Putting a Scene in Three Dimensions onto a Two-Dimensional Canvas

Introduction

OBJECTIVE: Learn to use linear transformations to map points from three dimensions onto a two-dimensional space.

Since time immemorial, artists have faced a problem that computer graphics artists are facing today, and that is how to picture three-dimensional objects on a two-dimensional plane. The representation of the objects may vary, depending on the position of the "eye of the beholder." In the project that follows, we analyze two aspects of this problem. Both involve mapping points in three-dimensional space to a plane. Part I addresses a problem suggested in the text (Section 12.5, Exercise 73) in which the viewpoint belongs to a single beholder. The remainder of the lab focuses on parallel projections that resemble a situation in which the results of X-rays demonstrate the need for a CAT-Scan. The linear algebra behind such mappings is introduced in Part V.

Technology Guidelines

NOTE: If you have just finished a module, restart *Mathematica* or close the *Kernel* before executing a new module.
TO OPEN CELLS, put your cursor on the right cell bracket and double click.
TO STOP AN EXECUTION
Select the *Kernel* pull-down menu and click on *Abort Evaluation.*
ORDER OF EXECUTION
Execute cells in the order given. Do not skip any Input cells within a given notebook.
SAVING NOTEBOOKS
You can save anytime to any directory you choose, and it is wise to save often.
However, before you do your final save, it is a good idea to delete all your output by selecting the
Delete All Output selection under the *Kernel* pull-down menu.
EXPERIENCING MAJOR PROBLEMS

Save if appropriate, then shut down *Mathematica* and start it up again.

Part I: Mapping to the Y-Z Plane from a Single Point

First, consider the problem from the text, where all points are mapped to the y-z plane. The viewer is at the point (x_0, 0, 0), and any point (x, y, z) between the viewer and the y-z plane appears at the point (0, t y, t z) on the y-z plane, where *t* will be a function of the viewer's position and the *x*-coordinate of the point in 3-space. Verify that $t = \frac{x_0}{x_0 - x}$.

The following commands demonstrate this mapping by computing image points as specified after randomly generating points in the domain that lie between the observer and the y-z plane.

In[1]:=

```
Off[General::spell]

Off[General::spell1]

Clear[x, y, z, x0, t, newx, newy, newz, domain,

Print["The number of domain and image points
 numberofpoints = 1000]

x0 = 500;

t[x_, x0_] := x0 / (x0 - x)

newx[x_, y_, z_] := 0

newy[x_, y_, z_] := t[x, x0] y

newz[x_, y_, z_] := t[x, x0] z

domain =
 Table[{Random[Real, {0, 100}], Random[Real,
   Random[Real, {-100, 100}]}, {i, 1, numbero
```

```
image =
 Table[{newx[domain[[i, 1]], domain[[i, 2]], (
   newy[domain[[i, 1]], domain[[i, 2]], domai
   newz[domain[[i, 1]], domain[[i, 2]], domai
  {i, 1, Length[domain]}];

partimage = Table[image[[i]], {i, 1, 10}];

PrependTo[partimage,
 {"x coordinates", "y coordinates", "z coordi

Print["The first ten image points are: ",
 partimage // TableForm]
```

```
The number of domain and image points is
 1000
```

```
The first ten image points are:

 x coordinates      y coordinates      z coor
 0                  27.0269            -89.32
 0                  16.8416            -18.00
 0                  -45.4927           2.7491
 0                  -44.5291           -91.93
 0                  -101.103           65.987
 0                  69.4717            77.610
 0                  -45.0085           57.971
 0                  -40.1864           67.532
 0                  -73.5658           -57.32
 0                  15.7038            46.979
```

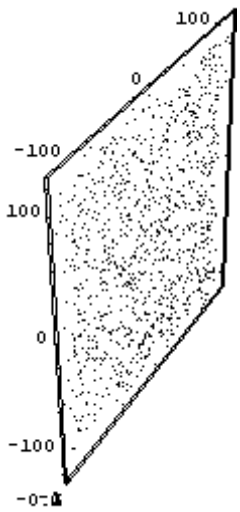
To view the points in the image, first load a graphics package. This command must be executed before using the **ScatterPlot3D** command.

