MOLECULAR FOUNDRY

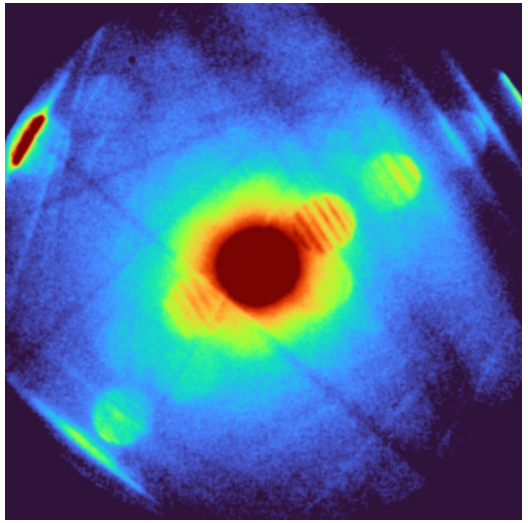
Scientific Figure Making Workshop

Colin Ophus
NCEM, Molecular Foundry, Lawrence Berkeley National Laboratory

5 Jan 2024 – STROBE Retreat
CNSI Auditorium – University of California, Los Angeles

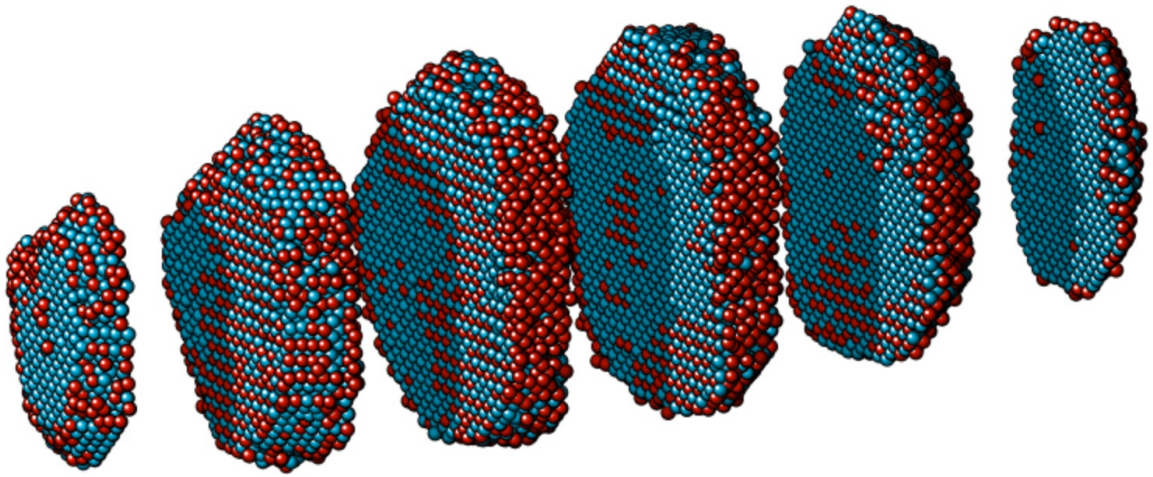
All files and slides available at – github.com/cophus/Programmatic_Figures_Tutorial

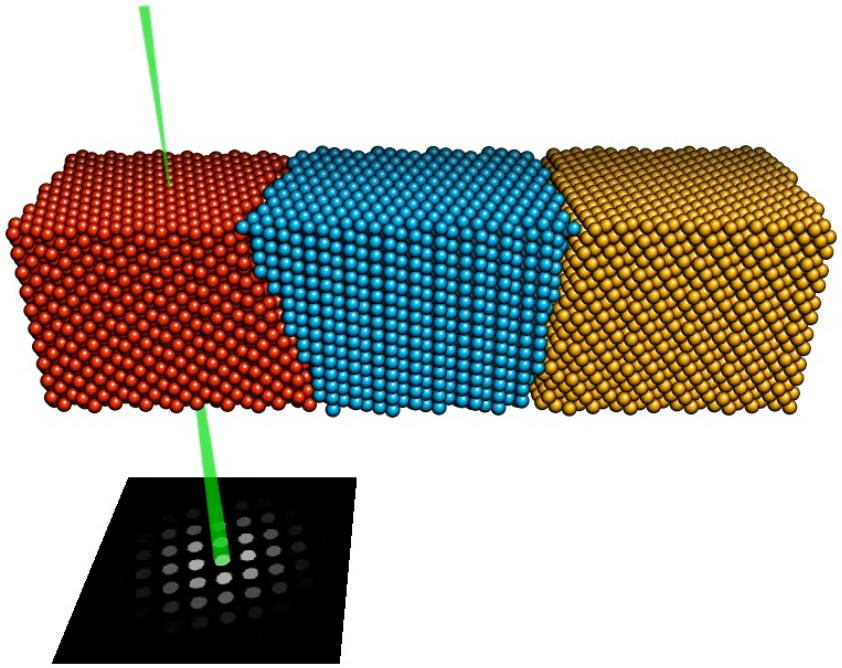
Outline



1 Programmatic figure making **concepts** and **examples** slides.

2 Tutorial on images plotted with automatic contrast scaling. images01_scaling.ipynb





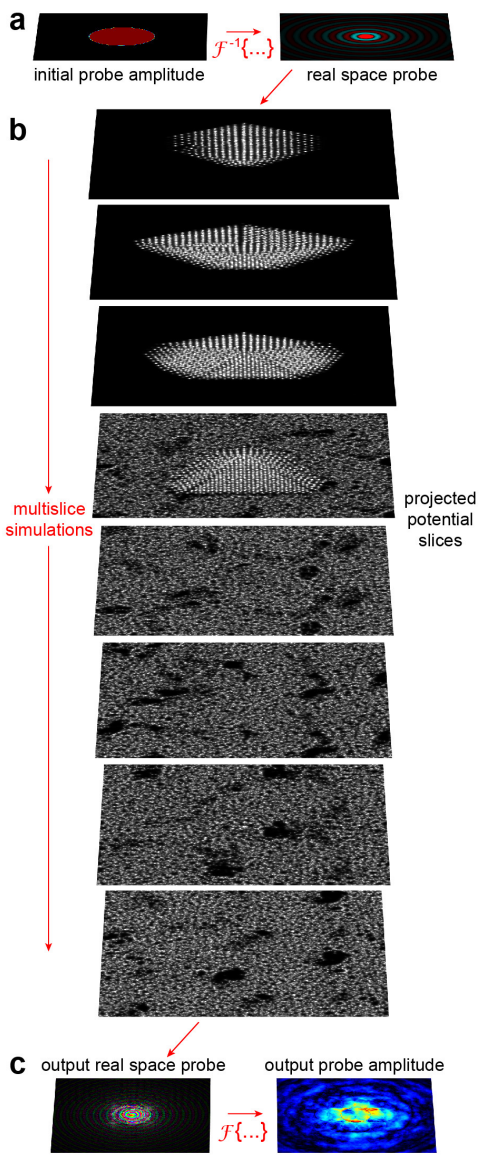
3 Tutorial on plotting **3D atomic** coordinates. atoms01.ipynb

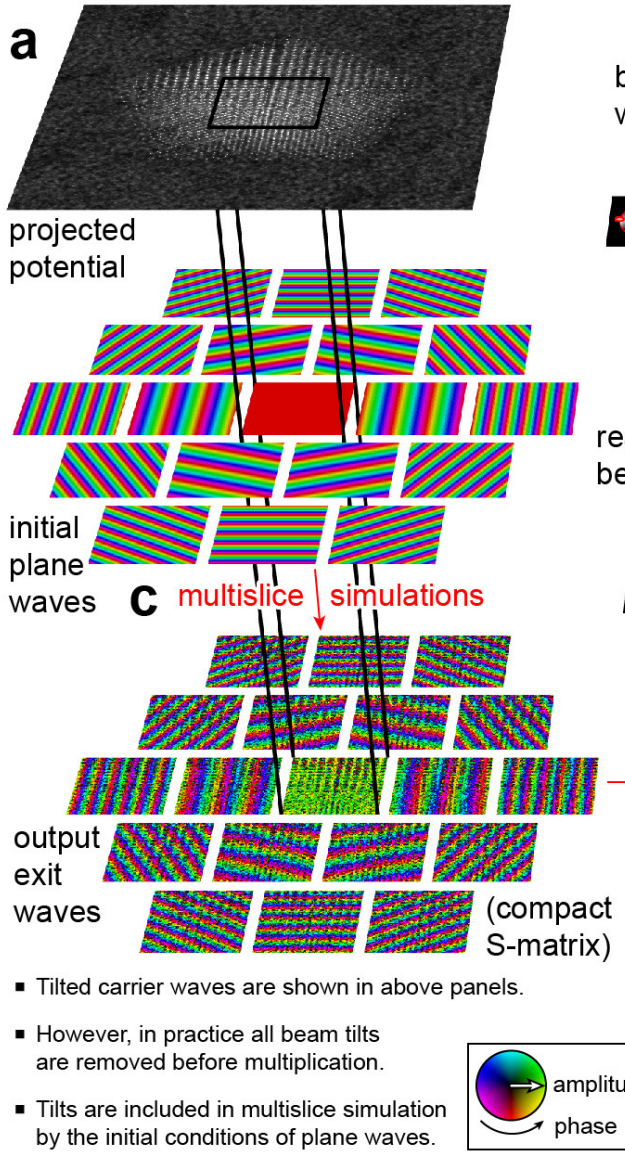
4 Tutorial on rendering **movies** to **demo concepts** in physics. physics01.ipynb

