UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Herramientas de ROS2 y el simulador Turtlesim

Eugenio Ivorra

Objetivos

- En esta práctica se aprenderá a usar un simulador de robótica 2D para el aprendizaje de conceptos de ROS2 y las herramientas de consola para la depuración:
  - Conocer y utilizar el simulador turtlesim.
  - Depurar nodos y topics con instrucciones de consola.
  - Aprender a usar el mensaje de velocidad.

Índice general

1 Simulador 2D Turtlesim

1.1 Introducción al paquete Turtlesim

El simulador `turtlesim` es un paquete en ROS 2 diseñado para ser una herramienta educativa y de demostración. Ofrece una introducción simplificada a ROS 2 y a sus conceptos fundamentales a través de un ambiente gráfico 2D (figura 1.1). En este ambiente, se muestra una tortuga que los usuarios pueden controlar mediante comandos de ROS 2.

Las características Clave de Turtlesim son las siguientes:

- **Entorno de Simulación 2D:** Se presenta una tortuga en un espacio bidimensional que el usuario puede navegar.
- **Control mediante ROS 2:** Utiliza nodos, topics y servicios de ROS 2 para controlar la tortuga, lo que permite experimentar con estos conceptos fundamentales en un entorno controlado.
- **Interactividad:** Capacidad de controlar la tortuga mediante el teclado o incluso programar un nodo para controlarla automáticamente.
- **Personalizable:** Permite añadir más tortugas al ambiente, cada una controlada por su propio nodo de ROS 2.
- **Facilidad de Instalación:** Es un paquete ligero y fácil de instalar.

1.1.1 Instalación de Turtlesim

Si aún no tiene instalado el paquete *turtlesim*, puede instalarlo fácilmente. Abra un terminal y ejecute el siguiente comando:

```
sudo apt update
sudo apt install ros-humble-turtlesim
```

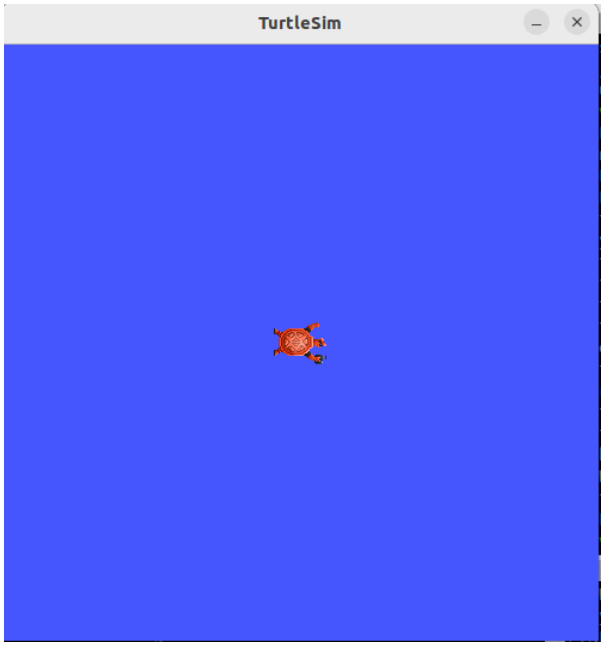

Figura 1.1: Ventana gráfica Turtlesim

1.1.2 Ejecutando Turtlesim

Una vez instalado, necesitamos decirle a nuestro terminal que hemos instalado un nuevo paquete. Para ello, utilizamos el siguiente comando:

```
source /opt/ros/humble/setup.bash
```

Ahora podemos lanzar *turtlesim*:

```
ros2 run turtlesim turtlesim_node
```

Esto abrirá una ventana 2D con una tortuga en el medio.

1.1.3 Controlando la Tortuga

Podemos controlar el robot tortuga desde la línea de comandos con los siguientes comandos:

```
# Avanzar
ros2 topic pub /turtle1/cmd_vel geometry_msgs/Twist '{linear: {x: 1.0}}'

# Girar a la izquierda
ros2 topic pub/turtle1/cmd_vel geometry_msgs/Twist '{angular: {z: 1.0}}'

# Girar a la derecha
ros2 topic pub /turtle1/cmd_vel geometry_msgs/Twist '{angular: {z: -1.0}}'
```

Prueba a dibujar un cuadrado y un circulo utilizando los comandos anteriores.

