IIC2685 - Robótica Móvil
I – 2022

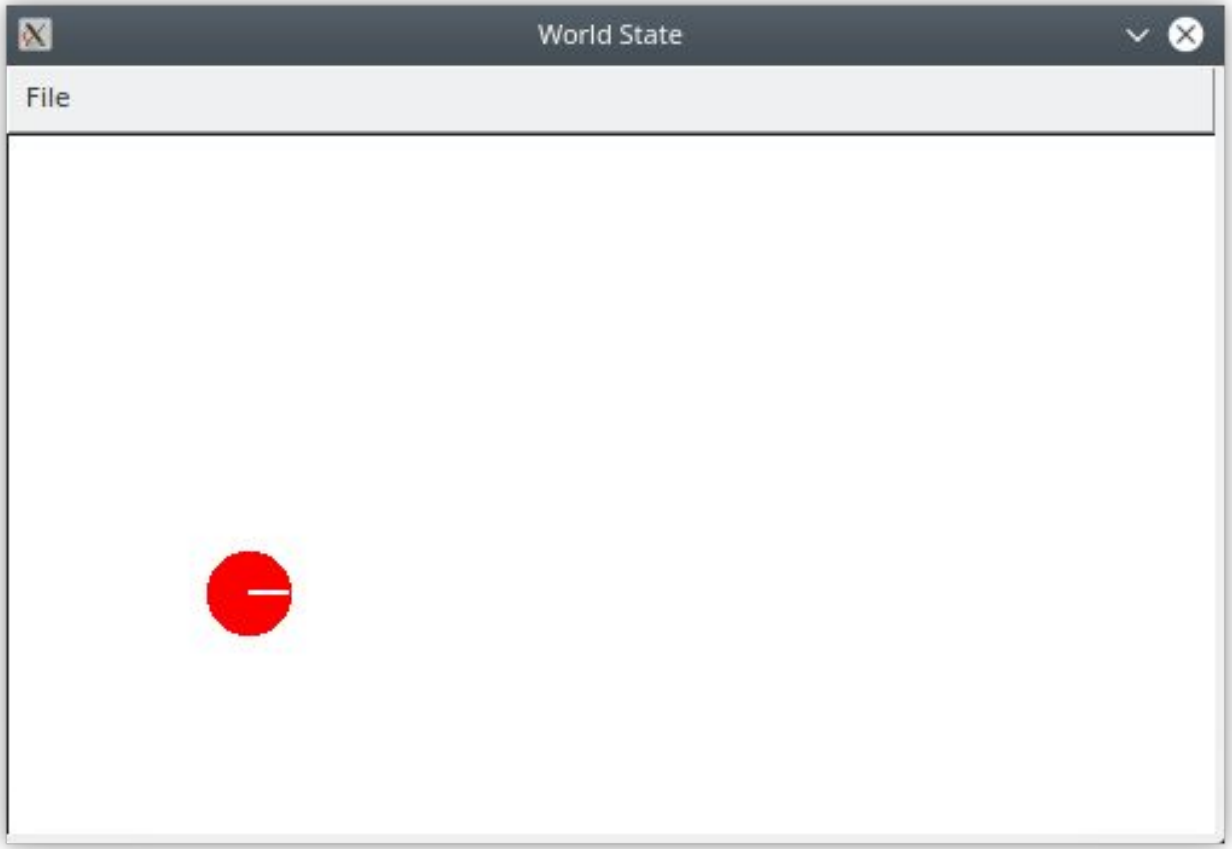
Clase Práctica 3

A Very Simple Robot Simulator

Profesor: Gabriel Sepúlveda V.
grsepulveda@ing.puc.cl

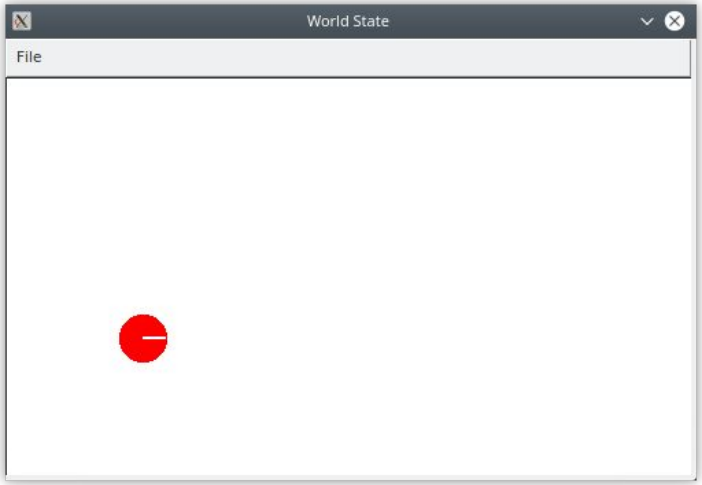
A Very Simple Robot Simulator

- Vista de interfaz gráfica
- Robot parte en posición fija: [1.0, 1.0, 0.0]



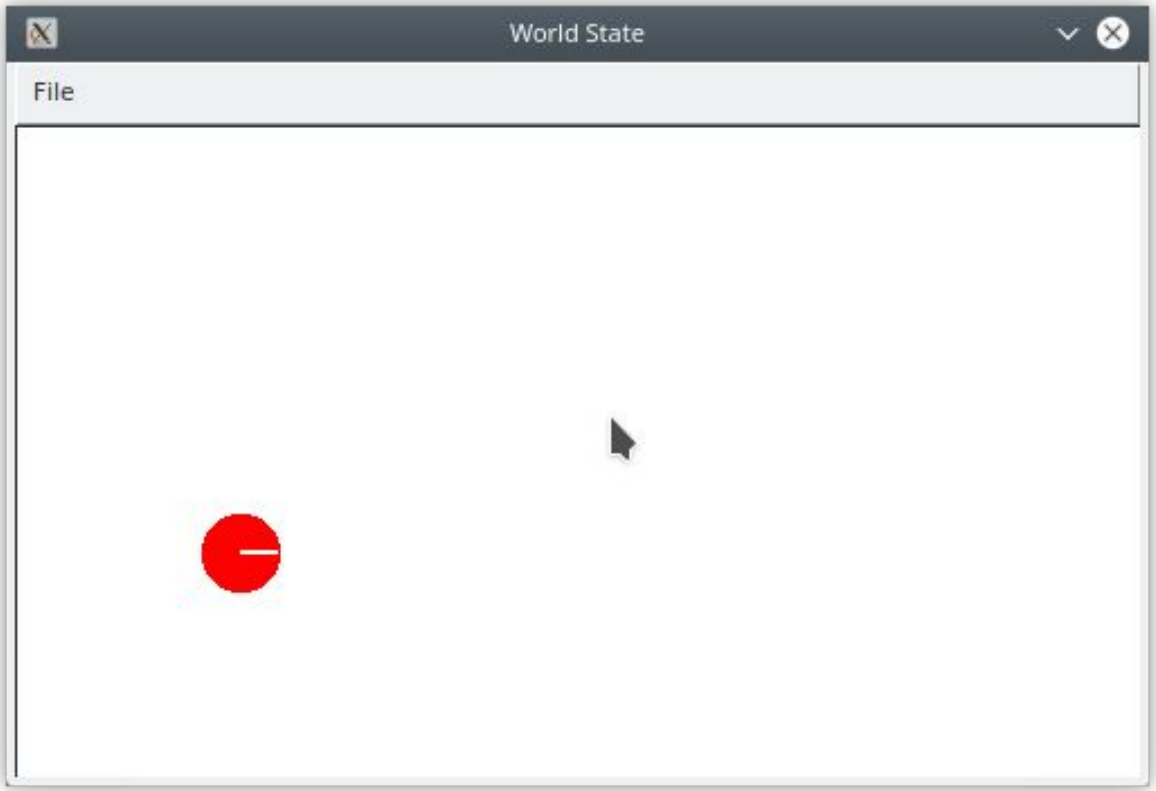
A Very Simple Robot Simulator

- Vista por defecto simula dimensiones del laboratorio: 2.9x5.5 [m]
- Tamaño de robot en proporción al espacio: 0.355 [m] de diámetro
- Inicialmente sin paredes interiores
- Margenes son considerados paredes
- Alto de paredes: 0.5 [m]
- Origen de sistema de coordenadas métrico: esquina inferior izquierda