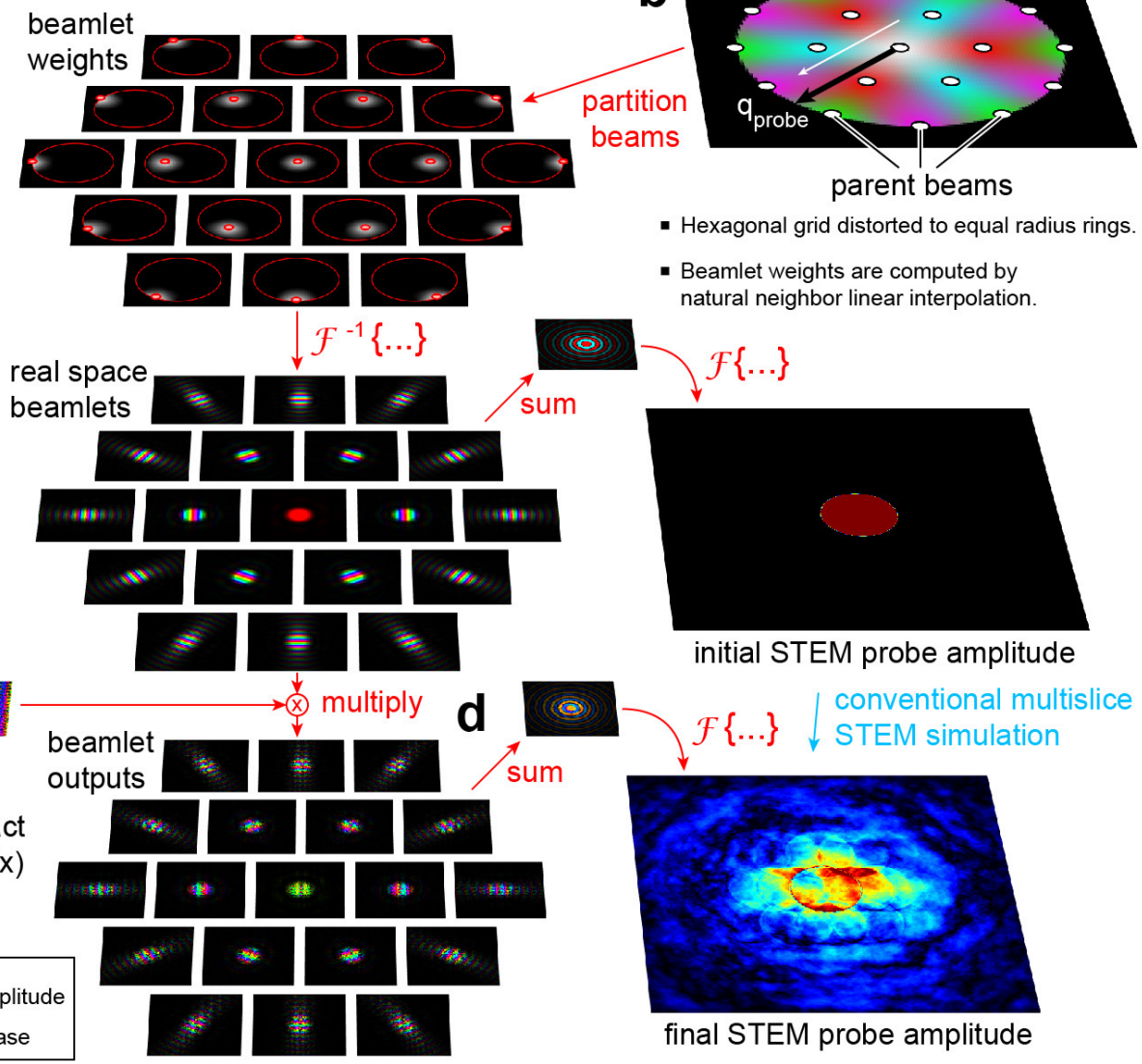
Molecular Foundry – Proposals due Mar 2024 – foundry.lbl.gov

- We are a **user facility** at the Berkeley Lab, operated by the US Department of Energy.
- **Anyone** can submit a proposal (including for **computation**, **simulation** or **analysis**!).
- If accepted by independent review board, access to microscopes and staff is **free**.

