

MCS 320 Project Three: Wire-frame modeling with MATLAB due Wednesday 28 April 2004, at 2PM.

The purpose of the project is to explore the capabilities of MATLAB in Computer Aided Design. We will see how with small scripts we can build 3-D models.

0. Wire-frame Models of Polyhedrons

To keep everything very simple and basic, we only work with polyhedra defined by vertices and edges. For example, a tetrahedron with triangular base spanned by vertices $[0\ 0\ 0]$, $[2\ 0\ 0]$, $[1\ 2\ 0]$, and top $[1\ 1\ 1]$ will be represented by two matrices v and e . The matrix v stores in its rows the coordinates of the vertices and the rows of the matrix e contain the labels of vertices spanning the edges. The MATLAB commands

```
>> v = [0 0 0; 2 0 0; 1 2 0; 1 1 1];  
>> e = [1 2; 2 3; 3 1; 1 4; 2 4; 3 4];
```

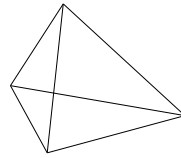
define the four vertices and six edges of a tetrahedron.

1. The wireplot function

The first assignment is to provide an implementation for the `wireplot` function, using the following prototype:

```
function wireplot(v,e)  
%  
% DESCRIPTION:  
%   Makes a wire plot of the polyhedron defined by vertices  
%   in the rows of v and edges in the rows of e.  
%   The plot is in the default 3-D view,  
%   without axis labeling, tick marks and background.  
%  
% REQUIRED:  
%   size(v,2) = 3, i.e.: v has three columns;  
%   size(e,2) = 2, i.e.: e has two columns; and  
%   all elements in e are in the range 1:size(v,1).  
%  
% EXAMPLE:  
%   v = [0 0 0; 2 0 0; 1 2 0; 1 1 1];  
%   e = [1 2; 2 3; 3 1; 1 4; 2 4; 3 4];  
%   wireplot(v,e) % shows a tetrahedron  
%
```

If implemented correctly, the commands in the EXAMPLE section of the prototype produce the following picture:



2. Some Basic Operations

The most general operations on a polyhedron are translations and general transforms. Write a function `translate.m` with following prototype:

```
function tv = translate(v,t)
%
% DESCRIPTION:
%   Translates the vertices in v by adding the vector t
%   to every vector in v.
%
% REQUIRED:
%   size(t,2) = size(v,2), i.e.: t and v have same #columns
%
```

If we multiply the vertices with a matrix, we can implement general transforms. Special matrices lead to special transforms. For example, with a diagonal matrix, we scale the coordinates of the vertices. If we use a rotation matrix we turn the polyhedron. Implement the following function:

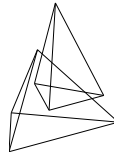
```
function tv = transform(v,t)
%
% DESCRIPTION:
%   Transforms the vertices in v, multiplying every row in v
%   by the matrix t.
%
% REQUIRED:
%   size(t,1) = size(t,2) = size(v,2) = 3,
%   i.e.: t is 3-by-3 matrix and v has three columns.
%
```

When these operations are implemented correctly, the sequence

```
>> v = [0 0 0; 2 0 0; 1 2 0; 1 1 1]; % vertices of tetrahedron
>> e = [1 2; 2 3; 3 1; 1 4; 2 4; 3 4]; % edges of tetrahedron
>> t = [3 2 0]; % displacement vector
>> tv = translate(v,t); % translate polyhedron
```

```
>> s = diag([0.5 0.8 1.2]);           % scaling matrix
>> sv = transform(tv,s);             % scale the polyhedron
>> wireplot(v,e);                    % original polyhedron
>> wireplot(sv,e);                   % scaled polyhedron
```

leads to the following picture:



3. Scaling

Like a set of Russian dolls which all neatly fit into each other we can build a sequence of polyhedra one inside the other. The algorithm to scale the vertices is simple: we first compute the barycenter of the vertices (i.e.: average the vertices), translate the polyhedron so that the barycenter becomes the origin, scale the polyhedron and then translate the scaled polyhedron back to the original coordinates.

Implement the following function:

```
function b = barycenter(v)
%
% DESCRIPTION:
%   Computer the barycenter of the vertices in the rows of v.
%
```

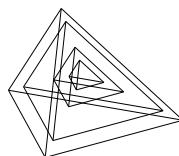
Use `barycenter` in the implementation of `scale`:

```
function tv = scale(v,f)
%
% DESCRIPTION:
%   Scales the polyhedron spanned by the vertices in the rows
%   of v by a (positive) factor f.
%
```

When implemented correctly, the sequence

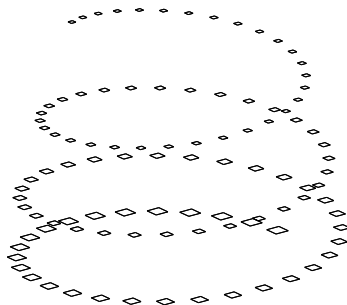
```
>> v = [0 0 0; 2 0 0; 1 2 0; 1 1 1];
>> e = [1 2; 2 3; 3 1; 1 4; 2 4; 3 4];
>> v8 = scale(v,0.8); v4 = scale(v,0.4); v2 = scale(v,0.2);
>> wireplot(v,e); wireplot(v8,e); wireplot(v4,e); wireplot(v2,e);
```

produces the following plot:



4. An Application

With the operations as defined above, we can build a model like



The picture consists of 100 cubes along a narrowing helix. The cubes are getting smaller as they are getting higher.

In this assignment, you need to give the code and picture of this or a similar model. The main requirements are that you use at least one of the operations defined above and that the picture uses about 100 polyhedra.

5. The deadline is Wednesday 28 April 2004, at 2PM.

The solutions to the project will be collected at the beginning of our class meeting on Wednesday 28 April at 2PM. If you cannot come to class that day, then you must arrange to hand in your solution before the deadline. Otherwise, your solution will be discounted with 10 points if it is turned in on the same day before 5PM, and will no longer be accepted afterwards.

Bring to class the printout of your MATLAB session which illustrate the good working of your m-files. With the command `type` you can bring the listing of an m-file in your session, i.e.: execute as `type wireplot.m` to show the content of your m-file.

It is good practice to use one m-file which generates all answers to the assignments.

Also provide printout of the plots to demonstrate the correctness of the functions.

To avoid any misunderstanding, this project must be solved *individually*. Under no circumstances is it allowed to copy or to collaborate. Regardless of who copied from whom, all caught in the act of plagiarism will be penalized.

If you have questions, comments, or difficulties, feel free to come to my office for help.