In[15]:=

```
<< Graphics`Graphics3D`
```

In[16]:=

```
sp = ScatterPlot3D[image, PlotStyle -> PointSi
```



Let's suppose that there is an object in the region bounded approximately by z=100-0.5 $\sqrt{x^2+y^2}$ and z=0.005(x^2+y^2). We will first plot the object in three-space and then separate out the image points that have been mapped from that region and plot them in pink. We must first load a different graphing package.

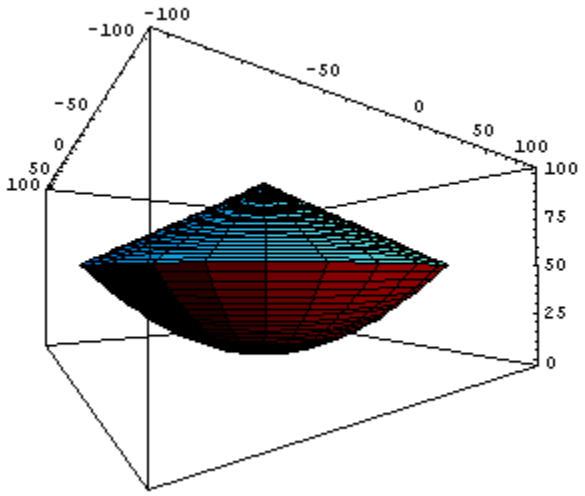
In[17]:=

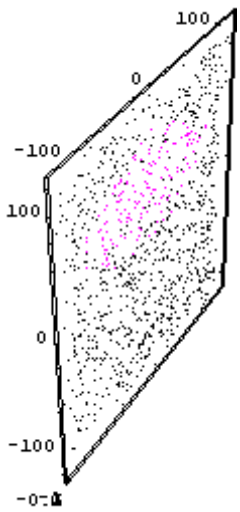
```
<<Graphics`ParametricPlot3D`
```

In[18]:=

```
Clear[r]

p1 = CylindricalPlot3D[100 - .5 r, {r, 0, 100},
  DisplayFunction → Identity];

p2 = CylindricalPlot3D[.005 r^2, {r, 0, 100}, {θ
  DisplayFunction → Identity];

Show[p1, p2, DisplayFunction → $DisplayFuncti
 ViewPoint -> {-0.637, -0.917, -0.018}];

listout = {};

listin = {};
```

```
Do[
 If[
  domain[[i, 3]] <
    100 - .5 Sqrt[(domain[[i, 1]])^2 + (domain[[:
   domain[[i, 3]] > .005 ((domain[[i, 1]])^2 + (
  AppendTo[listin, {image[[i, 1]], image[[i, :
    image[[i, 3]]}],
  AppendTo[listout, {image[[i, 1]], image[[i,
    image[[i, 3]]}]], {i, 1, Length[domain]}]

pout = ScatterPlot3D[listout, DisplayFunction

pin = ScatterPlot3D[listin,
  PlotStyle -> {RGBColor[1, 0, 1], PointSize[.
  DisplayFunction -> Identity];

Show[pin, pout, DisplayFunction -> $DisplayFu
```





You Try It: Part I

If you had a parallel projection onto the y-z plane, any (x, y, z) would map to (0, y, z). In the above problem, let x_0 get larger and larger, and compare your image points to your domain points to see if you are approaching this state. Do this by making the number in red bigger and bigger.

