

Visualizing Electron and X-Ray Transport in Complex 3-D Samples using NISTMonte

Nicholas W. M. Ritchie & John Henry Scott
Microanalysis Group
National Institute of Standards & Technology
Gaithersburg, MD 20899

Introduction to NISTMonte

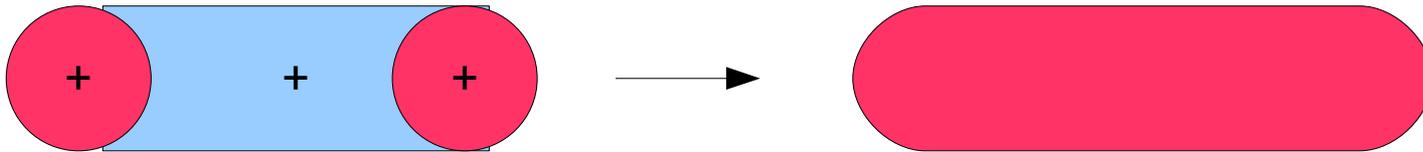
- Monte Carlo simulation of electron transport and x-ray generation and transport
 - Written in Java (J2SE 1.5 or higher)
 - Available with source code
- Features
 - Arbitrarily complex sample geometries
 - Interchangeable physics
 - Mix & match detection schemes
 - Scriptable in Jython, Java or ?



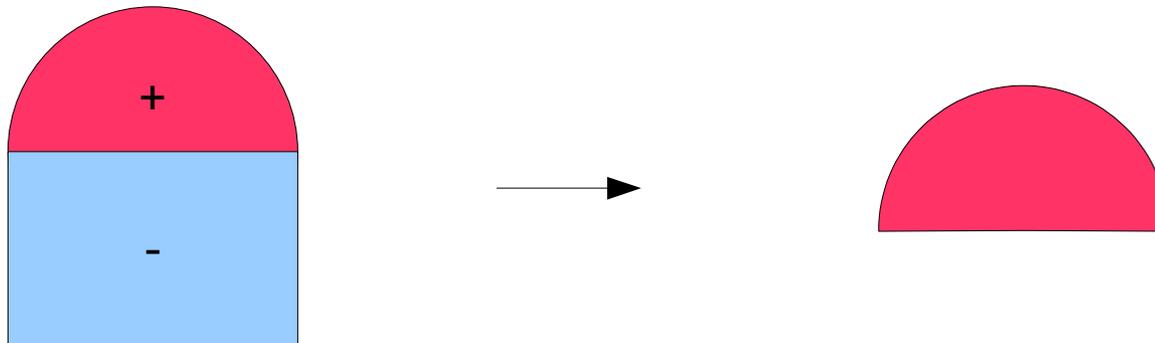
Sample primitives



- Constructed from basic 3D shapes
 - Sphere, Cylinder, Block, Intersection of Planes
- 3D shapes may be combined or differenced
 - 2 spheres + cylinder -> cylinder with rounded ends



- Sphere - plane -> hemisphere



Interchangeable physics

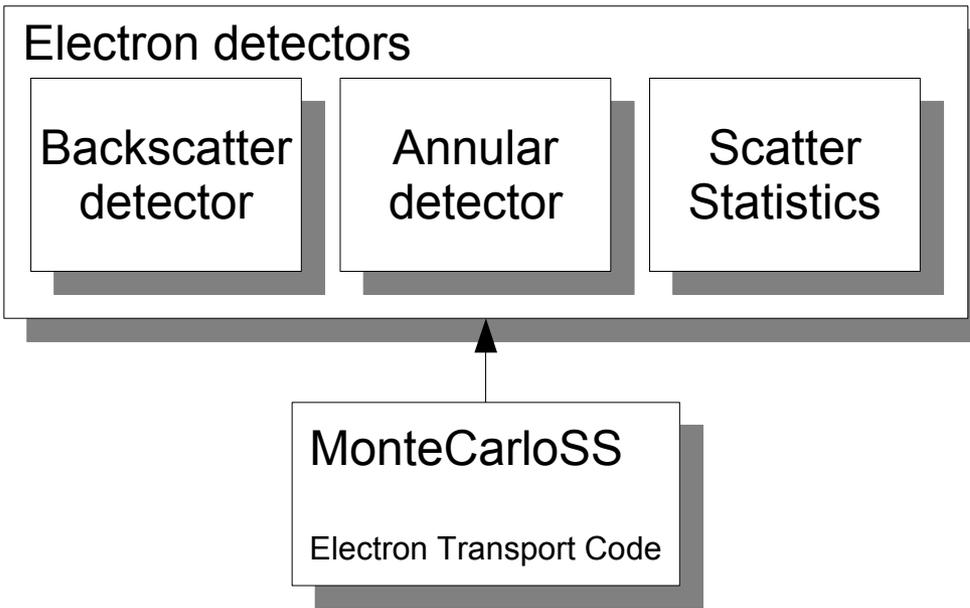
- Elastic scattering models
 - Screened Rutherford
 - NIST SRM-64 (Jablonski, Salvat & Powell)
 - Czyzweski (Czyzewski, MacCallum, Romig & Joy)
- Mass absorption coefficients
 - NIST FFAST
 - Heinrich IXCOM 11
 - Many more...
- Energy loss models
- Characteristic x-ray generation model

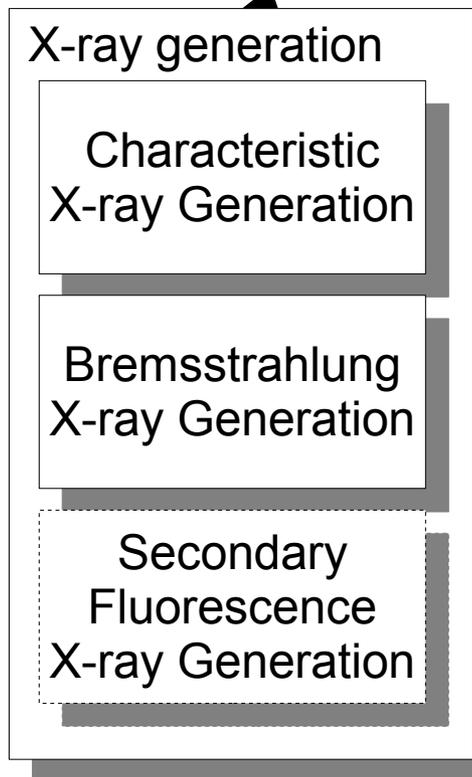
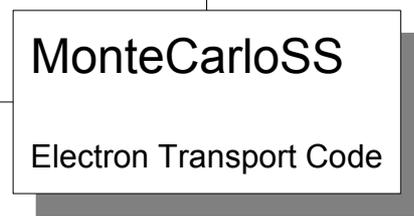
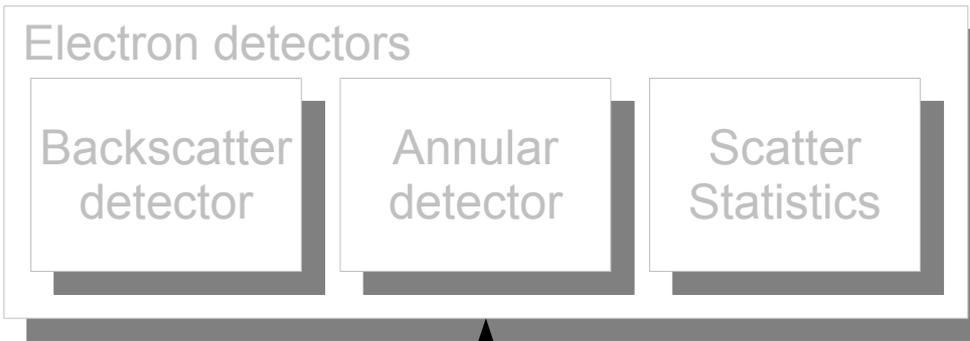
Detection Schemes

