Lab 1 Outline

Links:

Object Types: https://www.coppeliarobotics.com/helpFiles/en/objects.htm

Dyanmics Engines: https://www.coppeliarobotics.com/helpFiles/en/dynamicsModule.htm

Child Scripts: https://www.coppeliarobotics.com/helpFiles/en/childScripts.htm

Threaded and non-threaded script code:
https://www.coppeliarobotics.com/helpFiles/en/threadedAndNonThreadedCode.htm

API functions: https://www.coppeliarobotics.com/helpFiles/en/apiFunctions.htm

Module 1: Introduction and Installation

Goals:

Install Coppelia Sim

Welcome to Robot Academy. This lab will serve as an introduction to CoppeliaSim. CoppeliaSim is a free robotics simulator that's fantastic for getting started in the robotics field. This first module in this lab will show the installation process for CoppeliaSim. The following modules will teach a variety of basic CoppeliaSim functions and techniques that should leave you in a great spot to continue with more advanced labs. You can read more about CoppeliaSim on the home page at coppeliarobotics.com

You can click the Features link on the nav-bar to read about the many features CoppeliaSim has to offer. Here you can see that CoppeliaSim is cross-platform, so any operating system is usable and any simulations you create can be transferred to other platforms. There is a very wide array of features listed here, and we will just scratch the surface in this lab. Future labs will expand on the features learned in this lab and will also teach new features.

Click the Downloads link to start the installation process. Notice on the top it explains we are downloading CoppeliaSim for Windows. Make sure to click choose a different platform if you are not on Windows. Choose the Edu download link, as it is free and contains all the features you'll need to edit and simulate a robotics scenario. This dialog notes that the software should only be used for educational purposes, and is not for commercial use. Check that you understand and agree with the terms, and continue downloading.

Once your download is complete you can run the installer.

That's it, you're all setup to start simulating. In the next lab, we'll go over the the basic UI of coppeliasim, as well as how to open, save, and work with scenes.

Module 2: User Interface and Scenes

Goals:

Understand the default User Interface of CoppeliaSim

Learn how to Open, save, and work with scenes.

Welcome back to Robot Academy. In this module, we'll go over the standard user interface you see when opening CoppeliaSim for the first time, as well as how to work with different scenes.

Firstly, we'll go over the main simulation window. This is where your simulation environment takes place. On the bottom-right, you can see your x, y, and z axis's relative to your current camera view. The bottom left will show the flavor of CoppeliaSim you have installed. If you followed the previous video, this should show the EDU version like mine. On the top left, there's some text showing the selected objects. To select an object, directly to the left of the main simulation window is the Scene hierarchy. Simply click an object in the Scene hierarchy, for instance the floor, and you will now see that there is 1 selected object in the simulation text. You can see a list of properties for this object here, including the last selected object name, type, position, and orientation. Clicking the square will cycle through different text and background styles for the object properties. Clicking the dash will minimize this element of the user interface altogether.

I recommend keeping this open, as it can come in handy, especially when working with precise coordinates. More on the scene hierarchy, by default Coppelia sim loads a new scene when you open the software. You can open a new scene by going up to the menu bar, clicking file, and clicking open scene. CoppeliaSim comes free with a wide array of scenes for you to play around with. I encourage you to play around with these once we have a decent understanding of the user interface. Go ahead and select the first 3DofHolonomicPathPlanning.

From here, we can learn more about the Scene hierarchy. The tabs on the top show the different scenes we have open. We can freely switch between scenes by clicking the different tabs. Try not to have too many taps open and simulations running, as having too many open could cause potential crashes. In the scene hierarchy, you'll see all the objects in the current scene. Select the MazeOutline object. You'll see white lines surrounding the object, representing the bounding box the object is bound by. You'll also see the different axes of the object. Since the object's orientation is (0, 0, 0), the orientation of the object's axes matches the environments axes. Notice that all objects below the MazeOutline are indented in the hierarchy. This shows that these objects are children of the MazeOutline object. This tree continues on, L_goal is a child of GoalConfiguration, which is a child of MazeOutline. We'll explain more on that later when we get into moving objects. Each object has its object type shown in the object icon. I will attach a link in the documentation to show all the different object types and their associated icons.
https://www.coppeliarobotics.com/helpFiles/en/objects.htm

Note that we can also switch scenes by selecting the Scenes from the menu bar. For now, we can close

out of the scene we just opened. Don't worry, we'll come back to maze pathfinding in a future lab. To close the scene, go back up to the menu bar, select file, then select close scene. If it asks you if you wish to save the changes, just select no as we didn't make any meaningful changes for an experiment.