Otra opción para controlar la tortuga es utilizar un paquete de `teleop` que lo que hace es enviar mensajes de `cmd_vel` mediante el uso del teclado. Para ello, abra un nuevo terminal y ejecute:

```
ros2 run turtlesim turtle_teleop_key
```

Ahora puede usar las teclas de flecha del teclado para mover la tortuga en la ventana.

1.1.4 Comunicación entre Nodos

Si ejecuta `ros2 node list`, verá los dos nodos: *turtlesim* y *turtle_ teleop_ key*. Estos nodos se comunican entre sí para controlar el movimiento de la tortuga.

1.1.5 Renombrando Nodos

También puedes renombrar los nodos usando el argumento `-ros-args -r`. Por ejemplo:

```
ros2 run turtlesim turtlesim_node --ros-args -r __node:=my_turtle
```

Esto cambiará el nombre del nodo a *my_ turtle*.

1.2 Rqt

En ROS2, `rqt` es una herramienta de software que proporciona una interfaz gráfica de usuario (GUI) y facilita la interacción con el sistema de robots, permitiendo a los usuarios visualizar la información del nodo, los topics, los servicios y otros aspectos del sistema ROS2. `rqt` puede ser utilizado para monitorizar, controlar y debuggear de una manera más intuitiva y visual, ofreciendo varios plugins que los usuarios pueden emplear para adaptar la interfaz a sus necesidades específicas.

Al ejecutar `rqt` por primera vez, la ventana estará en blanco. No hay problema; simplemente seleccione *Plugins* > *Servicios* > *Llamador de Servicio* en la barra de menú en la parte superior.

Puede tomar algún tiempo para que `rqt` localice todos los plugins. Si haces clic en *Plugins* pero no ves *Servicios* u otras opciones, deberías cerrar `rqt` e ingresar el comando `rqt -force-discover` en tu terminal.

Teniendo lanzado el simulador `turtlesim`. Usa el botón de refrescar a la izquierda de la lista desplegable de *Service* para asegurarte de que todos los servicios de tu nodo `turtlesim` están disponibles.

Haz clic en la lista desplegable de *Service* para ver los servicios de `turtlesim`, y selecciona el servicio `/spawn`.

