

Scientific Visualization

Lecture 7: Other Visualization software

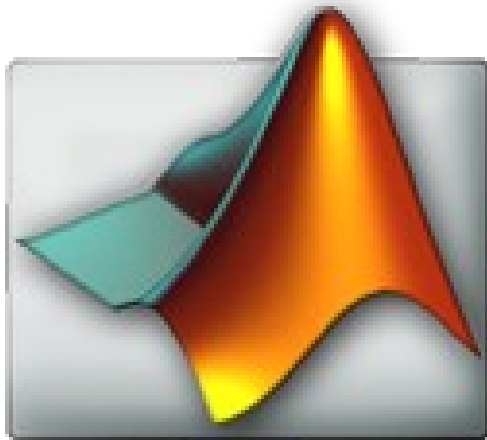
Patrik Malm

Centre for Image Analysis
Swedish University of Agricultural Sciences
Uppsala University



Today's lecture

- Ways to create visualizations using
 - Matlab
 - Blender



Opposite ends of the scale

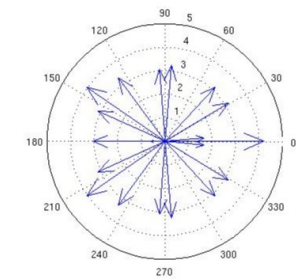
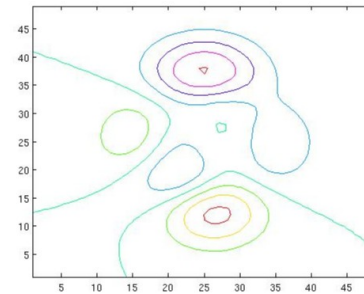
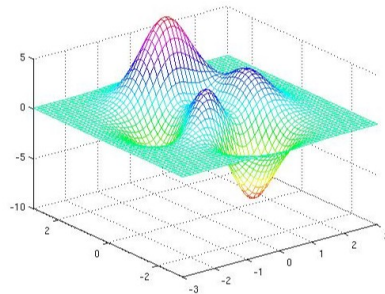
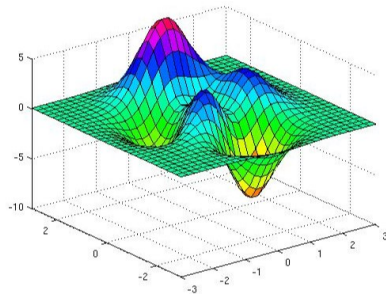
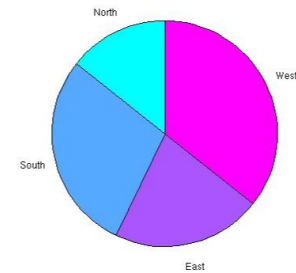
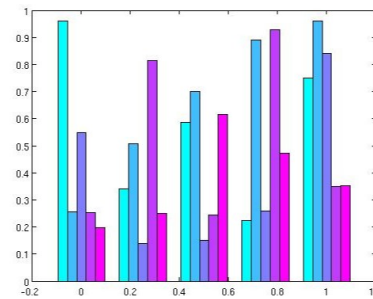
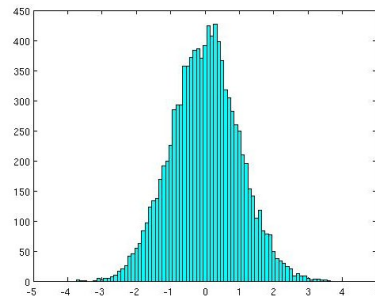
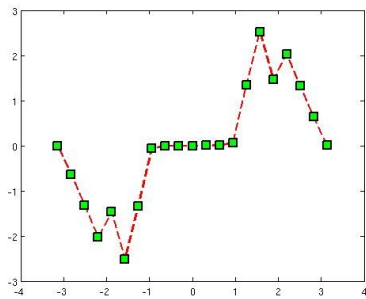
- Matlab
 - Scientific tool
 - Good for visualizing distributions of data
 - Expensive
- Blender
 - Open source 3D-modeling software
 - Free
 - Harder (but NOT impossible) to do "exact" scientific visualizations

Matlab

- Developed by The MathWorks
- First appeared in the late 1970s
- Short for "Matrix Laboratory"
- All variables are matrices
- Several modules, e.g., "Image processing toolbox"
- Cross-platform
- Can be slow or fast depending on application type

Matlab – Visualizations

- Plot options



And many, many more...

Matlab – Visualizations

- Available plot-types depend on which toolboxes are installed.
- Huge amount of user written functions at Mathwork file exchange
- Easy to modify and/or create own visualization functions

Matlab – Import and Export

- Import and export of 'special' data very much dependent on user scripts
- Example: writeVTK.m

```
function [] = writeVTK(vol,vtkfile)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Usage: writeVTK(vol,vtkfile)
%
%   vol:      The 3D matrix to be saved to file
%   vtkfile:  The output filename (string)
%
% Erik Vidholm 2005
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Matlab – Import and Export

```
% dimensions
volinfo = whos('vol');

sz = volinfo.size;
X = sz(1); Y = sz(2); Z = 1;
if( length(sz) == 3 )
    Z = sz(3);
end

% open file
fid = fopen(vtkfile,'w');