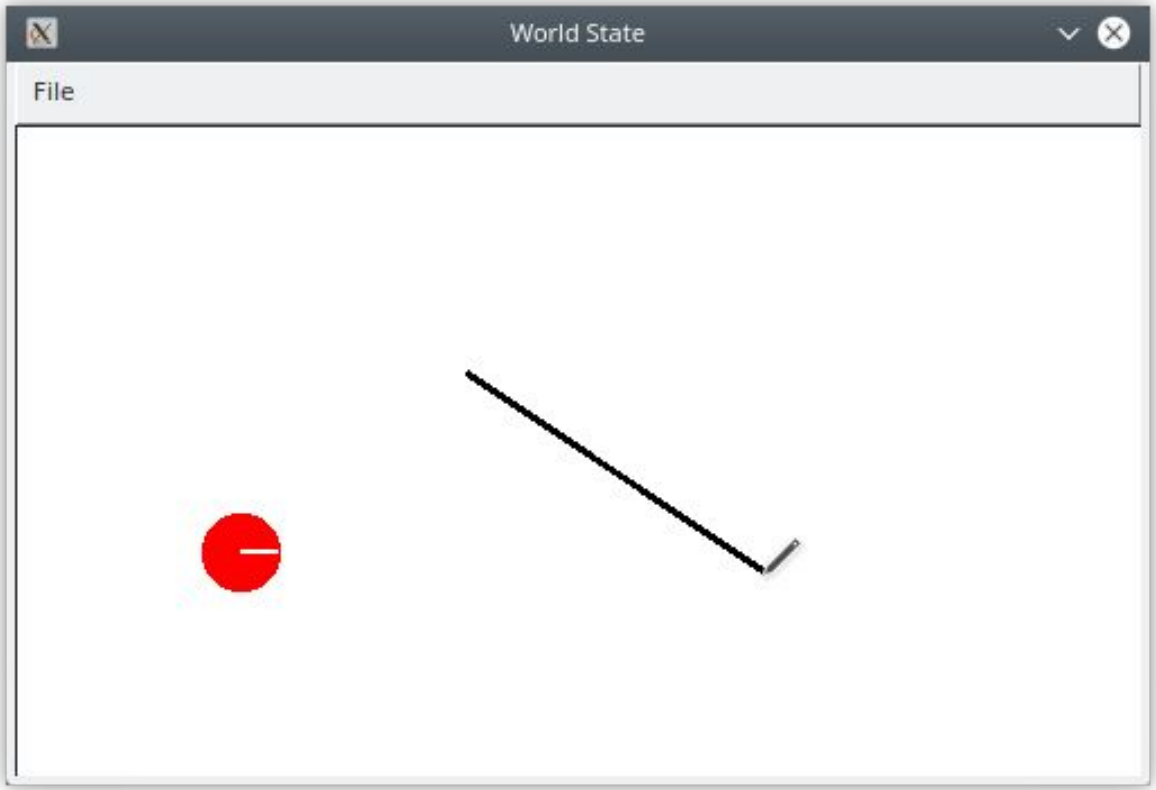
Modos de Operación

- Idle mode
- Modo por defecto al iniciar el programa
- No hay interacción con el mouse