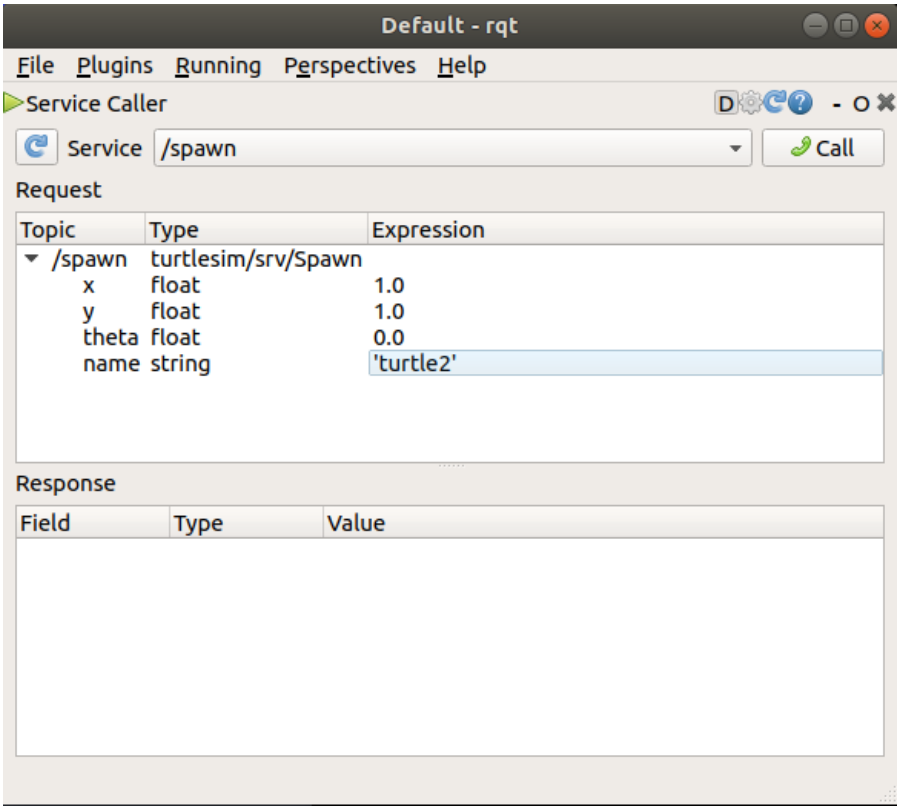

Figura 1.2: Herramienta rqt

Vamos a usar `rqt` para llamar al servicio `/spawn`. Puedes adivinar por su nombre que `/spawn` creará otra tortuga en la ventana de `turtlesim`.

Dale a la nueva tortuga un nombre único, como `turtle2`, haciendo doble clic entre las comillas simples vacías en la columna *Expression*. Puedes ver que esta expresión corresponde al valor de `name` y es de tipo *string*. Se debería de ver así figura 1.2

Luego, ingresa algunas coordenadas válidas en las que aparecerá la nueva tortuga, como `x = 1.0` y `y = 1.0`.

Si intentas crear una nueva tortuga con el mismo nombre que una tortuga existente, como la tortuga predeterminada `turtle1`, recibirás un mensaje de error en el terminal que ejecuta `turtlesim_node`:

```
[ERROR] [turtlesim]: A turtle named [turtle1] already exists
```

Para crear `turtle2`, necesitas llamar al servicio haciendo clic en el botón *Call* en la parte superior derecha de la ventana de `rqt`.

Si la llamada al servicio fue exitosa, deberías ver una nueva tortuga (de nuevo con un diseño aleatorio) aparecer en las coordenadas que ingresaste para `x` y `y`.

Si refrescas la lista de servicios en `rqt`, también verás que ahora hay servicios relacionados con la nueva tortuga, `/turtle2/...`, además de `/turtle1/...`.

1.2.1 set_pen servicio

Al llamar al servicio `/turtle1/set_pen` se debería ver como en esta ímagen figura 1.3. Los valores para `r`, `g` y `b`, que están entre 0 y 255, establecen el color del lápiz con el que `turtle1` dibuja, y `width` establece el grosor de la línea.

Para hacer que `turtle1` dibuje con una línea roja distintiva, cambia el valor de `r` a 255, y el valor de `width` a 5. No olvides llamar al servicio después de actualizar los valores.

Si regresas al terminal donde se está ejecutando `turtle_teleop_key` y presionas las teclas de flecha, verás que el lápiz de `turtle1` ha cambiado. Si la habías cerrado vuélvelo a lanzar.

