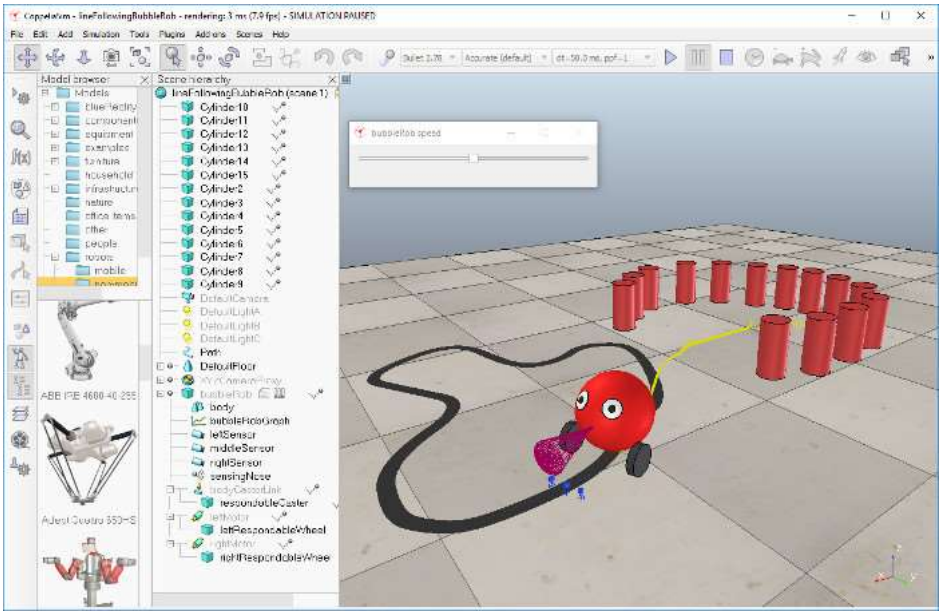
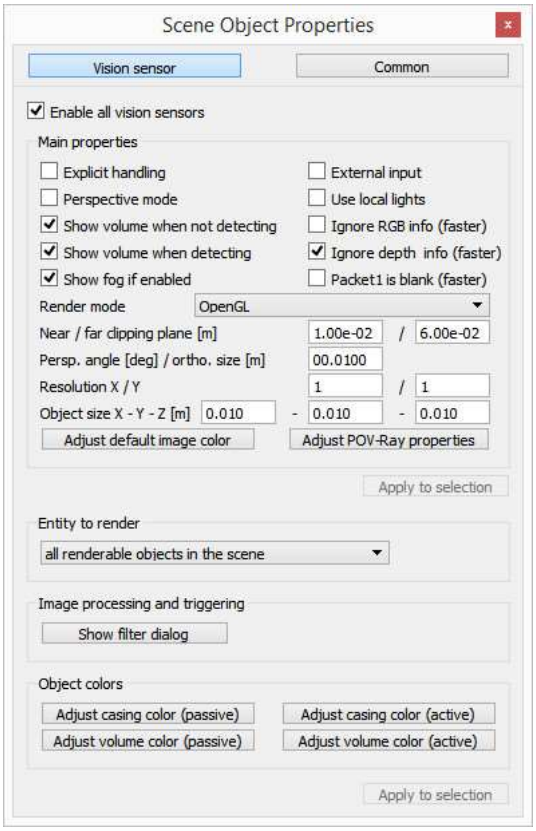
Line following BubbleRob tutorial

In this tutorial we aim at extending the functionality of BubbleRob to let it follow a line on the ground. Make sure you have fully read and unde
BubbleRob tutorial. This tutorial is courtesy of Eric Rohmer.

Load the scene of the first BubbleRob tutorial located in *scenes/tutorials/BubbleRob*. The scene file related to this tutorial is located in *scenes/tutorials/LineFollowingBubbleRob*. Following figure illustrates the simulation scene that we will design:

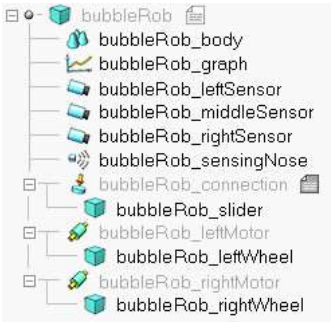


We first create the first of 3 vision sensors that we will attach to the *bubbleRob* object. Select [Menu bar --> Add --> Vision sensor --> Orthog
properties, by double-clicking on the newly created vision sensor icon in the scene hierarchy, and change the parameters to reflect following d



The vision sensor has to be facing the ground, so select it, and in the orientation dialog, on the **orientation** tab, set [180;0;0] for the *Alpha-B*