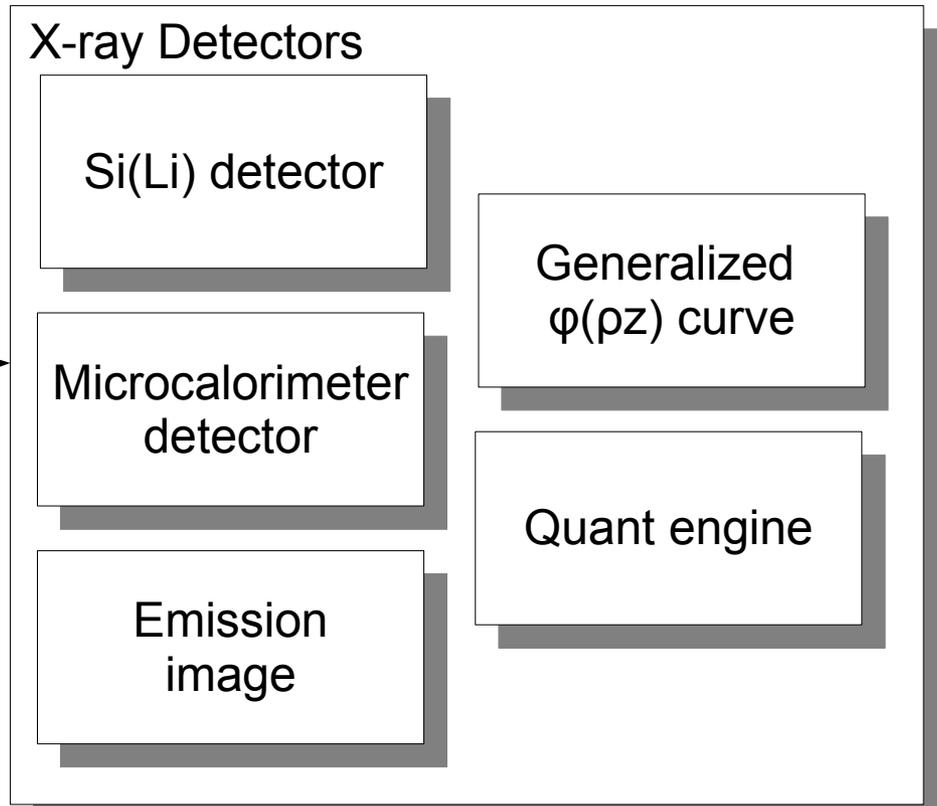
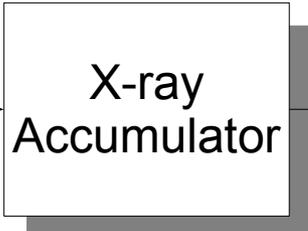
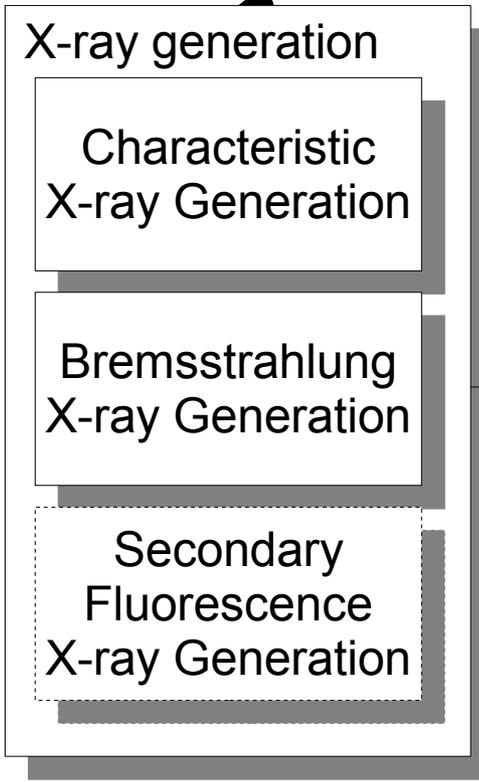
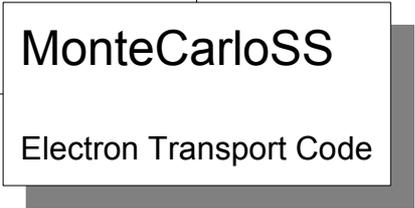
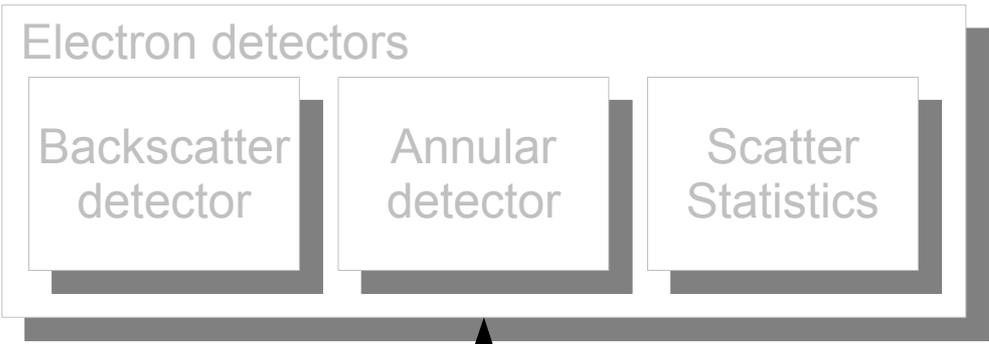
- Backscatter
 - Energy resolved backscatter detector
- Annular detector
 - Records electrons passing through a concentric set of planar rings
- X-ray emission image
 - Image per x-ray line showing the spatial dependence of generation and emission
- Trajectory image
 - Showing electron trajectories projected into a plane
- Trajectory VRML
 - Shows sample geometry and electron trajectories in a 3D CAD-like view
- Spectrum generation
 - Models an EDS detector measuring characteristic & bremsstrahlung emission

MonteCarloSS

Electron Transport Code







Scriptable

```
tw=jio.OutputStreamWriter(jio.FileOutputStream(baseDir+"result.csv"),cs.Charset.  
forName("UTF-8"))  
for pressure in [5.0, 10.0, 30.0, 50.0, 100.0]: # pascal ← Loop over pressure  
    print "Pressure = %g Pa" % (pressure)  
    # create an instance of the model  
    monte = nm.MonteCarloSS() ← Initialize the model  
    monte.setPhysics(nm.VPCompatiblePhysics())  
    monte.setBeamEnergy(epq.ToSI.keV(25.0))  
    # create a region of N2 above the substrate (15 mm)  
    mat = epq.Gas([epq.Element.N],[2], pressure,epq.ToSI.centigrade(25.0), "N2")  
    shape = nm.SimpleBlock([width/2.0,width/2.0,-15.0e-3],[-width/2.0,-  
width/2.0,0.0]) ← Define the geometry  
    vpRegion = monte.addSubRegion(monte.getChamber(),mat, shape)  
    # place an annular detector above the primary particle  
    annular1 = nm.AnnularDetector(epq.ToSI.micrometer(1000.0), 100, ← Add detector(s)  
[0.0,0.0,0.0], [0.0,0.0,-1.0])  
    monte.addActionListener(annular1)  
    monte.runMutipleTrajectories(10000) ← Run the analysis  
    tw.write("Pressure\t%g Pa\n" % (pressure))  
    annular1.dump(tw) ← Write the results  
tw.close()
```

Pro: Flexible, permanent record, iterable
Con: Unintuitive, intimidating to newcomers

Documentation

[Overview](#) [Package](#) **[Class](#)** [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

gov.nist.microanalysis.NISTMonte

Interface MonteCarloSS.Shape

All Known Implementing Classes:

[CylindricalShape](#), [MultiPlaneShape](#), [ShapeDifference](#), [SimpleBlock](#), [Sphere](#), [SumShape](#)

Enclosing interface:

[MonteCarloSS](#)

public static interface **MonteCarloSS.Shape**

Title: EPQLibrary.MonteCarloSS.Shape

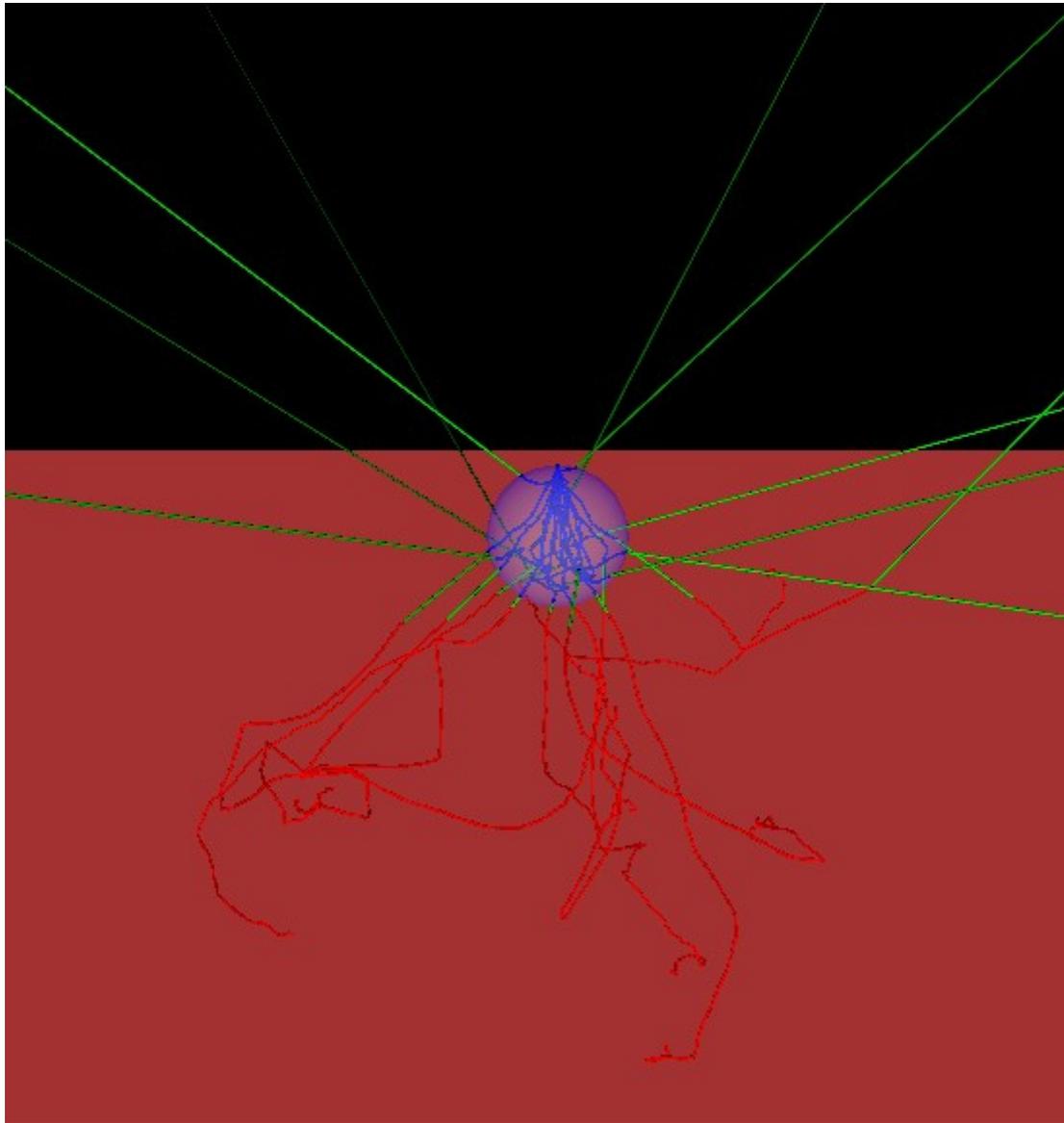
Description: An interface defining sufficient a sufficiently rich set of methods to implement a three dimensional volume. A Region is defined by a Shape and a Material and fully contained child Region objects.

Method Summary

boolean	contains (double[] pos) contains - Is the specified point inside the item represented by this Shape interface? A point on the interface between two Shapes is considered to be inside both Shapes.
double	getFirstIntersection (double[] pos0, double[] pos1) getFirstIntersection - Consider a ray starting at pos0 towards pos1.

Example 1

X-ray map of a K411 particle



0.5 μm radius particle of
K411 on a C substrate

By weight, K411 is

- 42% O
- 9% Mg
- 25% Si
- 11% Ca
- 11% Fe

Conditions:

- 25 keV
- 1000 e^- per pixel

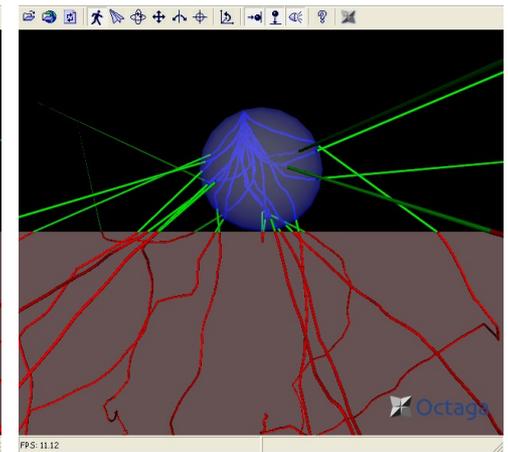
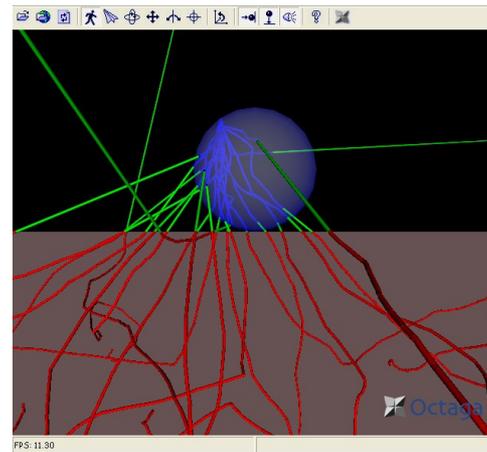
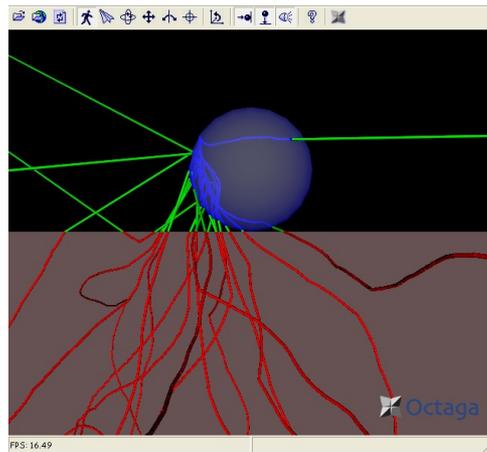
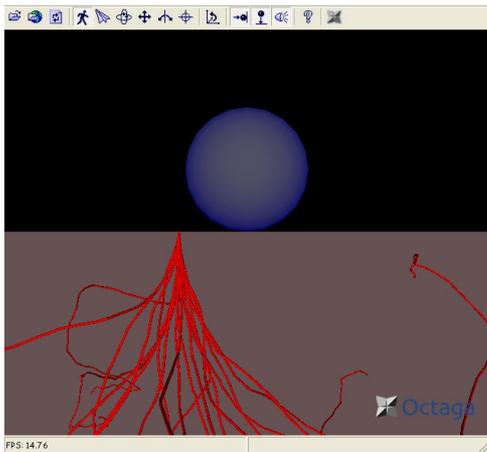
```

dest = jio.File(ScriptFile.getParentFile(),"Example 3")
dest.mkdirs()

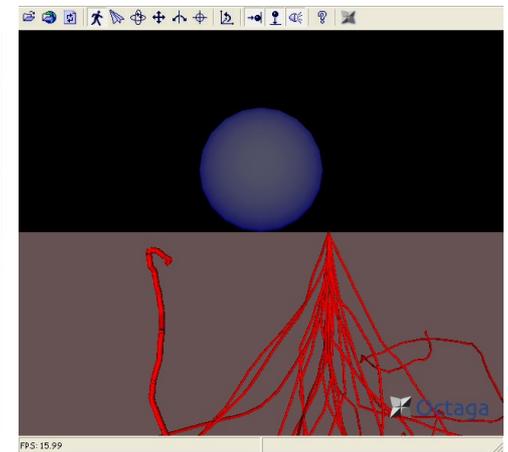
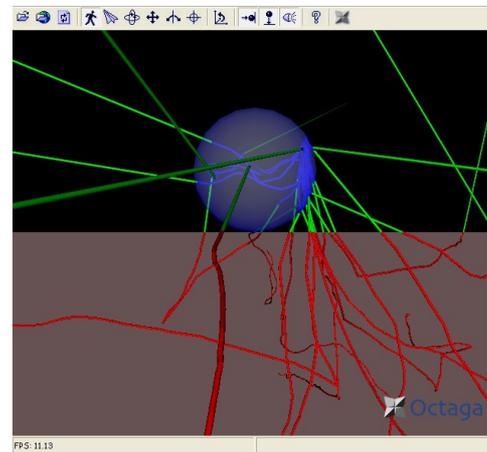
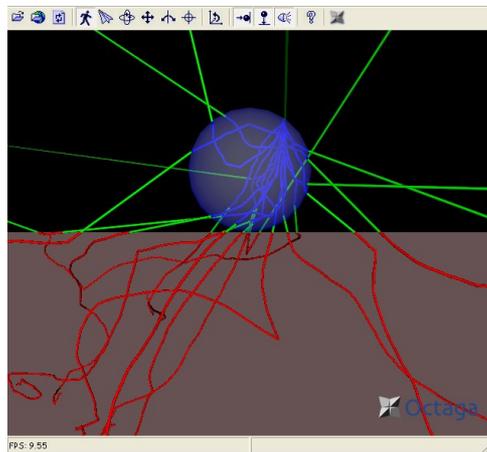
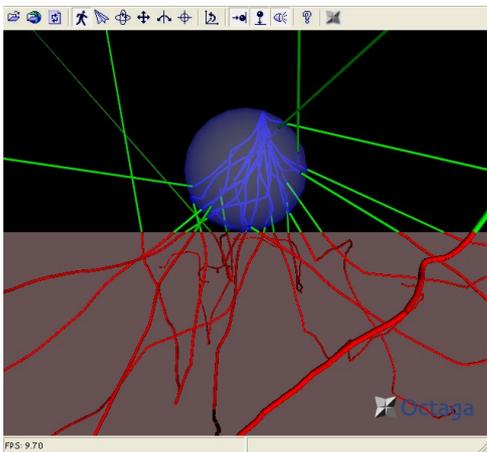
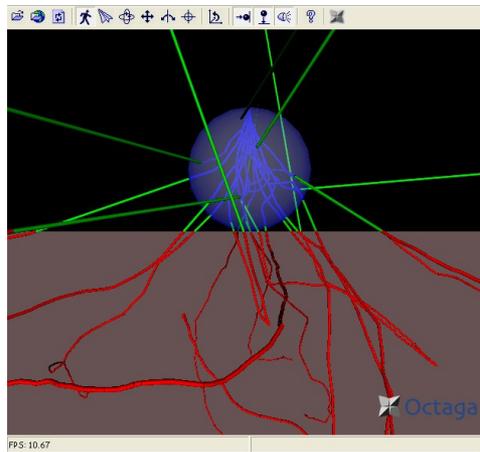
nTraj=1000
characteristic=1
brem=1
radius = 5.0e-7

res=et.RippleFile(64, 64, 2048, et.RippleFile.FLOAT, 4, et.RippleFile.BIG_ENDIAN, dest.toString()+"\K411.rpl",dest.toString()+"\K411.raw")
for x in range(-32,33):
    for y in range(-32,33):
        print "%d, %d" % (x,y)
        # create an instance of the model
        monte=nm.MonteCarloSS()
        monte.setBeamEnergy(epq.ToSI.keV(25.0))
        beam=nm.MonteCarloSS.GaussianBeam(monte,1.0e-9)
        monte.setElectronGun(beam)
        beam.setCenter([1.1*radius*x/32.0,1.1*radius*y/32.0,-0.05])
        # create the sample
        k411=epq.MaterialFactory.createMaterial("K411")
        part=nm.Sphere([0.0,0.0,-radius],radius)
        monte.addSubRegion(monte.getChamber(),k411,part)
        c=epq.MaterialFactory.createPureElement(epq.Element.C)
        block=nm.MultiPlaneShape.createBlock([20.0e-6,20.0e-6,20.0e-6],[0.0,0.0,10.0e-6],0.0,0.0,0.0)
        monte.addSubRegion(monte.getChamber(),c,block)
        if (x%8==0) and (y%8==0):
            # Add a VRML detector
            print "Creating VRML for %d, %d" % (x,y)
            fos=jio.FileOutputStream("%s/K411 (%d,%d).vrml" % (dest,x,y))
            tw=jio.OutputStreamWriter(fos,cs.Charset.forName("UTF-8"))
            vrml = nm.TrajectoryVRML(monte,tw)
            vrml.setMaxTrajectories(20)
            vrml.setTrajectoryWidth(1.0e-8)
            vrml.setDisplayBackscatter(1)
            vrml.addView("Gun",[0.0,0.0,-5.0e-5],[0.0,0.0,0.0])
            vrml.addView("X-Axis",[5.0e-5,0.0,0.0,0.0],[0.0,0.0,0.0])
            vrml.addView("Y-Axis",[0.0,5.0e-5,0.0,0.0],[0.0,0.0,0.0])
            vrml.renderSample()
            monte.addActionListener(vrml)
            monte.runMultipleTrajectories(20)
            tw.flush()
            fos.close()
        if 1:
            # add a detector (to collect a spectrum)
            det = epq.EDSDetector(2048,10.0,130.0)
            det.setGain(1.0e-8/epq.PhysicalConstants.ElectronCharge)
            detPos = monte.computeDetectorPosition(40.0*3.1415/180.0,0.0)
            # add an x-ray event listener
            if characteristic:
                xrel=nm.XRayEventListener(monte,detPos);
                allLines = jutil.HashSet()
                ass = monte.getAtomicShellSet()
                for a in ass:
                    allLines.addAll(epq.XRayTransition.createByDestinationShell(a))
                xrel.setTransitions(allLines);
                monte.addActionListener(xrel)
                xrel.addActionListener(det);
            if brem:
                bre1 = nm.BremsstrahlungEventListener(monte,detPos);
                monte.addActionListener(bre1);
                bre1.addActionListener(det);
            monte.runMultipleTrajectories(nTraj)
            res.seek(x+32,y+32)
            res.write(epq.SpectrumUtils.toDoubleArray(det))
print "Done!"

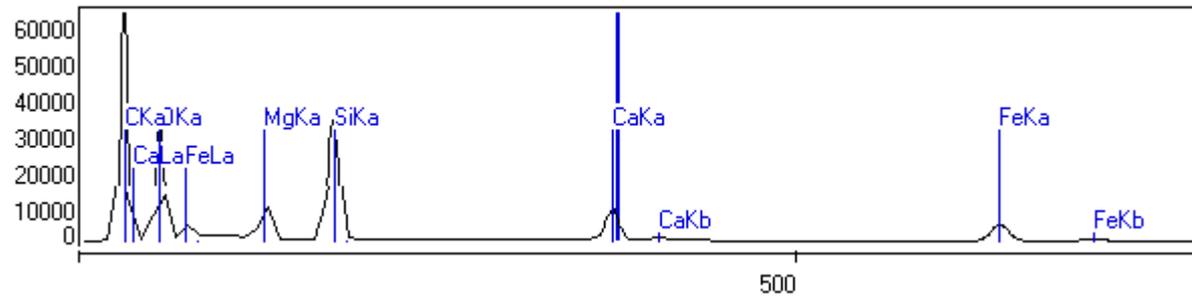
```