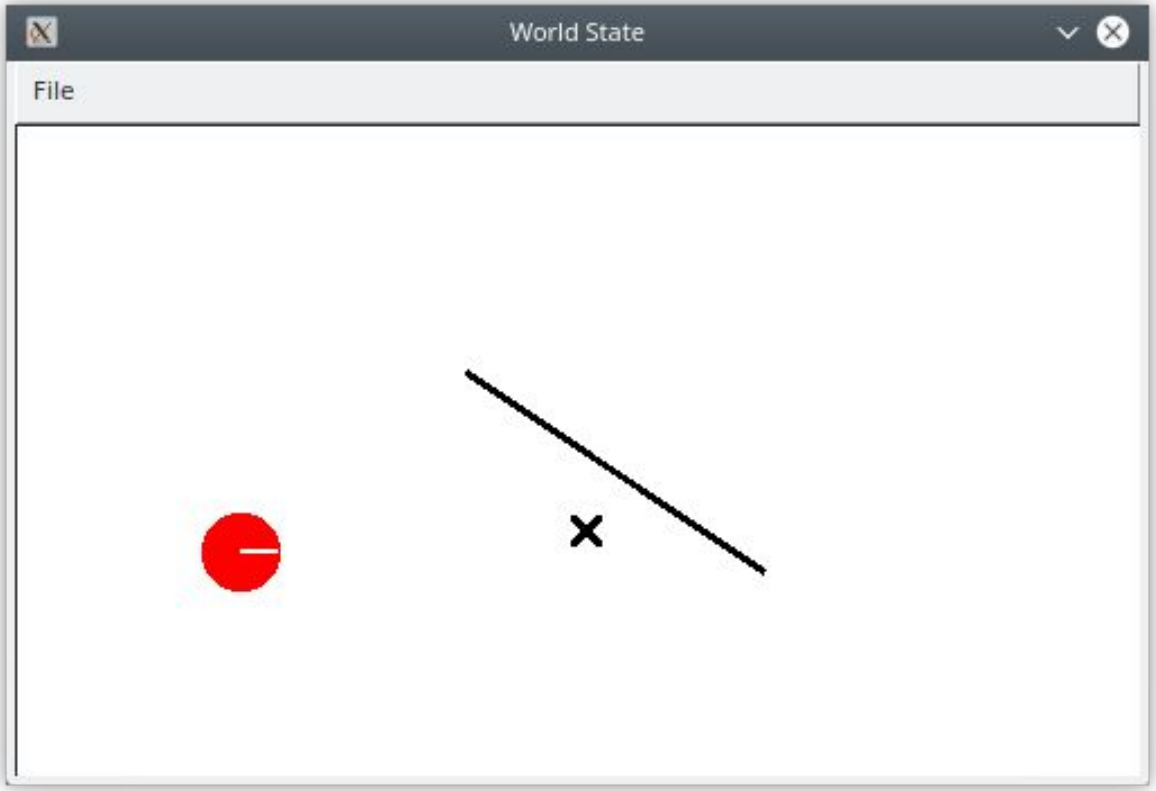
Modos de Operación

- Add wall mode: ‘w’
- Permite dibujar paredes
- Altura es fija para todas las paredes: 0.5 [m]



