

Scientific Visualization in MATLAB

>> 1997

>> **matlab** conference

David Foti

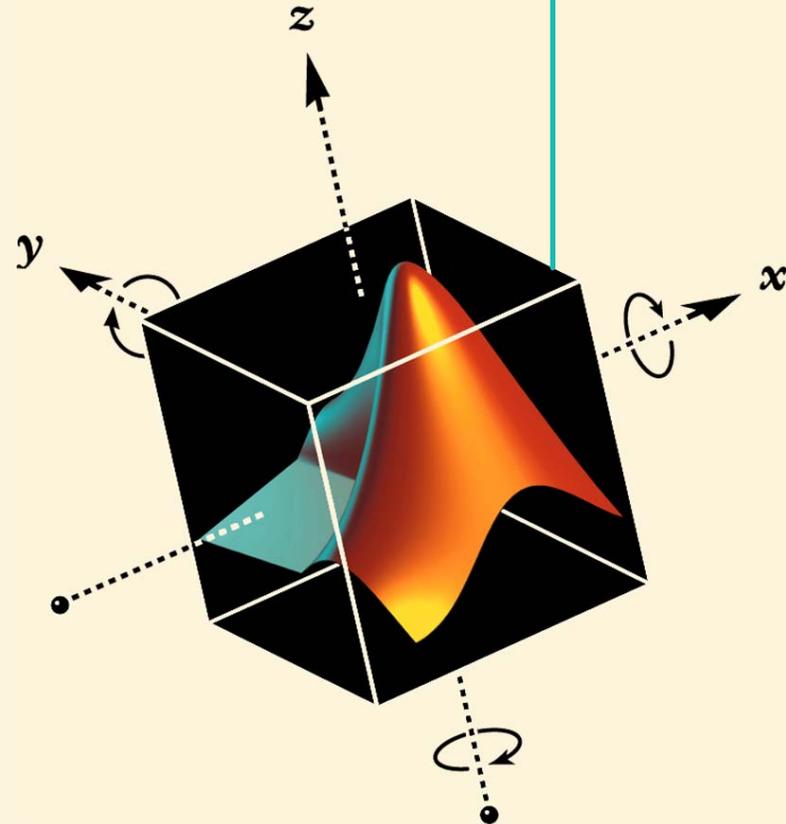
daf@mathworks.com

The MathWorks Inc.

24 Prime Park Way

Natick, MA 01760-1500

USA



Outline

- ◆ **What is Scientific Visualization?**
- ◆ **Classifying Data Sets**
 - ◆ Dimension of data - the number of independent variables
 - ◆ Number of dependent variables
- ◆ **Creating a Virtual World**
 - ◆ How to plot data
 - ◆ Making 3-dimensional surfaces
 - ◆ Modeling geometric solids
- ◆ **Lights, Camera, Action!**
 - ◆ Illuminating the virtual world
 - ◆ Viewing the world through a camera
 - ◆ Animation

What is Scientific Visualization?

The graphical representation of data

How is scientific visualization useful?

- ◆ finding patterns
- ◆ identifying trends
- ◆ comparing complex information
- ◆ examining objects that can not be examined physically

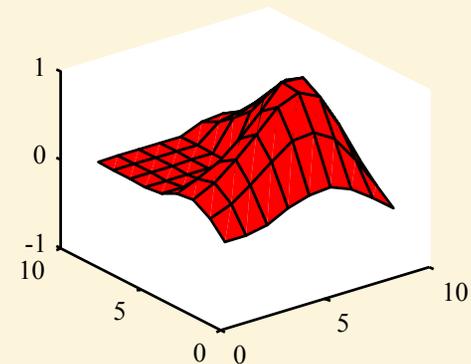
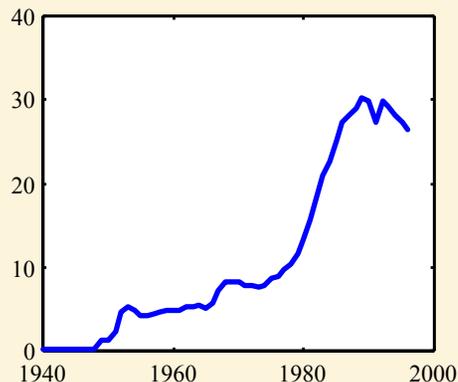
Classifying Data Sets

- ◆ **Determine the dimension of the data**
 - ◆ Count the number of independent variables
 - ◆ Dimension determines what type of plot to use
 - Use lines to plot 1-Dimensional data
 - Use surfaces to plot 2-Dimensional data

