# Lab 4: Linear Transformations and Graphics

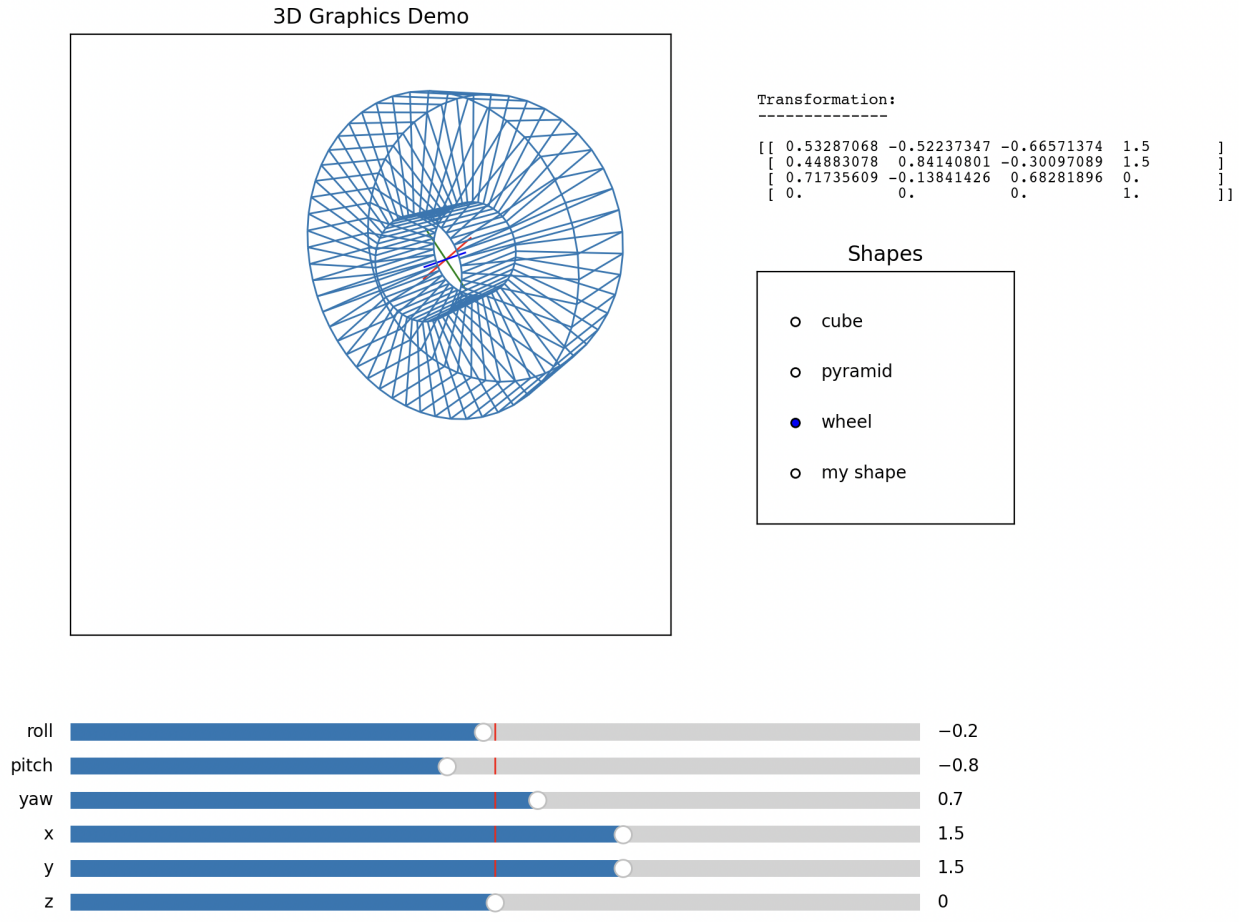
CAS CS 132: *Geometric Algorithms*

Due November 6, 2025 by 8:00PM

This week you'll be building the graphics demo from lecture. You won't be required to submit a lab write-up. Instead you'll submit your code to Gradescope and it will be autograded. There will be a very small extra credit part that can be optionally submitted as the write-up.

## Outline

The objective of this lab is to build an interactive `matplotlib` widget that manipulates 2D renderings of 3D wireframes. Once you're done, it should like something like this.




The starter code in given in `lab4.zip` and has two files, one called `lab4.py` that you will edit, and the other called `demo.py` that contains the code for the widget itself. Don't change the names of these files or the

names of any functions included in the starter code. The only changes you should make are to fill in the provided TODO items. At a high-level the code works as follows:

1. 3D wireframes are represented by 3D line segments, which are represented as pairs of 3D points, which are represented as triples of floating point-numbers. The function `shape_to_matrix` takes a 3D wireframe with $n$ line segments and converts it into a $4 \times 2n$ matrix in which each column is an endpoint of a line segment in the wireframe in *homogeneous coordinates*.

2. The output of `shape_to_matrix` will be transformed by the output of `transform_matrix`, which should implement a combination of rotation and translation. See the docstring for details. (*Hint.* Note that translation should happen *after* rotation.)

3. 2D line segments are represented as pairs of 2D points, which are represented as pairs of floating-point numbers. The function `matrix_to_shape` converts a $4 \times 2n$ matrix into a list of $n$ 2D line segments that will be rendered on the screen. This is done by

   (a) applying the projection matrix with viewing position `(0, 0, 10)` to the given matrix,

   (b) then converting each column from homogeneous coordinates to 2D Cartesian coordinates via the transformation discussed in lecture and lab,

   (c) then grouping every *two* points to create a list of line segments.

4. The above steps are combined into a single function called `full_transform`. This function should:

   (a) convert the 3D shape to a matrix,

   (b) apply the transformation matrix, and

   (c) convert the matrix to a 2D rendering.

Once you've completed every TODO item, you should be able to run `python3 demo.py` to see the widget in action.

## Extra Credit

For a small amount of extra credit, you can build a wireframe (of some reasonable complexity) using the variable `extra_credit_shape` which can be viewed in the widget. In order to get credit, you must submit a pdf write-up that includes the following.

1. A *high-quality* screenshot of the demo with your shape in a non-default orientation. This means no blurry images, no pictures of your computer screen.

2. You must name your shape, using the variable `extra_credit_name`. If your picture says "my shape" you will receive no credit.

3. One or two sentences about how you generated your wireframe, e.g., whether you did it programmatically or by hand, where you got the inspiration, etc.

4. You must clearly state in your write-up if you're comfortable making the image public (in case we collection the results on a public facing website). If you do not explicitly state this, you will receive no credit.

## Submission

This week, you'll upload the single file lab4.py to Gradescope under the assignment **Lab 4 (Programming)**, where you can verify that it passes some (but not all) autograder tests. Test your system early, here may be system dependent issues that we'd like to address as early as possible. For the extra credit, you'll submit a pdf write-up (as you did in previous weeks) under the assignment **Lab 4 (Extra Credit)**.