In[28]:=

```
Off[General::spell]

Off[General::spell1]

Clear[x, y, z, x0, t, newx, newy, newz, domain,

Print["The number of domain and image points
 numberofpoints = 100]

x0 = 10000;
t[x_, x0_] := x0 / (x0 - x)

newx[x_, y_, z_] := 0

newy[x_, y_, z_] := t[x, x0] y

newz[x_, y_, z_] := t[x, x0] z
```

```
domain =
 Table[{Random[Real, {0, 100}], Random[Real,
   Random[Real, {-100, 100}]}, {i, 1, numbero

image =
 Table[{newx[domain[[i, 1]], domain[[i, 2]], (
   newy[domain[[i, 1]], domain[[i, 2]], domai
   newz[domain[[i, 1]], domain[[i, 2]], domai
  {i, 1, Length[domain]}];

Clear[list]

list = Table[{domain[[i, j]], image[[i, j]]},
  {j, 1, 3}];

PrependTo[list, {"x coordinates", "y coordina
  "z coordinates"}];

Print["The first ten domain and image points
 list // TableForm ]
```

The number of domain and image points is
100

The first ten domain and image points are:

x coordinates	y coordinates	z coor
86.253	10.9948	66.168
0	11.0905	66.743
60.5617	-27.3693	56.678
0	-27.536	57.024
76.2735	64.5636	-5.351
0	65.0598	-5.393
32.2878	81.7829	70.909
0	82.0478	71.139
18.5145	-14.0704	18.652
0	-14.0965	18.686
59.1015	-54.4335	-55.17
0	-54.7572	-55.50
79.3736	53.2518	-1.095
0	53.6778	-1.103
26.3817	83.4809	49.403
0	83.7018	49.534
63.1995	41.7686	-82.68
0	42.0343	-83.21
14.1401	-46.2318	85.089
0	-46.2972	85.210

Are your image values for y and z close to your domain values for y and z respectively?

Part II: Parallel Mapping to Any Plane Using a Linear Transformation - Identifying the Image

We now consider another type of linear mapping onto a plane. Instead of projections from a single point, we consider parallel projections onto a plane. You could think of this as a generalization of the above case, where the beholder moves farther and farther away ($x_0 \to \infty$). Your domain is the set of all points in three-dimensional space. Begin by finding the image for several points in your domain, and use any three of these distinct noncollinear points in your image to write the equation of the plane in which they lie. Verify that other image points you found satisfy the equation of this plane, and then plot the image to verify visually that the image is a plane.

The transformation described below is defined as follows:
newx = x + y + 2z newy = 2x + y - z newz = 3x + 2y + z
Other transformations will work, provided that exactly one of the right-hand functions can be written as a linear combination of the other two. In this case, the first right-hand side function equals the third one minus the second one. This restriction guarantees that all your image points will lie in a plane.
Without loss of generality, we focus on domain points with integer coordinates between -100 and 100. We use integers to avoid making roundoff errors when we check to see that image points lie on the plane. One-thousand image points are computed, and all are plotted, but only the first 10 are listed.

In[42]:=

```
Clear[x, y, z, newx, newy, newz, domain, image]

newx[x_, y_, z_] := x + y + 2 z

newy[x_, y_, z_] := 2 x + y - z

newz[x_, y_, z_] := 3 x + 2 y + z

Print["The number of domain and image points
 numberofpoints = 1000]
```

```
domain =
 Table[{Random[Integer, {-100, 100}],
   Random[Integer, {-100, 100}], Random[Inte
  {i, 1, numberofpoints}];

image =
 Table[{newx[domain[[i, 1]], domain[[i, 2]], (
   newy[domain[[i, 1]], domain[[i, 2]], domai
   newz[domain[[i, 1]], domain[[i, 2]], domai
  {i, 1, Length[domain]}];

Clear[partimage]

partimage = Table[image[[i]], {i, 1, 10}];

PrependTo[partimage,
 {"x coordinates", "y coordinates", "z coordi

Print["The first ten image points are: "]

Print[partimage // TableForm]


The number of domain and image points is
 1000


The first ten image points are:
```

x coordinates	y coordinates	z coordinates
-113	-121	-234
-75	175	100
-217	111	-106
11	75	86
-38	10	-28
-105	-8	-113
-115	-12	-127
-8	67	59
102	88	190
-248	-136	-384

To determine the equation of the plane in which the first three image points lie, we specify two vectors in the plane and compute their cross product. Knowing that one point in the image

is {0, 0, 0} guarantees that the plane passes through the origin.