% write header
fprintf(fid, '%s\n', '# vtk DataFile Version 3.0');
fprintf(fid, '%s\n', 'created by writeVTK (Matlab implementation by Erik Vidholm)');
fprintf(fid, '%s\n', 'BINARY');
fprintf(fid, '%s\n', 'DATASET STRUCTURED_POINTS');
fprintf(fid, '%s%d%c%d%c%d\n', 'DIMENSIONS ', X, ' ', Y, ' ', Z);
fprintf(fid, '%s%f%c%f%c%f\n', 'ORIGIN ', 0.0, ' ', 0.0, ' ', 0.0);
fprintf(fid, '%s%f%c%f%c%f\n', 'SPACING ', 1.0, ' ', 1.0, ' ', 1.0);
fprintf(fid, '%s%d\n', 'POINT_DATA ', X*Y*Z);

tp = volinfo.class;
if( strcmp(tp, 'uint8') > 0 )
    fprintf(fid, '%s\n', 'SCALARS image_data unsigned_char');
elseif( strcmp(tp, 'uint16') > 0 )
    fprintf(fid, '%s\n', 'SCALARS image_data unsigned_short');
elseif( strcmp(tp, 'uint32') > 0 )
    fprintf(fid, '%s\n', 'SCALARS image_data unsigned_int');
elseif( strcmp(tp, 'single') > 0 )
    fprintf(fid, '%s\n', 'SCALARS image_data float');
elseif( strcmp(tp, 'double') > 0 )
    fprintf(fid, '%s\n', 'SCALARS image_data double');
end

fprintf(fid, '%s\n', 'LOOKUP_TABLE default');

% write data as binary
fwrite(fid,vol,tp);