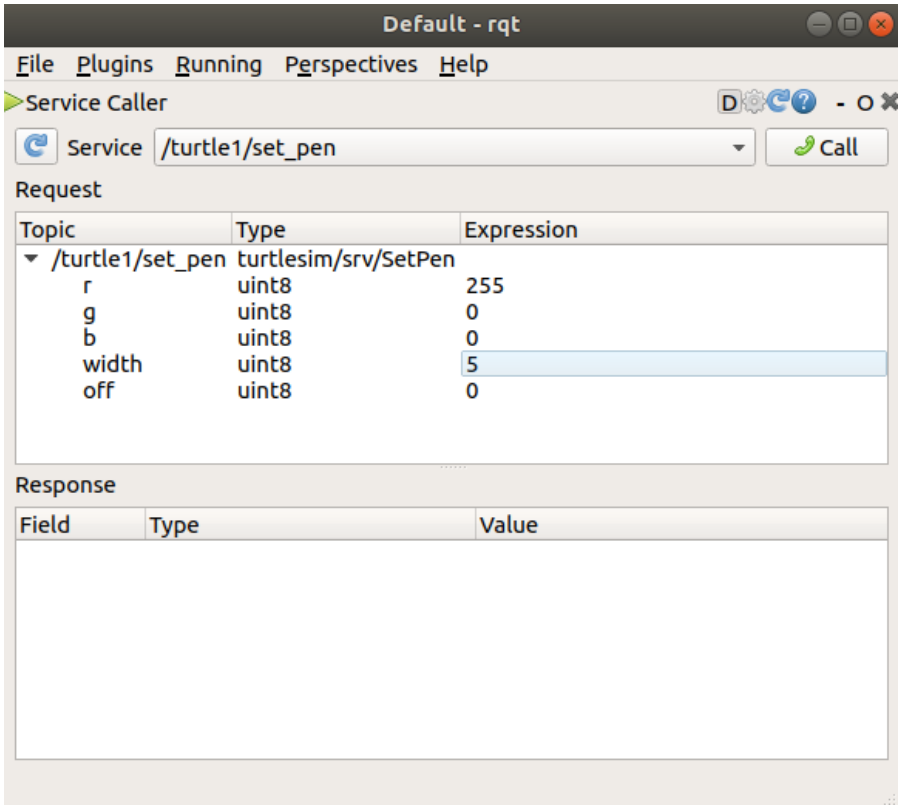

Figura 1.3: Rqt: Servicio set_pen

Probablemente también hayas notado que no hay forma de mover `turtle2`. Eso es porque no hay un nodo `teleop` para `turtle2` y habría que remapear el topic del original, esto lo veremos en una sección posterior.

2 Herramientas depuración de ROS2

Se presupone en esta sección que tenemos arrancado el simulador de turtlesim con una sola tortuga y el nodo de teleop_turtle.

2.1 rqt_graph

`rqt_graph` es una herramienta de ROS que proporciona una interfaz gráfica para visualizar la estructura de la red de nodos de ROS en tiempo real. Su propósito principal es ayudar a los desarrolladores a entender y diagnosticar la interacción y la conectividad entre diferentes nodos en el sistema, mostrando gráficamente cómo los nodos se comunican entre sí a través de topics, servicios y acciones, permitiendo así una inspección más intuitiva y eficiente del sistema. Para ejecutarlo pondremos el siguiente comnado en la terminal:

```
rqt_graph
```

Si seleccionamos en la pestaña de arriba `Nodes/Topics (active)` se muestran los topics y los nodos activos. En esta imagen (figura 2.1 tenemos lanzado el nodo `/turtlesim` y `/teleop_turtle` y podemos ver como se comunican entre sí.

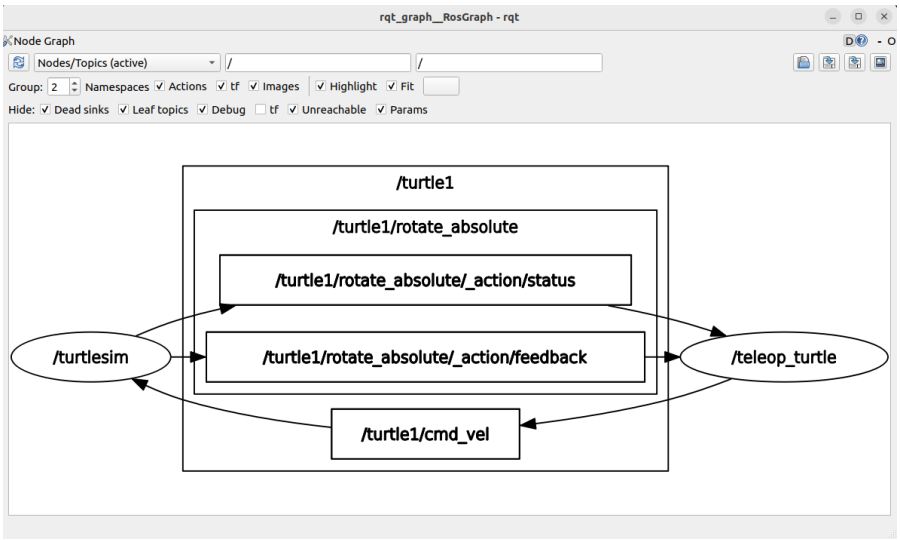

Figura 2.1: Rqt_graph

El gráfico representa cómo los nodos `/turtlesim` y `/teleop_turtle` se comunican entre sí a través de un topic. El nodo `/teleop_turtle` publica datos (las teclas que ingresan para mover

la tortuga) al topic `/turtle1/cmd_vel`, y el nodo `/turtlesim` está subscrito a ese topic para recibir los datos.

