Gazebo

Verónica E. Arriola-Rios

Robótica móvil

22 de octubre de 2024

Gazebo

Instalación de Gazebo

- Instalar **Gazebo harmonic(LTS)** según
  https://gazebosim.org/docs/harmonic/install_ubuntu/
- Probar instalación https://gazebosim.org/docs/harmonic
- Ejecutar demos:

```
$ gz sim shapes.sdf
```

- Tutoriales https://gazebosim.org/docs/harmonic/tutorials/

Integración con ROS

- Para la integración de ROS con gazebo se requiere `ros-gz`

```
$ sudo apt install ros-jazzy-ros-gz
```

Demos

Mundo vacío

- Gazebo **harmonic** incluye algunos mundos de muestra en el directorio:

```
/usr/share/gz/gz-sim8/worlds/
```

- El mundo vacío carga *plug in*s que permiten enviar mensajes para aparecer objetos en la escena.

Código: Terminal 1

```
$ gz sim empty.sdf
```

Aparecer (*Spawn*)

Temas

Mundo vacío

- Veamos los servicios del mundo vacío:

Código: Terminal 1

```
$ gz sim empty.sdf
```

- Para listar los servicios disponibles y sus requerimientos se usa:

Código: Terminal 2

```
$ gz service -l
$ gz service -is /world/empty/create
```

En el ejemplo anterior se obtiene:

```
Service providers [Address, Request Message Type, Response
    ➥Message Type]:
  tcp://192.168.1.70:46647, gz.msgs.EntityFactory, gz.msgs.
    ➥Boolean
```

Temas

3 Aparecer (*Spawn*)
- Servicio
- sdf
- xacro y urdf de ROS

Aparecer objetos directamente

```
$ gz service -s /world/empty/create \
--reqtype gz.msgs.EntityFactory \
--reptype gz.msgs.Boolean \
--timeout 300 \
--req 'sdf: "<?xml version=\"1.0\" ?>'\
'<sdf version=\"1.6\">'\
'<model name=\"spawned_sphere_model\"><link name=\"link\">'\
'  <visual name=\"sphere_visual\">'\
'    <geometry>'\
'      <sphere><radius>1.0</radius></sphere>'\
'    </geometry>'\
'  </visual>'\
'  <collision name=\"sphere_collision\">'\
'    <geometry><sphere><radius>1.0</radius></sphere></
    ➥geometry>'\
'  </collision></link></model></sdf>" '\
'pose: {position: {z: 10}} '\
'name: "black_sphere" '\
'allow_renaming: true'
```

https://answers.gazebosim.org//question/26037/how-to-spawn-sdf-object-in-ignition-gazebo/

Aparecer objetos directamente

Ejemplo anterior, versión en una línea:

```
$ gz service -s /world/empty/create --reqtype gz.msgs.
  ➥EntityFactory --reptype gz.msgs.Boolean --timeout 300
  ➥--req 'sdf: "<?xml version=\"1.0\" ?><sdf version
  ➥=\"1.6\"><model name=\"spawned_model\"><link name=\"
  ➥link\"><visual name=\"visual\"><geometry><sphere><
  ➥radius>1.0</radius></sphere></geometry></visual><
  ➥collision name=\"visual\"><geometry><sphere><radius
  ➥>1.0</radius></sphere></geometry></collision></link></
  ➥model></sdf>" pose: {position: {z: 10}} name: "
  ➥new_name" allow_renaming: true'
```

Aparecer objetos de un sdf

- En lugar de escribir la descripción `sdf`, es posible pasar la dirección absoluta de un archivo como parámetro.
- El archivo `sdf` sólo puede contener un elemento:
  - `<model>`
  - `<light>`
  - `<actor>`

Ejemplo:

```
$ gz service -s /world/empty/create --reqtype gz.msgs.
    ➥EntityFactory --reptype gz.msgs.Boolean --timeout 1000
    ➥ --req 'sdf_filename:"/home/muyser/blue_robot.sdf",␣
    ➥name:␣"robot_model"'

$ $(ros2 pkg prefix --share ros_gz_sim_demos)
/opt/ros/jazzy/share/ros_gz_sim_demos

$ gz service -s /world/empty/create --reqtype gz.msgs.
    ➥EntityFactory --reptype gz.msgs.Boolean --timeout 1000
    ➥ --req 'sdf_filename:"/opt/ros/jazzy/share/
    ➥ros_gz_sim_demos/models/vehicle/model.sdf",␣name:␣"
    ➥vehicle_model"'
```

Temas

3 Aparecer (*Spawn*)
- Servicio
- sdf
- xacro y urdf de ROS

Aparecer objetos de un xacro de ros I

1. Gazebo no puede abrir archivos `.xacro` pero los `.urdf` ya expandidos sí.
2. Para generar el `.urdf` se debe ejecutar el comando `xacro`. Si el archivo no importa otros recursos el comando es:

```
$ xacro <archivo>.xacro > <archivo>.urdf
```

   1. Si el `.xacro` importa recursos como mallas ubicadas en otra dirección, el comando debe ejecutarse justo afuera del directorio del paquete que contiene estos recursos.

```
$ xacro paquete/xacro/<archivo>.xacro > paquete/
    ➥urdf/<archivo>.urdf
```

   2. El `.urdf` tendrá direcciones relativas a los recursos importados.

Aparecer objetos de un xacro de ros II

3. Al aparecer estos modelos en gazebo, las direcciones serán calculadas con respecto al directorio donde se ejecute `ign gazebo`. Alternativamente se puede asignar la variable de ambiente

```
$ export IGN_GAZEBO_RESOURCE_PATH="$HOME/<ruta␣a␣
    ➥recursos>"
```

Ejemplo: Kobuki

- Comenzar con el paquete `kobuki_description` de
  https://github.com/kobuki-base/kobuki_ros/tree/devel/kobuki_description
- Clonar del repo dentro de `src` en un espacio de trabajo, compilarlo (`colcon build`) y cargar el espacio (`./install/setup.bash`).
- Desde afuera del directorio ejecutar

```
$ xacro kobuki_description/urdf/kobuki_standalone.urdf.
  ➥xacro > kobuki_description/urdf/kobuki_standalone.
  ➥urdf
$ ign gazebo empty.sdf
```

- Desde otra terminal enviar la solicitud de servicio:

```
ign service -s /world/empty/create --reqtype ignition
  ➥.msgs.EntityFactory --reptype ignition.msgs.
  ➥Boolean --timeout 300 --req 'sdf_filename:"
  ➥kobuki_description/urdf/kobuki_standalone.urdf",␣
  ➥pose:␣{position:␣{z:␣10}}␣name:␣"kobuki"'
```

(la posición es opcional)

Referencias I

 (s.f.). URL: https://gazebosim.org/docs/fortress.