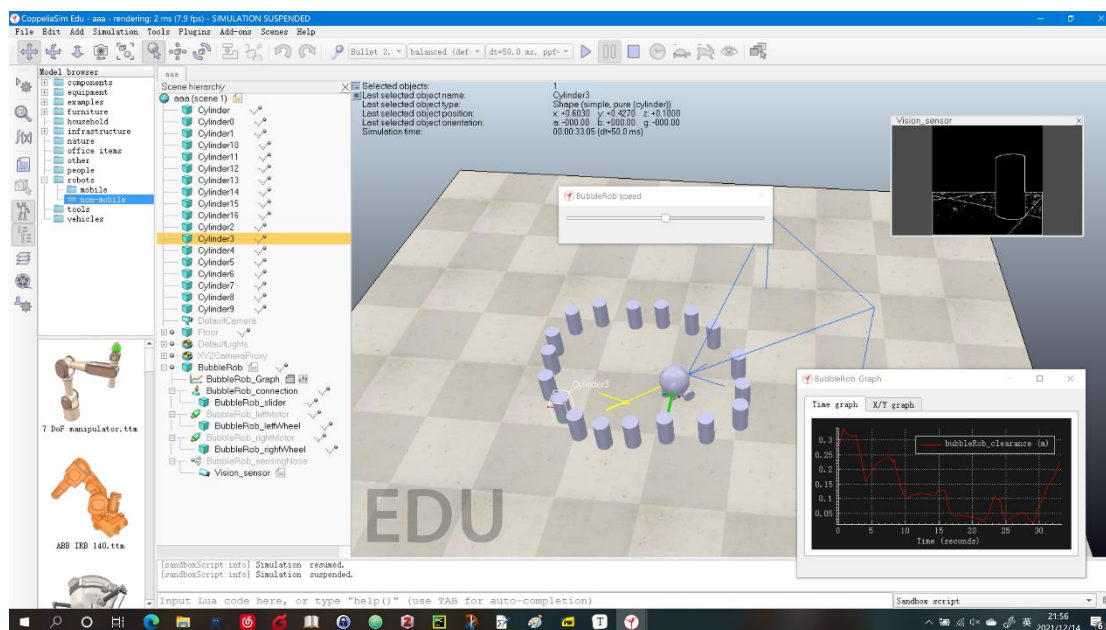




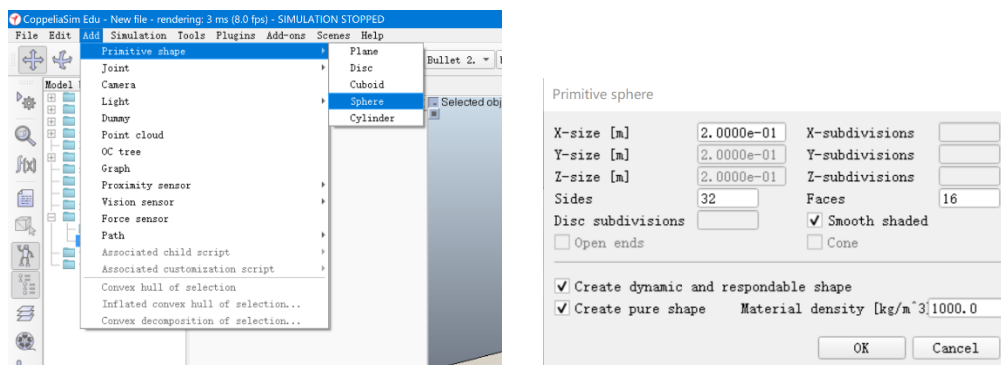
算法类问题实践课程教程

BubbleRob 机器人创建与避障控制教程




BubbleRob 模型创建

1. 通过菜单[Menu bar --> Add --> Primitive shape --> Sphere], 在场景中加入一球形。将选项 **X-size** 调整为 0.2, 然后单击确定。



2. 动力学仿真测试。点击开始仿真按钮进行仿真。仿真过程中, 选中刚刚创建的球形, 按键 Ctrl-C (或[Menu bar --> Edit --> Copy selected objects]) 对其进行复制, 按键 Ctrl-V (或[Menu bar --> Edit -> Paste buffer]) 将其粘贴到相同位置。可以看到两个球形相互碰撞, 并快速滚动开。点击关闭仿真, 可以看到刚才在仿真过程中