La función de resaltado de `rqt_graph` es muy útil cuando se examinan sistemas más complejos con muchos nodos y topics conectados de diferentes formas. `rqt_graph` es una herramienta de inspección y depuración gráfica. A continuación, analizaremos algunas herramientas de línea de comandos para inspeccionar topics.

2.2 Listado e Información de Nodos

2.2.1 Listar los nodos activos

El comando `ros2 node list` mostrará los nombres de todos los nodos en ejecución. Esto es especialmente útil cuando se desea interactuar con un nodo, o cuando se tiene un sistema con múltiples nodos y es necesario mantener un registro de ellos.

Abre un nuevo terminal mientras `turtlesim` sigue en ejecución en el otro, e introduce el siguiente comando:

```
ros2 node list
```

El terminal devolverá los nombres de los nodos activos:

```
/turtlesim
/teleop_turtle
```

2.2.2 Mostrar información de los nodos

Ahora que conoces los nombres de tus nodos, puedes acceder a más información sobre ellos con:

```
ros2 node info <node_name>
```

Para examinar una de tus nodos, `turtlesim`, ejecuta el siguiente comando:

```
ros2 node info /turtlesim
```

`ros2 node info` devuelve una lista de suscriptores, publicadores, servicios y acciones, es decir, las conexiones del grafo ROS que interactúan con ese nodo. La salida debería ser similar a esto:

```
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
```

```
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically:
     ↪  rcl_interfaces/srv/SetParametersAtomically
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
  Service Clients:
  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

Ahora intenta ejecutar el mismo comando en el nodo `/teleop_turtle`, y observa cómo sus conexiones difieren de `turtlesim`.

2.2.3 Remapear nodos

El remapeo de nombres de nodo permite reconfigurar propiedades predefinidas del nodo, como los nombres de nodos, nombres de topics, nombres de servicios, etc., asignando valores personalizados.

Ahora procederemos a reasignar el nombre de nuestro nodo `/turtlesim`. En un nuevo terminal, introduce el siguiente comando:

```
ros2 run turtlesim turtlesim_node --ros-args --remap __node:=mi_tortuga
```

Dado que estás invocando a `ros2 run` en `turtlesim` de nuevo, se generará otra ventana de `turtlesim`. No obstante, si regresas al terminal donde ejecutaste `ros2 node list` y lo vuelves a ejecutar, observarás tres nombres de nodos:

```
/mi_tortuga
/turtlesim
/teleop_turtle
```

2.3 Listado e Información de topics

2.3.1 Listado de topics

Ejecutar el comando `ros2 topic list` en un terminal nuevo devolverá una lista de todos los topics actualmente activos en el sistema.

```
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

Ejecutar `ros2 topic list -t` devolverá la misma lista de topics, esta vez con el tipo de topic añadido entre corchetes:

```
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/turtle1/cmd_vel [geometry_msgs/msg/Twist]
/turtle1/color_sensor [turtlesim/msg/Color]
/turtle1/pose [turtlesim/msg/Pose]
```

Estos atributos, en particular el tipo, son cómo los nodos saben que están hablando de la misma información.