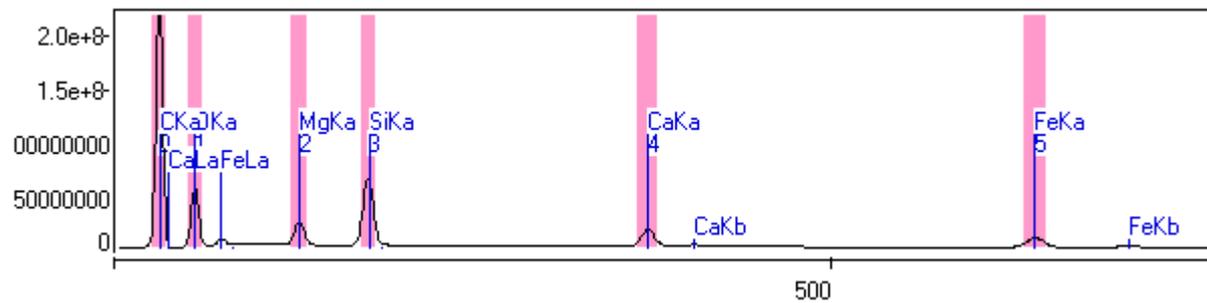
Rastering the beam across the center of the particle.



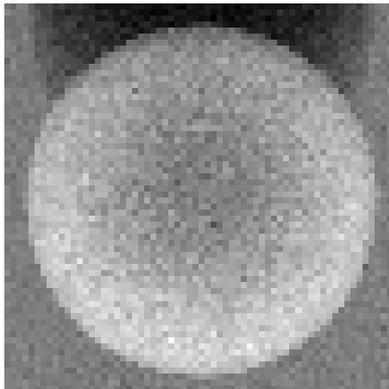
Simulated x-ray spectra



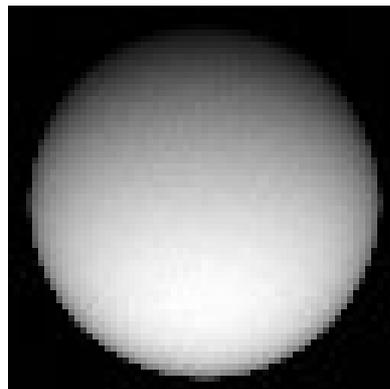
'Max-pixel' derived spectrum



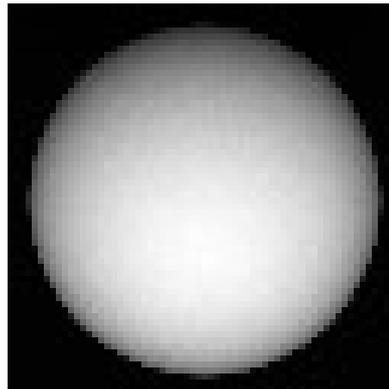
Sum spectrum



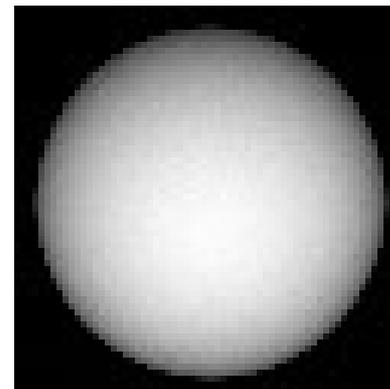
C-K (& Ca-L)



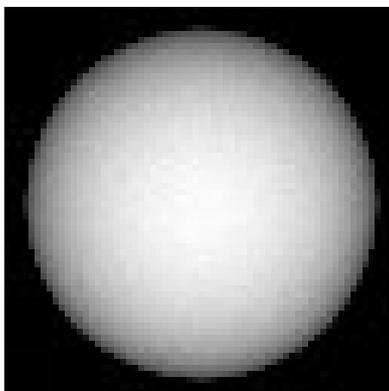
O-K



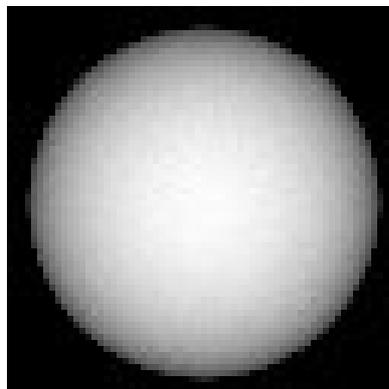
Mg-K



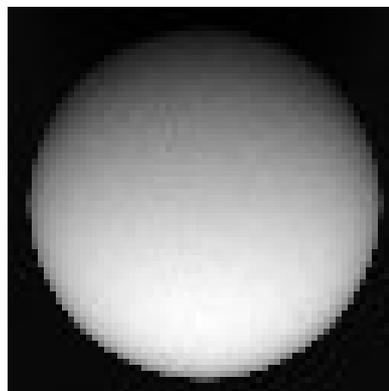
Si-K



Ca-K



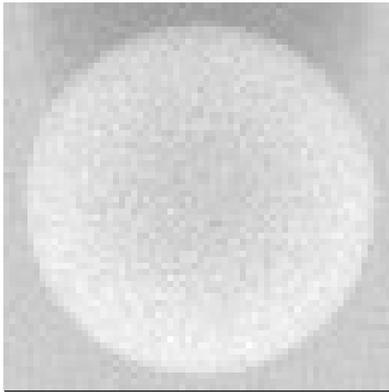
Fe-K



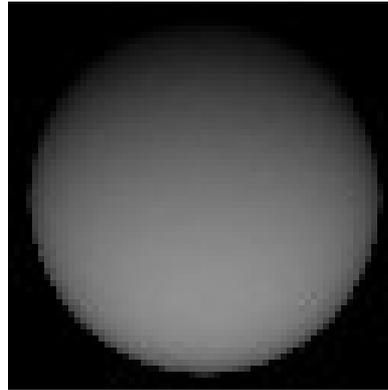
Fe-L

NISTMonte simulated x-ray maps for a 0.5 μm K411 particle

Created using David Bright's LISPIX software



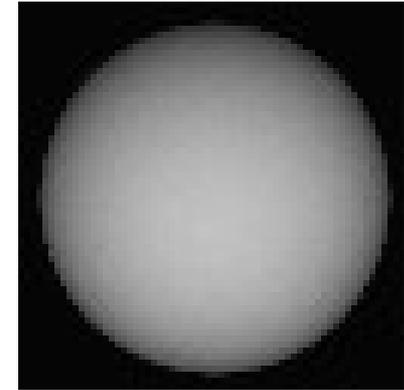
C-K (& Ca-L)