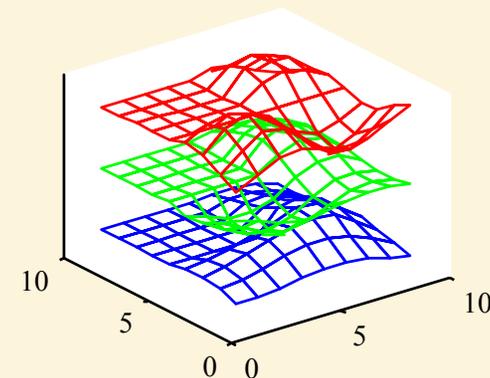
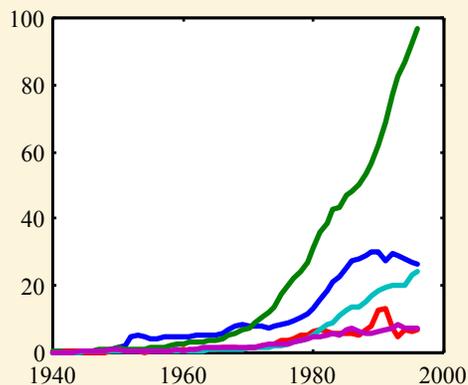
Classifying Data Sets

- ◆ Determine the number of dependent variables

- ◆ One dependent variable

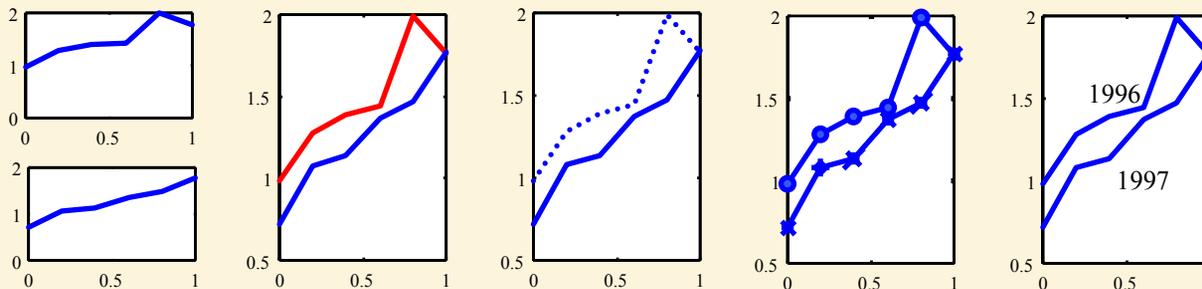


- ◆ Multiple dependent variables



Classifying Data Sets

- ◆ Ways to differentiate multiple variables
 - multiple plots
 - color
 - line style
 - marker style
 - annotation



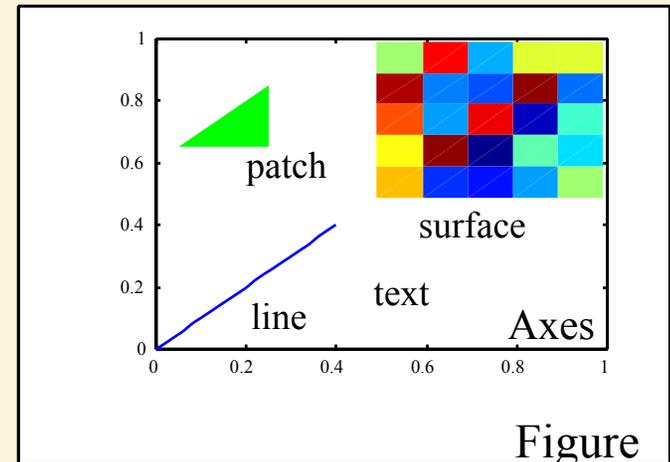
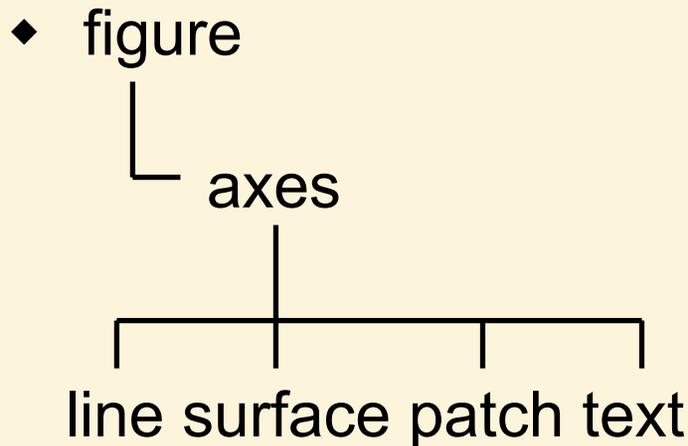
Classifying Data Sets