In[54]:=

```
vector1 = image[[1]] - image[[2]];
vector2 = image[[1]] - image[[3]];
normal = Cross[vector1, vector2];
Print["The normal to the plane is ",
  normaltoplane = normal / normal[[1]]]
Clear[x, y, z]
Print["The equation of the plane onto which
    points are projected is ",
  equationofplane[x_, y_, z_] =
   normaltoplane[[1]] x + normaltoplane[[2]] y
     normaltoplane[[3]] z == 0]
```

```
The normal to the plane is {1, 1, -1}
```

```
The equation of the plane onto which the
   points are projected is x + y - z == 0
```

Check to see if all points you found in the domain satisfy the equation of the plane. We start our iterator at 0 and then increase it by one each time an image point satisfies the equation of the plane.

In[60]:=

```
works = 0;
Do[If[equationofplane[image[[i, 1]], image[[:
    image[[i, 3]]], works = works + 1], {i, 1, I
works
```

Out[62]=

```
1000
```

It looks as though, as predicted, every point in the image lies on the specified plane.

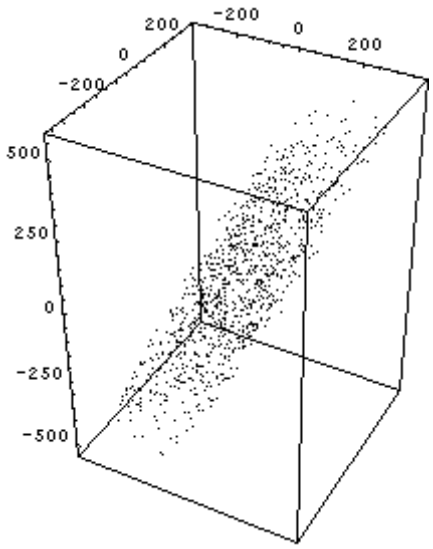
To view the points in the image, first load a graphics package if you have not done so previously.

In[63]:=

```
<< Graphics`Graphics3D`
```

In[64]:=

```
sp = ScatterPlot3D[image, PlotStyle -> PointSi
```



You Try It: Part II

Try the transformation described below:

newx = 2x + 2y + z newy = 4x + 4y - 7z newz = 3x + 3y + 9z

Change the expressions in red.

In[65]:=

```
Clear[x, y, z, newx, newy, newz, domain, image]

newx[x_, y_, z_] := x + y + 2 z

newy[x_, y_, z_] := 2 x + y - z

newz[x_, y_, z_] := 3 x + 2 y + z

Print["The number of domain and image points
 numberofpoints = 1000]
```

```
domain =
 Table[{Random[Integer, {-100, 100}],
   Random[Integer, {-100, 100}], Random[Inte
  {i, 1, numberofpoints}];

image =
 Table[{newx[domain[[i, 1]], domain[[i, 2]],
   newy[domain[[i, 1]], domain[[i, 2]], domai
   newz[domain[[i, 1]], domain[[i, 2]], domai
  {i, 1, Length[domain]}];

Clear[partimage]

partimage = Table[image[[i]], {i, 1, 10}];

PrependTo[partimage,
 {"x coordinates", "y coordinates", "z coordi

Print["The first ten image points are: "]

Print[partimage // TableForm]
```

```
The number of domain and image points is
 1000

The first ten image points are:
```

x coordinates	y coordinates	z coordinates
104	-119	-15
20	34	54
163	254	417
301	104	405
-86	-11	-97
-70	119	49
17	-122	-105
-150	120	-30
64	-173	-109
185	-12	173

Find the plane in which the image points lie.