% close file
fclose(fid);
```


Matlab - Links

- www.mathworks.com
- www.mathworks.com/matlabcentral/fileexchange

Blender

- Free 3D modeling and animation software
- Features include
 - Physics and Particles
 - Liquid and softbody simulations
 - Game creation
 - Python support for scripting
 - Video sequencer
- Available for all common OS (installation file sized around 10-20 Mb)



Blender – History

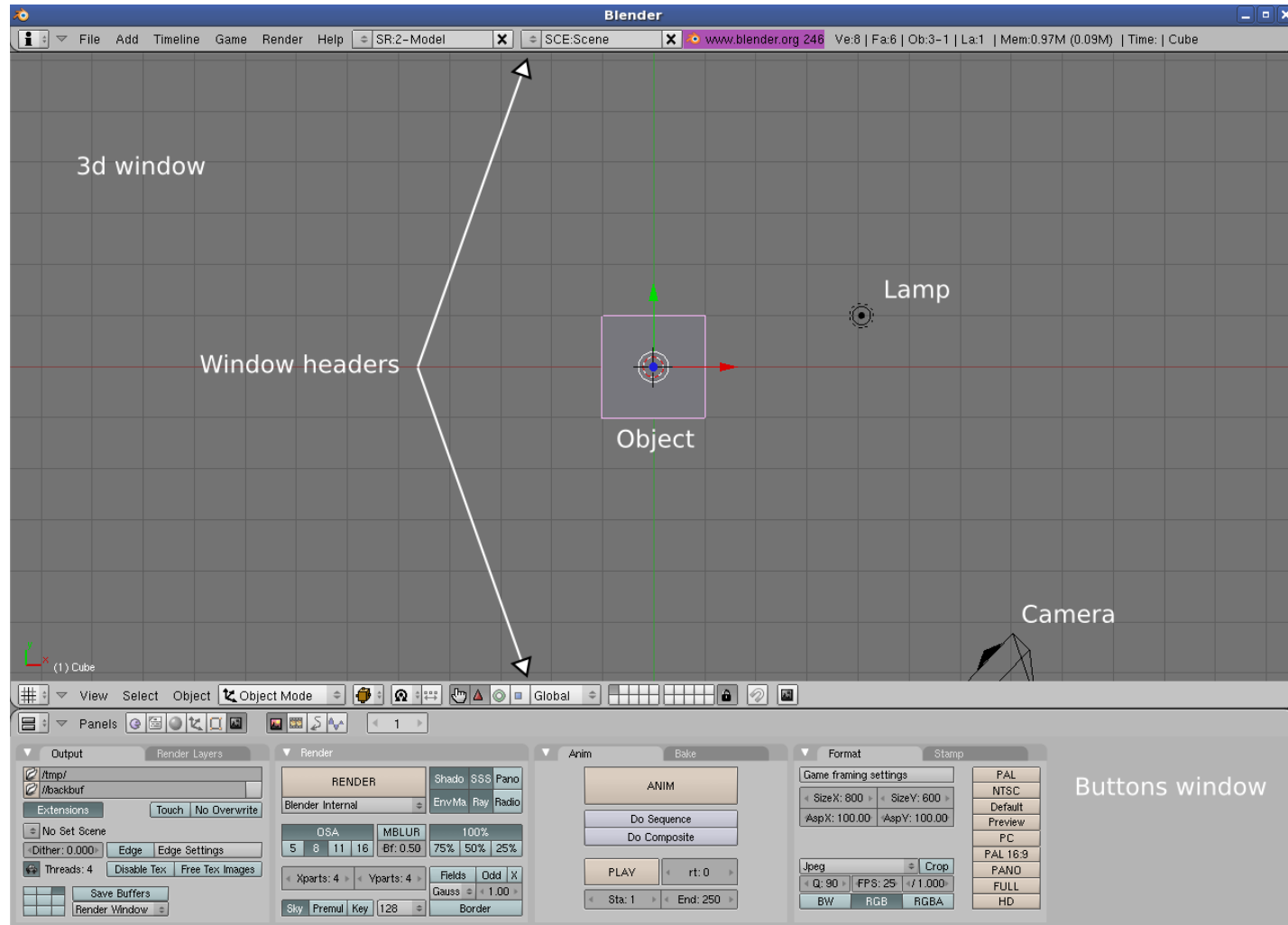
- Initially developed as an in-house application by the Dutch animation studio NeoGeo and Not a Number Technologies (NaN) during the late 90's
- NaN bankrupt 2002 (Blender v 2.25)
- Released under the GNU GPL for a one time payment of 100,000 Euros
- Blender is now managed by the Blender Foundation
- Version 2.5 is soon to be released



Blender

- Strengths:
 - Free (!!!)
 - Powerful