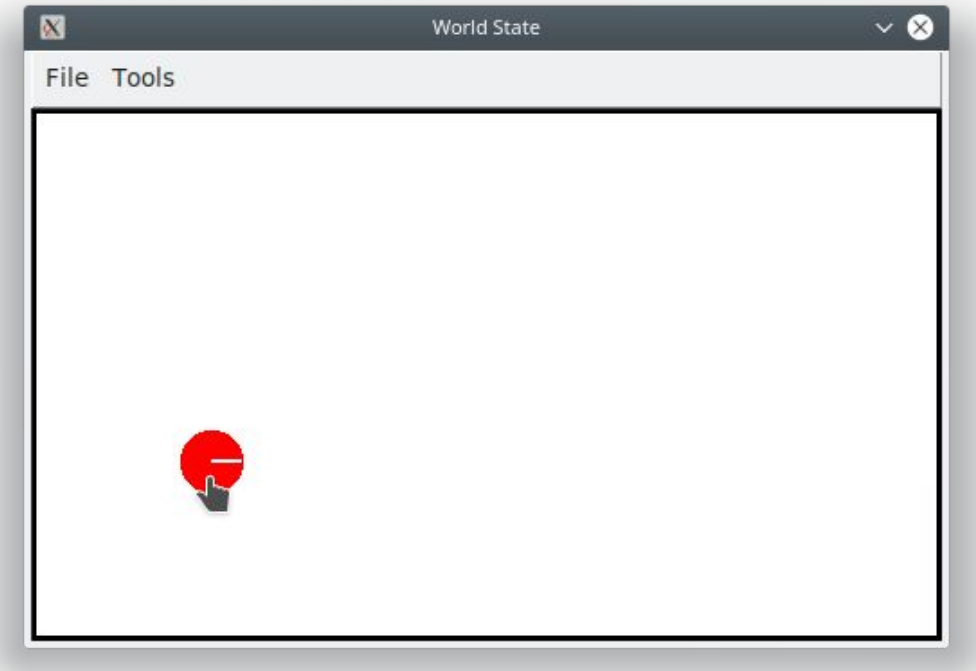
Modos de Operación

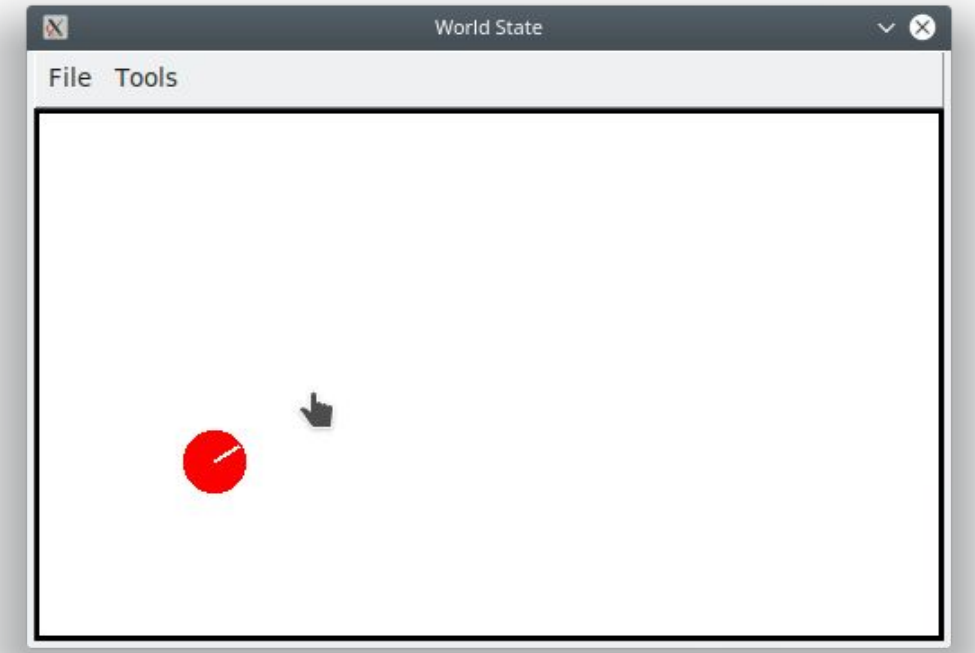
- Delete wall mode: ‘d’
- Elimina pared bajo el cursor
- Eliminación es una a una



Modos de Operación

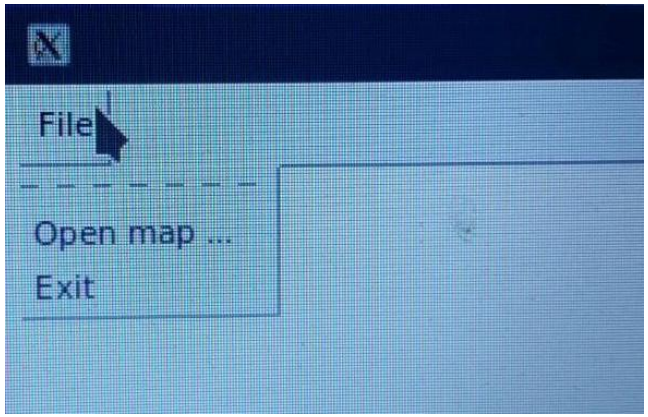
- Set Robot Pose mode: ‘p’
- Permite mover el robot mediate *drag and drop*
- Permite orientar el robot haciendo click fuera del ícono del robot