In the scene hierarchy, we can modify an object's properties by doubleclicking the object's icon. Let's modify the scene object itself. Double click the scene icon. This will bring up the Environment properties dialog box. The background color in the environment is a gradient. Select the background (up) button, and play around with the color sliders to change the top gradient of the background. You can do the same with the background (down) button. Click the Ambient light button to adjust the lighting in the scene, and click the adjust fog parameters to add fog to the scene. That's all we'll play around with for now, note that there are some potentially useful options in the scene, such as disabling shape textures, and the ability to lock the scene after a save. We can leave these unchecked for now, though.

To the left of the Scene hierarchy, we'll find the model browser. CoppeliaSim has a bunch of different models pre-loaded that you can simply drag into the scene. In the top half of the model browser, you'll see all the different folders you can select. In the bottom half, you'll see all the models that are in the currently selected folder. Select the robots folder, and select the mobile folder, and drag the Asti.ttm robot into the scene. You can drag any robot you'd like for the purpose of this module. Most of these robots come pre-loaded with scripts, and this dialog lets you know you can modify these scripts to fit your simulation's needs. You should now be able to see the robot you dragged in, in your scene hierarchy. To the right of your robot, there should be a script icon. Double click the script icon will bring up the script, which you are free to modify. Be careful modifying pre-loaded robots' scripts, as they can easily be broken.

On the bottom of the user interface, we'll find the simulation console. You can drag the resizable bar on top to increase the size of this. In the input box, you can input Lua code like you would in a command line. For instance, let's print a statement to the console. Type print("Hello World!"), and press enter. You'll see your message show up in the console.

To save your scene, click file on the menu bar, then select save scene as, and save as a coppelia sim scene. Type a meaningful name for your project, then click save. If you close your scene like we did earlier by selecting file, and close scene, you can re-open your scene quickly by selecting file, then open recent, and selecting your scene.

That's it for this module. We went over the main elements in the user interface you'll be interacting with. In the next lab, we'll go over camera manipulation, how to move objects, and running a simulation. Thanks for watching and I'll see you in the next module.

Module 3: Camera Manipulation, and running a simulation

Goals:

Learn how to manipulate the camera

Learn how to run a simulation, and the adjust different parameters we can change for the simulation.

Welcome back to Robot Academy. In this lab, we'll be going over how to manipulate the camera in various ways, as well as how to run the simulation, and the different modifications we can make to our simulation. Firstly, let's disable the fog we enabled in the last lab, as it makes it kind of hard to see. Double click the Scene icon, Click the adjust fog parameters, then deselect the check box.

Currently, we can only see our robot from the shoulders down. We can zoom the camera in and out by scrolling the mouse wheel up and down. We can also achieve the same result by selecting the Camera Shift button, and clicking, holding, and dragging the mouse up and down.

To pan the camera, select the Camera Pan button. There are two ways to pan the camera. The first way is to click, hold, and drag the mouse on an object in the scene environment. This will pan the camera relative to the selected object in the scene. The other way to pan the camera is to click, hold, and drag the mouse, with the mouse hovered over no objects. This will pan the camera relative to the actual camera position.

To rotate the camera, select the Camera Rotate button. We can rotate the camera with the same methods we used to pan the camera. Click, hold, and drag the mouse on an object in the scene environment. This will rotate the camera around the selected object. Next, click, hold, and drag the mouse without a selected object. This will rotate the actual camera rotation in the direction of the mouse.

If you want the camera to focus on a particular object, select the object in the scene hierarchy. In this case, I'll select the robot we dragged into the scene. Next, select the fit to view button to focus the camera on the selected object.

While I'm in this area, I may as well go over the undo and redo buttons. Simply click the undo button if you wish to undo your last action. And click redo to redo the action.