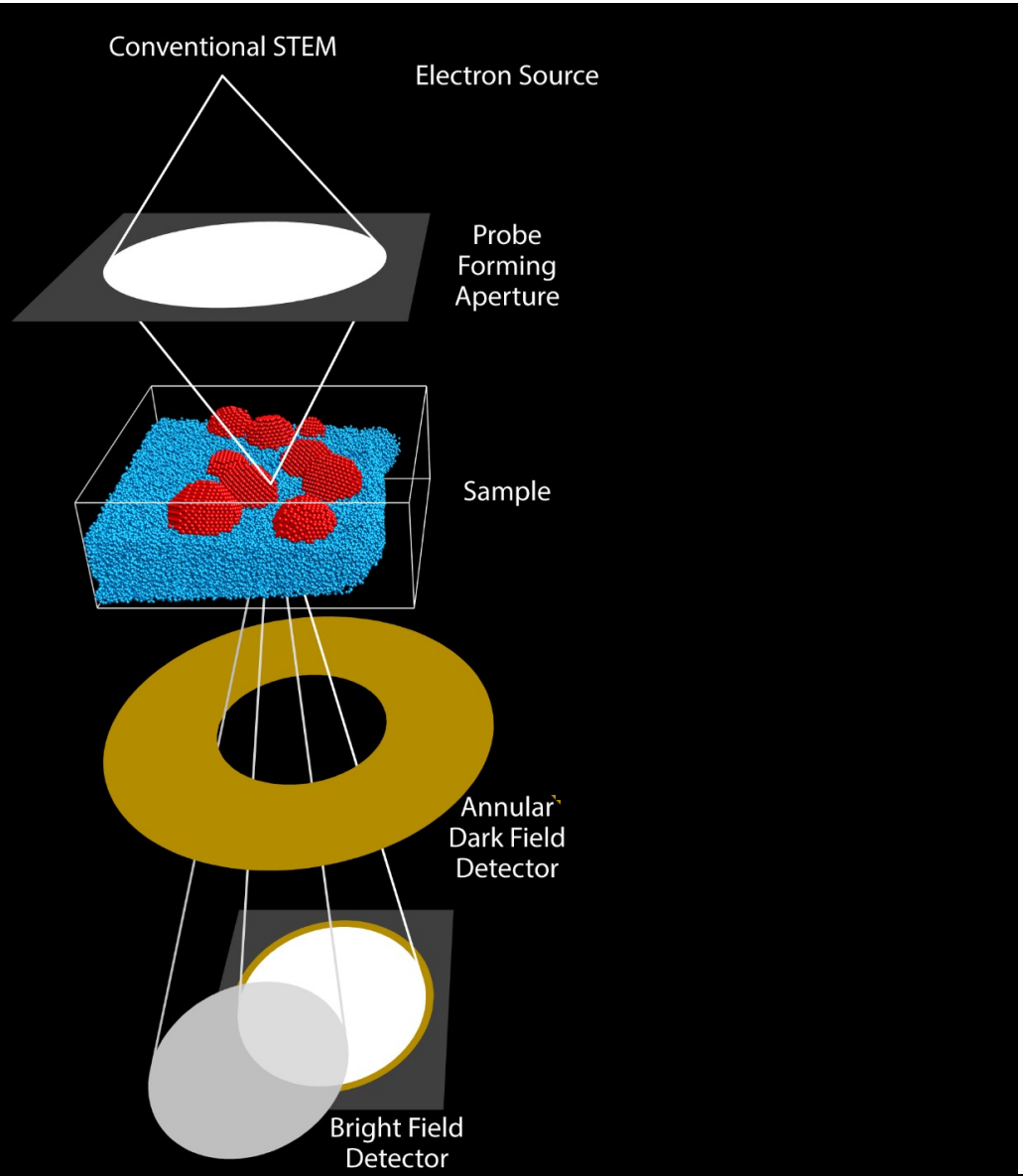
Programmatic Figure Examples

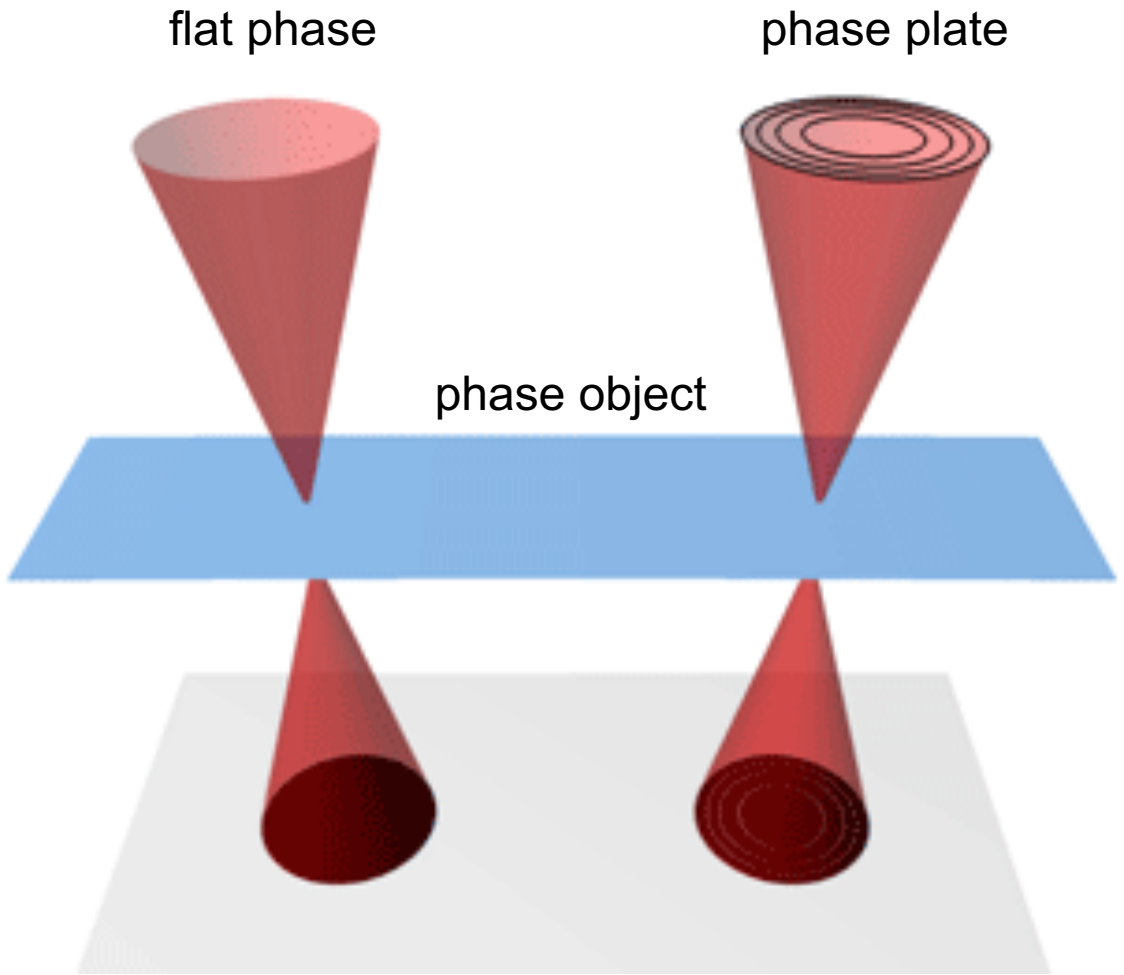






Programmatic Visualization Examples

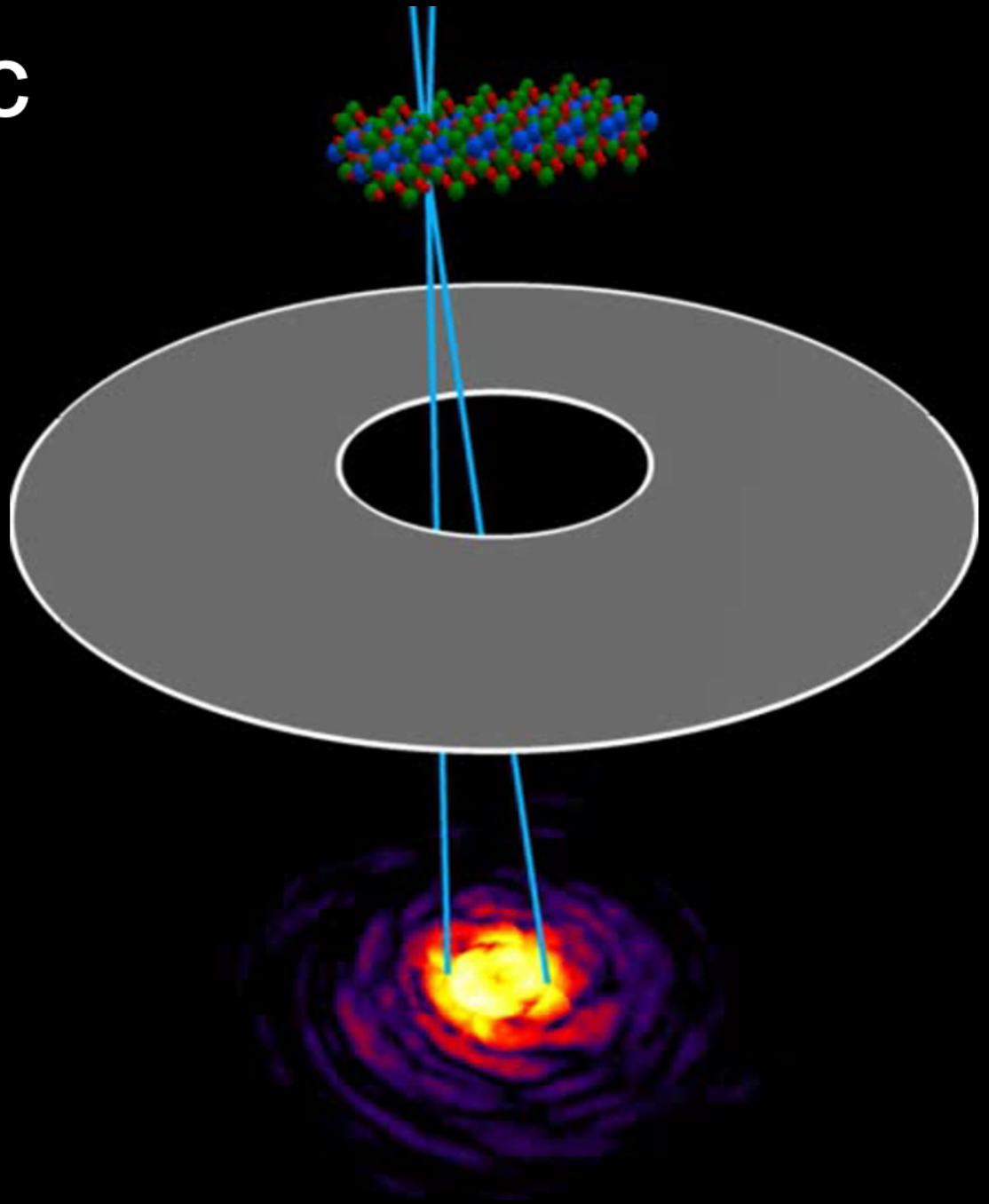




Phase plate interference pattern can generate linear interference.

C Ophus et al., Nature Communications 7, 10719 (2016)

Programmatic Visualization Examples

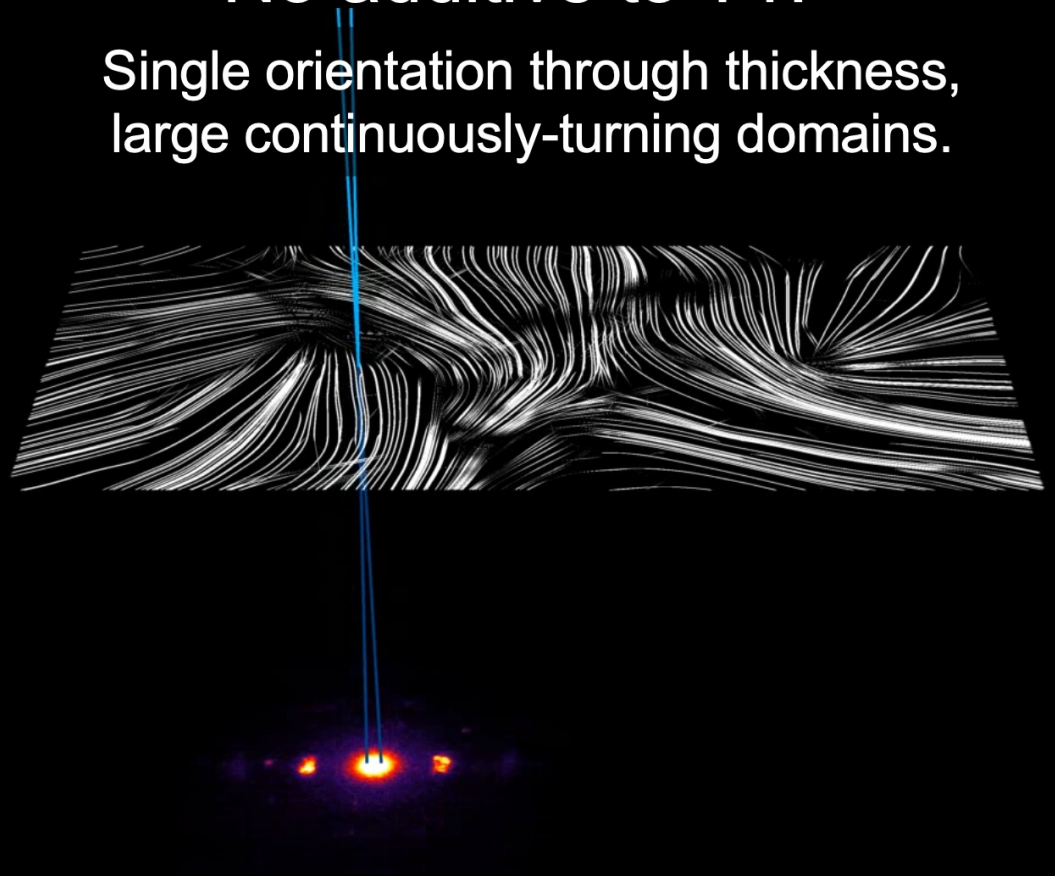


Programmatic Visualization Examples

Visualization of the two morphologies:

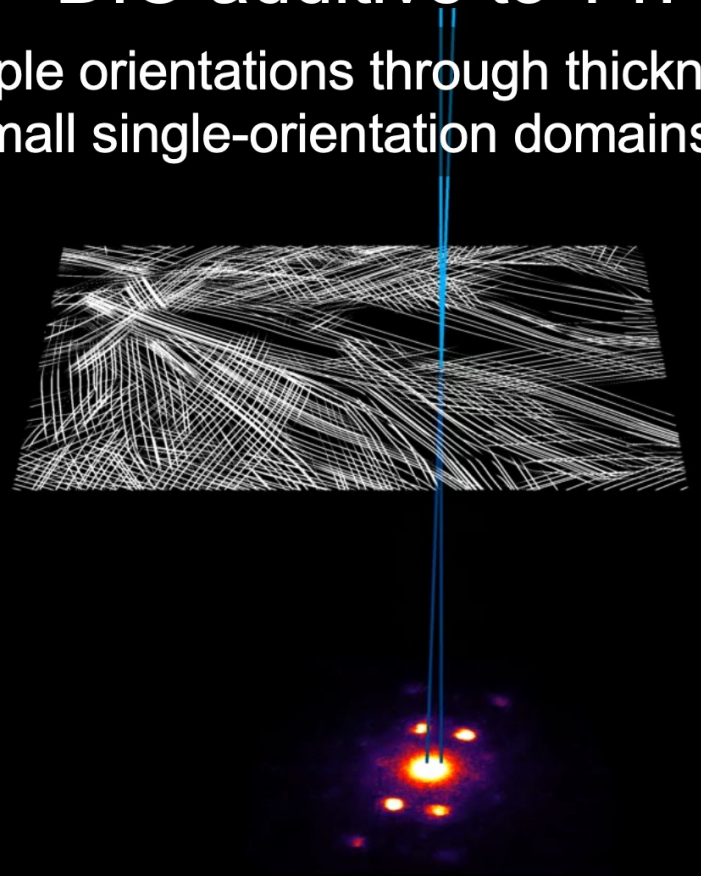
No additive to T1:

Single orientation through thickness,
large continuously-turning domains.



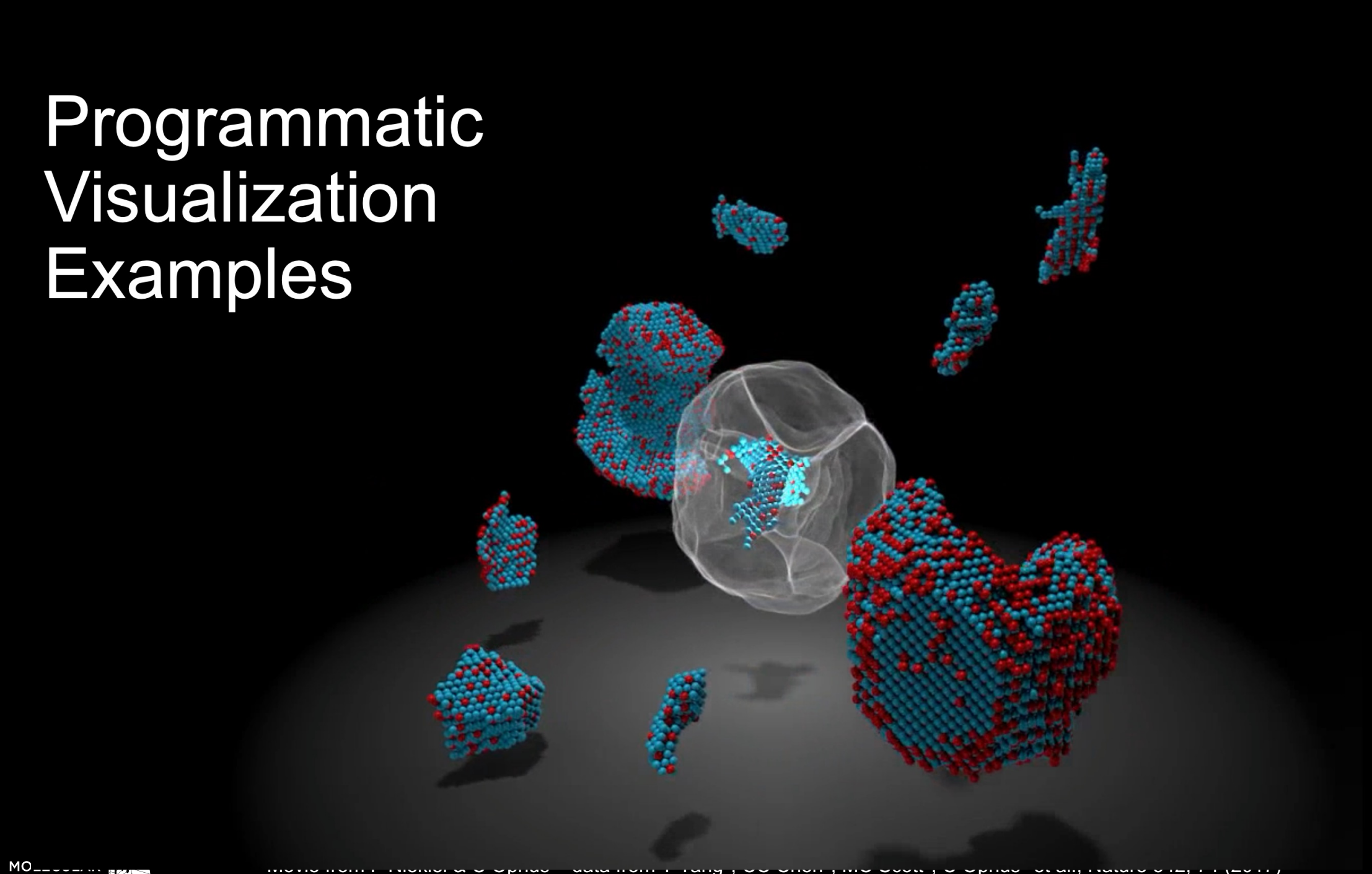
DIO additive to T1:

Multiple orientations through thickness,
small single-orientation domains.