 - Interface for python scripting
 - Large user community -> tons of online material
- Weaknesses:
 - Confusing user interface (Under major reconstruction)
 - Sometimes hard to find good documentation

Blender – User interface



Blender – Getting around

- 'Middle mouse button' (MMB) – Rotate view
- 'Shift' + 'MMB' – Pan view
- 'Scroll wheel' – Zoom
- 'Ctrl' + 'MMB' – Zoom
- 'Numpad 1, 3, 7' – Front, right and top view
- 'Left mouse button' (LMB) – Move 3D cursor

Blender – Object selection

- Right mouse button (RMB) – Selects objects
- 'Shift' + RMB – Select multiple objects, deselect objects
- 'A' – Select all, clear selection
- 'B' – Box select
- 'B' + 'B' – Brush select (edit mode only)



Blender – Edit mode vs. Object mode

- 'TAB' key used to switch between modes for selected object
- In 'Object mode' the user can perform actions that affect the entire object (translation, scale, rotation,...)
- In 'Edit mode' local changes can be made by manipulating the vertices that the object consists of

Blender – Object manipulation

- 'G' – Grab (Move objects)
- 'R' – Rotate
- 'S' – Scale
- 'X' – Delete
- 'E' – Extrude (Edit mode)
- 'Z' – Shaded/Unshaded view
- 'Shift' + 'D' – Duplicate
- 'Ctrl' + 'Z' – Undo

Golden Rule
Always keep one hand on the keyboard!

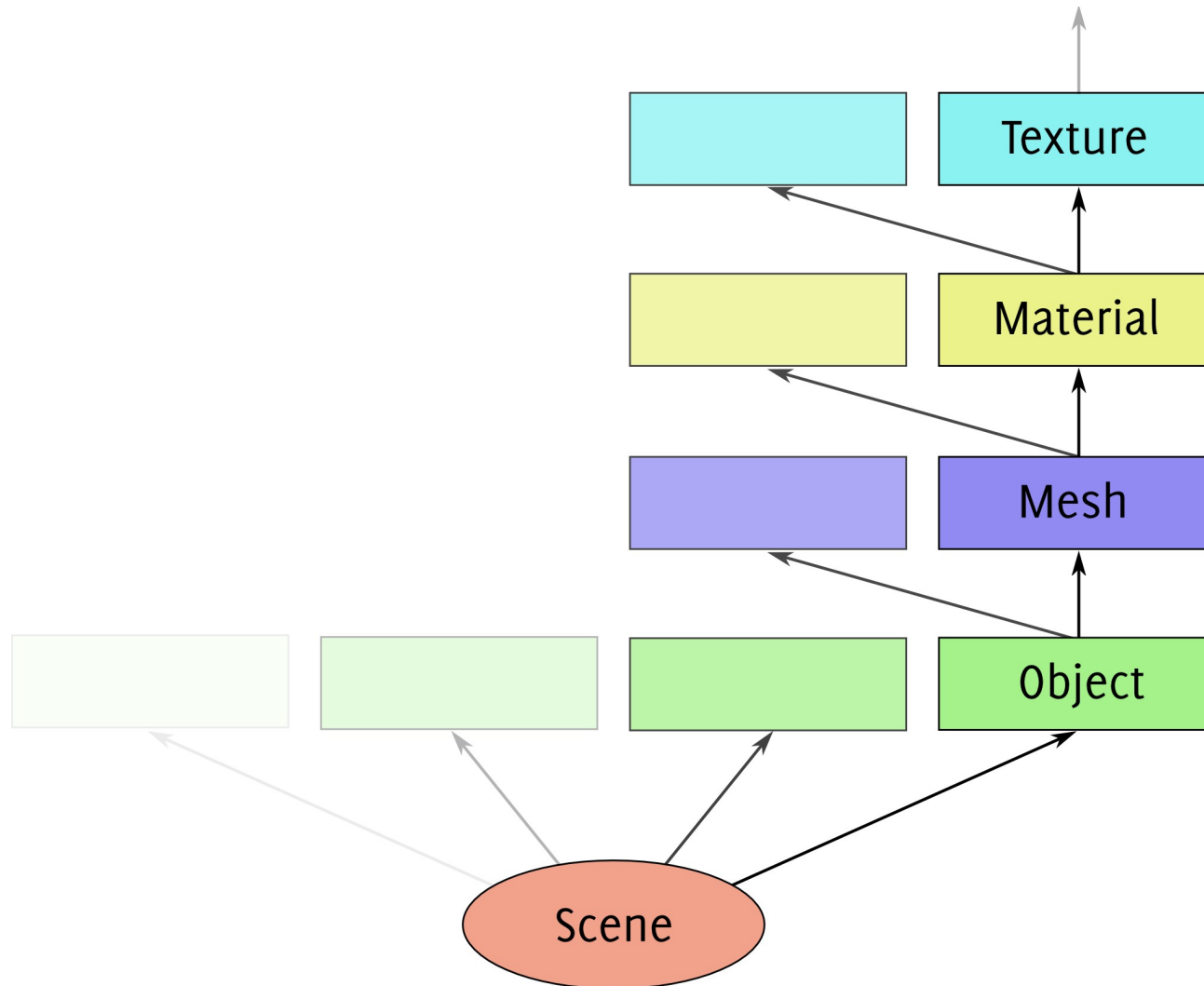
Blender – Rendering

- Numpad '0' – Shows the camera view
- 'F12' renders using current settings and camera, light positions
- Internal renderer supports ray-tracing
- Several external rendering engines supported (Yafray, Indigo)
- It is possible to make image composition (effects) in Blender based on the render layers

Blender – Loading and saving