Key Points

- ◆ **The number of independent variables determines whether to plot with lines or surfaces**
- ◆ **Dependent variables can be distinguished by height, color, line style, marker style, etc.**

Creating a Virtual World

- ◆ MATLAB can be used to create a database of graphics primitives which store and display data.
- ◆ MATLAB's principal plotting primitives



How to Plot 1-Dimensional Data

- ◆ **MATLAB general plotting commands:**

- ◆ `plot`

- `plot(Y)` - plots the dependent variable(s) against matrix indices
- `plot(X,Y)` - plots the dependent variable(s) in Y against the independent variable X
- `plot(X,Y, '<style>')` - uses the specified color, line style, and marker style

- ◆ `loglog`, `semilogx`, `semilogy`

- ◆ `polar`

- ◆ use `help graph2d` for more information

Demonstration

```
>> budget1
```

How to Plot 1-Dimensional Data

- ◆ **Specialized plotting commands**
 - ◆ `bar`, `bar3` - make bar graphs
 - ◆ `pie`, `pie3` - make pie charts
 - ◆ `area` - produces a plot with filled areas under the curve(s)
 - ◆ `hist` - make a histogram
 - ◆ `stairs` - plots discrete values as steps
 - ◆ use `help specgraph` for more information and even more special plotting commands

Demonstration

```
>> budget2
```

```
>> budget3
```

```
>> budget4
```

Plotting multiple lines

- ◆ `plot(X,Y,'<style>', x2, Y2, '<style2>', ...)`
- ◆ `plot(Y1)`
`hold on`
`plot(Y2)`

Demonstration

```
>> phase1
```

Annotation

- ◆ The `title`, `xlabel`, and `ylabel` commands
`title('This is a title')`
`xlabel('Time in seconds')`
- ◆ The `legend` command
`legend('1994', '1995', '1996', '1997');`
- ◆ The `gtext` command allows you to add an annotation anywhere on a plot by clicking on a point in the plot with the mouse
`gtext('Transition point here')`

Creating a Virtual World

Exercises

Part 1: 1-Dimensional Data

Plotting 2-Dimensional Data

◆ Surfaces

- `surf(Z)` plots each value in the matrix **Z** as the height of a rectangular surface having the same dimensions as matrix **Z**
`surf(X, Y, Z)` uses the coordinate pairs defined by each (x,y) pair from the vectors X and Y to define the surface whose height values are contained in matrix **Z**.
- `mesh(Z)` plots surface defined by **Z** as a “wire” grid
`mesh(X, Y, Z)` is also available
- type `help graph3d` for more information

Demonstration

```
>> membrane1
```

Plotting 2-Dimensional Data

◆ Contour plots

- `contour(Z)` plots contour lines for the matrix **Z**
`contour(Z,n)` computes *n* contour lines
`[C, H] = contour(Z,n)` draws a contour plot and returns the contour matrix **C** and vector of line object handles **H**
- `contourf(Z)` fills in the areas between contour lines with the color of the higher contour line
- `clabel(C, H)` adds height labels to a contour plot

Demonstration

```
>> eltemp1
```

Modeling Geometric Solids

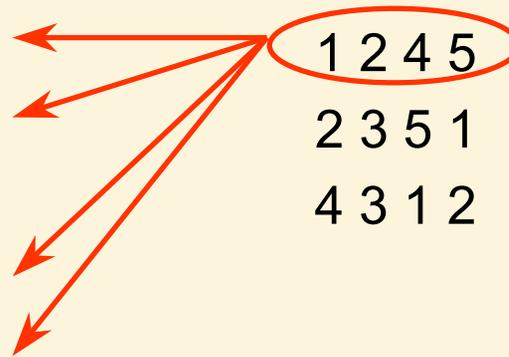
- ◆ A geometric solid is turned into a list of polygons which approximate the original object
- ◆ Polygon lists can be displayed in MATLAB using the `patch` command
`h = patch('faces',F, 'vertices',v)`

Vertices

0 0 0 (vertex #1)
3 0 0 (vertex #2)
3 2 0 (vertex #3)
0 0 3 (vertex #4)
5 4 5 (vertex #5)

Faces

1 2 4 5
2 3 5 1
4 3 1 2



Demonstration

```
>> engine
```

Creating a Virtual World

Key Points

- ◆ **plot command is used to plot 1-Dimensional data with one or more dependent variables**
- ◆ **specialized commands exist to produce a variety of 1-Dimensional graphs**
- ◆ **surface and contour commands can plot 2-Dimensional data**
- ◆ **The patch command can be used for modeling geometric solids**
- ◆ **By changing property values one can manipulate the objects created with the above commands**

Creating a Virtual World

Exercises

Part 2: 2-Dimensional Data

Lights!