In[77]:=

```
vector1 = image[[1]] - image[[2]];
vector2 = image[[1]] - image[[3]];
normal = Cross[vector1, vector2];
Print["The normal to the plane is ",
  normaltoplane = normal / normal[[1]]]
Clear[x, y, z]
Print["The equation of the plane onto which
    points are projected is ",
  equationofplane[x_, y_, z_] =
   normaltoplane[[1]] x + normaltoplane[[2]] y
     normaltoplane[[3]] z == 0]
```

```
The normal to the plane is {1, 1, -1}
```

```
The equation of the plane onto which the
   points are projected is x + y - z == 0
```

Check to see if all points you found in the domain satisfy the equation of the plane. Start your iterator at 0 and then increase it by one each time an image point satisfies the equation of the plane.

In[83]:=

```
works = 0;
Do[If[equationofplane[image[[i, 1]], image[[
    image[[i, 3]]], works = works + 1], {i, 1, I
works
```

Out[85]=

```
1000
```

Does every point in the image lie on the plane specified?

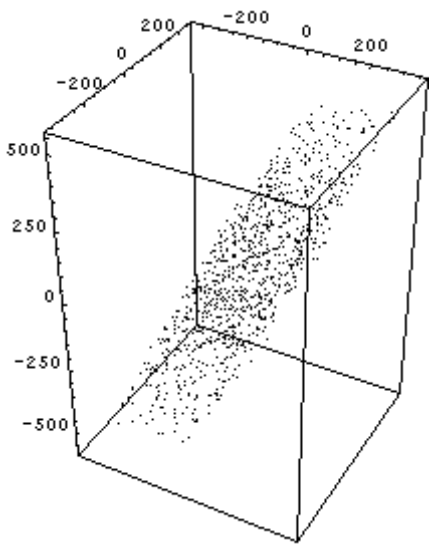
To view the points in the image, load a graphics package if you have not done so previously.

In[86]:=

```
<< Graphics`Graphics3D`
```

In[87]:=

```
sp = ScatterPlot3D[image, PlotStyle -> PointSi
```



Do your points look as though they lie on a plane? If not, can you see where there is a problem?

Part III: Analyzing the Mapping

We return to the first mapping defined in Part II. Identify the points in the domain that map to the origin O{0, 0, 0}, and verify that they all lie along the same line. Write the equations for this line.

In[88]:=

```
Off[Solve::svars]

Clear[x, y, z, newx, newy, newz, domain, image]

newx[x_, y_, z_] := x + y + 2 z

newy[x_, y_, z_] := 2 x + y - z

newz[x_, y_, z_] := 3 x + 2 y + z

mapto0 =
 Solve[{newx[x, y, z] == 0, newy[x, y, z] == 0,
  {x, y, z}]
```

Out[93]=

```
{{x → 3 z, y → -5 z}}
```

Now select another point in the image, say P{10,-20,-10}, and find the domain points that map to P. Note that they all fall along a line, and observe how this line is related to the line of points that map to the origin.

In[94]:=

```
maptoP =
 Solve[{newx[x, y, z] == 10, newy[x, y, z] == -
   newz[x, y, z] == -10}, {x, y, z}]
```

Out[94]=

```
{{x → -30 + 3 z, y → 40 - 5 z}}
```

Check this out for another point in the image Q(-40,70,30) and see if you can make a generalization about the mapping in terms of sets of points in the domain that map to a given point in the image.

In[95]:=

```
maptoQ =
 Solve[{newx[x, y, z] == -40, newy[x, y, z] ==
   newz[x, y, z] == 30}, {x, y, z}]
```

Out[95]=

```
{{x → 110 + 3 z, y → -150 - 5 z}}
```

Part IV: Application to Medical X-Ray and CAT-Scan Technologies

We will now demonstrate how points within a tumor would show on an X-ray slide.