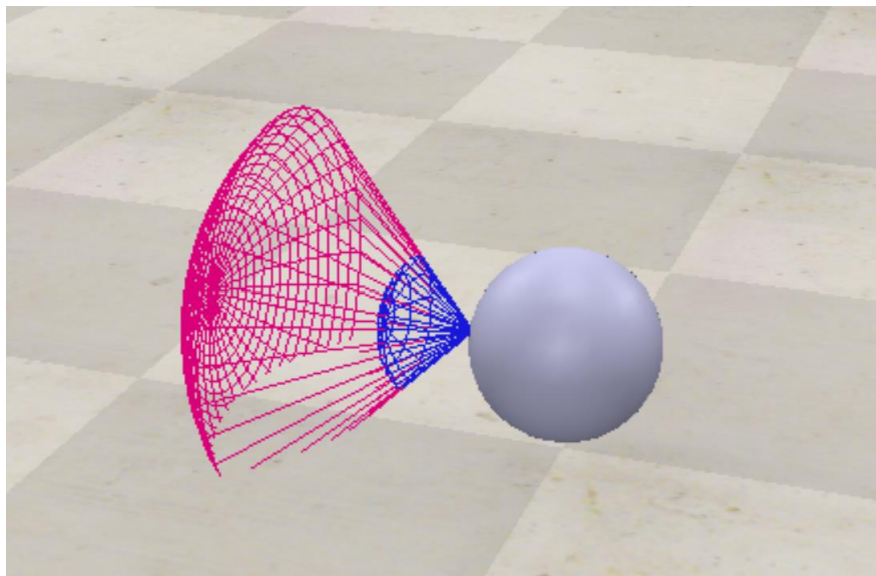
创建的第二个球形自动消失。


- 我们还希望 BubbleRob 的模型可供其他计算模块使用（例如距离计算）。因此，在 **scene hierarchy** 中双击创建的球形名称部分，将其重命名为：bubbleRob。双击名称左侧形状标志可打开其属性设置窗口。在其 **common** 属性中启用（勾选）**Collidable**、**Measurable**、**Renderable** 和 **Detectable** 选项。此外，我们现在还可以在其 **shape** 属性中更改球体的视觉外观。
- 选中 BubbleRob 后，点击  打开位置对话框。切换到 **translation** 标签，在 **Along Z** 右侧输入 0.02，并确认位置变换为相对于世界坐标系，如下图所示。之后点击 **Translate selection**。

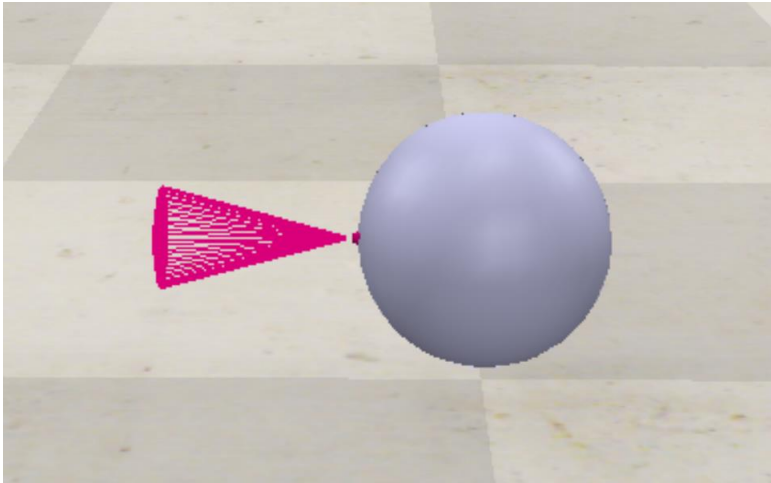
Relative to: ☒ World ☐ Parent frame ☐ Own frame