2.3.2 Mostrar topics por consola

Para ver los datos que se están publicando en un topics, se utiliza:

```
ros2 topic echo <topic_name>
```

Ejemplo:

```
ros2 topic echo /turtle1/cmd_vel
```

Al principio, este comando no devolverá ningún dato. Esto se debe a que está esperando que `/teleop_turtle` publique algo. Regresa al terminal donde `turtle_teleop_key` está corriendo y usa las flechas para mover la tortuga. Observa el terminal donde tu eco está corriendo al mismo tiempo, y verás datos de posición siendo publicados para cada movimiento que haces.

```
linear:
  x: 2.0
  y: 0.0
  z: 0.0
```

```
angular:
  x: 0.0
  y: 0.0
  z: 0.0
```

2.3.3 Información del topic

Los topic no sólo tienen que ser de comunicación uno a uno; pueden ser de uno a muchos, de muchos a uno, o de muchos a muchos. Otra forma de ver esto es ejecutando:

```
ros2 topic info /turtle1/cmd_vel
```

Lo cual devolverá:

```
Type: geometry_msgs/msg/Twist
Publisher count: 1
Subscription count: 2
```

2.3.4 Información de las interfaces

Los nodos envían datos sobre topics usando mensajes. Los publicadores y subscriptores deben enviar y recibir el mismo tipo de mensaje para poder comunicarse. Podemos ejecutar el comando `ros2 interface show <msg type>` para obtener más información sobre estos tipos de mensajes.

```
ros2 interface show geometry_msgs/msg/Twist
```

Este comando devolverá:

```
Vector3  linear
    float64 x
    float64 y
    float64 z
Vector3  angular
    float64 x
    float64 y
    float64 z
```

Esto indica que el nodo `/turtlesim` está esperando un mensaje con dos vectores, *linear* y *angular*, de tres elementos cada uno que describen la velocidad lineal y la velocidad angular en tres dimensiones.

2.3.5 Publicación en topics

Ahora que conoces la estructura del mensaje, puedes publicar datos en un topic directamente desde la línea de comandos usando:

```
ros2 topic pub <topic_name> <msg_type> '<args>'
```

Ejemplo:

```
ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x:
↪  2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```

`-once` es un argumento opcional que significa "publicar una vez". Si no lo incluyes, se publicará a una tasa predeterminada hasta que canceles el comando. El formato de los argumentos `<args>` debe coincidir exactamente con la estructura de los mensajes del topic. La publicación de datos directamente a topics puede ser útil para probar nodos y comportamientos específicos en tu sistema.

2.3.6 Remapear topics

Es posible cambiar el topic destino de un nodo mediante parámetros. Por ejemplo, antes nos hemos encontrado que necesitábamos un segundo nodo `teleop` para poder controlar `turtle2`. Sin embargo, al intentar ejecutar otro comando de `teleop`, lo que ocurría es que este también controla `turtle1`. La forma de cambiar este comportamiento es remapeando el `topic cmd_vel` del segundo nodo de `teleop`. Vamos a solucionarlo continuación. Lanza una segunda tortuga llamada `turtle2` tal y como hemos visto antes.

En un nuevo terminal, inicializa ROS 2, y ejecuta:

```
ros2 run turtlesim turtle_teleop_key --ros-args --remap
↪  turtle1/cmd_vel:=turtle2/cmd_vel
```

Ahora, puedes mover `turtle2` cuando este terminal está activo, y `turtle1` cuando el otro terminal que está ejecutando `turtle_teleop_key` está activo.

2.4 Actividad

Objetivo: Lanzar cuatro tortugas en `turtlesim` con los nombres de las Tortugas Ninja como nombres de nodo, cada uno en una esquina diferente y configurar sus *pens* con los colores correspondientes. Haz que cada tortuga dibuje una forma geométrica. Por último abre cuatro terminales con `turtle_teleop_key` para controlar cada tortuga individualmente después de que se hayan dibujado sus formas geométricas.

Es posible borrar el simulador llamando al servicio `/clear` con `rqt`.

Nombre nodo	Esquina	Forma geométrica	Color del lápiz
Leonardo	Superior izquierda	Cuadrado	Azul
Raphael	Inferior izquierda	Triángulo	Roja
Michelangelo	Superior derecha	Círculo	Naranja
Donatello	Inferior derecha	Rectángulo	Púrpura

Bibliografía

Web de los tutoriales de ROS (2023). https://docs.ros.org/en/humble/Tutorials.html.

Web de ROS2 Humble (2023). https://docs.ros.org/en/humble/index.html.