- ◆ **Lighting models**
 - ◆ **none** - no lighting
 - ◆ **flat** - fastest, faceted appearance
 - ◆ **gouraud** - fast, low quality smooth appearance
 - ◆ **phong** - slow, high quality smooth appearance



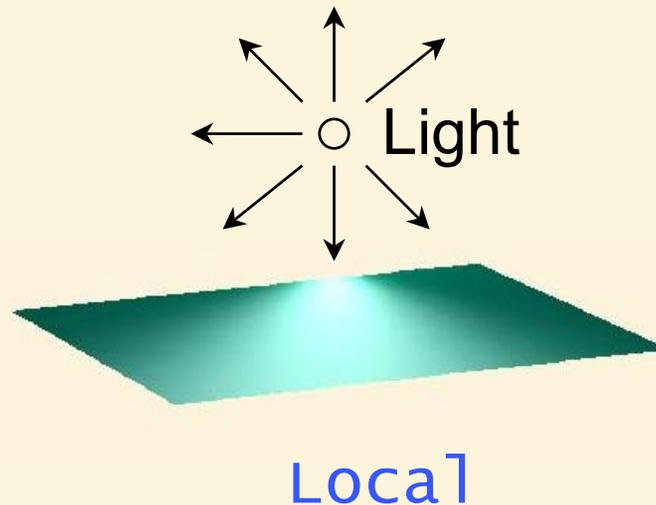
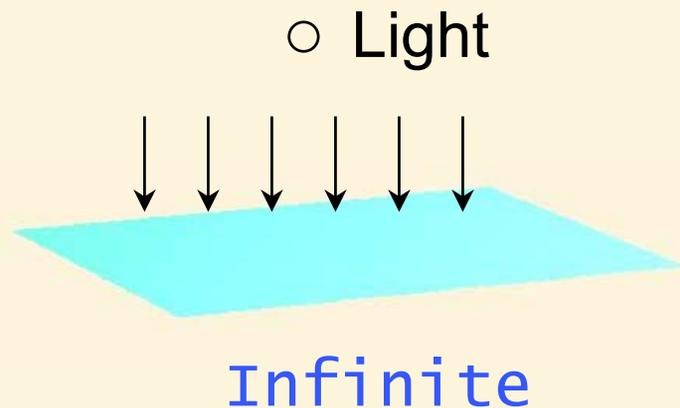
Demonstration

```
>> eltemp2
```

Lights!