该操作会把 bubblerob 沿世界坐标系 Z 轴向上移动 0.02 米。

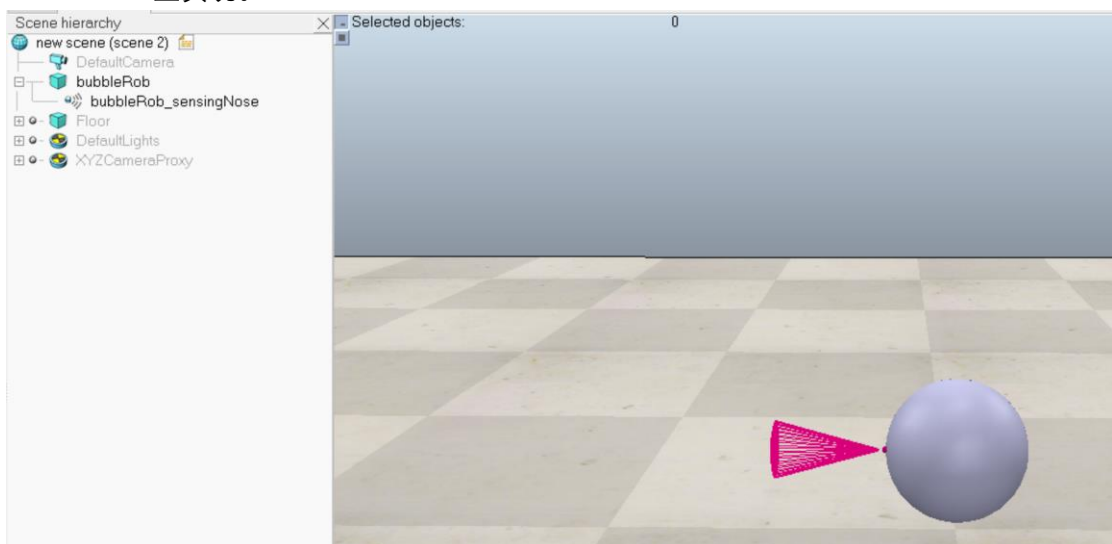
- 为 BubbleRob 增加接近传感器（[proximity sensor](#)），使其具有感知障碍物能力。选择 [Menu bar --> Add --> Proximity sensor --> Cone type]。点击  打开 **orientation** 对话框，在 **Rotation** 标签栏中的 **Around Y** 和 **Around Z** 分别输入 90，然后点击 **Rotate selection**。点击  打开位置对话框，切换到 position 标签栏，分别在 **X-coord**，**Y-coord**，**Z-coord** 输入 0.1，0，0.12。按下 Enter 按键执行。现在，接近传感器被正确的放置在 BubbleRob 上面，如下图所示。





- 修改接近传感器属性。在 **scene hierarchy** 中双击接近传感器的图标 ，打开其属性修改对话框。点击 **Show volume parameter** 打开 **Detection volume properties** 对话框。进行以下修改：**Offset**: 0.005, **Angle**: 30, **Range**: 0.15。然后关闭 **Detection volume properties** 对话框，在属性修改对话框中点击 **Show detection parameters** 打开 **Detection parameters** 对话框，取消勾选 **Don't allow detections if distance smaller than**，然后关闭对话框。在 **scene hierarchy** 中双击接近传感器的名字，将其名称修改为：bubbleRob_sensingNose。



7. 将传感器**连接**到 BubbleRob。在 **scene hierarchy** 中首先选中 bubbleRob_sensingNose，然后 **Ctrl** 同时选中 BubbleRob，接着点击[**Menu bar --> Edit --> Make last selected object parent**]。该动作将把传感器**连接**到机器人的身体上，该动作也可以通过在 **scene hierarchy** 中将 bubbleRob_sensingNose 拖到 bubbleRob 上实现。



8. 下面为机器人添加轮子。点击[**Menu bar --> File --> New scene**]新建一个场景。在多个场景间工作通常比较方便对特定元素进行可视化和工作。点击[**Menu bar --> Add --> Primitive shape --> Cylinder**]加入圆柱形，在弹出对话框中的外形尺寸中 **X-size** 输入 0.08，**Z-size** 输入 0.02，点击 **OK**。同 bubbleRob 一样，双击其图标，打开 **common** 属性对话框，启用（勾选） **Collidable**、**Measurable**、

