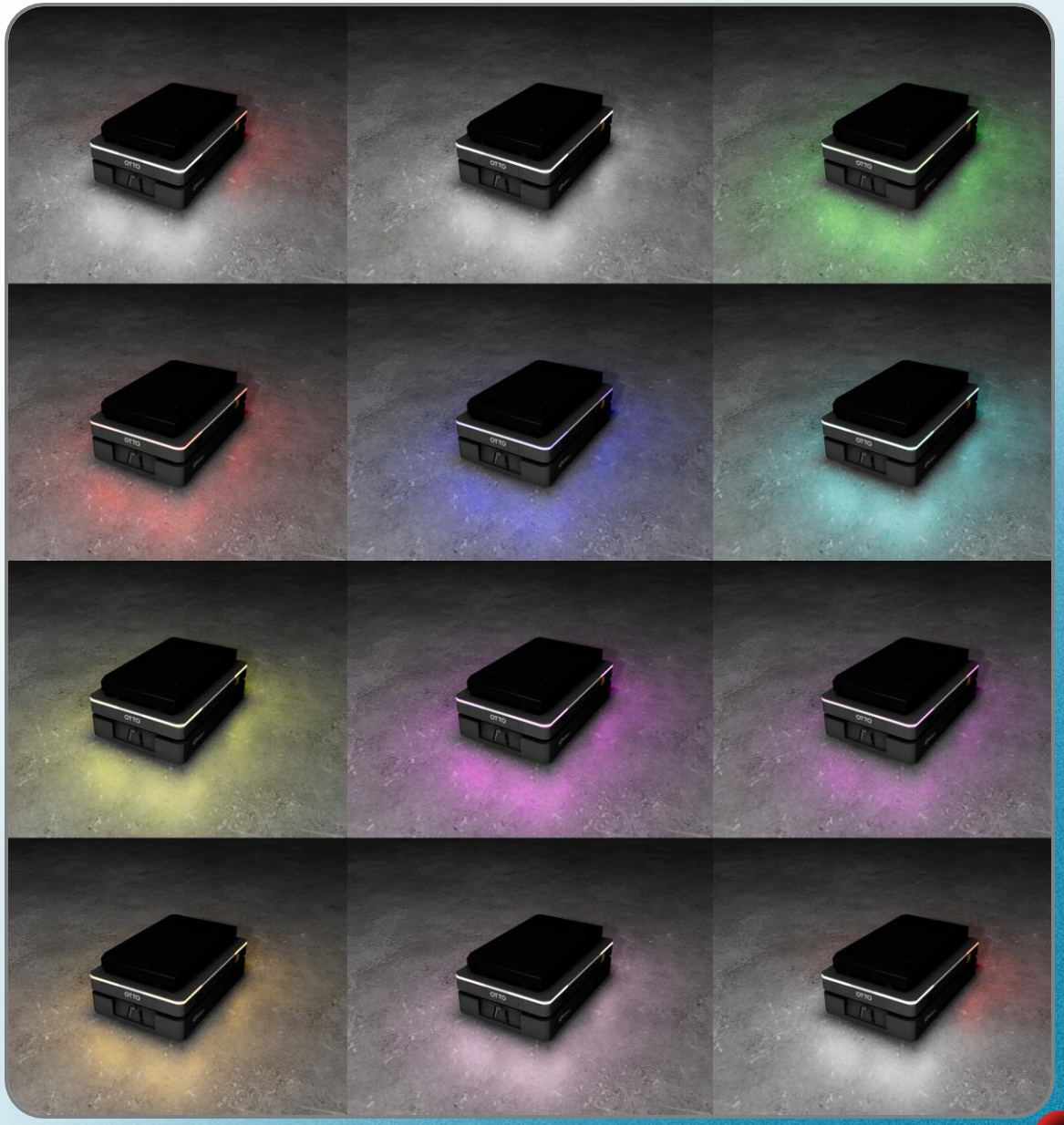


C++ Example

Implement a Gem, that:

- publishes current light intensity via a ROS 2 topic
- subscribes to a ROS 2 topic to allow intensity changes
- creates a service to change the intensity

Don't get fooled by the picture ;–)