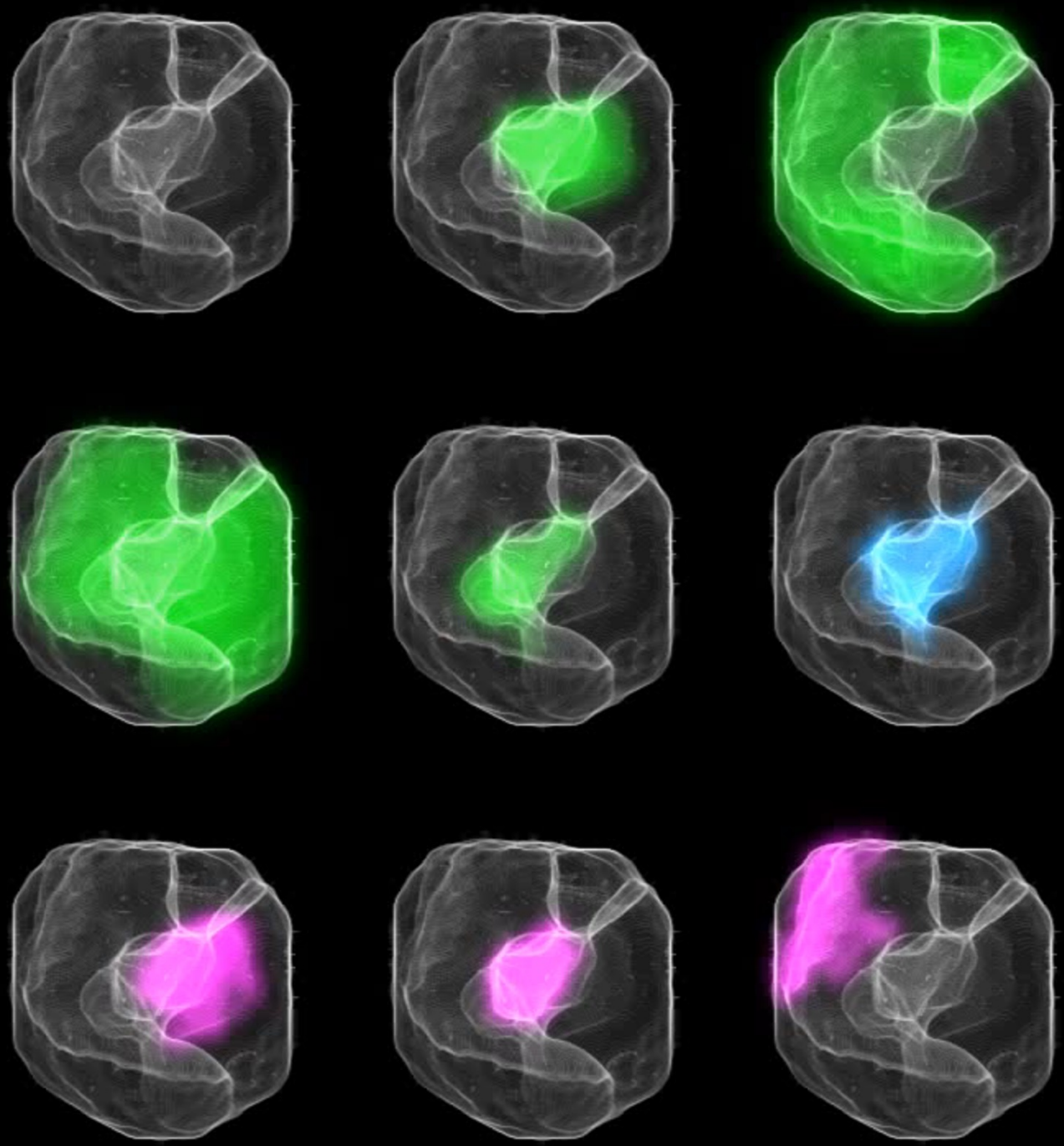
C Ophus*, O Panova* et al., Nature Materials 18, 860 (2019)

Programmatic Visualization Examples



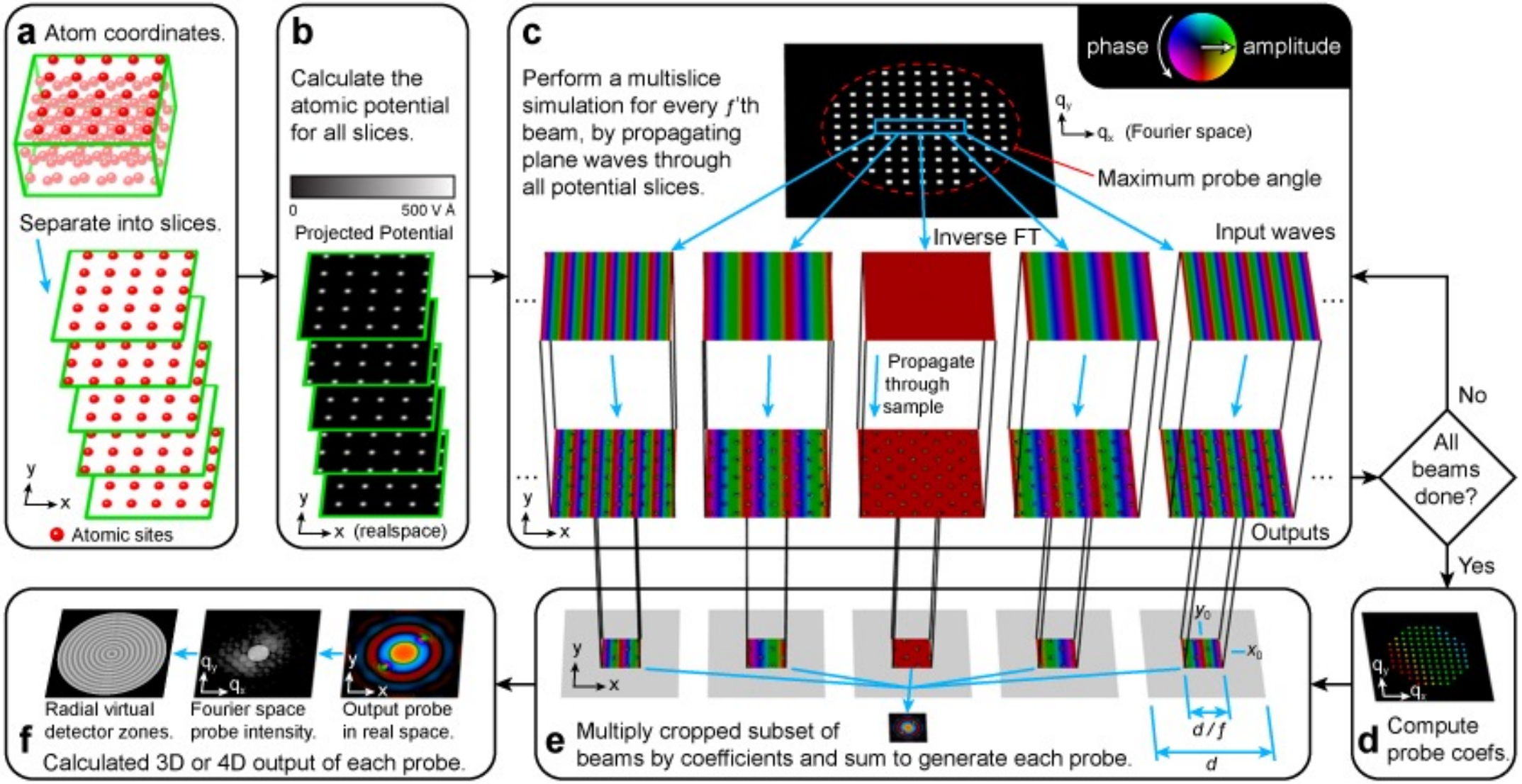
Movie from P. Nickels & J. Ephrussi — data from P. Tang, JC Chen, MC Scott, J. Ephrussi et al., Nature 512, 77 (2014)

Programmatic Visualization Examples



- Same data as previous slide, but visualized in a completely different way – grains become volumetric renders.

Basics – Vector Versus Raster (Bitmap)

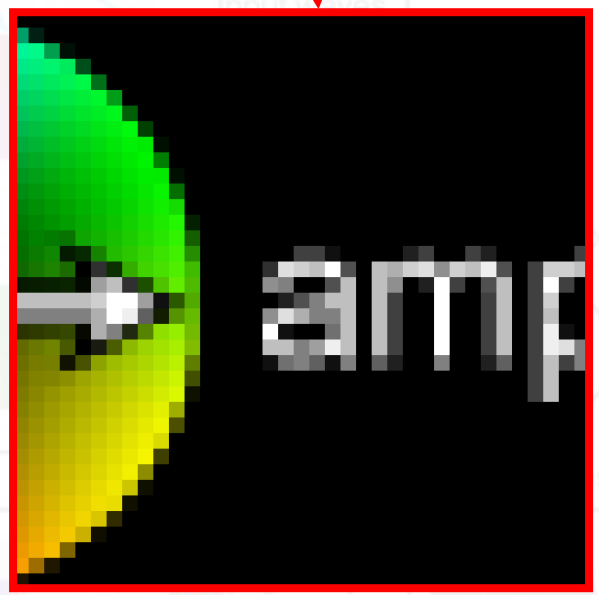


C Ophus, Advanced Structural and Chemical Imaging 3, 13 (2017).

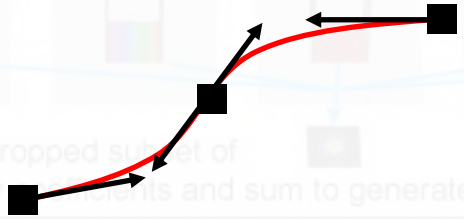
Basics – Vector Versus Raster (Bitmap)

Vector formats can easily be “rendered” into raster.

The reverse direction is not possible however!



Vector format – objects stored as vertices, curves, faces, patches …



Raster format – RGB values stored in a grid.