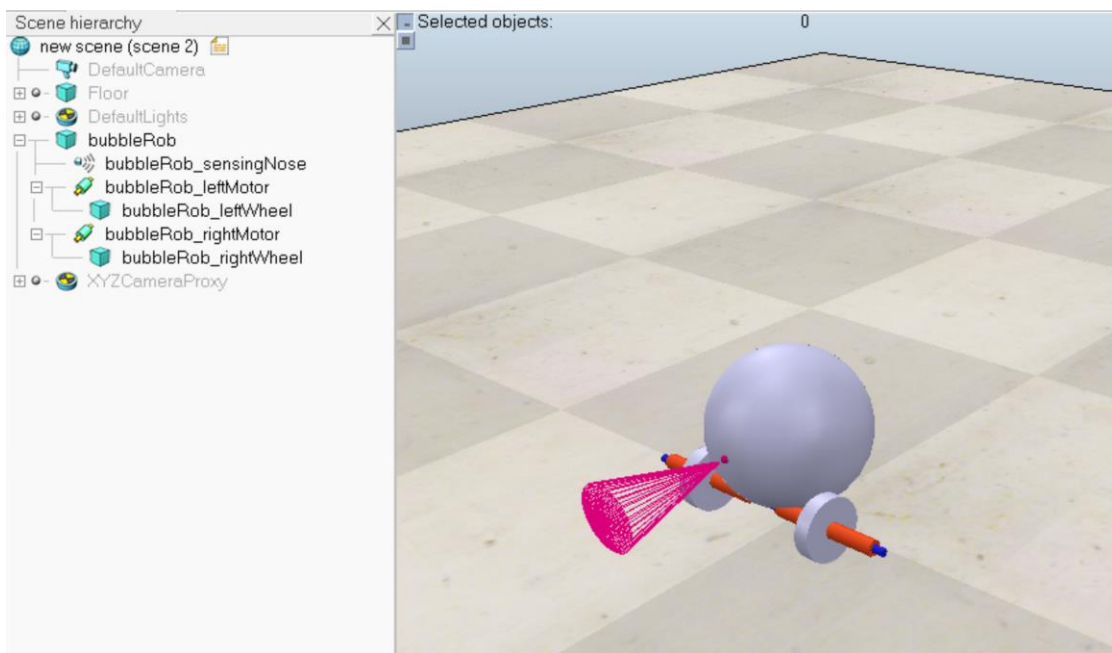
Renderable 和 **Detectable** 选项。然后点击  打开位置对话框，在 **Position** 标签栏中，选择 **Relative to: World**，这样可以对圆柱形在世界坐标系中的绝对位置进行修改，将其绝对位置修改为：(0.05,0.1,0.04)。同样道理，点击  打开 **orientation** 对话框，在 **orientation** 标签栏中选择 **Relative to: World**，这样可以对圆柱形在世界坐标系中的绝对姿态进行修改，将其绝对姿态修改为(-90,0,0)。将其重命名为 bubbleRob_leftWheel。将其复制粘贴，将复制的轮子的绝对 Y 坐标修改为：-0.1。将复制的轮子重命名为 bubbleRob_rightWheel。选中两个轮子，复制 (Ctrl-C)，**切换回**