- ◆ **Light Properties**

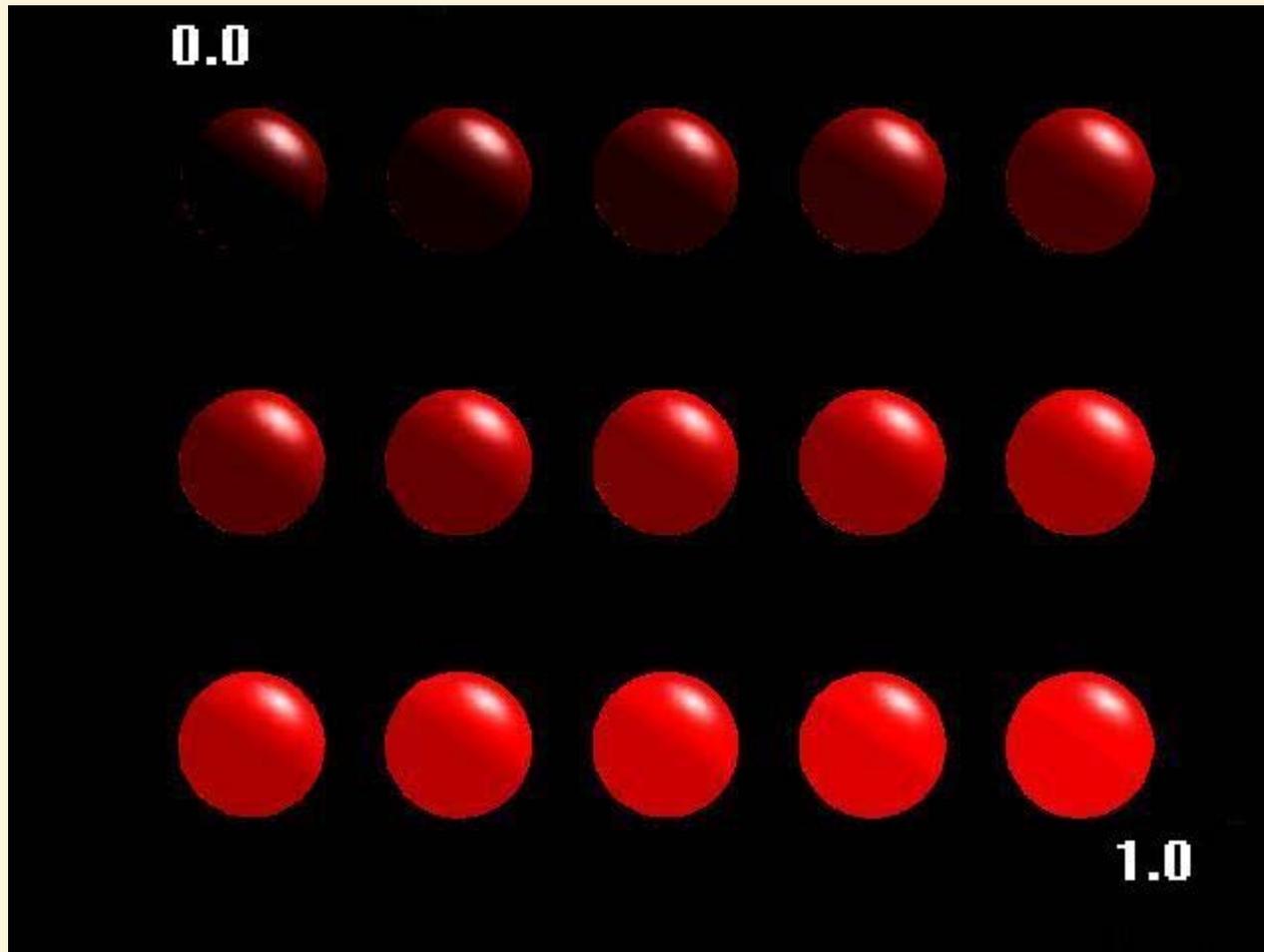
- ◆ color
- ◆ position