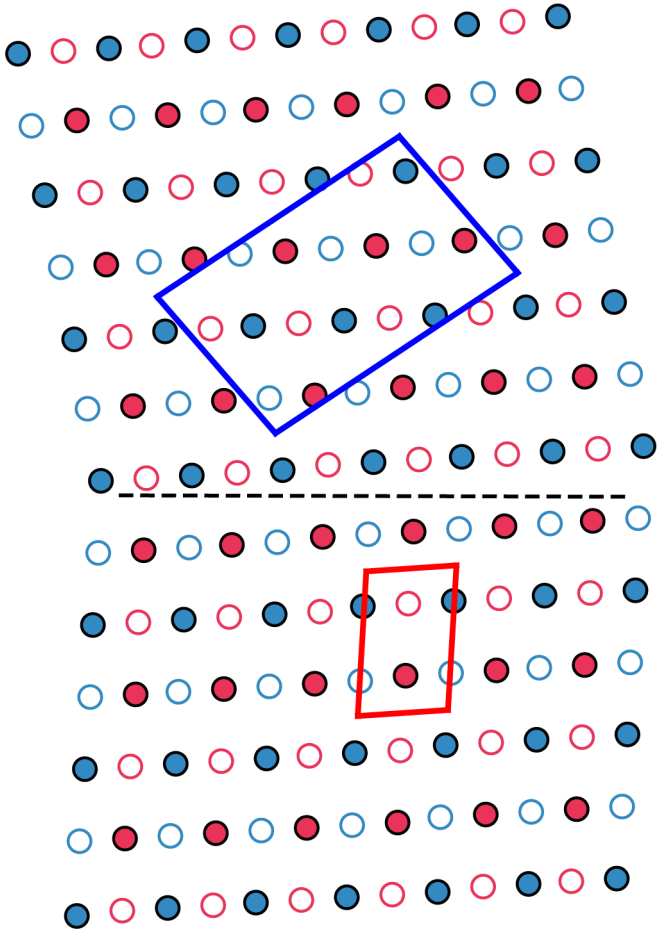
Basics – Vector and Raster File Formats

	Vector	Raster
ideal format	PDF (portable document format)	PNG (for lossless images) JPG (for lossy images)
runner up formats	SVG (scalable vector graphics) EPS (encapsulated post script)	TIFF (can contain metadata) MPG, MP4, AVI, WEBM
don't use formats	AI (adobe illustrator) CD (CorelDraw) EMF / WMF (Windows vectors) DOC, PPTX, Keynote DXF (or any other drafting format)	BMP, PSD, TGA, … GIF (Maybe the worst format ever? Terrible for animations too! Does support transparency …)

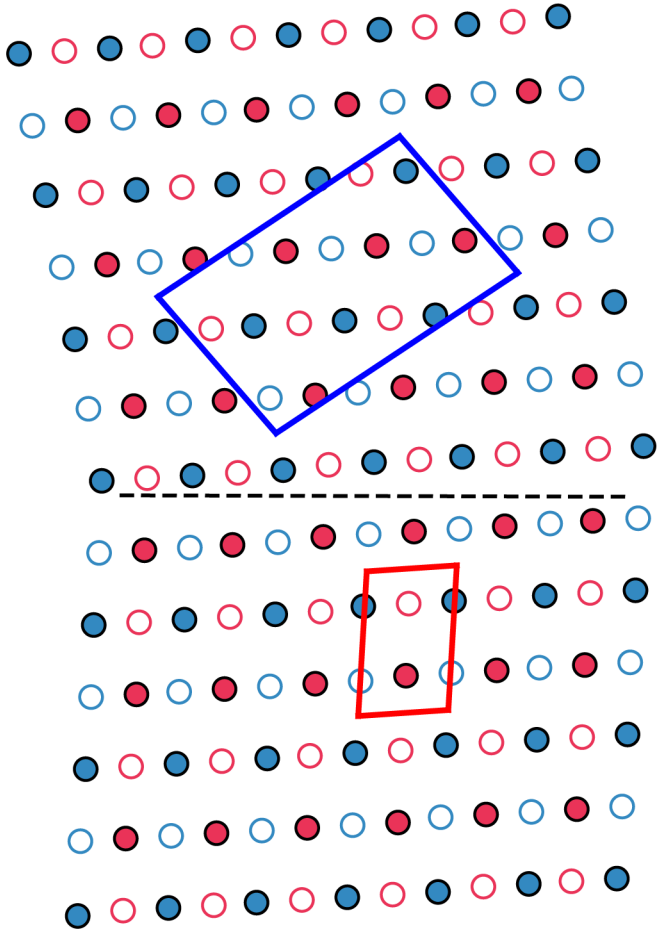
Basics – Vector and Raster Editing Software

	Vector		Raster	
Free programs	Inkscape	powerful, some quirks, SVG files	Gimp	powerful, duplicates most photoshop features
			www.photopea.com	
Good programs	Illustrator	high learning curve	Photoshop	powerful, easy to use
	CorelDraw	easy to learn		
Bad programs (for editing figures)	Microsoft Office Suite (Powerpoint)	low quality output, inconsistent support, missing most advanced tools, cannot control resolution		
	OfficeLibre (Open Office)	low quality output, inconsistent support, missing most advanced tools, cannot control resolution		

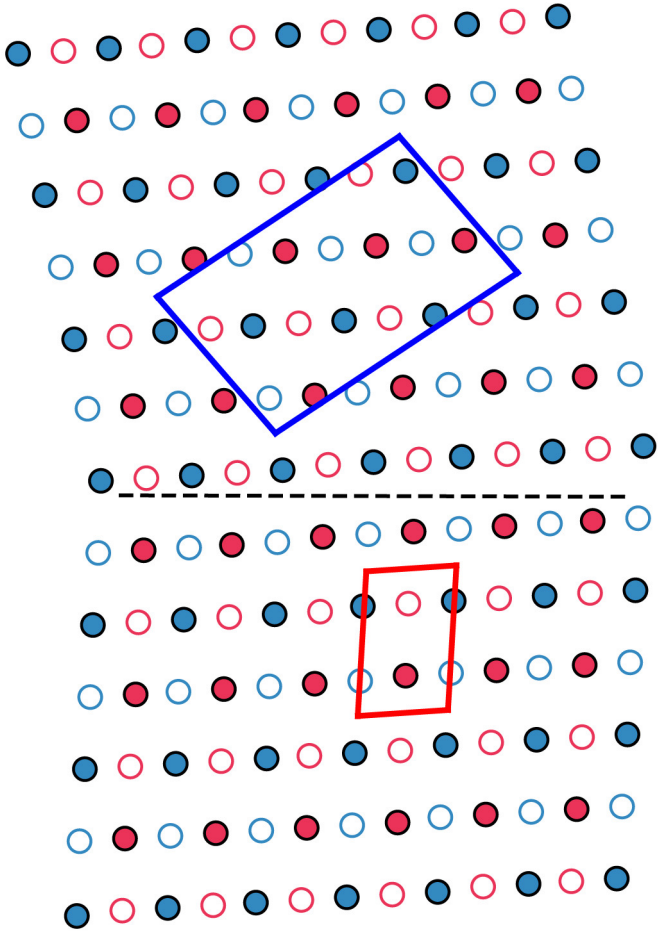
Basics – Vector and Raster File Sizes



PDF – 21 kilobytes



PNG – 187 kilobytes

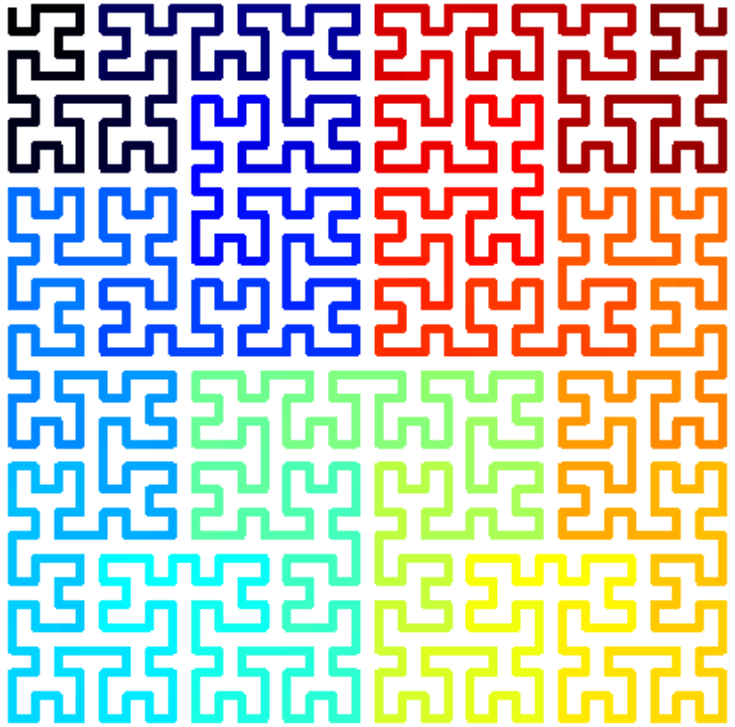


JPG – 352 kilobytes

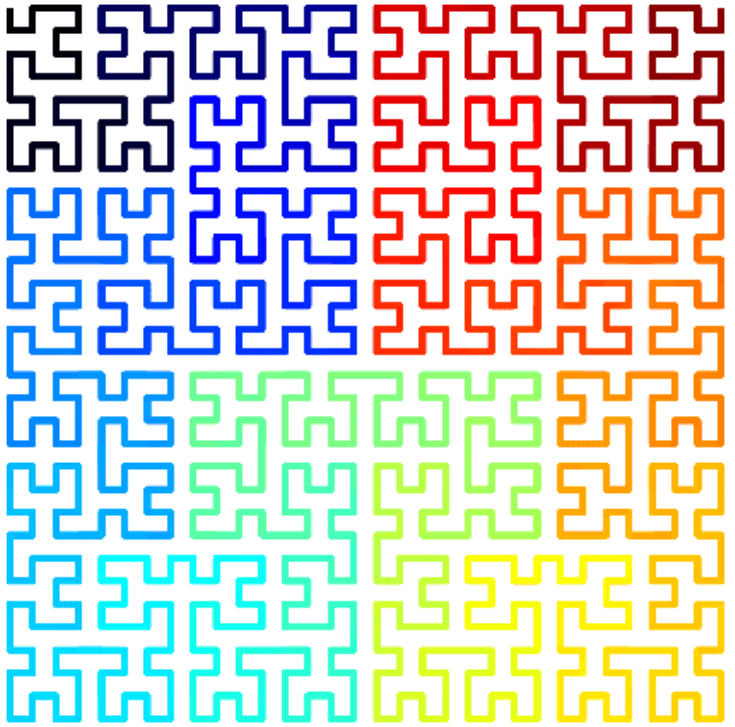
For relatively simple vector objects, PDF is the best (in general vector is also better than raster!)

X Chen et al., Physical Review Materials 4, 103611 (2020).