之前的场景，将他们粘贴 (Ctrl-V) 进来。

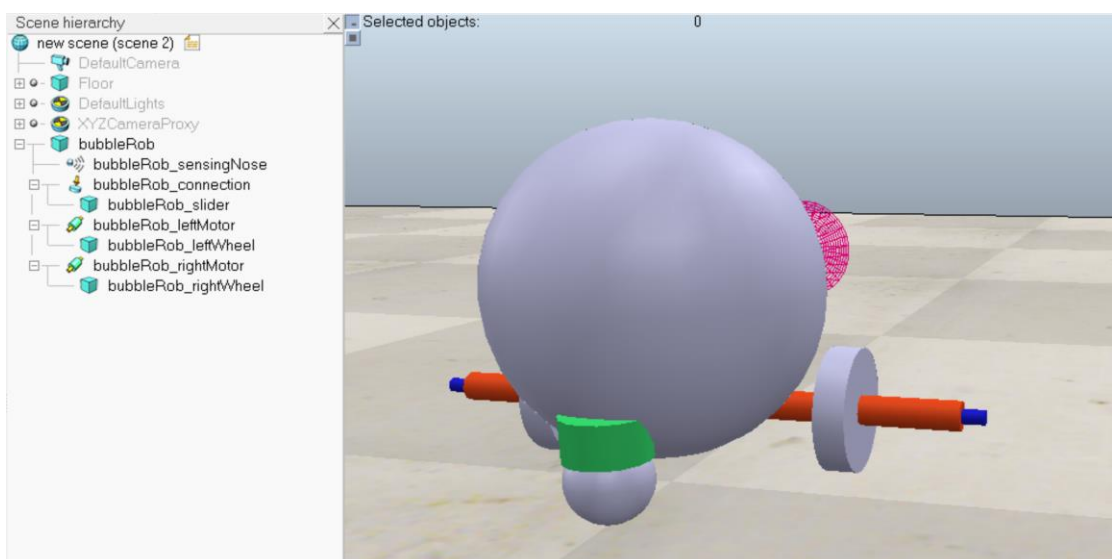
9. 为轮子增加旋转关节。点击[Menu bar --> Add --> Joint --> Revolute]在场景中增加一个旋转关节。大多数情况下，当向场景中添加新对象时，该对象会出现在世界的原点。我们保持关节处于选中状态，按下 Ctrl 再选中 bubbleRob_leftWheel，然后点击  打开位置对话框，在 **Position** 标签栏中，点击 **Apply to selection** 按钮：把关节的中心和轮子中心重合。同样的，点击  打开 **orientation** 对话框，在 **Orientation** 标签栏中，点击 **Apply to selection** 按钮：把关节姿态与轮子保持一致。将关节重命名为：bubbleRob_leftMotor。



10. 双击 bubbleRob_leftMotor 图标，打开其属性修改对话框。点击 **Show dynamic properties dialog** 按钮，勾选 **enable the motor**，勾选 **item Lock motor when target velocity is zero**。
11. 新建第二个轮子，重复以上过程，将其命名为：bubbleRob_rightMotor。
12. 将左轮**连接**到左旋转电机，右轮**连接**到右旋转电机。再把两个电机**连接**到 bubbleRob。得到如下图所示。

