

```

%=====
% Name:          hw2_1.m
%
% Author:        Kairi Kozuma
%
%=====

% a) Field of view
% Focal length
f = .004;

% Sensor dimensions
w = .0048;
h = .0036;

% Field of view in degrees
fov = (180 / pi) * 2 * [atan(w / (2 * f)), atan(h / (2 * f))];
fprintf('Field of view\n');
fprintf('Horizontal: %f degrees\nVertical: %f degrees\n',fov(1),fov(2));

% b) Lines to project
numpoint = 1000;
line1 = [linspace(-36.6,23.1,numpoint);linspace(-25.7,0.1,numpoint);linspace(66.0,77.0,numpoint)];
line2 = [linspace(-45.0,-81.0,numpoint);linspace(49.7,89.5,numpoint);linspace(150.0,270.0,numpoint)];

line1proj = camera(line1);
line2proj = camera(line2);

figure(1);
plot(line1proj(1,:), line1proj(2,:));
axis([0, 800, 0, 600]);
title('Projection of line 1');

figure(2);
plot(line2proj(1,:), line2proj(2,:), '-*');
axis([0, 800, 0, 600]);
title('Projection of line 2');

fprintf('Line 1 collapses onto a single point in the projection, while line 2 appears to be a line\n');

% c) Plot squares
slpt = 1000; % Number of points for linspace
slw = 7.5; % Square 1 width

square1 = makeSquare([-30.0, -15.0, 100.0],7.5,1000);
square2 = makeSquare([0.0, 27.6, 160.0],12,1000);

square1proj = camera(square1);
square2proj = camera(square2);

figure(3);
plot(square1proj(1,:), square1proj(2,:));
title('Projection of square 1');
axis('equal');

figure(4);
plot(square2proj(1,:), square2proj(2,:));
title('Projection of square 2');
axis('equal');

% Relative size
fprintf('The two squares have the same relative size in the projection');

%===== makeSquare =====
%
% Creates vector of squares with same z coordinate from bottom-left point
% and width of square
%
%
% function coords = makeSquare(lbcoords, width, n)
%
% Input:
% lbcoords - Coordinates of bottom left point, [x,y,z]
% width    - Width of square
% n        - number of points per side
%
% Output:
% coords   - coordinates of all points on the square
%
%===== makeSquare =====

```

```

function coords = makeSquare(lbcoords, width, n)

side1 = [linspace(lbcoords(1),lbcoords(1)+width,n);linspace(lbcoords(2),lbcoords(2),n);linspace(lbcoords(3),lbcoords(3),n)];
side2 = [linspace(lbcoords(1),lbcoords(1),n);linspace(lbcoords(2),lbcoords(2)+width,n);linspace(lbcoords(3),lbcoords(3),n)];
side3 = [linspace(lbcoords(1)+width,lbcoords(1)+width,n);linspace(lbcoords(2),lbcoords(2)+width,n);linspace(lbcoords(3),lbcoords(3),n)];
side4 = [linspace(lbcoords(1),lbcoords(1)+width,n);linspace(lbcoords(2)+width,lbcoords(2)+width,n);linspace(lbcoords(3),lbcoords(3),n)];

coords = [side1, side2, side4, side3];
end

%===== camera =====
%
% Implements the camera projection equations to a CCD-type sensor.
% Takes in the real world coordinates in meters of a point and returns
% the image coordinate location in pixels of the point.
%
%
% function icoords = camera(wcoords)
%
% Input:
%   wcoords   -The world coordinates of a point in meters as a column
%               vector.
%
% Output:
%   icoords   -The image coordinates of the point in pixels as a
%               column vector.
%
%
% If done properly, then passing a matrix of column vectors should perform
% the projection equations for all of the associated vectors and return
% a matrix of column vectors representing the projected 3D points.
% This requires some savviness with Matlab (using .* and ./).
%
%===== camera =====

%
% Name:        camera.m
%
% Author:      Patricio A. Vela, pvela@ece.gatech.edu
%
% Created:     2006/01/19
% Modified:    2013/01/17
%
%===== camera =====

function icoords = camera(wcoords)

[M, N] = size(wcoords);

if (M ~= 3)
    error('Input needs to be 3D column vector(s).');
end

xvals = wcoords(1,:);
yvals = wcoords(2,:);
zvals = wcoords(3,:);

% Focal length
f = .004;

% Sensor dimensions
w = .0048;
h = .0036;

% Sensor resolution
W = 800;
H = 600;

% Pixel size
dr1 = w / W;
dr2 = h / H;

% View-centered projection equations
r1 = f * xvals ./ zvals;
r2 = f * yvals ./ zvals;

% Sensor array equations
R1 = ceil(r1./dr1) + (W / 2);
R2 = ceil(r2./dr2) + (H / 2);

% Camera coordinates
icoords = [R1; R2];

```

end

Field of view
Horizontal: 61.927513 degrees
Vertical: 48.455491 degrees

Line 1 collapses onto a single point in the projection, while line 2 appears to be a line
The two squares have the same relative size in the projection