- ◆ style - 'infinite' or 'local'



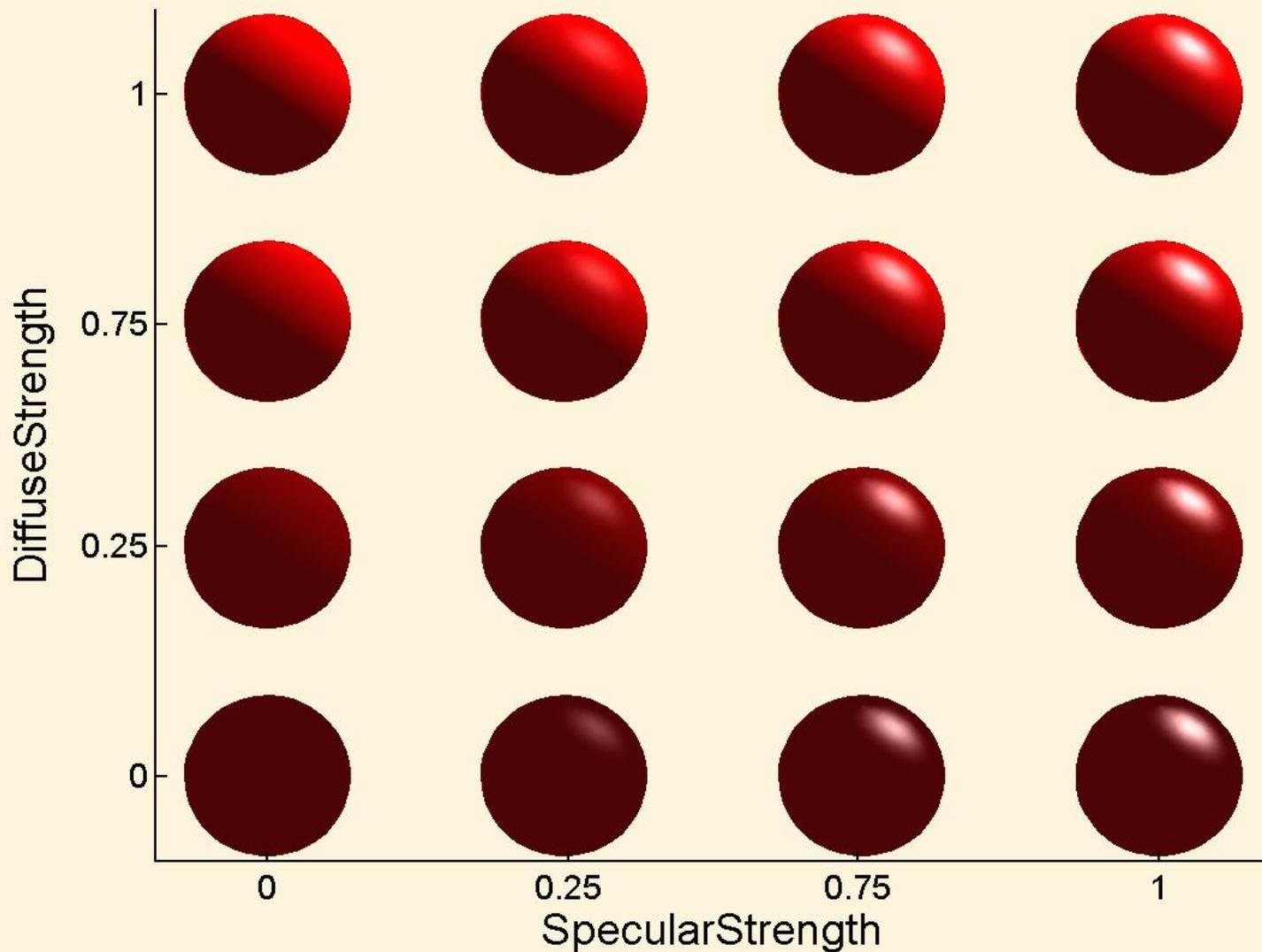
Lights!