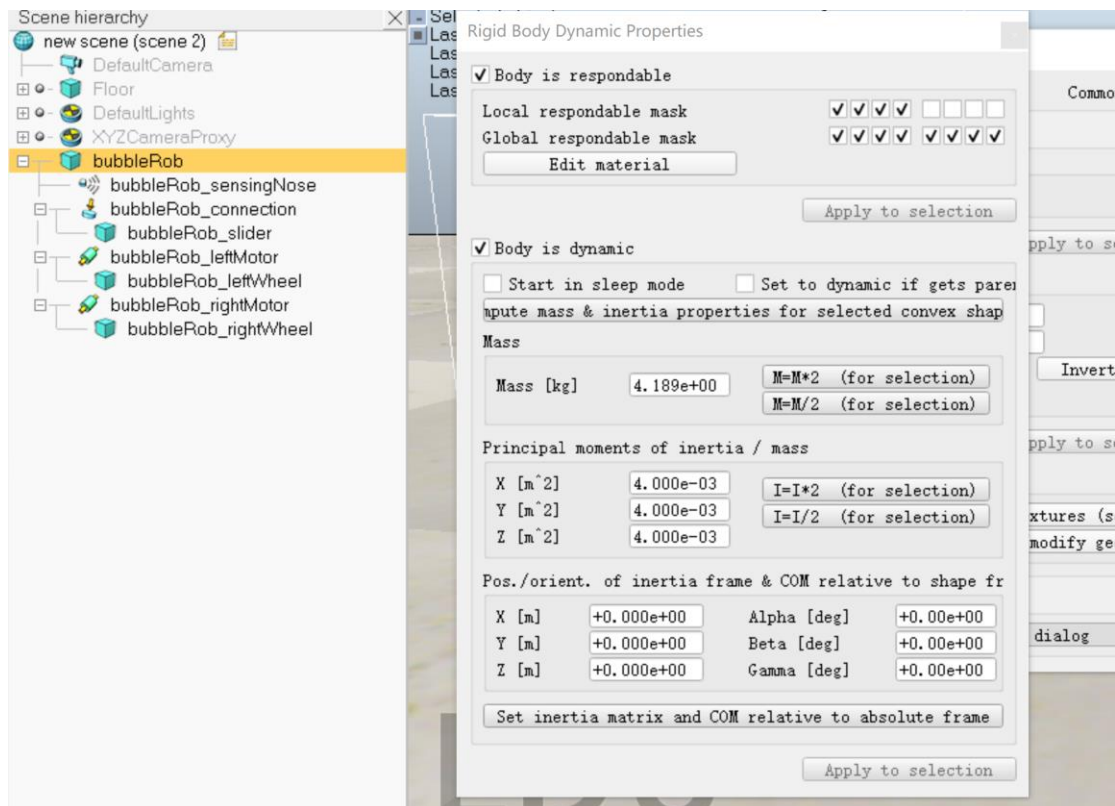


13. 点击开始仿真按钮进行测试。可以看到机器人向后倒下，原因是其缺少与地面的第三个接触点。现在我们为其添加一个脚轮。
14. 在一个新场景中，我们添加一个直径为 0.05 的球体，并使球体 **Collidable**、**Measurable**、**Renderable** 和 **Detectable**，然后将其重命名为 bubbleRob_slider。打开其属性修改对话框。点击 **Show dynamic properties dialog** 按钮，点击 **Edit Material**，在 **Apply predefined settings** 下拉菜单中选择 **noFrictionMaterial**。为了将脚轮与机器人的其余部分刚性连接，我们通过[Menu bar --> Add --> Force sensor] 添加一个力传感器对象。我们将其重命名为 bubbleRob_connection 并将其位置上移 0.05m。将滑块**连接**到力传感器，然后复制两个对象，切换回之前的场景并粘贴它们。将力传感器沿绝对 X 轴移动 **-0.07m**，然后将其**连接**到机器人身体上。得到如下图所示。




15. 如果我们现在运行模拟，我们可以注意到脚轮会相对于机器人身体轻微移动：这是因为两个物体（即 bubbleRobRob_slider 和 bubbleRob）正在相互碰撞。为了去除在动力学模拟过程中出现这种奇怪的效果，打开 bubbleRob 的 **Show dynamic**

properties dialog 对话框。在将其 **local responsible mask** 设置为 11110000。同理，将 bubbleRob_slider 的 **local responsible mask** 设置为 00001111。再次进行仿真，可以看到两个对象不再互相干扰。



- 再次运行仿真并可以注意到 BubbleRob 在轻微移动，即使两个轮子的电机已锁定。尝试使用不同的物理引擎运行仿真，发现结果会有所不同。动态模拟的稳定性与所涉及的非静态形状的质量和惯性密切相关。为了消除这种不良影响，对于两个轮子和脚轮，分别打开其属性设置中的 **Show dynamic properties dialog** 对话框，点击三次 **M=M*2 (for selection)**。该操作会把选中的每个对象质量乘以 8。对于这三个对象，在同一对话框点击三次 **I=I*2 (for selection)**，将其惯性乘以 8。再次运行仿真，我们发现稳定性有所提高。在两个 joint 的动力学对话框中，我们将两个电机的 **Target velocity** 设置为 50。运行仿真，发现 BubbleRob 现在向前移动并最终从地板上掉下去。将两个关机电机的 **Target velocity** 设置为 0。