In[96]:=

```
Clear[x, y, z, newx, newy, newz, domain, image,
 listout, pin, pout]

newx[x_, y_, z_] := x + y + 2 z

newy[x_, y_, z_] := 2 x + y - z
```

```
newz[x_, y_, z_] := 3 x + 2 y + z

Print["The number of domain and image points
 numberofpoints = 1000]

domain =
 Table[{Random[Integer, {-100, 100}],
   Random[Integer, {-100, 100}], Random[Inte
  {i, 1, numberofpoints}];

image =
 Table[{newx[domain[[i, 1]], domain[[i, 2]], (
   newy[domain[[i, 1]], domain[[i, 2]], domai
   newz[domain[[i, 1]], domain[[i, 2]], domai
  {i, 1, Length[domain]}];

Clear[partimage]

partimage = Table[image[[i]], {i, 1, 10}];

PrependTo[partimage,
 {"x coordinates", "y coordinates", "z coordi

Print["The first ten image points are: "]

Print[partimage // TableForm]


The number of domain and image points is
 1000


The first ten image points are:
```

x coordinates	y coordinates	z coordinates
-100	-151	-251
10	22	32
172	-20	152
23	156	179
141	166	307
228	131	359
-52	104	52
-41	245	204

-41	-73	-114
218	-111	107

Let's suppose that there is a tumor in the region bounded approximately by z = 100 -0.5 $\sqrt{x^2 + y^2}$ and z = .005($x^2 + y^2$). We will first plot the object in three-space and then separate out the image points that have been mapped to that region and plot them in green. We must first load two graphing packages if not already loaded.

In[108]:=

```
<<Graphics`ParametricPlot3D`
```

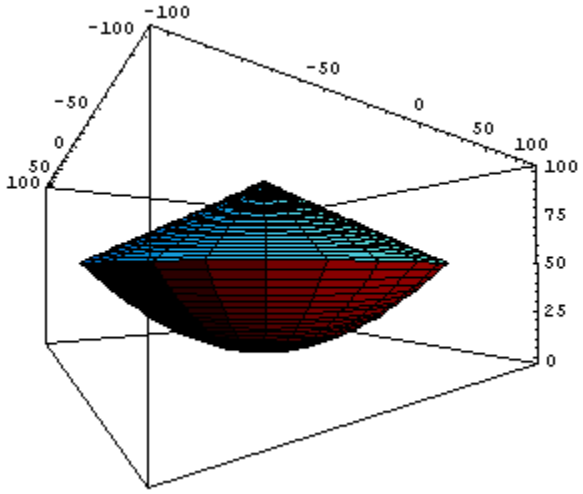
In[109]:=

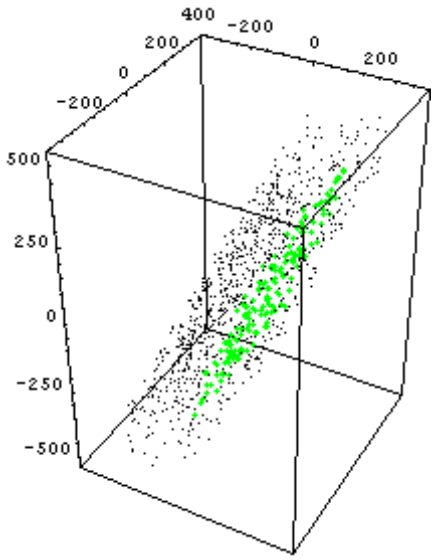
```
<<Graphics`Graphics3D`
```

In[110]:=

```
Clear[r]

p1 = CylindricalPlot3D[100 - .5 r, {r, 0, 100},
  DisplayFunction → Identity];

p2 = CylindricalPlot3D[.005 r^2, {r, 0, 100}, {θ
  DisplayFunction → Identity];

Show[p1, p2, DisplayFunction → $DisplayFuncti
 ViewPoint -> {-0.637, -0.917, -0.018}];

listout = {};

listin = {};

Do[
 If[
  domain[[i, 3]] <
    100 - .5 Sqrt[(domain[[i, 1]])^2 + (domain[[:
   domain[[i, 3]] > .005 ((domain[[i, 1]])^2 + (
  AppendTo[listin, {image[[i, 1]], image[[i, :
    image[[i, 3]]}],
  AppendTo[listout, {image[[i, 1]], image[[i,
    image[[i, 3]]}]], {i, 1, Length[domain]}]

pout = ScatterPlot3D[listout, DisplayFunction
```

```
pin = ScatterPlot3D[listin,
  PlotStyle -> {RGBColor[0, 1, 0], PointSize[.
  DisplayFunction -> Identity];

Show[pin, pout, DisplayFunction -> $DisplayFu
```





Suppose, for example, that a spot is found at the image point T1 (10,20,30). What does that tell you about the location of the possible growth? Based on this image, you cannot precisely identify the placement of the tumor, because this mapping is NOT one-to-one.