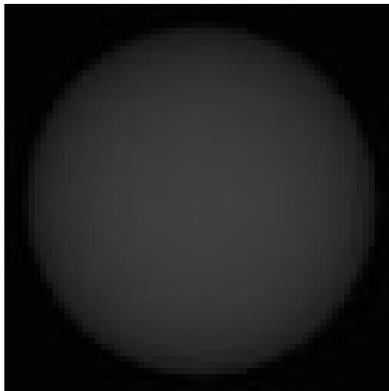
O-K



Mg-K



Si-K



Ca-K



Fe-K



Fe-L

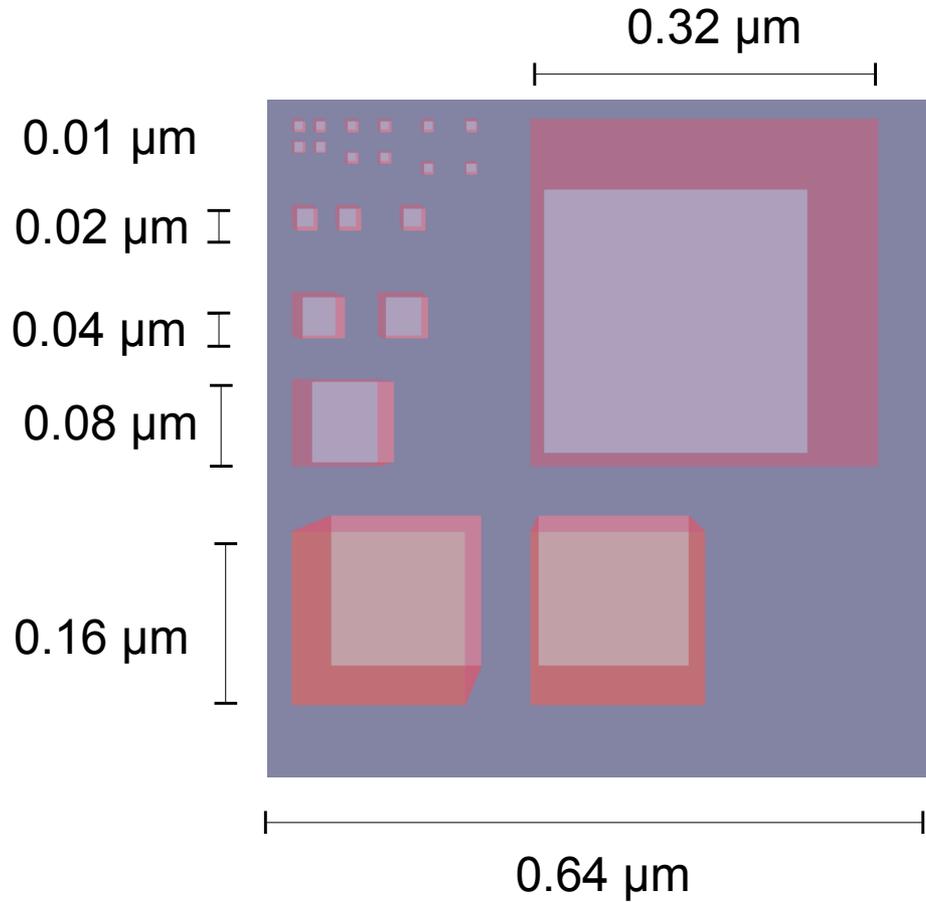
NISTMonte simulated x-ray
maps for a 0.5 μm K411 particle

Created using David Bright's LISPIX software

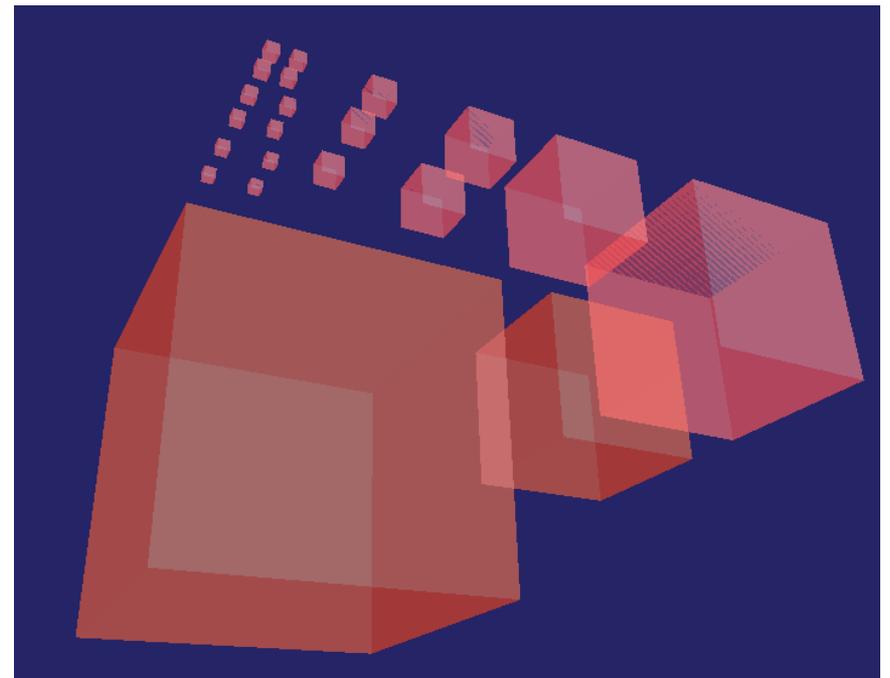
Example 2

How can we see features much smaller than the excitation volume?

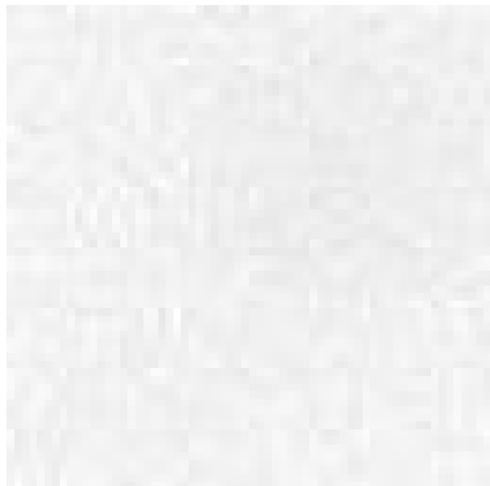
Sample geometry



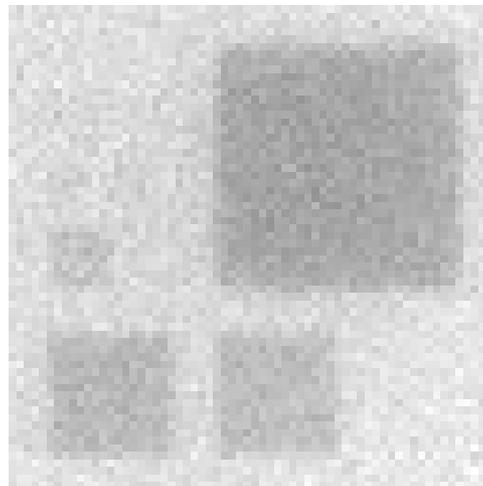
Looking up from within
the iron block.