- ◆ **Material properties (of surface and patch objects)**
 - ◆ AmbientStrength
 - ◆ DiffuseStrength
 - ◆ SpecularStrength
 - ◆ SpecularExponent
 - ◆ SpecularColorReflectance

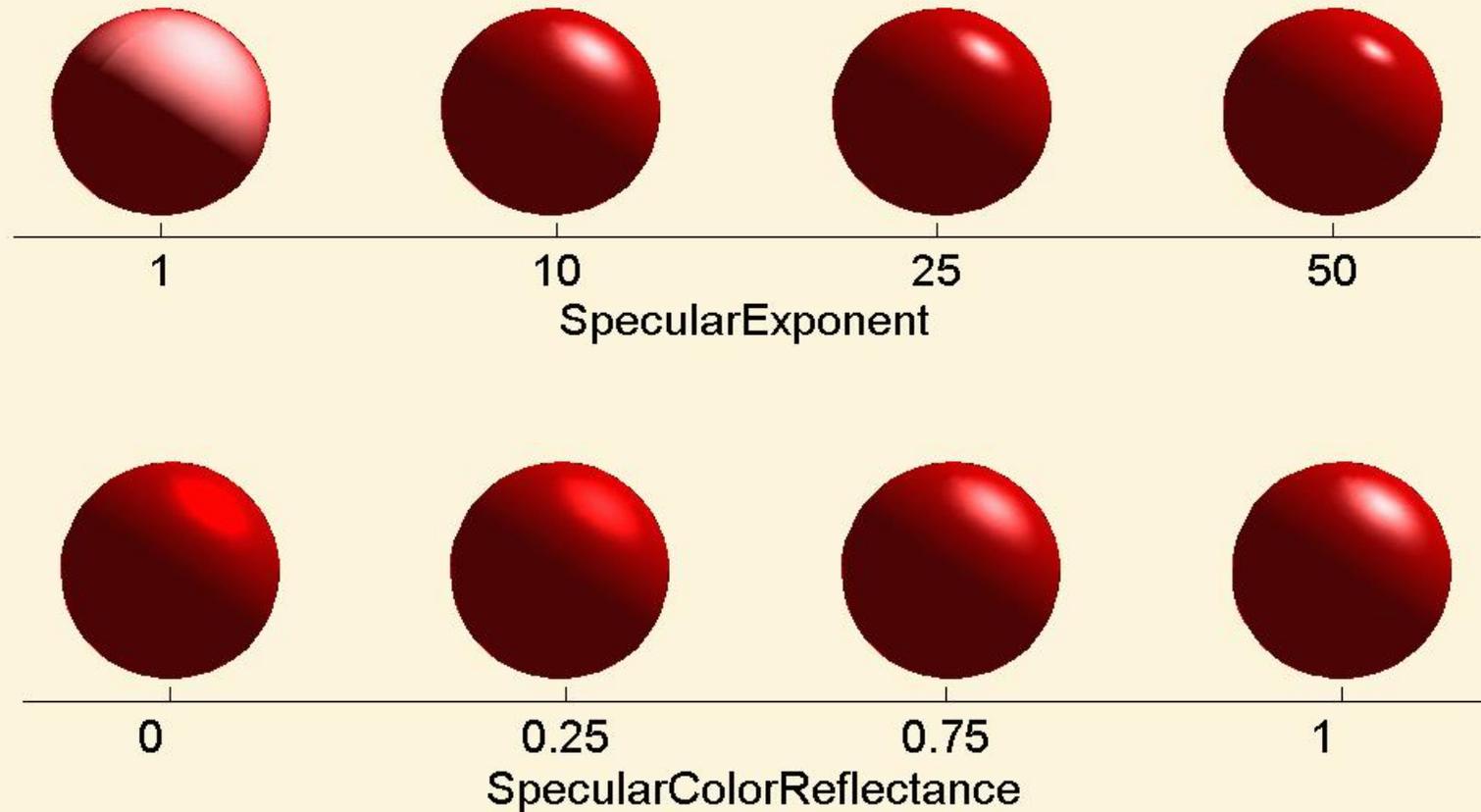
Ambient Strength



Diffuse and Specular Light



Specular Exponent and Color Reflectance



Demonstration

```
>> engine2
```

Camera!

- ◆ **Controlling the point of view**
 - ◆ the `view` command
 - ◆ the `rotate3d` command

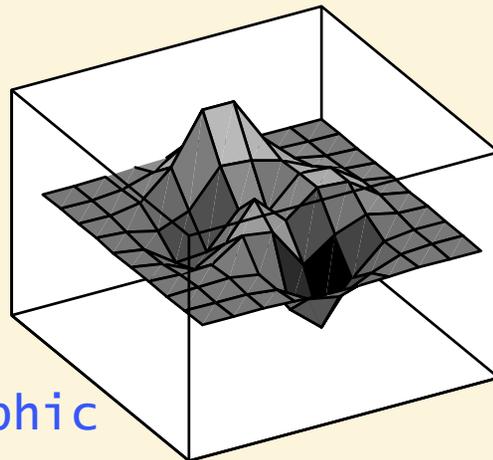
Demonstration

```
>> membrane2
```

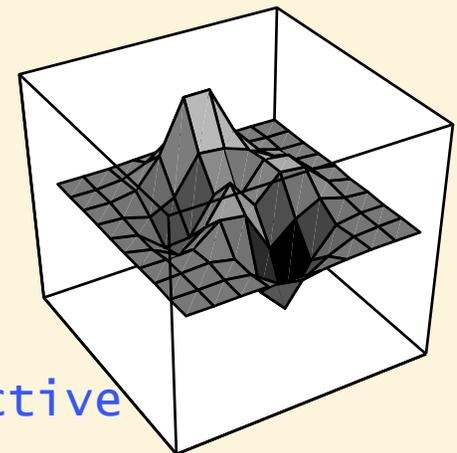
Camera!

◆ Projection

- ◆ orthographic -
`set(gca, 'projection', 'orthographic')`
- ◆ perspective -
`set(gca, 'projection', 'perspective')`