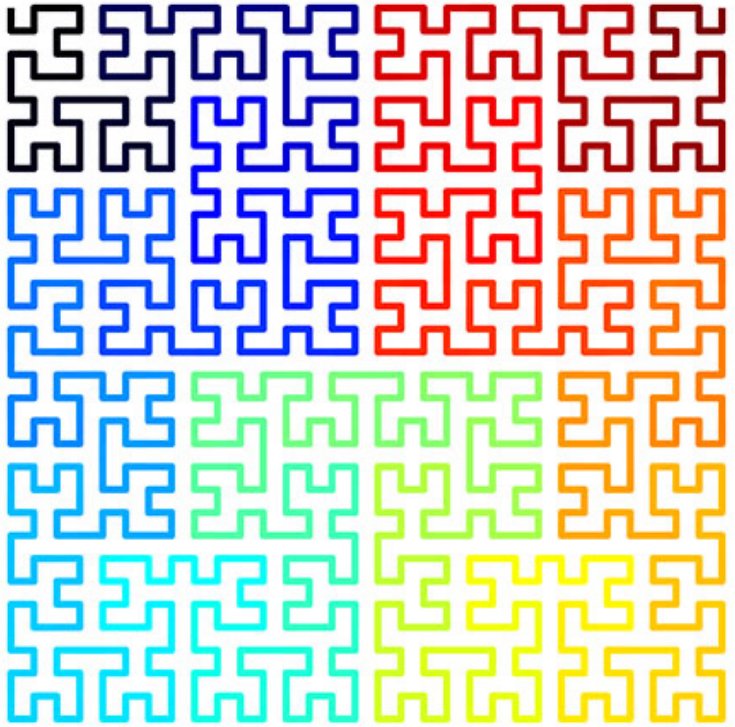
Basics – Vector and Raster File Sizes



PDF – 117 kilobytes



PNG – 15 kilobytes

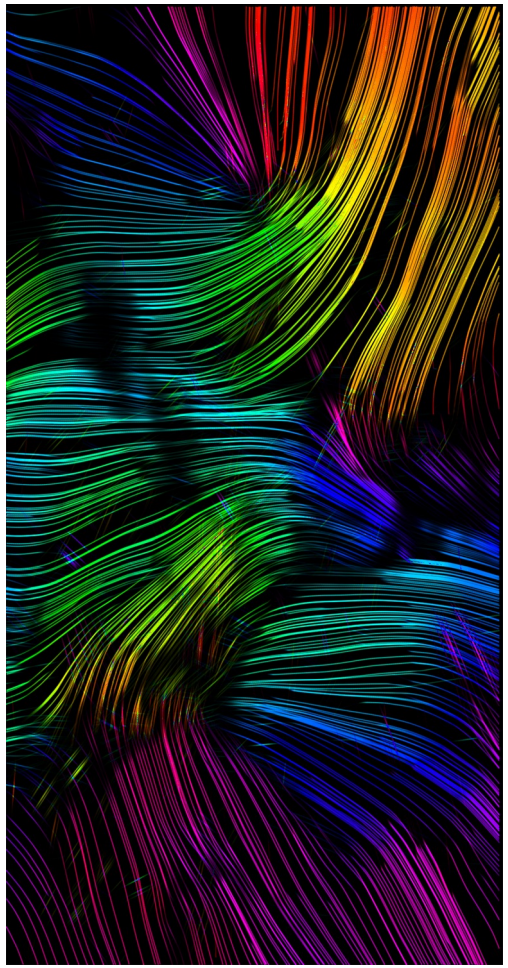


JPG – 123 kilobytes

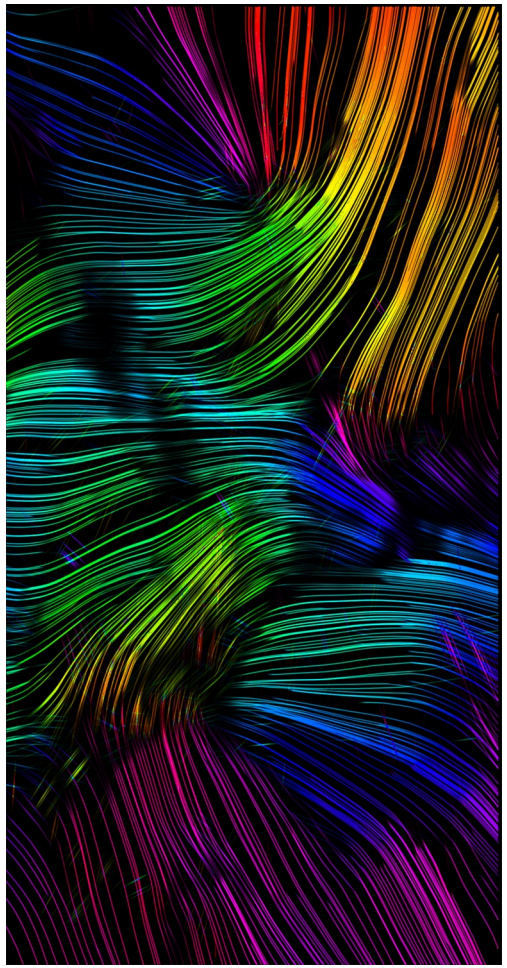
For images with few colors (<256), flat square regions, PNG compression is very good.

Hilbert patterns from – A Velazco et al., Ultrmiacroscopy 215, 113021 (2020).