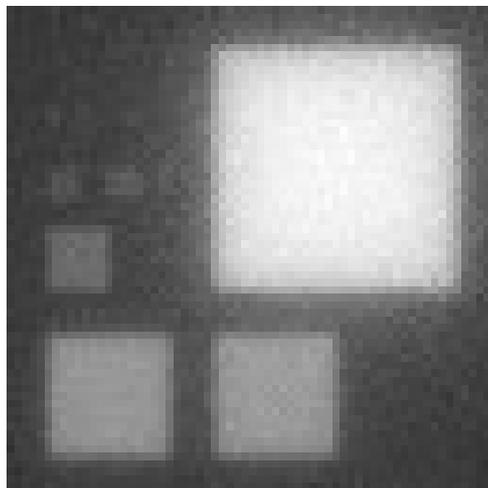
Simulated x-ray maps



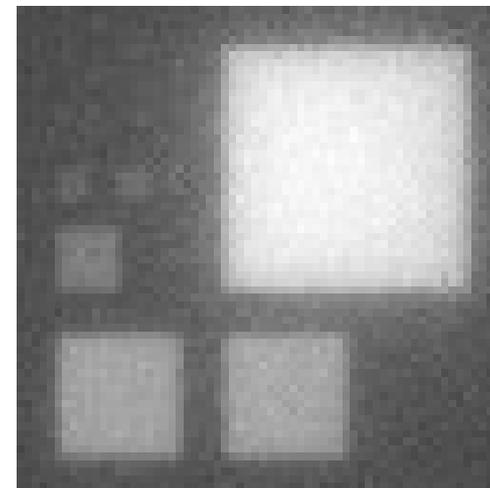
Fe K β



Fe L



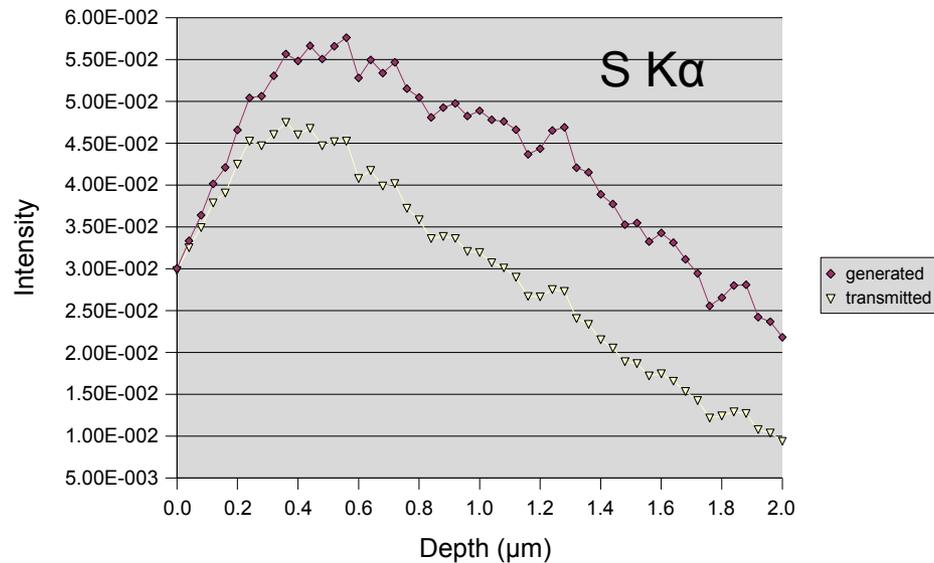
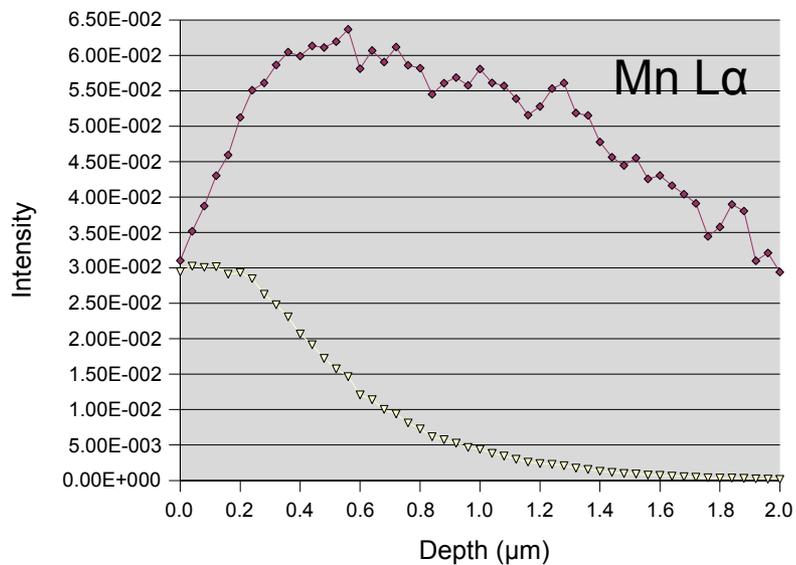
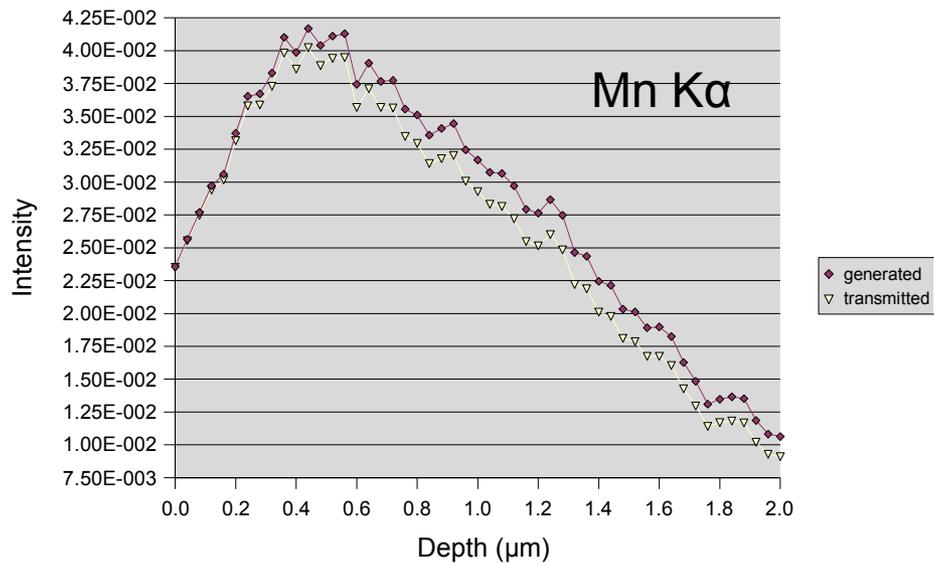
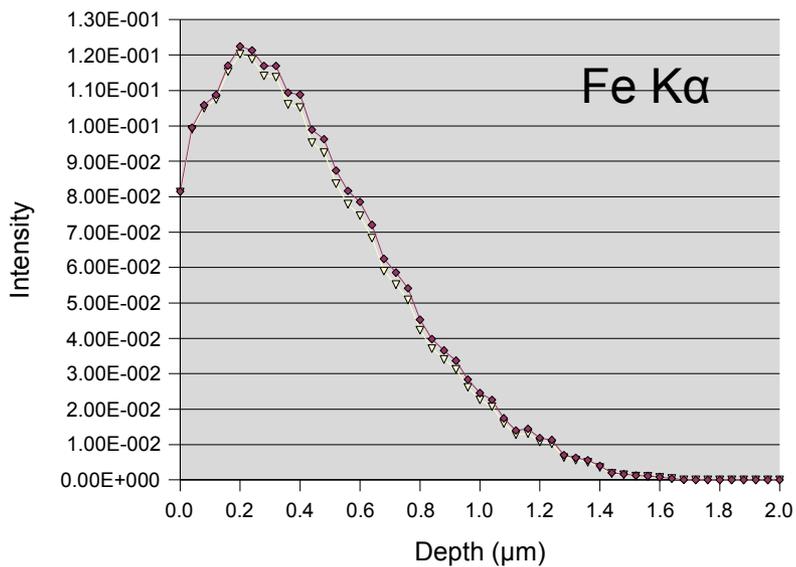
Mn K α