Menú

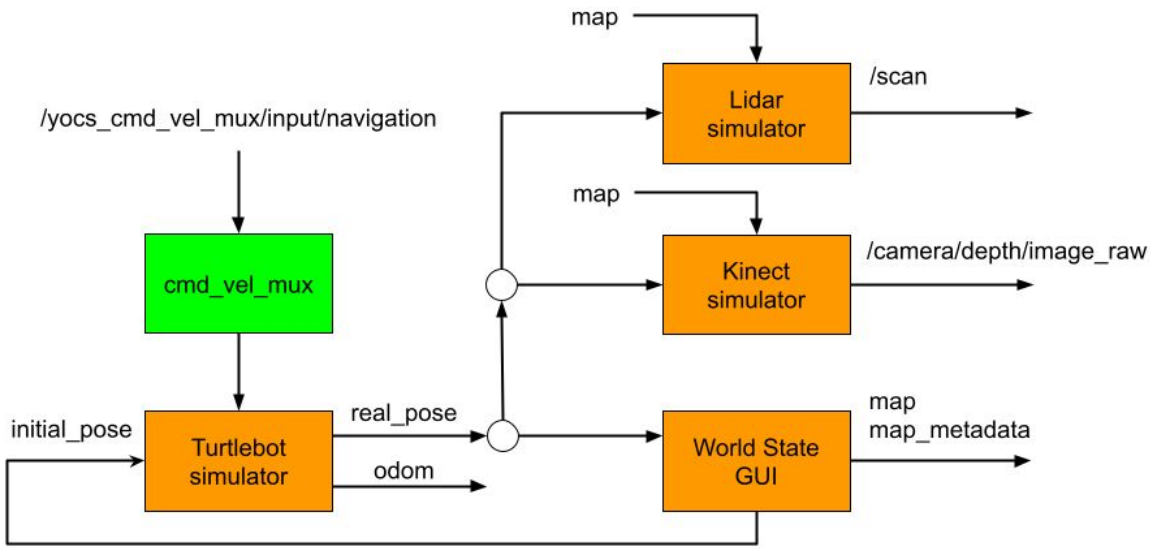
- Carga de mapa desde archivo: “Open map …”



- Archivo de entrada en formato *yaml*
  - image: simple_map.png
  - resolution: 0.006
  - origin: [0.0, 2.898, 0.0]

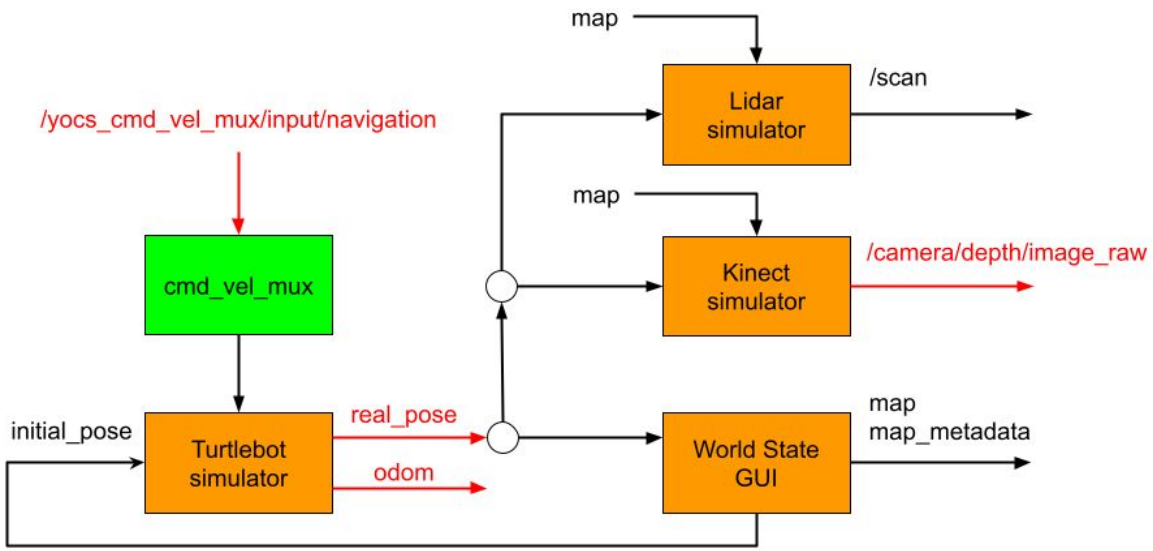
Arquitectura

- Compuesta por 4 nodos que simulan:
  - Turtlebot
  - Kinect
  - Lidar
  - Estado real del mundo
- Adicionalmente, utiliza el *Command Velocity Multiplexer* para recibir instrucciones de velocidad



