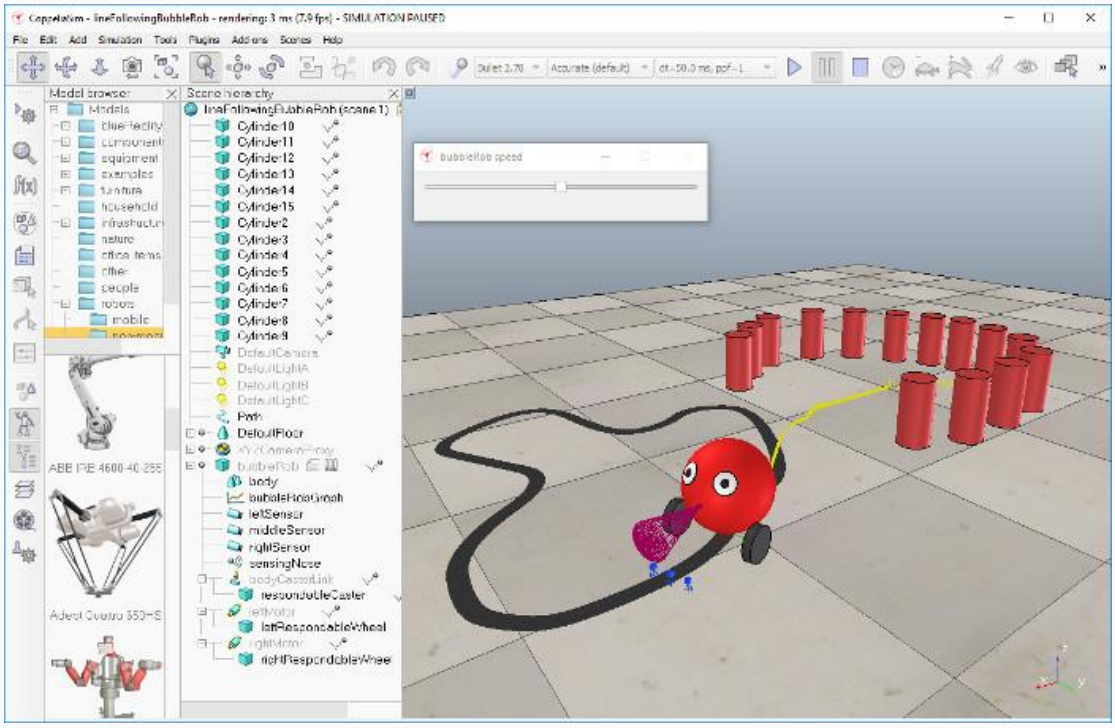
Line following BubbleRob tutorial

在本教程中，我們在擴展 BubbleRob 的功能，以使他遵循基礎上的規則。

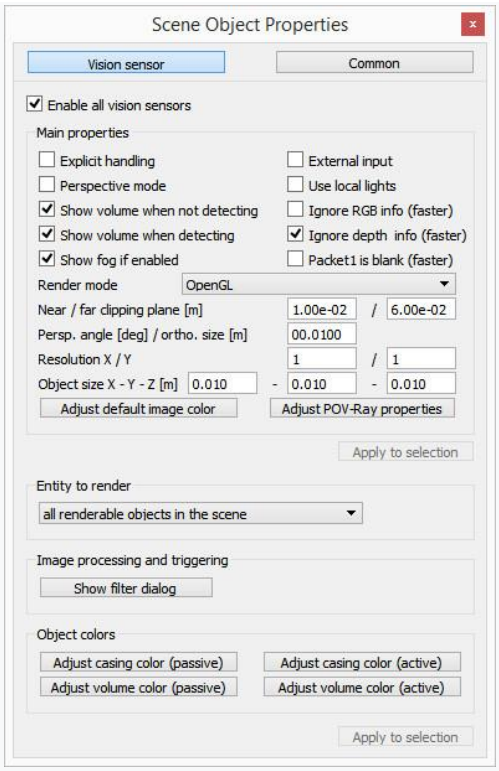
在 CoppeliaSim 的安裝文件夾中的 tutorials / BubbleRob 中加載第一個 BubbleRob 教程的場景。與本教程相關的內容文件位於 tutorials / LineFollowingBubbleRob 中。