We have several possibilities to read a vision sensor. Since our vision sensor has just one pixel and operates in an easy way, we will simply que intensity value of the image read by our vision sensor. For more complex cases, we could have set-up a vision callback function. Now copy and sensor twice, and adjust its names to *bubbleRob_leftSensor*, *bubbleRob_middleSensor* and *bubbleRob_rightSensor*. Make *bubbleRob* their pare the *bubbleRob* object). Your sensors should now look like this in the scene hierarchy:



Let's position the sensors correctly. For that use the position dialog, on the **position** tab, and set following absolute coordinates:

- left sensor: [0.2;0.042;0.018]
- middle sensor: [0.2;0;0.018]
- right sensor: [0.2;-0.042;0.018]

Now let's modify the environment. We can remove a few cylinders in front of BubbleRob. Next, we will build the path that the robot will try to --> Add --> Path --> Closed]. You have several possibilities to adjust the shape of the path, by manipulating its control points: you can delete you can shift/reorient them. Enable the object movement with the mouse, and adjust the path to your liking.

Once you are satisfied with the geometry of the path (you can always modify it at a later stage), open the customization script attached to it a with:

```
path=require('path_customization')

function path.shaping(path,pathIsClosed,upVector)
    local section={-0.02,0.001,0.02,0.001}
    local color={0.3,0.3,0.3}
    local options=0
    if pathIsClosed then
        options=options|4
    end
    local shape=sim.generateShapeFromPath(path,section,options,upVector)
    sim.setShapeColor(shape,nil,sim.colorcomponent_ambient_diffuse,color)
    return shape
end
```

Restart the customization script for the changes to take effect, then open the path's user configuration dialog and check the **Generate extru**

The last step is to adjust the controller of BubbleRob, so that it will also follow the black path. Open the child script attached to *bubbleRob*, an following code:

```
function speedChange_callback(ui,id,newVal)
    speed=minMaxSpeed[1]+(minMaxSpeed[2]-minMaxSpeed[1])*newVal/100
end

function sysCall_init()
    -- This is executed exactly once, the first time this script is executed
    bubbleRobBase=sim.getObjectAssociatedWithScript(sim.handle_self)
    leftMotor=sim.getObjectHandle("bubbleRob_leftMotor")
    rightMotor=sim.getObjectHandle("bubbleRob_rightMotor")
    noseSensor=sim.getObjectHandle("bubbleRob_sensingNose")
    minMaxSpeed={50*math.pi/180,300*math.pi/180}
    backUntilTime=-1 -- Tells whether bubbleRob is in forward or backward mode
    floorSensorHandles={-1,-1,-1}
    floorSensorHandles[1]=sim.getObjectHandle("bubbleRob_leftSensor")
    floorSensorHandles[2]=sim.getObjectHandle("bubbleRob_middleSensor")
    floorSensorHandles[3]=sim.getObjectHandle("bubbleRob_rightSensor")
    -- Create the custom UI:
        xml = '<ui title="'..sim.getObjectName(bubbleRobBase)..' speed" closeable="false" resizeable="false" activ
        <hslider minimum="0" maximum="100" onchange="speedChange_callback" id="1"/>
        <label text="" style="* {margin-left: 300px;}"/>
        </ui>
```

```
        ]]
    ui=simUI.create(xml)
    speed=(minMaxSpeed[1]+minMaxSpeed[2])*0.5
    simUI.setSliderValue(ui,1,100*(speed-minMaxSpeed[1])/(minMaxSpeed[2]-minMaxSpeed[1]))
end

function sysCall_actuation()
    result=sim.readProximitySensor(noseSensor)
    if (result>0) then backUntilTime=sim.getSimulationTime()+4 end

    -- read the line detection sensors:
    sensorReading={false,false,false}
    for i=1,3,1 do
        result,data=sim.readVisionSensor(floorSensorHandles[i])
        if (result>=0) then
            sensorReading[i]=(data[11]<0.3) -- data[11] is the average of intensity of the image
        end
        print(sensorReading[i])
    end

    -- compute left and right velocities to follow the detected line:
    rightV=speed
    leftV=speed
    if sensorReading[1] then
        leftV=0.03*speed
    end
    if sensorReading[3] then
        rightV=0.03*speed
    end
    if sensorReading[1] and sensorReading[3] then
        backUntilTime=sim.getSimulationTime()+2
    end

    if (backUntilTime<sim.getSimulationTime()) then
        -- When in forward mode, we simply move forward at the desired speed
        sim.setJointTargetVelocity(leftMotor,leftV)
        sim.setJointTargetVelocity(rightMotor,rightV)
    else
        -- When in backward mode, we simply backup in a curve at reduced speed
        sim.setJointTargetVelocity(leftMotor,-speed/2)
        sim.setJointTargetVelocity(rightMotor,-speed/8)
    end
end

function sysCall_cleanup()
        simUI.destroy(ui)
end
```

You can easily debug your line following vision sensors: select one, then in the scene view select [Right-click --> Add --> Floating view], then i floating view select [Right click --> View --> Associate view with selected vision sensor].