Arquitectura

- Para el laboratorio 1 utilizaremos los siguientes tópicos
  - /yocs_cmd_vel_mux/input/navigation ⇦ geometry_msgs/Twist
  - /odom ⇦ nav_msgs/Odometry
  - /real_pose ⇦ geometry_msgs/Pose
  - /camera/depth/image_raw ⇦ sensor_msgs/Image



Instalación y futuros updates

- Secuencia de comandos de instalación:

```
$ cd <your_catkin_ws>/src
$ git clone https://github.com/gasevi/yocs_cmd_vel_mux.git
$ git clone https://github.com/gasevi/very_simple_robot_simulator.git
$ cd ..
$ sudo apt-get install python3-pil python3-pil.imagetk
$ rosdep install --from-paths src --ignore-src --rosdistro $ROS_DISTRO
$ catkin_make
```

- Inclusión de paquete al ambiente ROS

```
$ echo "source <your_catkin_ws>/devel/setup.bash" >> .bashrc
$ source .bashrc
```

Inicialización de nodos

- Para inicializar los módulos utilizaremos archivos launch
- Este paquete cuenta con archivos launch que deben ser incluidos desde su aplicación de la siguiente forma:

```
<launch>

  <include file="$(find very_simple_robot_simulator)/launch/minimal_simulator.launch" />
  <include file="$(find very_simple_robot_simulator)/launch/openni_simulator.launch" />
  <include file="$(find very_simple_robot_simulator)/launch/world_state.launch" />

  ...

</launch>
```

Inicialización de nodos

- Los que utilicen WSL, no podrán compilar los módulos en C++
- En este caso, deberán utilizar la versión Python del simulador

```
<launch>

  <include file="$(find very_simple_robot_simulator)/launch/minimal_simulator_py.launch" />
  <include file="$(find very_simple_robot_simulator)/launch/openni_simulator_py.launch" />
  <include file="$(find very_simple_robot_simulator)/launch/world_state.launch" />

  ...

</launch>
```

Ejemplo 1: *Move in circles*

```
class MoveInCircles( object ):

    def __init__( self ):
        rospy.init_node( 'move_in_circles_node' )
        self.cmd_vel_mux_pub = rospy.Publisher( '/yocs_cmd_vel_mux/input/navigation', Twist,
                                                queue_size = 10 )
        self.rate_obj = rospy.Rate( 10 ) # 10 [Hz]

    def move( self, lin_speed, ang_speed ):
        speed = Twist()
        speed.linear.x = lin_speed
        speed.angular.z = ang_speed
        while not rospy.is_shutdown():
            rospy.loginfo( 'publishing speed (%f, %f)' % (lin_speed, ang_speed) )
            self.cmd_vel_mux_pub.publish( speed )
            self.rate_obj.sleep()
```

Ejemplo 2: *Read Odometry*

```
class OdometryReader( object ):

    def __init__( self ):
        rospy.init_node( 'read_odom' )
        self.odom_sub = rospy.Subscriber( '/odom', Odometry, self.odometry_cb )

    def odometry_cb( self, odom ):
        x = odom.pose.pose.position.x
        y = odom.pose.pose.position.y
        z = odom.pose.pose.position.z
        roll, pitch, yaw = euler_from_quaternion( ( odom.pose.pose.orientation.x,
                                                    odom.pose.pose.orientation.y,
                                                    odom.pose.pose.orientation.z,
                                                    odom.pose.pose.orientation.w ) )
        rospy.loginfo( 'Current pose - lin: (%f, %f, %f) ang: (%f, %f, %f)' % (x, y, z, roll, pitch, yaw) )
```