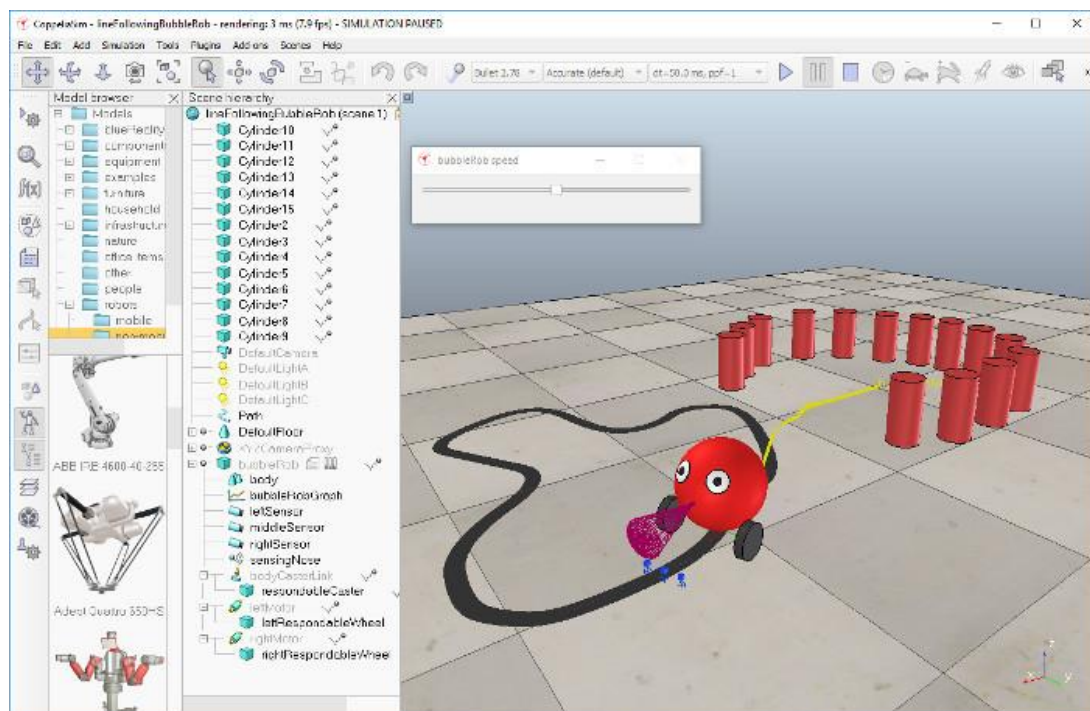


Line following BubbleRob tutorial

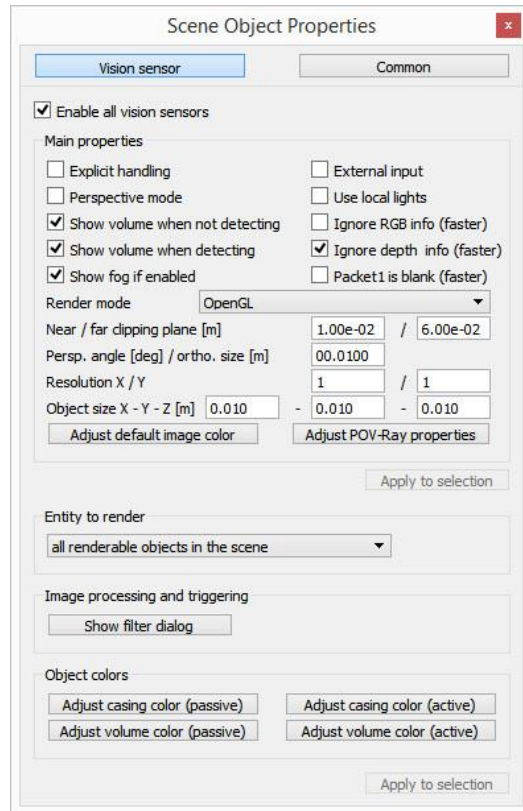
在本教程中，我們旨在擴展 BubbleRob 的功能，以使他/她遵循地面上的規則。確保您已經閱讀並理解了第一個 BubbleRob 教程。本教程由 Eric Rohmer 提供。

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

下圖說明了我們將設計的仿真場景：

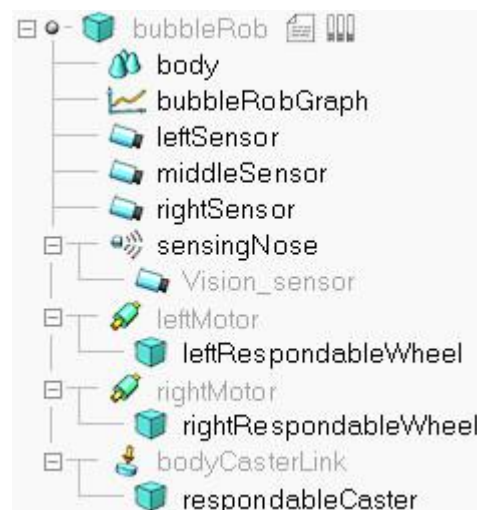


我們首先創建 3 個視覺傳感器中的第一個，並將其附加到 bubbleRob 對象。選擇[菜單欄->添加->視覺傳感器->正交類型]。通過雙擊場景層次中新創建的視覺傳感器圖標來編輯其屬性，並更改參數以反映以下對話框：



視覺傳感器必需面向地面，因此選擇它，然後在“方向”對話框中的“方向”選項上，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，我們可以設置視覺回調函數。現在，複製並貼上視覺傳感器兩次，並將其名稱調整為 `leftSensor`，`middleSensor` 和 `rightSensor`。將 `bubbleRob` 設置為其關係（即，將其附加到 `bubbleRob` 對象）。現在，您的傳感器在場景層次中應如下所示：



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： 左傳感器：[0.2; 0.042; 0.018] 中間傳感器：[0.2; 0; 0.018] 右傳感器：[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**。然後點擊[菜單欄->添加->路徑->圓圈類型]。使用鼠標啟用對象移動。您可以通過兩種方式調整路徑的形狀： 選擇路徑（並且只有路徑）後，按住 **Ctrl** 並點擊其控制點之一。然後可以將它們拖動到正確的位置。 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。

一旦對路徑的幾何形狀滿意（您隨時可以在以後的階段對其進行修改），請選擇它，然後在路徑屬性中取消選中“顯示點的方向”，“顯示路徑線”和“顯示路徑上的當前位置”。然後點擊顯示路徑整形對話框。這將打開路徑整形對話框。點擊啟用路徑變更，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。結果是有時我們會看到路徑，有時會看到地板（這種效果在 **openGl** 行話中被稱為 “**z-fighting**”）。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。為了避免與 **z** 戰鬥有關的問題，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.getObjectHandle(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.recreate(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(floorSensorH
andles[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
```

您可以輕鬆地調試以下視覺傳感器的生產線：選擇一個，然後在場景視圖中選擇[右鍵->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]。