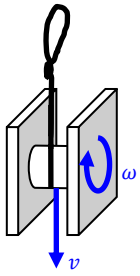
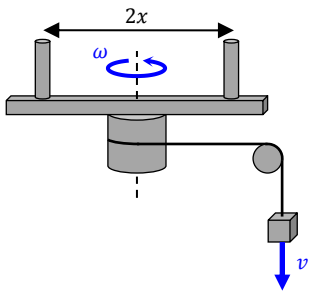
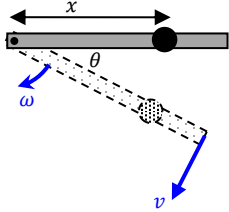


Coding Options for PHYS 161 Oral Presentation 3

Option 1	Option 2	Option 3
 <p>Simulate yo-yo-falling distance h. Input plate dimensions, masses, & h. Output the following plots:</p> <ul style="list-style-type: none"> • speed (v) versus time • speed (v) versus distance fallen • <i>final</i> speed versus axle radius 	 <p>Simulate above device as hanging mass falls distance h. Input masses, dimensions, h & x. Output the following plots:</p> <ul style="list-style-type: none"> • speed (v) versus time • speed (v) versus distance fallen • <i>final</i> speed versus x 	 <p>Simulate above device released from rest for one full oscillation. Input masses, dimensions, & x. Output the following plots:</p> <ul style="list-style-type: none"> • rotation speed (ω) versus time • end of rod speed (v) versus time • <i>max</i> ω versus x.

Students typically have trouble with strings & cylinders. I can help you get the position & axis vectors correct.

TIP: enter a bunch of *scalar* constants at the top of your code (i.e. `axle_radius`, `axle_height`, `pulley_radius`, `pulley_thickness`, `plate_mass`, etc). This will allow us maximum flexibility in initializing the various objects we need to draw.

Also, figure out how to add textures to any rotating objects. If you want to see the rotation this is pretty critical. In the past I have added a radius line (using the cylinder command) to my pulleys and cylinders that rotates with the cylinder to show rotation. This also works well but it is probably easier to just texture rotating objects.

It is possible to group objects using the COMPOUND function. This may or may not be useful. Potential issues with the compound function:

- Ensure you color and size objects separately before compounding. If I read correctly, at the time of writing this file all items in the compounded object must have the same texture...see the help file.
- After compounding, I think it may be difficult to resize things. For more read the help file.
- All objects will share a common position which is not necessarily at the location you expect! There is a way to manually redefine the objects position if you read the entire help file...
- Help file location: <https://www.glowscript.org/docs/VPythonDocs/compound.html>

The Euler-Cromer Method will still apply, but implementation is slightly different with rotations. I hope the table below helps you make some sense of the changes. In addition, I would watch the entire slow-paced training vid linked below. Many, many students have tried to use my shorter training vids but they all end up having to watch the slow paced one to get anywhere. Just accept the pain and play it at double speed until things start to get tricky for you.

<https://www.youtube.com/watch?v=RGzfWdh42qM&list=PL4S11ZPMcTDX-cOkqRjsVD4HvaLVvJPE8&index=13>

ECM for translation	ECM for rotation
1) compute <i>net</i> force vector ($\Sigma \vec{F} = \vec{F}_1 + \vec{F}_2 + \dots$)	1) compute <i>net</i> torque vector ($\Sigma \vec{\tau} = \vec{\tau}_1 + \vec{\tau}_2 + \dots$)
2) compute acceleration vector ($\vec{a} = \frac{\Sigma \vec{F}}{m}$)	2) compute angular acceleration vector ($\vec{\alpha} = \frac{\Sigma \vec{\tau}}{I}$)
3) <i>update</i> velocity vector ($\vec{v} += \vec{a} dt$)	3) <i>update</i> angular velocity vector ($\vec{\omega} += \vec{\alpha} dt$)
4) <i>update</i> position vector ($\vec{r} += \vec{v} dt$)	4) compute scalar angle increment ($d\theta = \pm \omega \cdot mag * dt$)
5) increment time ($t += dt$)	5) use rotate command (helpfile link)
	6) increment time ($t += dt$)

Notice you are supposed to create plots for this lab. It is easy to make plots in glowscript but the formatting is often annoying. You might revisit this quick training vid showing you how get data from glowscript into a CSV file.

<https://www.youtube.com/watch?v=oAbYQWcBj1w&list=PL4S11ZPMcTDX-cOkqRjsVD4HvaLVvJPE8&index=19>

Finally, the last plot in each case is a level up in difficulty. It is you using your code to do real science for once. To help you understand the process of taking a simple code that runs a *single* simulation to one which runs a *set* of simulations and outputs the plot I made this training video. I also linked some codes in the video comments you can open up and look at.

<https://www.youtube.com/watch?v=DifkriEiuU&list=PL4S11ZPMcTDX-cOkqRjsVD4HvaLVvJPE8&index=14>

The first 10 minutes explain a simple projectile code animation which output range for one velocity & launch angle. I then ask, "What angle gives max range?"

Rather than running the code 100 different times with slightly different inputs, can I make a loop do this for me?

The next code shows how I implement that.

The last code shows how I can get all that data into a plot.