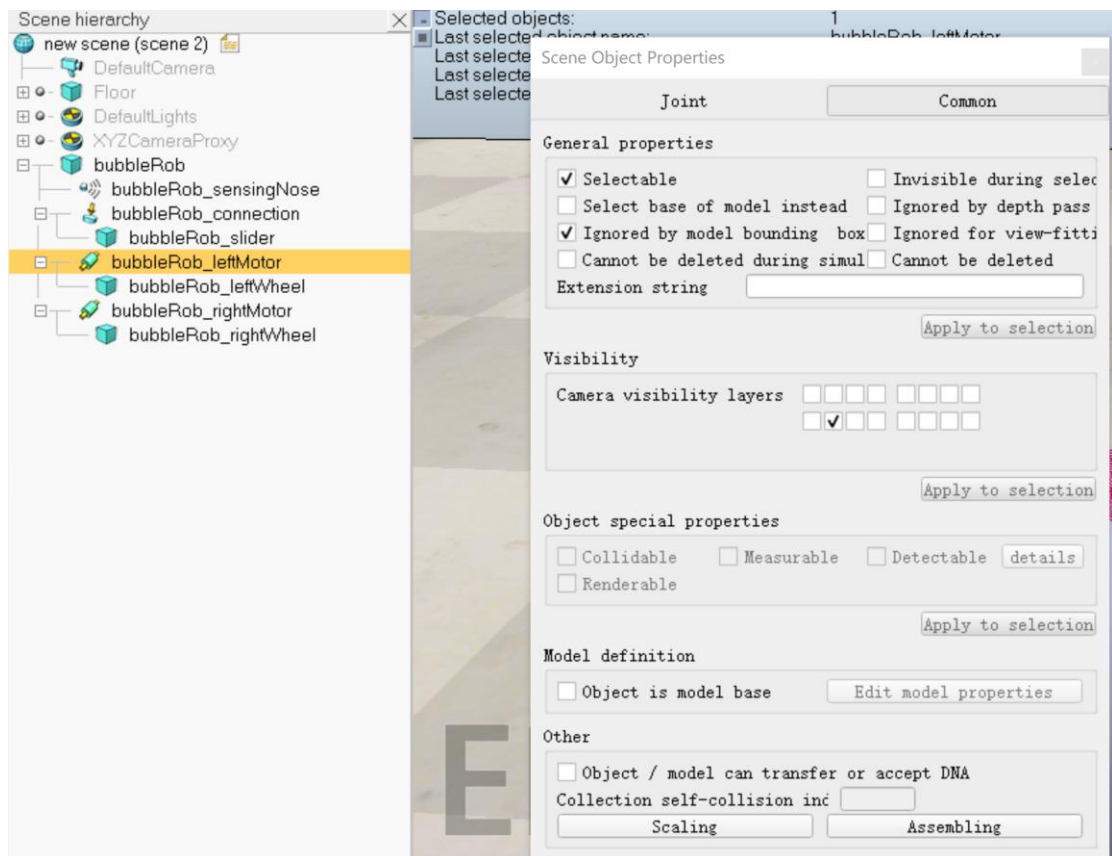
障碍物设置

- 增加一个圆柱体，尺寸设置为：(0.1, 0.1, 0.2)。我们希望这个圆柱体是静态的（即不受重力或碰撞的影响），但仍然对非静态可响应形状施加一些碰撞响应。为此，我们在其形状动态属性中禁用 **Body is dynamic**。然后将其设置为 **Collidable**、**Measurable**、**Renderable** 和 **Detectable**。
- 当圆柱体仍处于选中状态时，单击对象平移工具栏按钮 ，现在我们可以将其拖动到场景中的任意位置，同时其 Z 坐标仍保持不变。我们复制并粘贴圆柱体几次，然后将它们移动到 BubbleRob 周围的位置（从顶部查看场景时执行此操作最方便）。在对象移动期间，按住 shift 键可以执行较细的移动间隔。若按住 ctrl 键则允许其在与

常规方向正交的方向上移动。

完成机器人模型设置（finish BubbleRob as a model definition）

1. 为机器人增加一个 **Graph**，[Menu bar --> Add --> Graph]。将其重命名为：bubbleRob_graph。然后将其**连接**到 bubbleRob。将其绝对坐标设置为(0,0,0.005)。
2. 选中 bubbleRob，打开其属性修改对话框，切换到 **common** 标签页。选中 **Object is model base** 和 **Object/model can transfer or accept DNA**。之后可以看到在该模型中所有对象的周围出现了一个点状边界框。
3. 分别选择两个关节电机，接近传感器和 **Graph**，打开其属性修改对话框，切换到 **common** 标签页。勾选 **Ignored by model bounding box**。同时，在该对话框，我们为两个关节电机，接近传感器设置：取消 **camera visibility layer 2**，勾选 **camera visibility layer 10**。



4. 为了完成模型定义，我们选择**视觉**传感器、两个轮子、滑块和图形，然后启用 **Select base of model instead**：如果我们现在尝试在场景中的模型中选择一个对象，整个模型将被选中，这是将整个模型作为单个对象进行处理和操作的便捷方式。此外，这可以保护模型免受无意修改。模型中的单个对象仍然可以在场景中通过使用 control-shift 单击选择它们来选择，或者通常在场景层次结构中选择它们。
5. 接下来，我们将添加一个视觉传感器，其位置和方向与 BubbleRob 的接近传感器相同。点击[Menu bar --> Add --> Vision sensor --> Perspective type]，然后将视觉传感器**连接**到接近传感器上，将视觉传感器的局部位置和方向都设置为 (0,0,0)。在其

common 属性设置中取消所有 **camera visibility layer** 的勾选，将其设置为视觉传感器不可见。在属性对话框的 **vision sensor** 标签页中，将 **far clipping plane** 项设置为 1，将 **Resolution x** 和 **Resolution y** 项设置为 256 和 256。