In[120]:=

```
maptoT1 =
 Solve[{newx[x, y, z] == 10, newy[x, y, z] == 20
   newz[x, y, z] == 30}, {x, y, z}]
```

Out[120]=

```
{{x → 10 + 3 z, y → -5 z}}
```

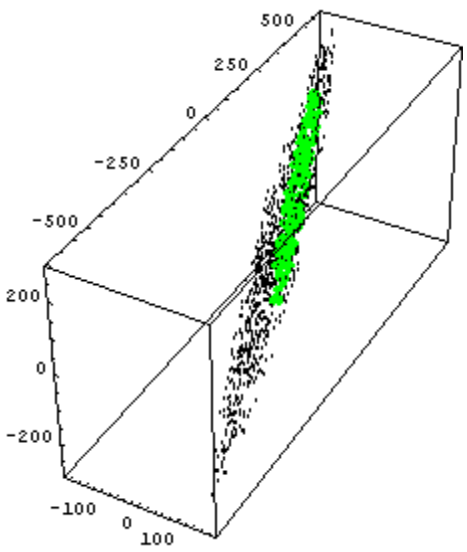
Since this result gives you some information, but not enough, you must take another X-ray from a different angle.
We will set up a mapping from a different perspective. In the example given below, we plot the new perspective and verify that the points all map to a plane.

In[121]:=

```
Clear[x, y, z, newx, newy, newz, domain, image,
 listout, pin, pout]

newx[x_, y_, z_] := -x + y

newy[x_, y_, z_] := 2 x - 3 y + 2 z

newz[x_, y_, z_] := -y + 2 z

Print["The number of domain and image points
 numberofpoints = 1000]

domain =
 Table[{Random[Integer, {-100, 100}],
   Random[Integer, {-100, 100}], Random[Inte
  {i, 1, numberofpoints}];

image =
 Table[{newx[domain[[i, 1]], domain[[i, 2]],
   newy[domain[[i, 1]], domain[[i, 2]], domai
   newz[domain[[i, 1]], domain[[i, 2]], domai
  {i, 1, Length[domain]}];

listout = {};

listin = {};
```

```
Do[
 If[
  domain[[i, 3]] <
    100 - .5 Sqrt[(domain[[i, 1]])^2 + (domain[[:
   domain[[i, 3]] > .005 ((domain[[i, 1]])^2 + (
  AppendTo[listin, {image[[i, 1]], image[[i, :
    image[[i, 3]]}],
  AppendTo[listout, {image[[i, 1]], image[[i,
    image[[i, 3]]}]], {i, 1, Length[domain]}]

pout = ScatterPlot3D[listout, DisplayFunction

pin = ScatterPlot3D[listin,
  PlotStyle -> {RGBColor[0, 1, 0], PointSize[.
  DisplayFunction -> Identity];

Show[pin, pout, DisplayFunction -> $DisplayFu
```

```
The number of domain and image points is
 1000
```



If you were somehow able to detect that the SAME spot now shows up on the image at T2(-90,250,70), would that tell you about the possible location of the tumor? As before, we can get information on the point in the domain that mapped to this image point.

In[134]:=

```
maptoT2 =
 Solve[{newx[x, y, z] == -90, newy[x, y, z] == 
   newz[x, y, z] == 70}, {x, y, z}]
```

Out[134]=

$$\{\{x \to 20 + 2\,z,\ y \to -70 + 2\,z\}\}$$

If we put this information together with the previous information, we can locate the point on the organ identified. Essentially, we are finding the intersection of two straight lines in three-dimensional space.