下圖說明了我們設計的仿真場景：



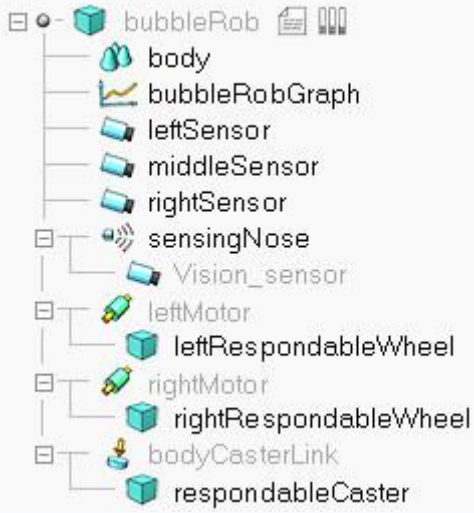
我們首先創建 3 個視覺傳感器中的第一個，並將其附加到 bubbleRob 物件。

選擇[菜單欄->添加->視覺傳感器->正交類型]，通過雙擊場景層次中新創建的視覺傳感器圖標來編輯其屬性，並更改參數以反映以下對話框：



視覺傳感器必須面向地面，因此選擇它，然後在“方向”對話框中的“方向”選項卡上，將“ Alpha-Beta-Gamma”項設置為[180; 0; 0]。

我們有幾種可能性可以讀取視覺傳感器。由於我們的視覺傳感器只有一個像素，並且操作簡單，因此我們只需查詢視覺傳感器讀取的圖像的平均強度值即可。對於更複雜的情況，我們可以設置一個視覺回調函數。將視覺傳感器複製並黏貼兩次，並將其名稱調整為 leftSensor，middleSensor 和 rightSensor。讓 bubbleRob 成為他們的母模組。



讓我們正確放置傳感器。為此，使用位置對話框，在位置選項卡上，並設置以下絕對坐標：

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

現在讓我們修改環境。我們可以移去 BubbleRob 前面的幾個圓柱體。接下來，我們將構建機器人將嘗試遵循的路徑。從現在開始最好切換到頂視圖：通過頁面選擇器工具欄按鈕選擇頁面 4。

然後單擊[菜單欄->添加->路徑->圓圈類型]。使用鼠標啟用物體移動。您可以通過兩種方式調整路徑的形狀：

- With the path (and only the path) selected, ctrl-click one of its control points. You can then drag them to the right position.

(選擇路徑後，按住 Ctrl 並單擊其控制點之一。然後可以將它們拖動到正確的位置。)

- With the path selected, enter the path edit mode. In there, you have much more flexibility to adjust the individual path control points.

(選擇路徑後，進入路徑編輯模式。在那裡，您可以靈活地調整各個路徑控制點。)

對路徑的幾何形狀滿意後(您隨時可以在以後修改)，選擇它，並取消選中路徑屬性中的“顯示點的方向”，“顯示路徑線”和“顯示路徑上的當前位置”。 然後單擊顯示路徑成形對話框。這將打開路徑成形對話框。單擊啟用路徑成形，將類型設置為水平線段，並將縮放因子設置為 4.0。最後將顏色調整為黑色。

我們必須對路徑進行最後一個重要的調整：當前路徑的 z 位置與地板的 z 位置重合。結果是有時我們看到路徑，有時看到地板，這不僅影響我們所看到的，而且還會影響視覺傳感器所看到的(這種效應在 openGl 術語中稱為“ z-fighting”)。為了避免與 z-fighting 有關的問題，只需將路徑對象的位置向上移動 0.5 毫米即可。

最後一步是調整 BubbleRob 的控制器，使其也將遵循黑色路徑。打開附加到 bubbleRob 的子腳本，並將其替換為以下代碼：

```
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("leftMotor")
    rightMotor=sim.getObjectHandle("rightMotor")
    noseSensor=sim.getObjectHandle("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("leftSensor")
    floorSensorHandles[2]=sim.getObjectHandle("middleSensor")
    floorSensorHandles[3]=sim.getObjectHandle("rightSensor")
    -- Create the custom UI:
        xml = '<ui title="'..sim.getObjectName(bubbleRobBase)..' speed" closeable="false" resizeable="false" activate="false">'..[[
        <hslider minimum="0" maximum="100" onchange="speedChange_callback" id="1"/>
        <label text="" style="* {margin-left: 300px;}"/>
        </ui>
        ]]
    ui=simUI.reate(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
```

您可以通過視覺傳感器輕鬆調試生產線：選擇一個，然後在場景視圖中選擇，[右鍵單擊->添加->浮動視圖]，然後在新添加的浮動視圖中，選擇[右鍵->視圖->將視圖與選定的視覺傳感器關聯]。

最後，刪除在第一個 BubbleRob 教程中添加的輔助項：卸下圖像處理視覺傳感器，其相關的浮動視圖，表示障礙物間隙的浮動視圖。通過距離對話框也刪除距離計算對象。