6. 在场景中右击，**[Add-->Floating view]**，添加一个浮动视图。先选中视觉传感器，再在新添加的浮动视图上右键点击**[Popup menu --> View --> Associate view with selected vision sensor]**，将视觉传感器的内容输出到该视图。
7. 为视觉传感器增加一个脚本程序。选中视觉传感器后右击，**[Add --> Associated child script --> Non threaded]**。双击其旁边新出现的脚本图标会打开脚本，将脚本中对应的代码修改如下。

```
function sysCall_vision(inData)
    simVision.sensorImgToWorkImg(inData.handle) -- copy the vision
sensor image to the work image
    simVision.edgeDetectionOnWorkImg(inData.handle,0.2) -- perform
edge detection on the work image
    simVision.workImgToSensorImg(inData.handle) -- copy the work
image to the vision sensor image buffer
end
```

```
function sysCall_init()
end
```

此时运行仿真可以看到视觉传感器的图形出现在视图中。

8. 最后一步是为 bubbleRob 增加脚本控制程序，实现避障。选择 bubbleRob，右击，**[Add --> Associated child script --> Non threaded]**。双击其旁边新出现的脚本图标会打开脚本，将脚本中对应的代码修改如下。

```
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)
-- this is bubbleRob's handle
    leftMotor=sim.getObjectHandle("bubbleRob_leftMotor") -- Handle of
the left motor
    rightMotor=sim.getObjectHandle("bubbleRob_rightMotor") -- Handle
of the right motor
    noseSensor=sim.getObjectHandle("bubbleRob_sensingNose") -- Handle
of the proximity sensor
    minMaxSpeed={50*math.pi/180,300*math.pi/180} -- Min and max
speeds for each motor
    backUntilTime=-1 -- Tells whether bubbleRob is in forward or
backward mode
    robotCollection=sim.createCollection(0)
```



```

sim.addItemToCollection(robotCollection,sim.handle_tree,bubbleRobBase
,0)
    distanceSegment=sim.addDrawingObject(sim.drawing_lines,4,0,-
1,1,{0,1,0})

robotTrace=sim.addDrawingObject(sim.drawing_linestrip+sim.drawing_cyc
lic,2,0,-1,200,{1,1,0},nil,nil,{1,1,0})
    graph=sim.getObjectHandle('bubbleRob_graph')
    distStream=sim.addGraphStream(graph,'bubbleRob
clearance','m',0,{1,0,0})
    -- Create the custom UI:
    xml = '<ui title="'.sim.getObject(bubbleRobBase)..
speed" closeable="false" resizable="false" activate="false">'..[[
        <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_sensing()
    local
result,distData=sim.checkDistance(robotCollection,sim.handle_all)
    if result>0 then
        sim.addDrawingObjectItem(distanceSegment,nil)
        sim.addDrawingObjectItem(distanceSegment,distData)
        sim.setGraphStreamValue(graph,distStream,distData[7])
    end
    local p=sim.getObjectPosition(bubbleRobBase,-1)
    sim.addDrawingObjectItem(robotTrace,p)
end

function sysCall_actuation()
    result=sim.readProximitySensor(noseSensor) -- Read the proximity
sensor
    -- If we detected something, we set the backward mode:
    if (result>0) then backUntilTime=sim.getSimulationTime()+4 end

    if (backUntilTime<sim.getSimulationTime()) then

```

```

        -- When in forward mode, we simply move forward at the desired
speed
        sim.setJointTargetVelocity(leftMotor,speed)
        sim.setJointTargetVelocity(rightMotor,speed)
    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

```

9. 运行仿真。 BubbleRob 将在尝试避开障碍物的同时向前移动。 在仿真中：(1) 更改 bubbleRob 的速度；(2) 将 bubbleRob 复制/粘贴几次，会发现多个机器人的计算需求使得仿真的速度受到影响。