- Save & Load .blend files
 - 'Ctrl' + 'w' – Open save file dialog
 - 'Ctrl' + 'o' – Open recent
 - 'F1' – Open load file dialog
- Save render result
 - 'F3' – Open save image dialog

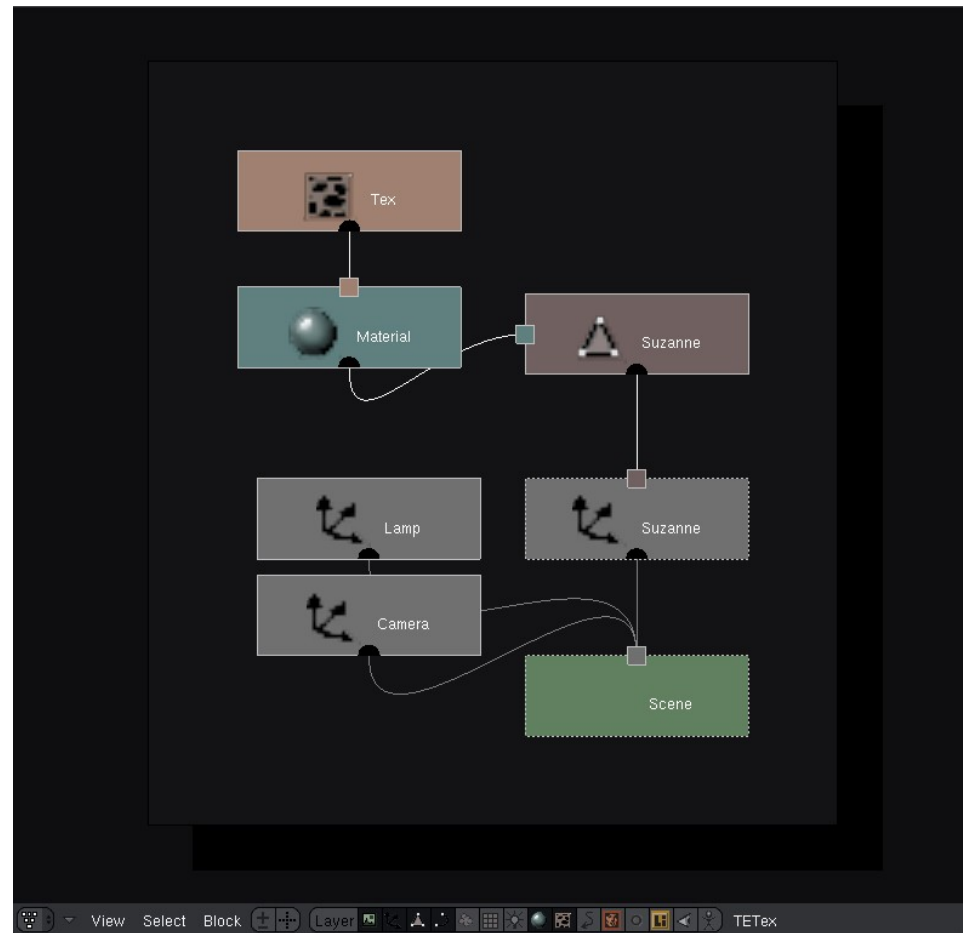
Blender - Object structure



Blender - Object structure

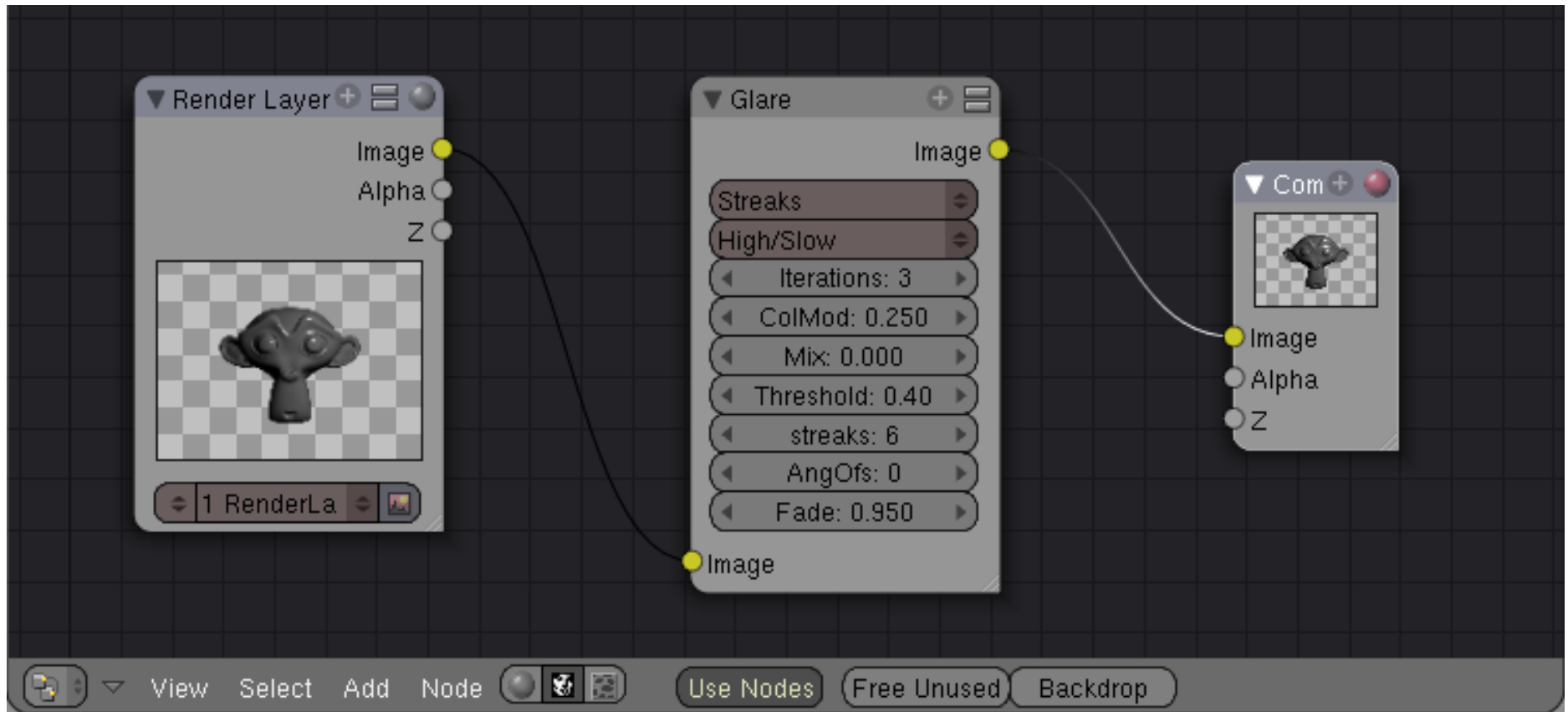
- Structure of a file can be seen using the OOPS-schematic view 

OOPS-schematic for a small scene



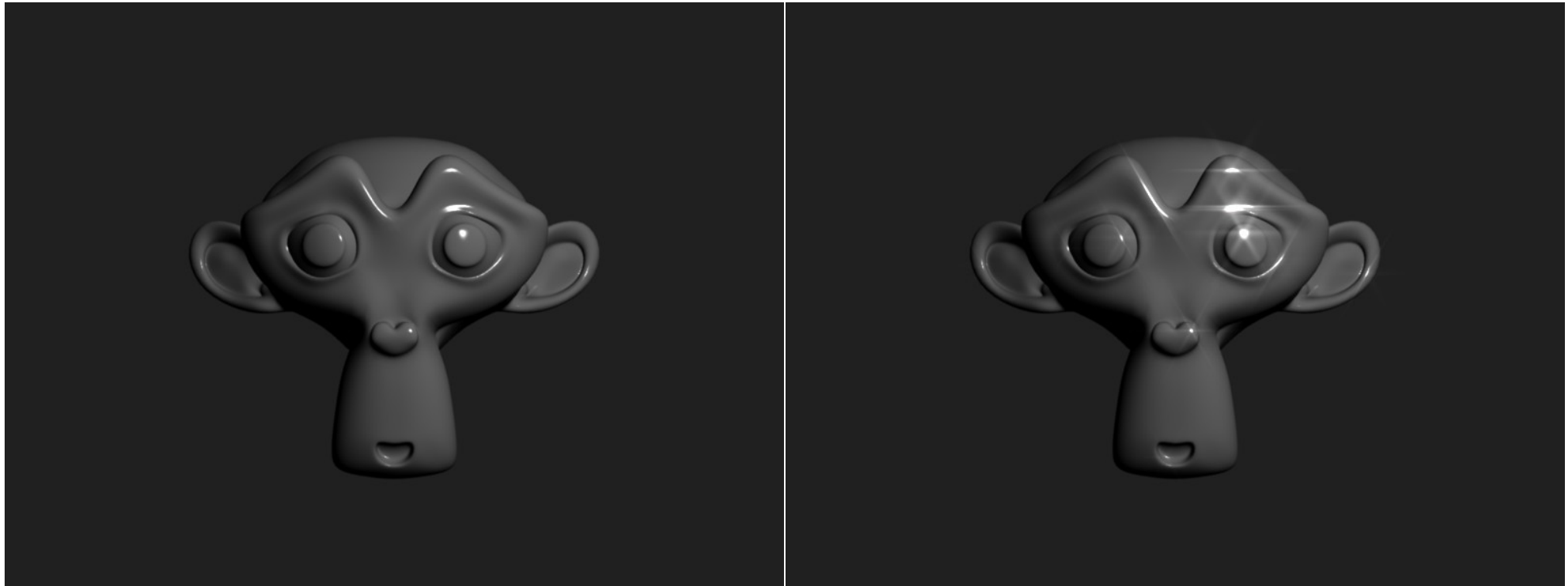
Blender – Node composition

- Built-in post-processing of your render



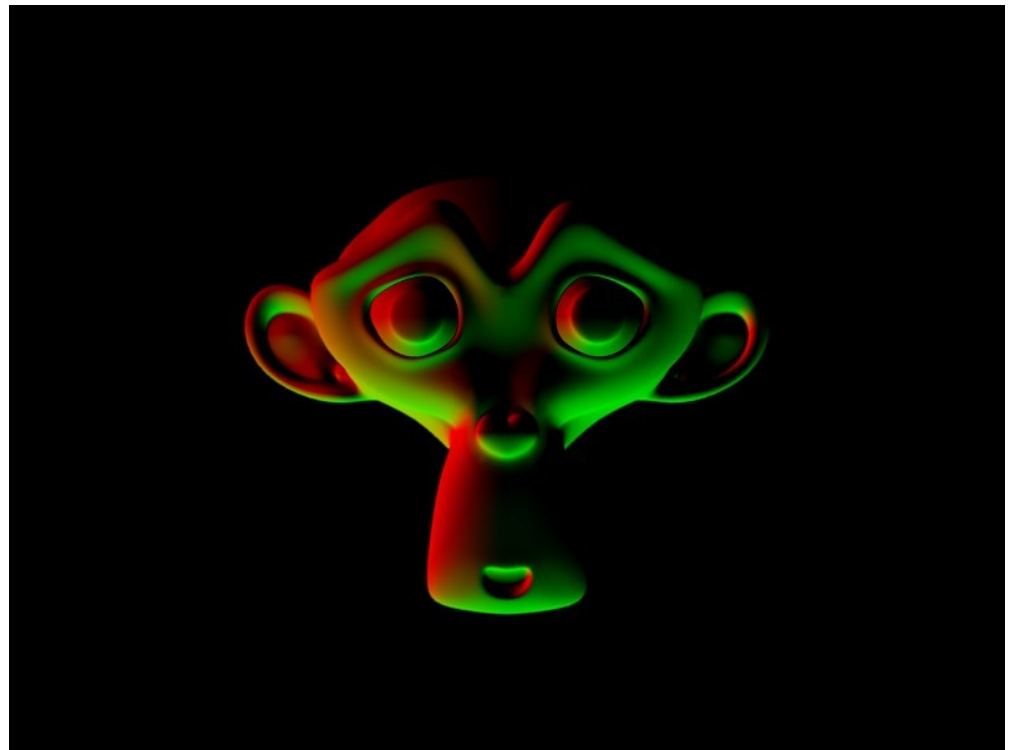
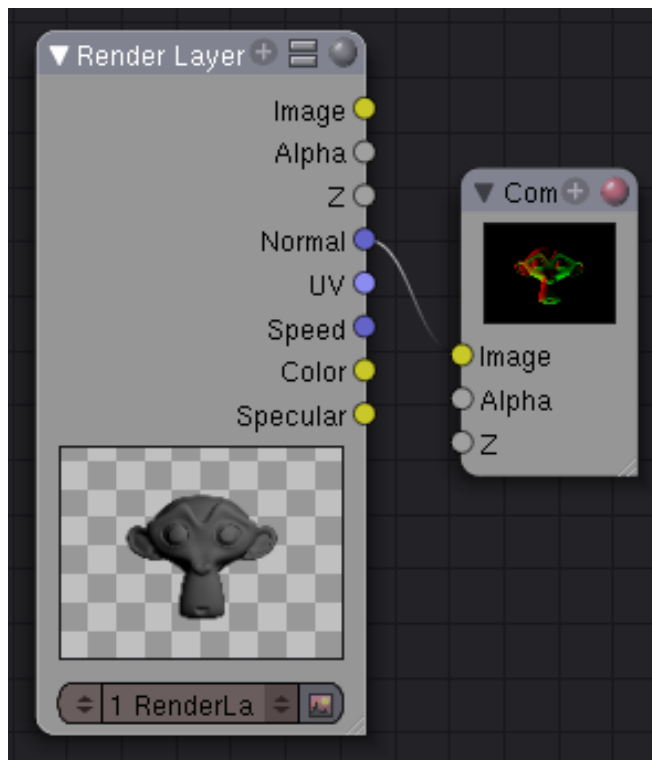
Blender – Node composition

Effect of the 'Glare'-node



Blender – Node composition

- Possible to separate and manipulate different layers of the render result

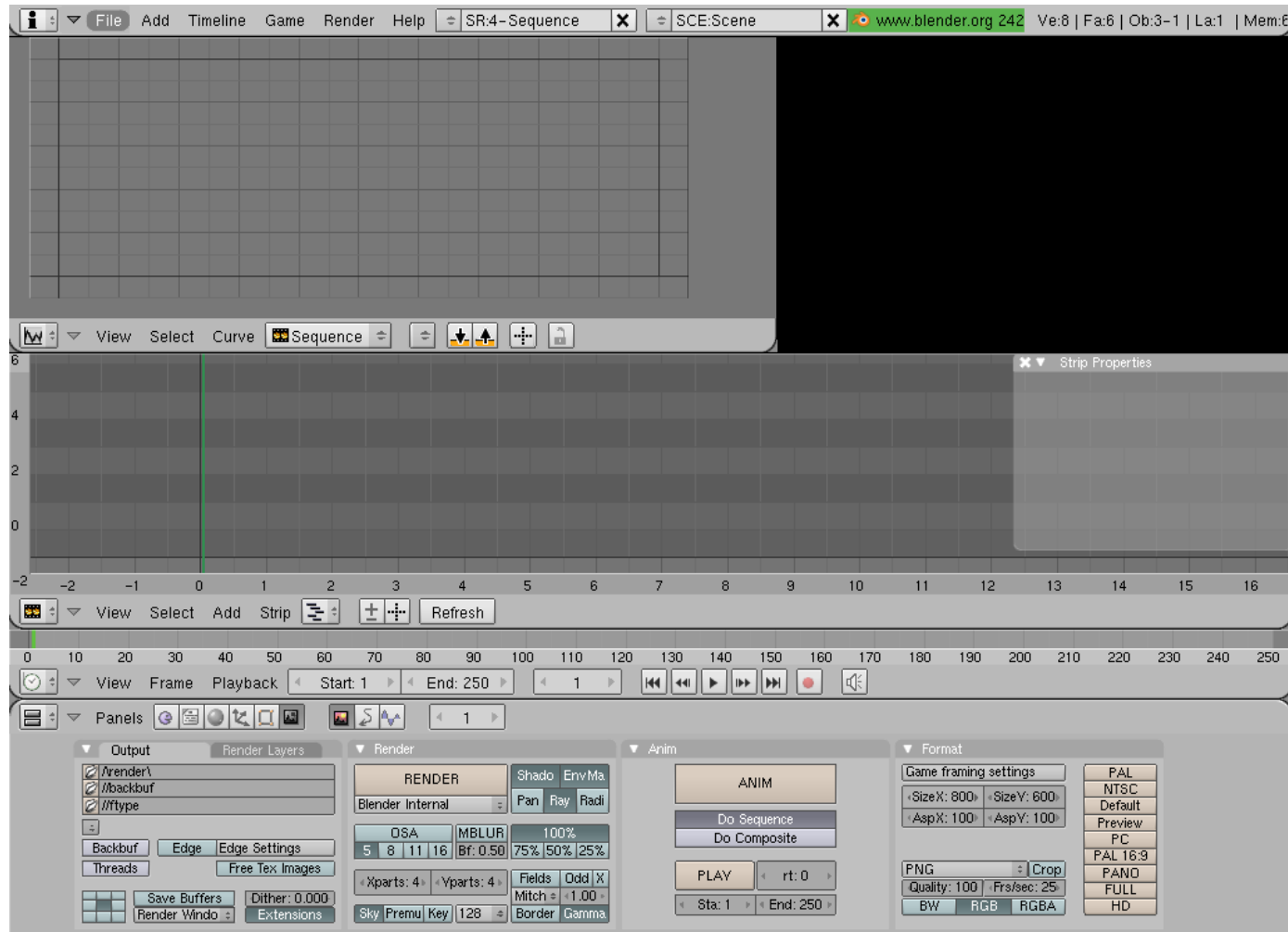


Blender – Sequence editor

- A very powerful tool for editing movies
- Can combine multiple video clips, sound and add effects to create a final result
- Can be used together with node editor to add more control



Blender – Sequence editor



Blender – Import and Export

- Import/Export scripts exist for several common 3D-formats
- “Open source quality”
- Writing custom scripts possible



Blender – Python scripting

- Blender has its own Python API
- Possible to control most of the functionality through Python scripts
- Learning threshold determined by ability to pick up Python

Blender – Python scripting

The screenshot shows the Blender 2.45 interface with a Python script editor open. The script is titled 'raw_export.py' and is used to export selected meshes to Raw format. The script includes the following code:

```
#!/usr/bin/python
'''
Name: 'Raw Faces (.raw)...'
Blender: 2.45
Group: 'Export'
Tooltip: 'Export selected mesh to Raw Format (.raw)'
'''
__author__ = "Anthony D'Agostino (Scorpius)"
__url__ = ("blender", "blenderartists.org",
"Author's homepage, http://www.redrival.com/scorpius")
__version__ = "Part of I0Suite 0.5"

__bpydoc__ = """
This script exports meshes to Raw file format.

The raw triangle format is very simple; it has no verts or faces lists.
It's just a simple ascii text file with the vertices of each triangle or quad
listed on each line. There were some very old utilities (when the PovRay
forum was in existence on CompuServe) that performed operations on these
files.

Usage:
Select meshes to be exported and run this script from "File->Export" menu.
'''

# $Id: raw_export.py 14597 2008-04-28 16:09:17Z campbellbarton $
#
# Copyright (c) 2002 Anthony D'Agostino
# http://www.redrival.com/scorpius
# scorpius@netzero.com
# april 28, 2002
# Read and write RAW Triangle File Format (*.raw)
#
# ***** BEGIN GPL LICENSE BLOCK *****
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software Foundation,
# Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
# ***** END GPL LICENSE BLOCK *****

import Blender
import BPyMesh

# =====
# Write RAW Triangle Format
# =====
def write(filename):
    start = Blender.sys.time()
    if not filename.lower().endswith('.raw'):
        filename += '.raw'

    scn = Blender.Scene.GetCurrent()
    ob = scn.objects.active
    if not ob:
        Blender.Draw.PupMenu('Error!|Select 1 active object')
        return

    file = open(filename, 'wb')

    mesh = BPyMesh.getMeshFromObject(ob, None, True, False, scn)
    if not mesh:
        Blender.Draw.PupMenu('Error!|Could not get mesh data from active object')
        return

    mesh.transform(ob.matrixWorld)

    file = open(filename, "wb")
    for f in mesh.faces:
```

A yellow box with the text **'Alt' + 'p' to run** is overlaid on the right side of the script editor.

Blender – Import example

```
# Import Demo

import Blender
from Blender import *
import bpy

import math
from math import *

f=open("/home/patrik/pythondata.dat")

coords=[]
edges=[]
counter = 0

for l in f:
    l=l.split(',')
    d=[float(l[0]),float(l[1]),float(l[2])]
    coords.append(d)

    counter = counter + 1
    print(counter)
    if counter > 1:
        edges.append([counter-2,counter-1])

me = Mesh.Primitives.UVsphere(6,6,0.2) # create a primitive

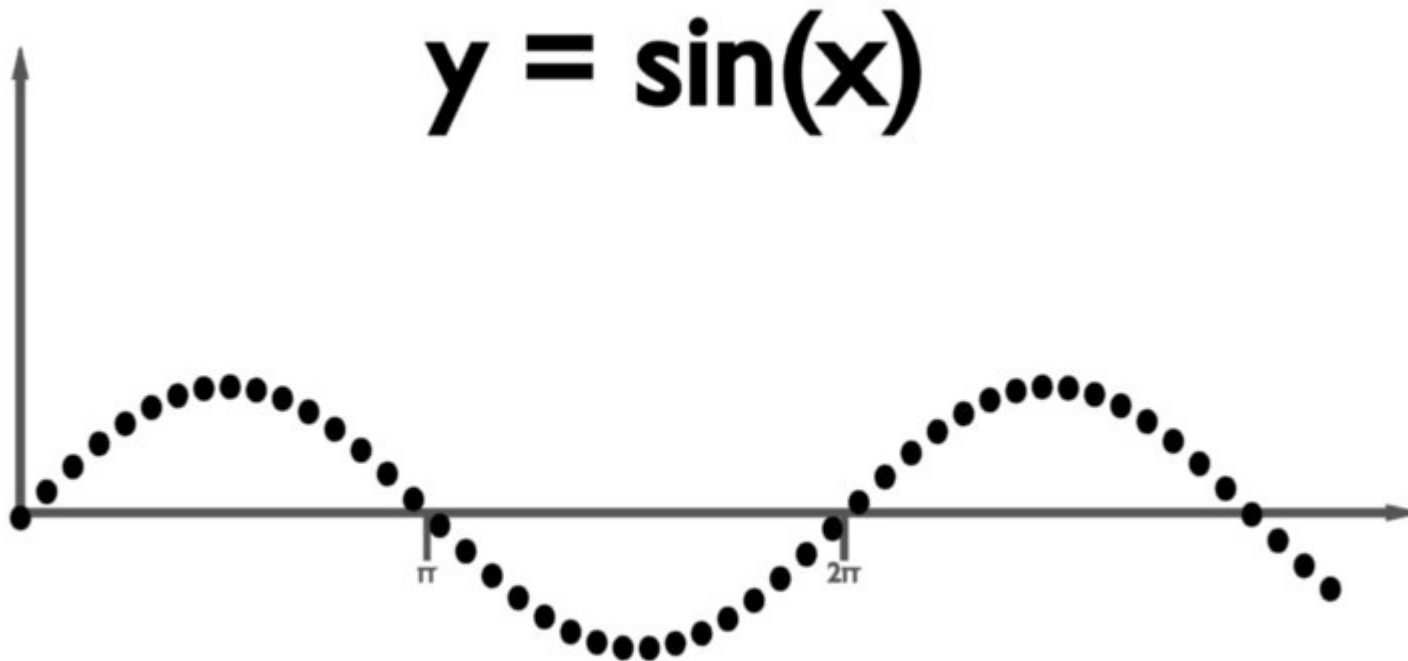
sc = Scene.GetCurrent()           # get current scene
ob2 = sc.objects.new(me)         # add a new mesh-type object to the scene
ob2.setLocation(d)

me = bpy.data.meshes.new()       # create empty mesh
me.verts.extend(coords)          # populate with vertices
me.edges.extend(edges)           # create edges
scn = bpy.data.scenes.active     # link object to current scene
ob = scn.objects.new(me)

Blender.Redraw()
```



Blender - Import example



Blender – SCIENTIFIC visualizations

- Blender is used at scientific groups over the world
- Free and open source but still powerful the main selling points
- Uses include
 - Electron microscope data visualizations
(<http://www.sciencemag.org/content/vol320/issue5872/cover.dtl>)
 - Population behavior simulation using the Blender game engine (<http://vimeo.com/2158835>)
 - Molecular visualization and simulation
(<http://www.scivis.ifc.cnr.it/>)



Blender – Some links

- A few links that could be interesting
 - www.blender.org
 - www.blendernation.com
 - www.blendercookie.com
 - www.blenderartists.org
 - www.elephantsdream.org
 - www.bigbuckbunny.org
 - www.open3dcourse.se (Gävle högskola summer course)