C++ Example

- Link to ROS 2 Gem in your gem's description
- Link to ROS 2 Gem in your `CMakeLists.txt`
- Link to ROS 2 packages to access the framework

```
34
35  # The ${gem_name}.Private.Object target is an internal target
36  # It should not be used outside of this Gems CMakeLists.txt
37  ly_add_target(
38      NAME ${gem_name}.Private.Object STATIC
39      NAMESPACE Gem
40      FILES_CMAKE
41          jho3detestgem_private_files.cmake
42          ${pal_dir}/jho3detestgem_private_files.cmake
43      TARGET_PROPERTIES
44          O3DE_PRIVATE_TARGET TRUE
45      INCLUDE_DIRECTORIES
46          PRIVATE
47              Include
48              Source
49      BUILD_DEPENDENCIES
50          PUBLIC
51              AZ::AzCore
52              AZ::AzFramework
53              Gem::ROS2.Static
54              Gem::Atom_AtomBridge.Static
55              Gem::CommonFeaturesAtom.Static
56  )
57
58  # Request ROS2 packages to be included
59  target_depends_on_ros2_packages(${gem_name}.Private.Object rclcpp tf2 std_msgs std_srvs)
60
```

C++ Example

Header implementation:

- Include ROS 2–related headers
- Define your publisher, subscriber and service

- Additional tools such as `ROS2::TopicConfiguration` are provided by the ROS 2 Gem

```
#include <ROS2/Communication/TopicConfiguration.h>
#include <rclcpp/rclcpp.hpp>
#include <std_msgs/msg/float32.hpp>
#include <std_srvs/srv/trigger.hpp>

namespace JHO3DETestGem
{
    class TestComponent
        : public AZ::Component
        , public AZ::TickBus::Handler
    {
    public:
        [redacted]

    private:
        ////////////////////////////////////////////////////////////////////////
        // AZ::TickBus::Handler overrides
        void OnTick(float deltaTime, AZ::ScriptTimePoint time) override;
        ////////////////////////////////////////////////////////////////////////

        AZStd::string m_serviceName{ "service_name" };
        ROS2::TopicConfiguration m_subscriberConfiguration;
        ROS2::TopicConfiguration m_publisherConfiguration;

        rclcpp::Service<std_srvs::srv::Trigger>::SharedPtr m_lightsOnService;
        rclcpp::Subscription<std_msgs::msg::Float32>::SharedPtr m_subscriber;
        rclcpp::Publisher<std_msgs::msg::Float32>::SharedPtr m_publisher;

        bool SetLightsIntensity(const float targetIntensity);
        AZ::EntityId m_lightsEntityId;
    };
} // namespace JHO3DETestGem
```

C++ Example

- Additional tools finding the namespace are provided by the Gem
- Creating services is unchanged compared to ROS 2 framework

```cpp
void TestComponent::Activate()
{
    auto ros2Node = ROS2::ROS2Interface::Get()->GetNode();
    if (!ros2Node)
    {
        AZ_Error("TestComponent", false, "ROS2 node is not available. ROS 2 services will not be created.");
        return;
    }

    auto ros2Frame = ROS2::Utils::GetGameOrEditorComponent<ROS2::ROS2FrameComponent>(GetEntity());
    if (!ros2Frame)
    {
        AZ_Error("TestComponent", false, "ROS2 frame is not available. ROS 2 services will not be created.");
        return;
    }

    AZStd::string serviceName = ROS2::ROS2Names::GetNamespacedName(ros2Frame->GetNamespace(), m_serviceName);
    m_lightsOnService = ros2Node->create_service<std_srvs::srv::Trigger>(
        serviceName.c_str(),
        [this](
            [[maybe_unused]] const std::shared_ptr<std_srvs::srv::Trigger::Request> request,
            std::shared_ptr<std_srvs::srv::Trigger::Response> response)
        {
            if (!m_lightsEntityId.IsValid())
            {
                response->success = false;
                response->message = "Cannot turn on lights, entity ID not found.";
                return;
            }

            constexpr float targetIntensity = 500.0f;
            response->success = SetLightsIntensity(targetIntensity);
            if (response->success)
            {
                response->message = "Lights turned on.";
            }
            else
            {
                response->message = "Failed to turn the lights on.";
            }
        });
```