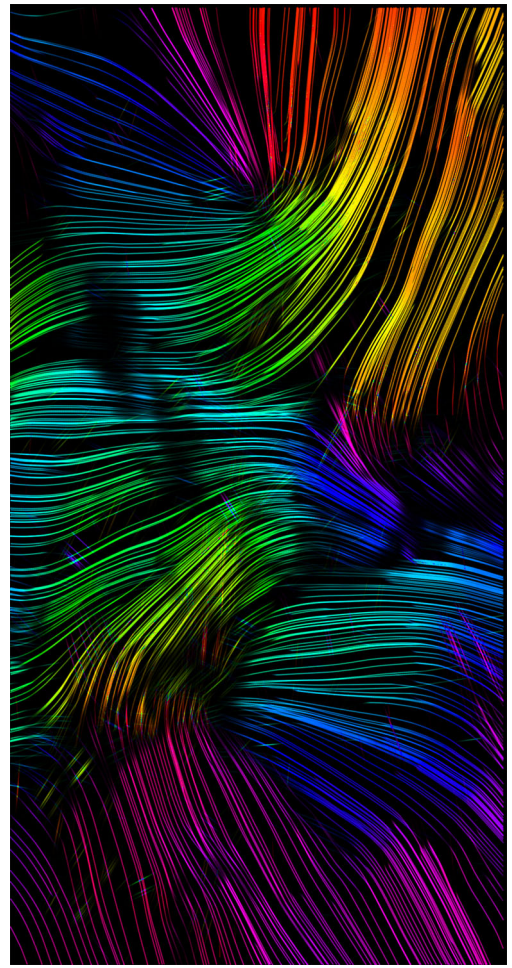
Basics – Vector and Raster File Sizes



PDF – 10 851 kilobytes



PNG – 1 710 kilobytes

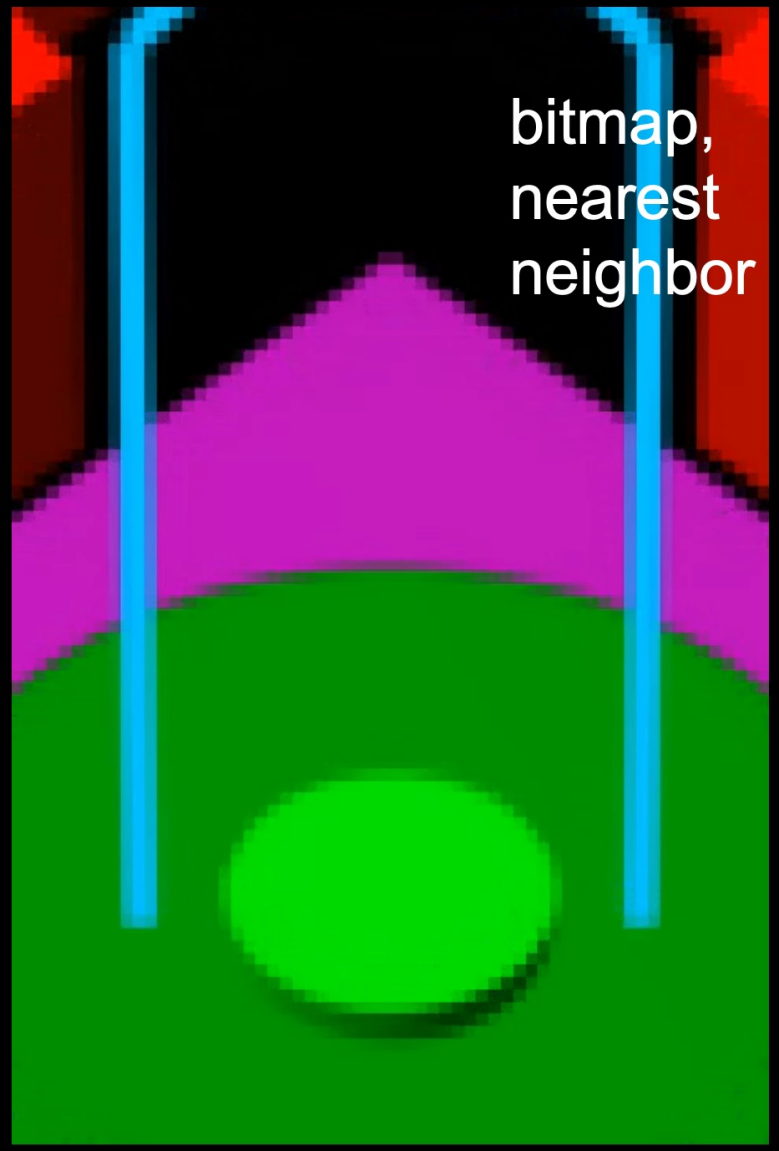


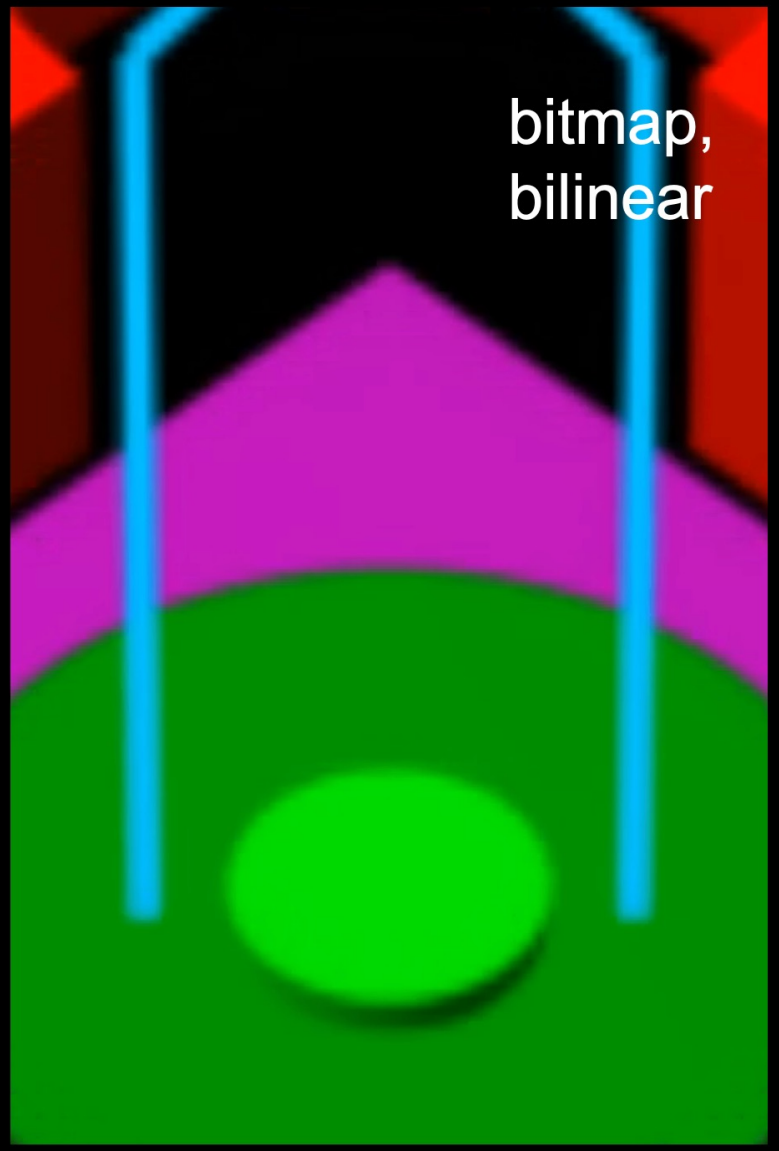
JPG – 576 kilobytes

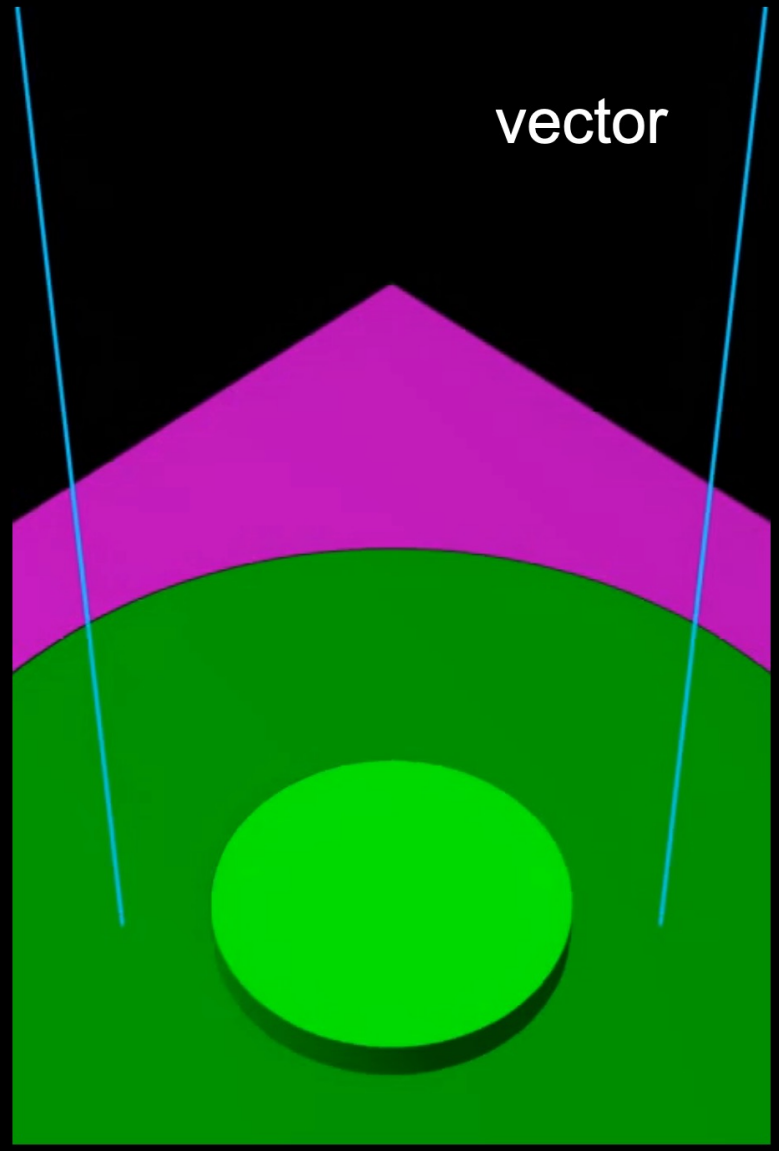
If the image contains many “high energy” regions, many colors, lossy compression such as JPG is best.

C Ophus*, O Panova*, et al., Nature Materials 18, 860 (2019).

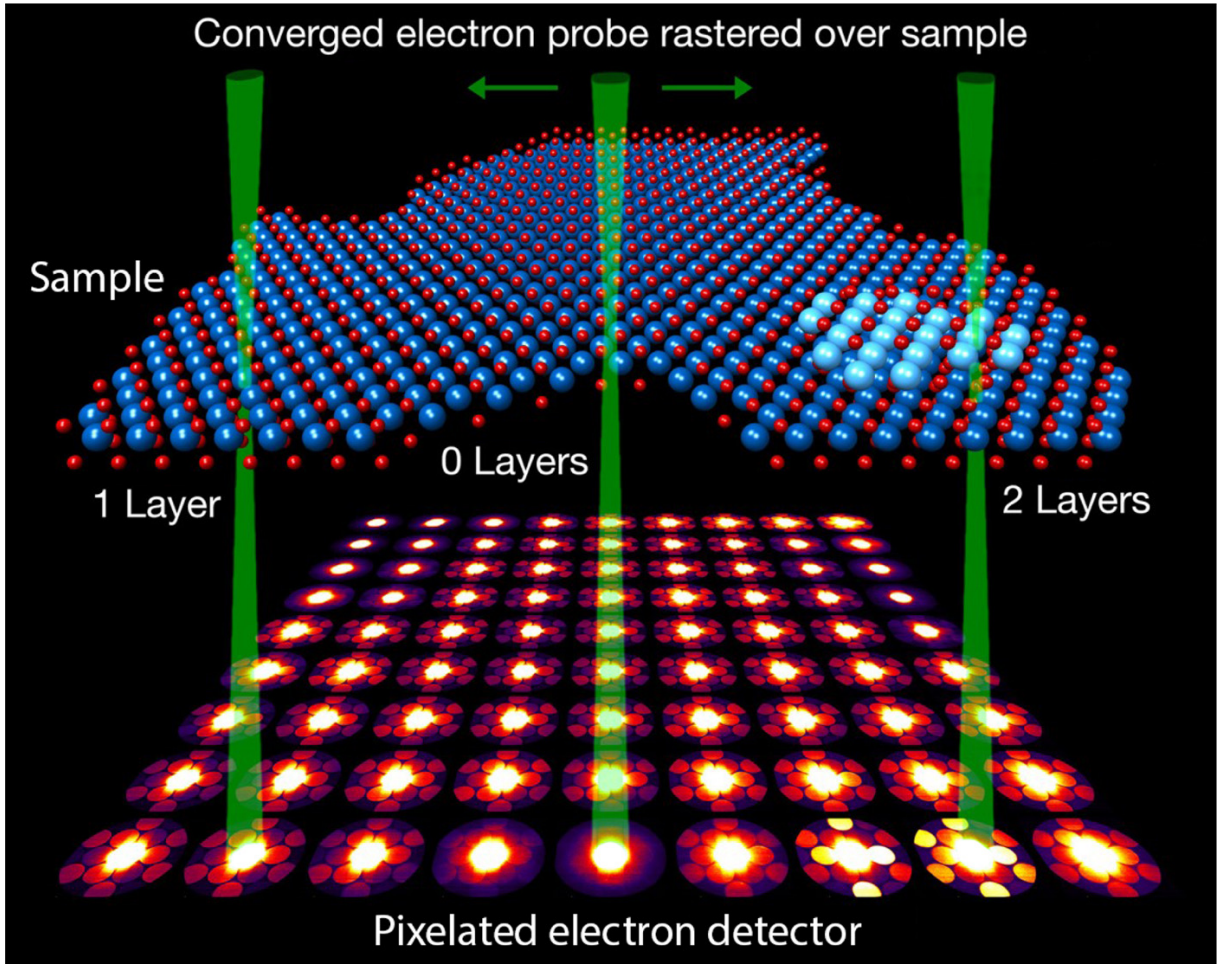
Basics – Zooming into Vector and Raster







Creating Visualization / Figures With Programming



What Language Should You Use For Figures?

Advantages

- Very large number of functions, libraries, toolboxes built in, even more on file exchange.
- Easy to use, quite fast (though not C speed)
- Very powerful 2D and 3D plotting tools.
- No compiler needed (interpreted).
- Widely used in scientific research.

- Very large number of functions and libraries included and available online.
- Easy to use, easy to extend, fast dev.
- Reasonable plotting (e.g. Matplotlib).
- No compiler needed (interpreted).
- Widely used in scientific research.

Disadvantages

- Expensive license (cheap for academics).
- Poor support for GUIs, Java front end crashes.
- Slow development relative to open source.

- Slow (offset by Cython, Numba, PyTorch).
- Poor memory efficiency.
- Unreliable versioning ... e.g. Python 2 vs 3

What Language Should You Use For Figures?

Jupyter
King of the Data
Visualization Pantheon?

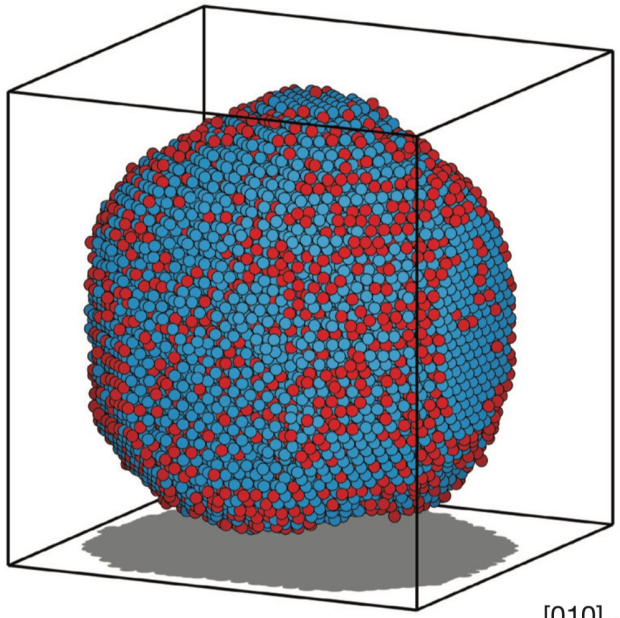
MATLAB
God-Emperor of the Data
Visualization Universe

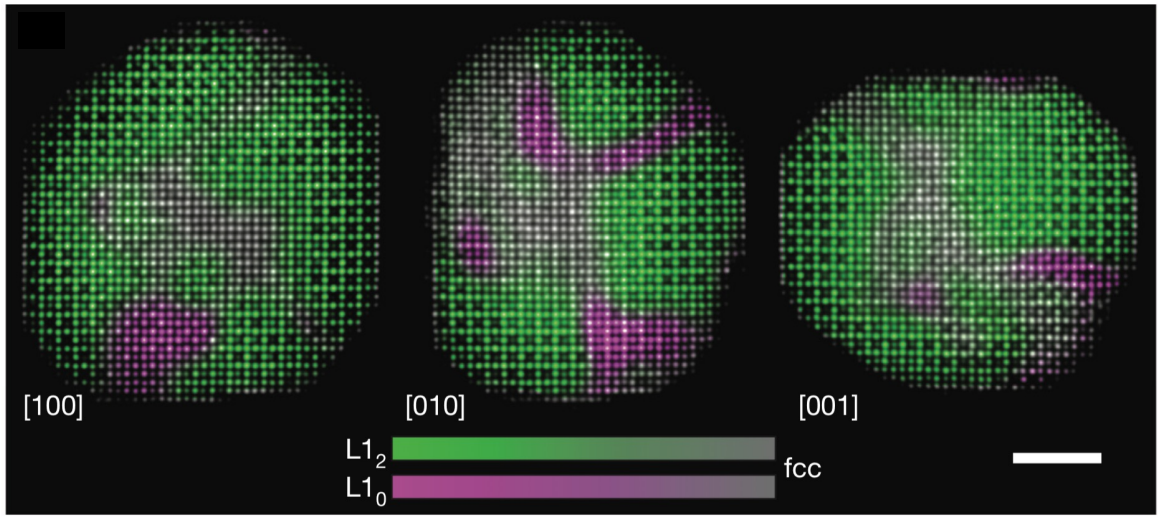
What Language Should You Use For Figures?