Next, we'll go over some simulation settings you can modify. In general I'd recommend keeping these at the default values. Nevertheless, the dynamics engine dropdown allows you to change the physics engine selected. I will provide links to the different dynamic engine properties in the documentation.

https://www.coppeliarobotics.com/helpFiles/en/dynamicsModule.htm

The dynamics settings drop down allows you to change whether you want the engine to prioritize speed or accuracy. The simulation time step allows you to change the increment at which the simulation steps.

Let's get setup to run the simulation. Rotate and pan the camera using the camera techniques we just learned so that we have a good view of the entire simulation.

Now, clicking the play button will start the simulation. Any user interface elements will pop up immediately when starting the simulation. For me, I'm going to reduce the Asti's step size so that he walks in place. You can click the pause button to pause the simulation. And you can click play again to start the simulation again. You can adjust the speed of the simulation. Click the turtle to slow the simulation down. And click the rabbit to speed the simulation up. Click the page selector to view the simulation from some pre-determined points of view. Click the stop simulation button to stop the simulation and return everything to its original position.

That's it for this module. We learned how to manipulate the camera and run a simulation. In the next lab, we'll learn about the different ways to move and rotate objects.

Module 4: Moving and Rotating Objects

Goals:

Learn how to move and rotate objects in different ways.

Learn how objects attach to one another on the scene hierarchy.

Welcome back to Robot Academy. In this module, we'll learn how to move and rotate objects in different ways, as well as how objects attach to one another. I've set up the scene so I can easily demonstrate some different ways to move and rotate objects. If you'd like to add a cuboid, simply select Add on the menu bar, select a primitive shape, and select a cuboid. Click OK to generate the cuboid. My cuboids are already setup, so I'm going to cancel this dialog.

Firstly, lets go over how to rotate objects. Select the the object either in the scene hierarchy or in the simulation environment, then click object/item rotate. The mouse translation tab allows us to rotate the object with our mouse. For this rotation, let's select relative to the world to rotate the cuboid relative to the world. You can adjust the step size to increase or decrease the amount of degrees change as our mouse moves. Lastly, we can select the axes we want to rotate the cuboid around. Let's select the z-axis for this rotation, and rotate it approximately 45 degrees. You can see the degree amount next to the object as you rotate it.

The Rotation tab allows you to input degree amounts into each axes. This allows for very precise rotations. This time, lets select a rotation relative to its own frame, as the objects axes no longer line up with the worlds axes. Let's rotate the object around the x axis by 85 degrees. We can see that the cuboid doesn't quite end up upright, as we expected from an 85 degree rotation. Add another 5 degrees to the rotation to straighten it out.

Next, let's try moving the object. With the object selected, select the object/item shift. The dialog is very similar to the rotation dialog, and behaves the same way. If we shift the item relative to the world, the object will move about the world's axes. If we shift the item relative to its own frame, we'll see it shifts the item about the object's axes.

Let's move the cuboid a bit. The position tab allows us to enter exact coordinates for the object to be placed. To move it back to the center, we can enter the coordinated 0, 0 to move the object to the exact center of the simulation.

The Translation tab allows us to move the the object a specific distance away from its current position. For example, if we select relative to its own frame, then apply 1 meter of x translation, the cuboid will move 1 meter in its x axis direction.

Now, we can actually apply the coordinates of another object to this object. To do this select the object you want to move, hold shift, then select the object you want to move to. Select object item/shift again, go to the position tab, and click apply selection. These two object's coordinates are now identical. We can apply the same process to the object's orientation, which I will demonstrate by coming to object item/rotate, coming to the orientation tab, and clicking apply to selection. The two objects now have the exact same position and orientation. In fact they're so neatly inside each other that there only appears to be a single cuboid.

Let's shift gears a bit to understand how objects attach to one another, and what this means for object shifts and rotations. We have two cuboids here, a parent cuboid and its child cuboid. Any shift or rotation done to the parent cuboid, will also be applied to its attached child.

You may have also been wondering what it means to move relative to its parent frame, as we haven't used this option yet. Basically, when moving a child object, selecting parent frame will move it relative to the parent frames axes. To demonstrate this, I'll first rotate the parent element. Then, I'll rotate the child cuboid back to its original orientation by rotating it relative to the world. Then I can shift the child object relative to its parent frame. You'll see that child moves in the direction of the parent frames axes. It should also be noted, that when moving a parent object, the attached object does not physically need to be attached to the parent. As long as the child object is a child in the scene hierarchy, the child will move along with its parent.