Orthographic



Perspective

Camera Controls

◆ Camera Properties of Axes

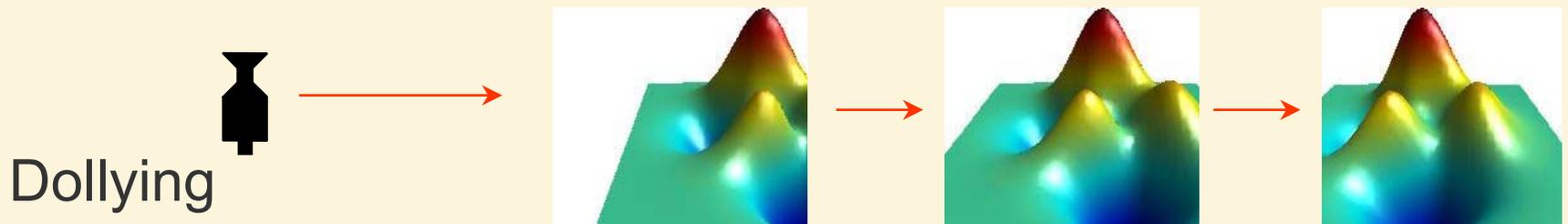
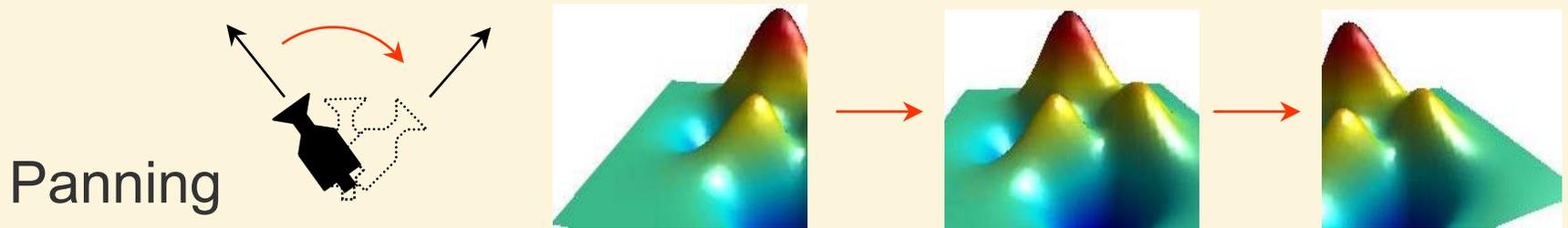
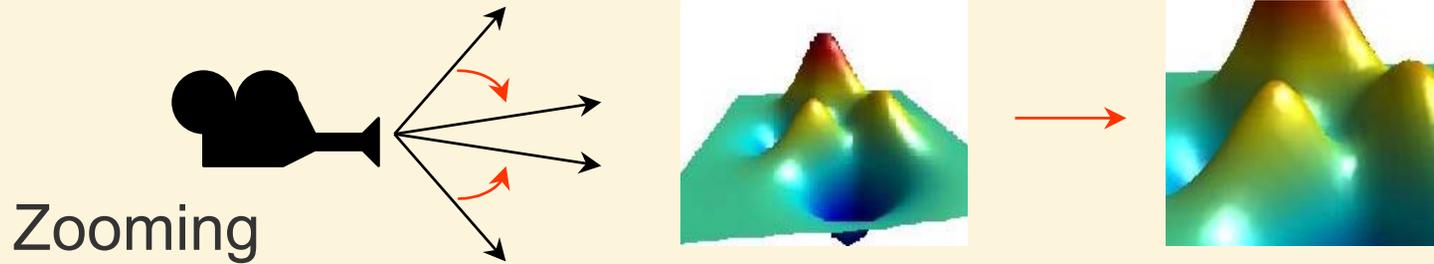
- `CameraPosition`
- `CameraTarget`
- `CameraViewAngle`
- `CameraUpVector`

example: `set(gca, 'CameraUpVector', [1 1 1])`

◆ Operations on the Camera

- zooming - change the `CameraViewAngle` property
- panning - change the `CameraTarget`
- dollying - move the `CameraPosition` and `CameraTarget` along two parallel lines

Diagram of Camera Operations



Action!

Animating Plots Using the Time Dimension

- ◆ **MATLAB's database of objects permits individual properties to change without affecting other properties of an object**
- ◆ **It's faster to just change the data of one object than to create a whole new plot**
- ◆ **Many times only the data is changing, but other attributes like color or linestyle may not**
- ◆ **Data properties of lines and surfaces**
 - ◆ XData
 - ◆ YData
 - ◆ ZData

Demonstration

```
>> sinewave1
```

Lights, Camera, Action!

Key Points

- ◆ **Lighting is used to provide depth cues allowing color and height to represent different dependent variables**
- ◆ **The axes object's camera properties can be used to view a scene from different locations and angles**
- ◆ **Once created, changes over time can be animated by modifying the values of an object's data properties**

Lights, Camera, Action!

Exercises

Summary

- ◆ **The number of independent and dependent variables determine how best to visualize data**
- ◆ **MATLAB has many ways to differentiate variables including color, line-style, and marker-style**
- ◆ **MATLAB can be used to plot lines and surfaces or to model geometric solids**
- ◆ **MATLAB plots are created from an interactive database representing a virtual world that can be explored and manipulated**

References

◆ From within MATLAB

- helpdesk
- help graphics

◆ Books

- *Using MATLAB Graphics* (comes with MATLAB 5).
- *Graphics and GUIs with MATLAB*, Patrick Marchand, CRC Press, 1996.
- *Numerical Analysis and Graphic Visualization with MATLAB*, Shoichiro Nakamura, Prentice Hall, 1996.
- *The Visual Display of Quantitative Information*, Edward R. Tufte, Graphics Press, 1983.
- *Visual Explanations*, Edward R. Tufte, Graphics Press, 1997.

◆ www.mathworks.com