C++ Example

- Creating publishers and subscribers is unchanged compared to ROS 2 framework

```
102         m_subscriber = ros2Node->create_subscription<std_msgs::msg::Float32>(
103             ROS2::ROS2Names::GetNamespacedName(ros2Frame->GetNamespace(), m_subscriberConfiguration.m_topic).c_str(),
104             m_subscriberConfiguration.GetQoS(),
105             [this](const std_msgs::msg::Float32::SharedPtr msg)
106             {
107                 SetLightsIntensity(msg->data);
108             });
109
110         m_publisher = ros2Node->create_publisher<std_msgs::msg::Float32>(
111             ROS2::ROS2Names::GetNamespacedName(ros2Frame->GetNamespace(), m_publisherConfiguration.m_topic).c_str(),
112             m_publisherConfiguration.GetQoS());
```

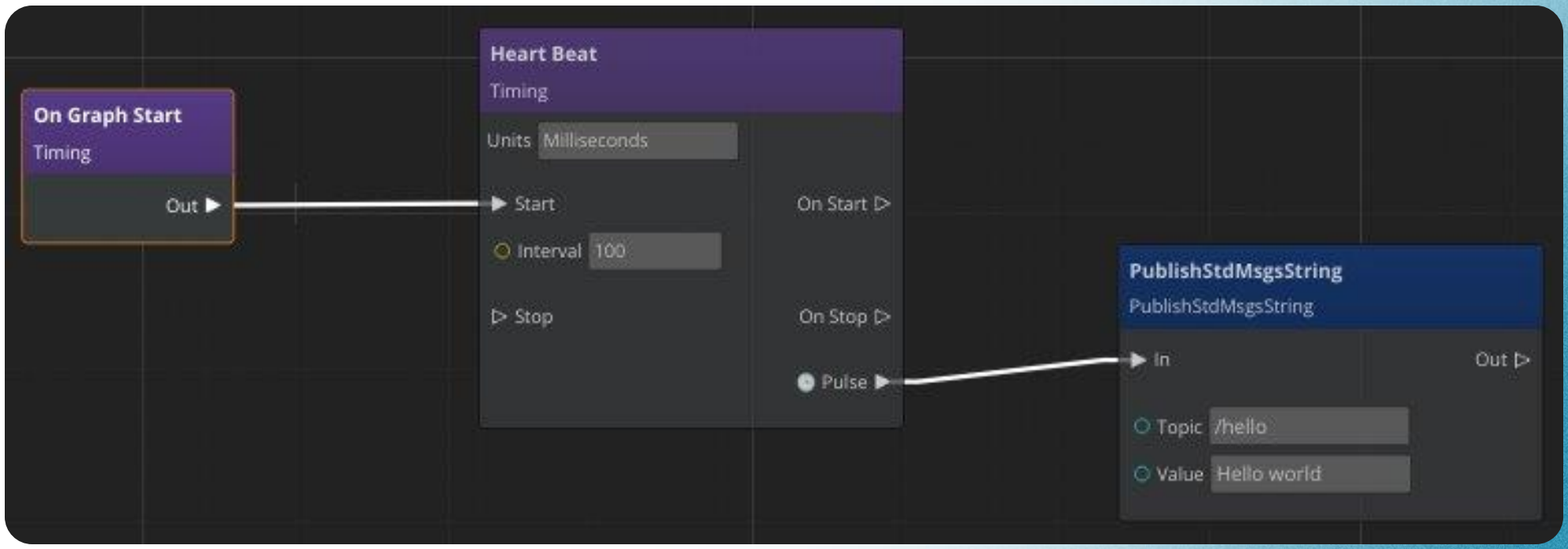
```
148         // Note: demo only, it makes not sense to publish info every OnTick()
149         float currentIntensity = -100.0f;
150         AZ::Render::AreaLightRequestBus::EventResult(currentIntensity, m_lightsEntityId, &AZ::Render::AreaLightRequests::GetIntensity);
151
152         std_msgs::msg::Float32 msg;
153         msg.data = currentIntensity;
154
155         m_publisher->publish(msg);
```

Bonus: Interoperability

- ROS2ScriptIntegration Gem: Hello from Lua:

```
PublisherRequestBus.Broadcast.PublishStdMsgString("/hello", "Hello World")
```

- ROS2ScriptIntegration Gem: Hello from ScriptCanvas:



Introduction to O3DE: demo

Contact

LinkedIn

Github

- Jan Hańca
- jan@hanca.pl

- https://discord.com/invite/o3de
- https://github.com/o3de