That's it for this module. This module was a comprehensive look at the different ways to shift and rotate objects. The next lab will teach you all about scripts, including the different scripts available, some LUA code, as well as the different API's at your disposal in CoppeliaSim.

Module 5: Scripts

Goals:

Learn about the different script types

Learn how to add a child script.

Learn about LUA code as well as API calls

Hello, and welcome back to Robot Academy. In this module, we'll learn how to write different scripts, learn some LUA code, as well as different API calls available through CoppeliaSim. Firstly, to add a script, click the script icon located on the left hand side of the software. Click Insert new script. Select the drop down and view the different script types available. We will be focusing on child scripts in this module. There are threaded and non-threaded child scripts. The difference on a very, very basic level is that threaded scripts run sequentially with a thread, and can used thread calls such as sim.wait() to instruct the thread to wait before proceeding. Non-threaded scripts use callback functions to determine if CoppeliaSim should halt. Non-threaded scripts are also technically less-resource intensive and should run faster than threaded scripts. Again, this is a very basic explanation, if you are interested in reading more you can click the link in the documentation. For the purpose of this lab, we'll select a threaded script.

Now that we've added the threaded child script, we actually need to click out of the script dialog box, so that the script properties become available to us. We're going to mostly leave these as the default values, but we want to associate an object with it. From our last module, let's select the ParentCuboid to associate the script with. You should now see a child script associated with the ParentCuboid. You can double click the child script to view the script.

The function sysCall_threadmain() will contain your main code, as this calls the thread to execute the given code. You can delete all of the commented code in here to make space for your main code. Now, let's do some API calls. To get an object in the scene, you can assign sim.getObjectHandle('objectHandleName') to a variable.

We can repeat this process to assign each object to a variable.

We've already moved the "CuboidToMove" in the last module. We can start off the script by removing the object from the scene. This can be done by calling sim.removeObject(). After calling the API, run the simulation and you'll notice the cuboid is deleted from the scene. Stop the simulation, and return to the script. We need to remove references of this object, so that the code doesn't throw any errors the next time its run.

For the learning purposes of this lab, we're going to replace the CuboidToMoveTo with a dummy object. A dummy object will serve the purpose of a position to move to much better than a cuboid will. We're then going to move the child, then parent objects to the dummy objects and observe the results.

So the first step is to create a dummy object, and move it to the current position of the

CuboidToMoveTo. Assign a new variable to the API call sim.createDummy(). Sim.createDummy takes a size parameter, and then a color variable. Assign it a size of .05, and set the color to nullptr to select the default color value.

You'll notice that when you stop the simulation, the dummy is not saved in the scene hierarchy. Navigate to the menu bar, select simulate, and select simulation settings. Here we can uncheck remove new objects at the end of the simulation.

Next, we want to get the cuboidToMoveTo's position, and set the dummys position to the cuboid's position. To get the cuboid's position, call sim.getObjectPosition() and assign it to a variable. Enter the objectHandle, and -1 for the parameters.

Set the dummys position to the cuboid position by calling sim.setObjectPosition(), passing the parameters with the dummy's object handle, -1, and the cuboid's position.

Now that the dummy object is setup in the right position, we can remove the code we used to get it there, and remove the cuboid object. Remove the calls to create the dummy, and remove the cuboid from the scene like we did in the last example.

Now that the dummy object is setup and all unnecessary elements are removed from the scene, let's move the child object to the dummy object, wait a bit, then move the parent object to the dummy, and observe the results.

Get the dummy objects handle, the dummy objects position, then set the parent cuboids position equal to the dummy position. Next we can run a command to make the simulation wait, and move the child object to the dummy position.

Notice that the child object does not move with the parent object. This is expected and due to the script being threaded. In addition, the child cuboid will not be able to occupy the space the parent cuboid takes up, so it's movement may appear strange.

That'll do it for this lab. This module taught you about adding a child script, the different types of scripts, and some LUA and API calls within CoppeliaSim. Thanks for sticking through the lab, I hope you learned a lot. Consider taking the quiz on your Robot Academy dashboard to test your knowledge. We'll see you in the next lab, take care.