In[135]:=

```
var = Solve[{maptoT1[[1, 1, 2]] == maptoT2[[1,
   maptoT1[[1, 2, 2]] == maptoT2[[1, 2, 2]]}, 
```

Out[135]=

$$\{\{z \to 10\}\}$$

Let's put this result for z into the specifications for the first line we found.

In[136]:=

```
Print["x = ", (maptoT1[[1, 1, 2]] /. var)[[1]]

Print["y = ", (maptoT1[[1, 2, 2]] /. var)[[1]]

Print["z = ", var[[1, 1, 2]]]
```

```
x = 40

y = -50

z = 10
```

This specifies one of the points in the three-dimensional object where the tumor is present.

You should notice in the analysis above how difficult it is to identify the point in the second mapping that is the image of the SAME point as the one we first found. Because of this problem, many more than simply two X-rays would need to be taken to determine the precise location of a tumor The technology that makes this process feasible is called a CT-Scan,

commonly referred to as a CAT-Scan. The procedure above reflects the fundamental mathematics used to interpret CAT-Scans.

Part V: Linear Algebra Approach (optional; not in calculus text)

"> About Mathematica

In[139]:=

```
Clear[m, eig]
m = {{1, 1, 2}, {2, 1, -1}, {3, 2, 1}};
MatrixForm[m]
eig = Eigensystem[m];
Print[
  "The eigenvalues and corresponding eigenve
    the given matrix are: ", N[eig]]
image = {};
Do[
  {new = m.{Random[Integer, {-100, 100}],
      Random[Integer, {-100, 100}],
      Random[Integer, {-100, 100}]}, AppendTo
  {1000}]
```

Out[141]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 2 \\ 2 & 1 & -1 \\ 3 & 2 & 1 \end{pmatrix}$$

```
The eigenvalues and corresponding
   eigenvectors for the given matrix are:
 {{3.79129, -0.791288, 0.},
  {{0.791288, 0.208712, 1.},
   {-3.79129, 4.79129, 1.}, {3., -5., 1.}}}
```

Note the presence of the 0 eigenvalue.

The kernel of a linear transformation is the set of elements in the domain that map to the 0 element (0, 0, 0) in the image.

In[146]:=

```
Clear[a, b, c]
```

```
kernel = {a, b, c};

solk = Solve[m.kernel == {0, 0, 0}, {a, b, c}]
```

Out[148]=

```
{{a → 3 c, b → -5 c}}
```

If c = 1, note how this vector compares to the eigenvector associated with the eigenvalue of 0.

In[149]:=

```
c = 1;

kernel /. solk
```

Out[150]=

```
{{3, -5, 1}}
```

You can find the normal to the plane formed by the image by taking the cross product of the eigenvectors associated with the nonzero eigenvalues. In this case, these are the first and second eigenvectors.

In[151]:=

```
norm = Cross[eig[[2, 1]], eig[[2, 2]]];

scalednorm = norm / norm[[1]] // Simplify
```

Out[152]=

```
{1, 1, -1}
```

Check to see if all the points in the image lie in the plane.

In[153]:=

```
works = 0;
Do[
  If[scalednorm[[1]] image[[i, 1]] +
     scalednorm[[2]] image[[i, 2]] +
     scalednorm[[3]] image[[i, 3]] == 0, works
  {i, 1, 1000}]
works
```

Out[155]=

```
1000
```

What do you suppose the existence of a 0 eigenvalue has to do with the fact that the image points all lie in a plane, even though the domain consists of points in three dimensions?

About *Mathematica*

If you have studied linear algebra, you can learn in this section some of the behind-the-scenes mathematics in the study of linear transformations and how *Mathematica* can help you with this analysis.
The command **Eigensystem[m]** in *Mathematica* gives a list consisting of two parts. The first part contains the eigenvalues of the matrix for **m** and the second part gives the set of eigenvectors for the eigenvalues in corresponding order.
Go back.

Created by *Mathematica* (April 2, 2005)