S K α

Created using David Bright's LISPIX software

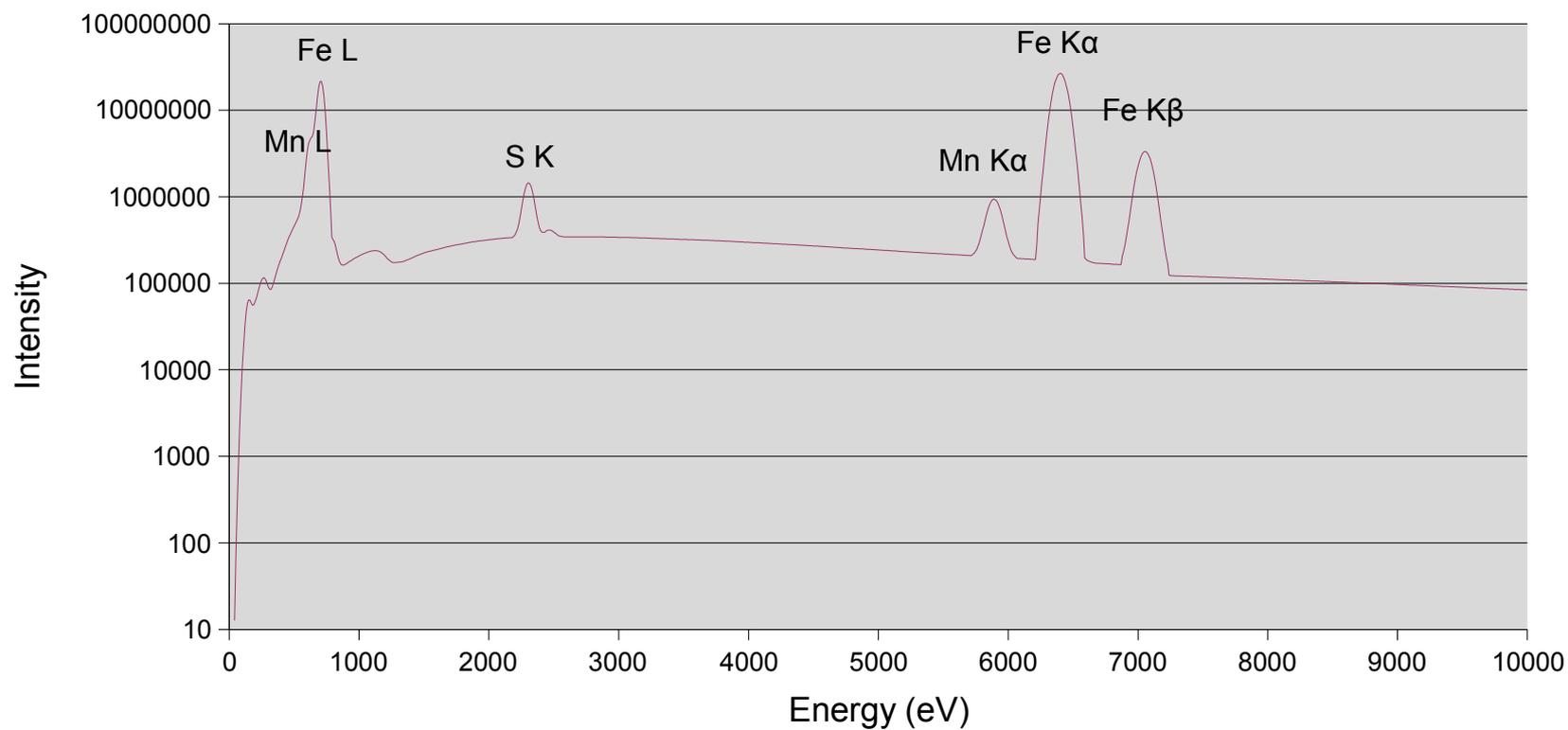
$\phi(\rho z)$ curves for bulk Fe & MnS



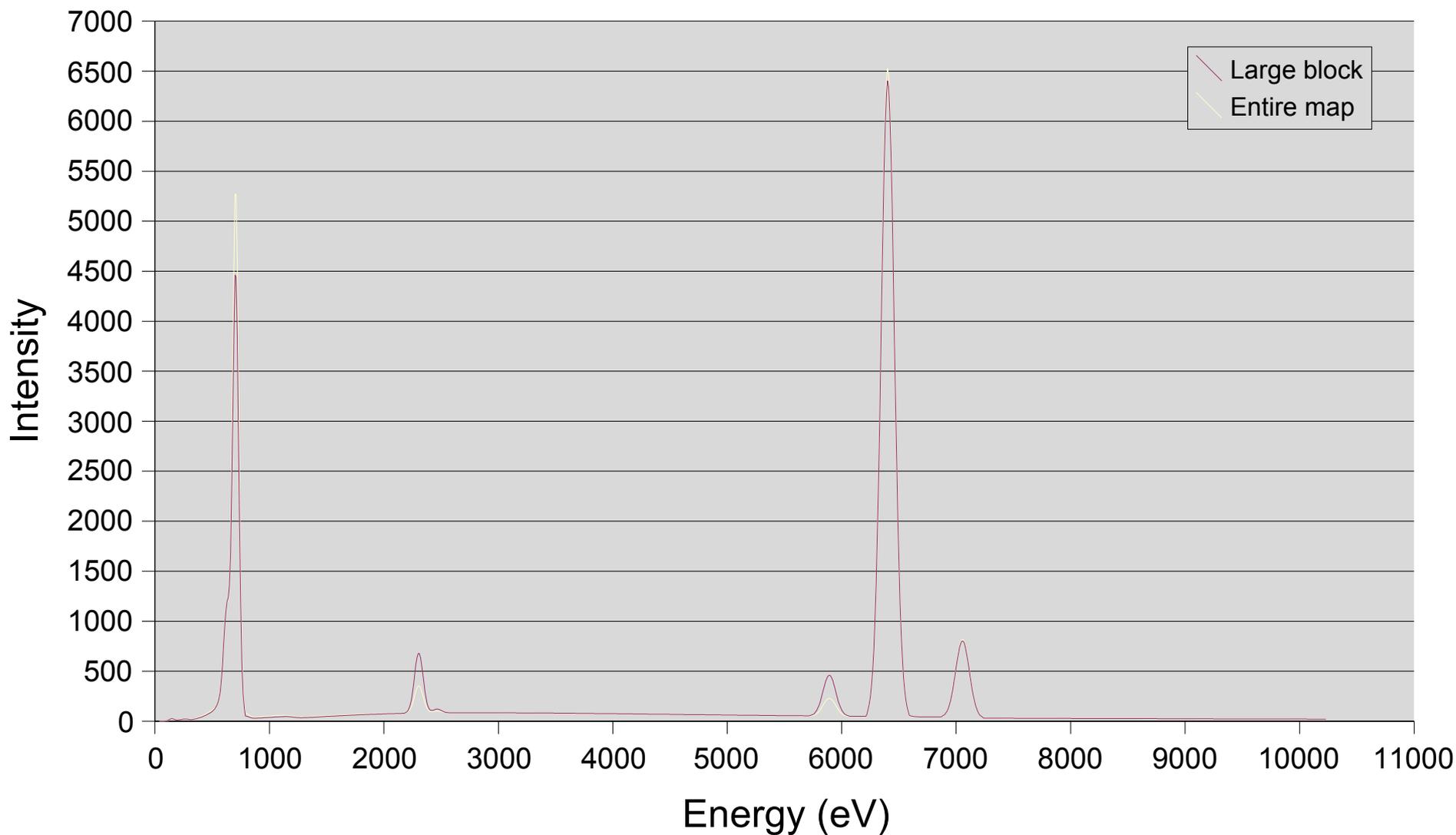
Conclusions

- NISTMonte is capable of simulating electron & x-ray transport in complex 3D samples
- NISTMonte can be scripted to simulate complex iterative experiments
- NISTMonte is currently available at <http://www.duck-and-cover.com> and a new version will soon be available on the NIST web site <http://www.nist.gov>
- .
- While the core NISTMonte functionality remains stable, new features are being added regularly

Sum spectrum for block map



Comparing the large block with the average spectrum



Spectra are normalized to a single pixel intensity