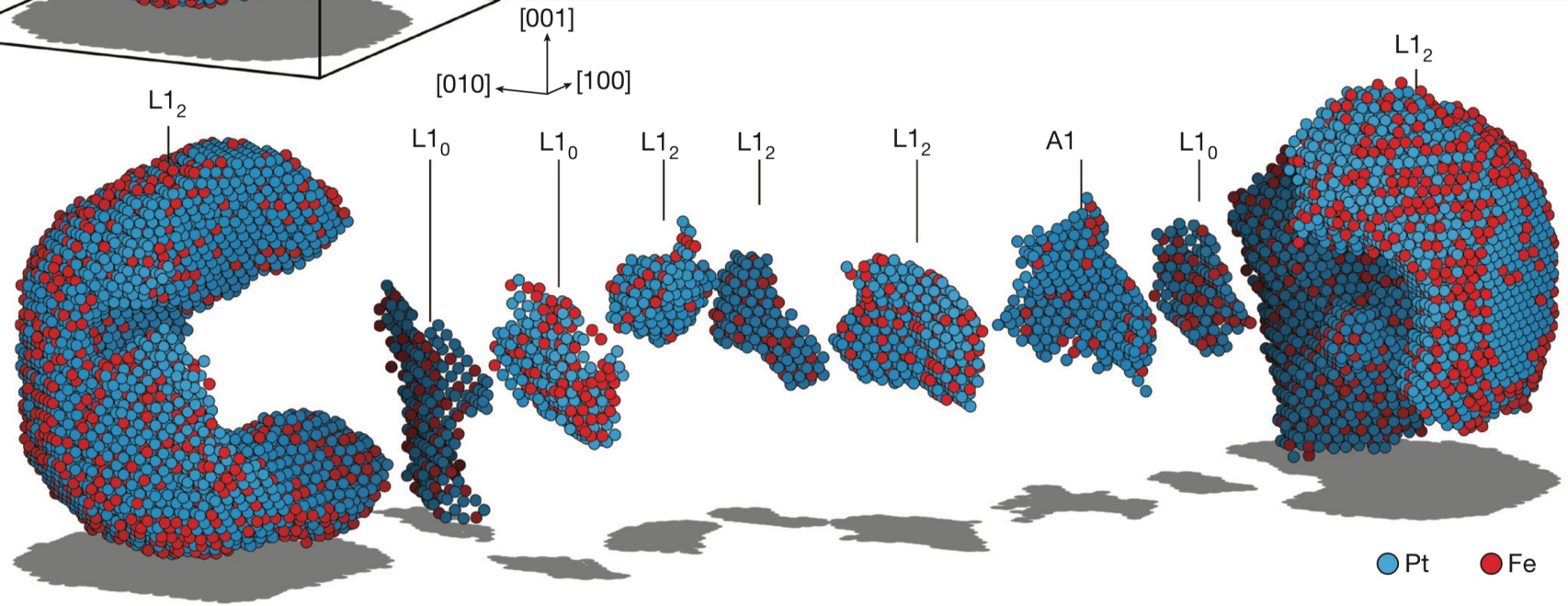
- This talk will show examples from matlab.
- Github repo contains many matlab examples.
- Our practical workshop will use python in Colab.
- Will need a google login: **tinyurl.com/progviz**

github.com/cophus/**Programmatic_Figures_Tutorial**

Common Vis Task – Plotting 3D Atoms





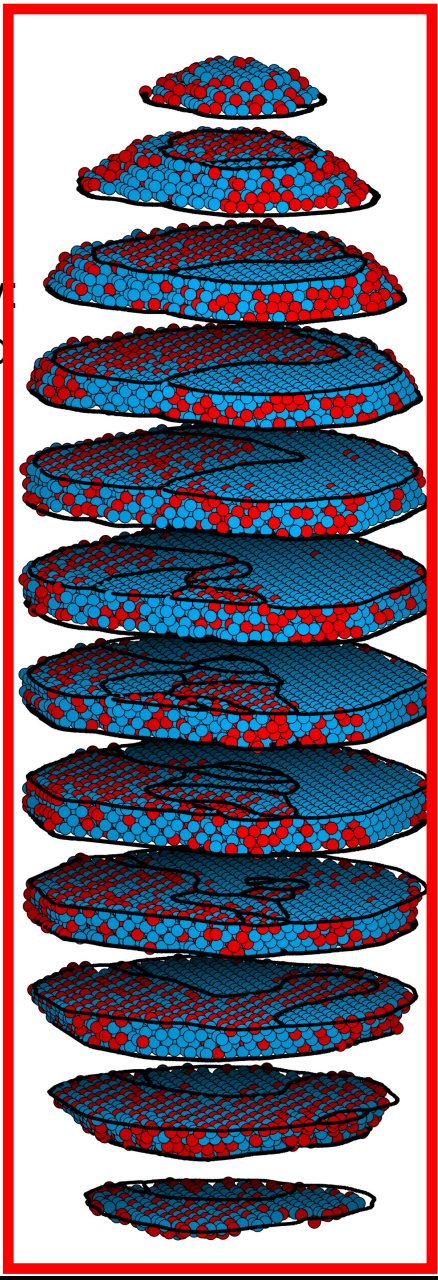


atomic electron tomography: position and species

- Fe
- Pt

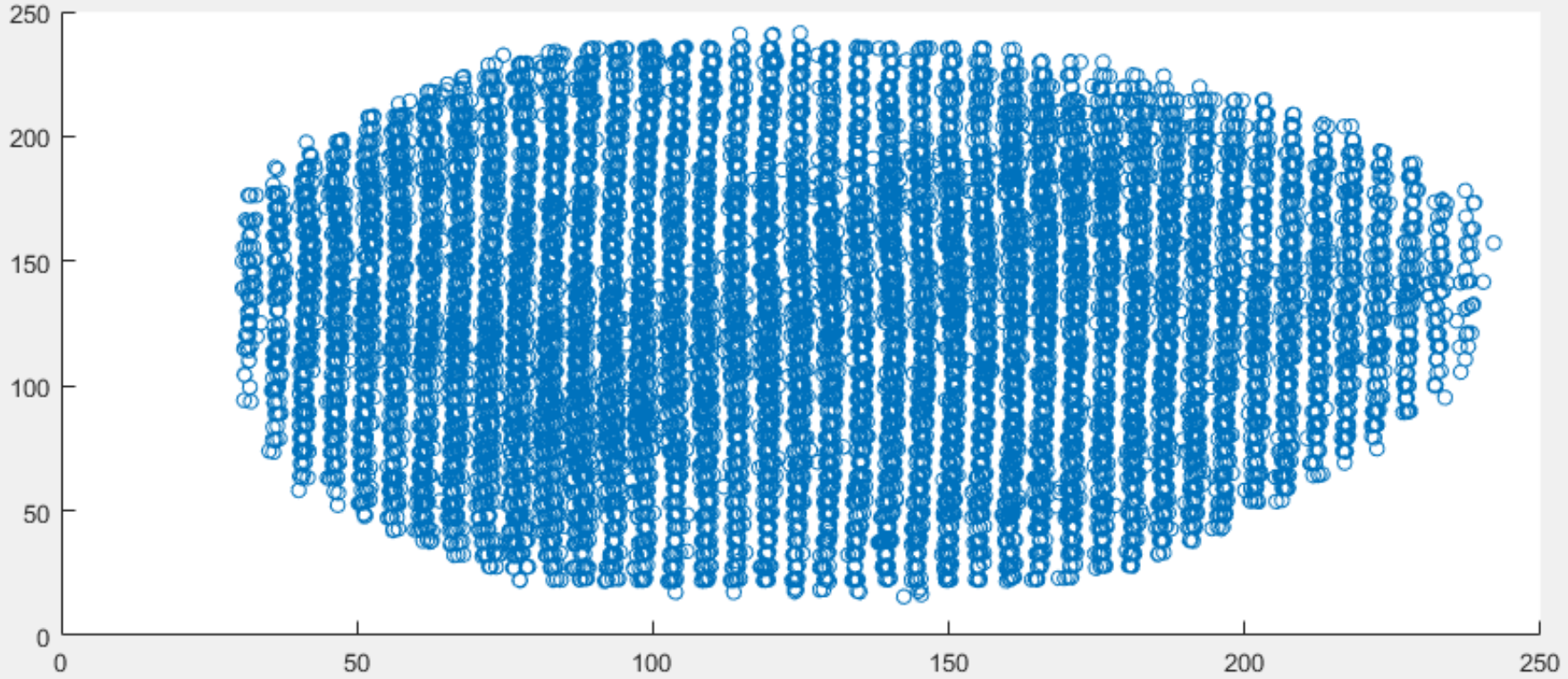
GBs: $L1_2$ and $L1_0$ grains

Examples drawing this panel



Y Yang*, CC Chen*, MC Scott*, C Ophus*, et al., Nature 542, 74 (2017)

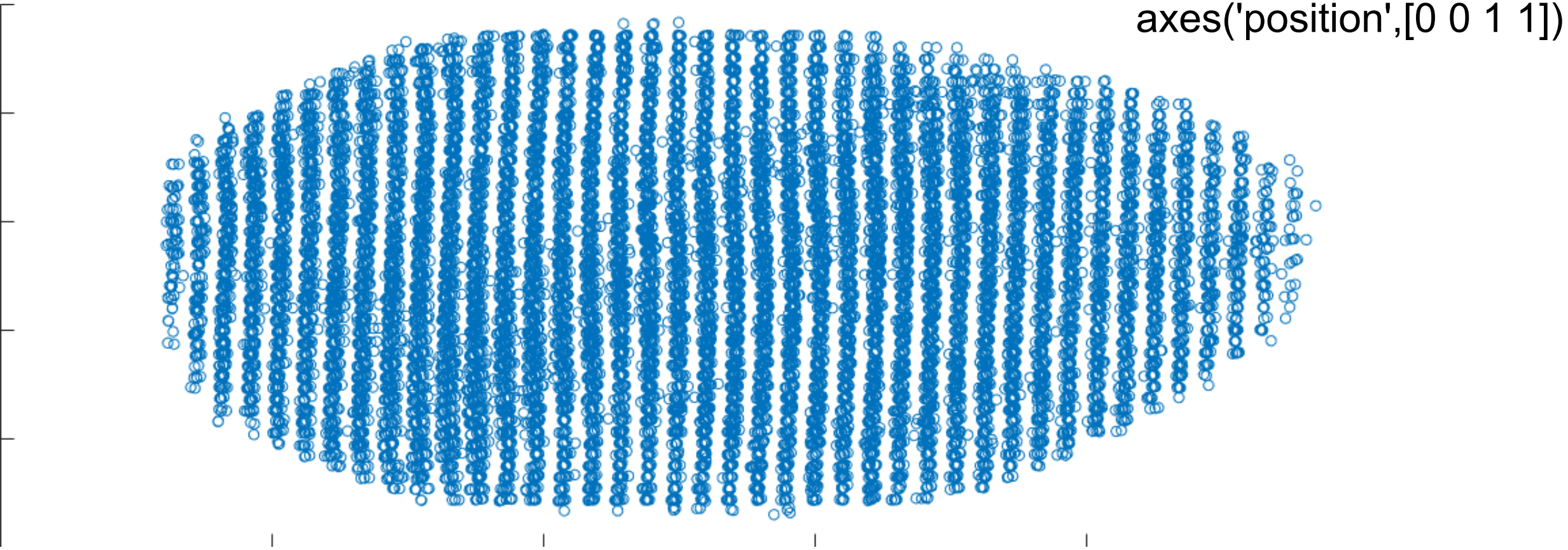
Common Visualization Task – 3D Atoms




```
scatter3( ...
    atomPos(:,1),...
    atomPos(:,2),...
    atomPos(:,3));
```

A 3D scatter plot of the coordinates, default appearance – pretty ugly!

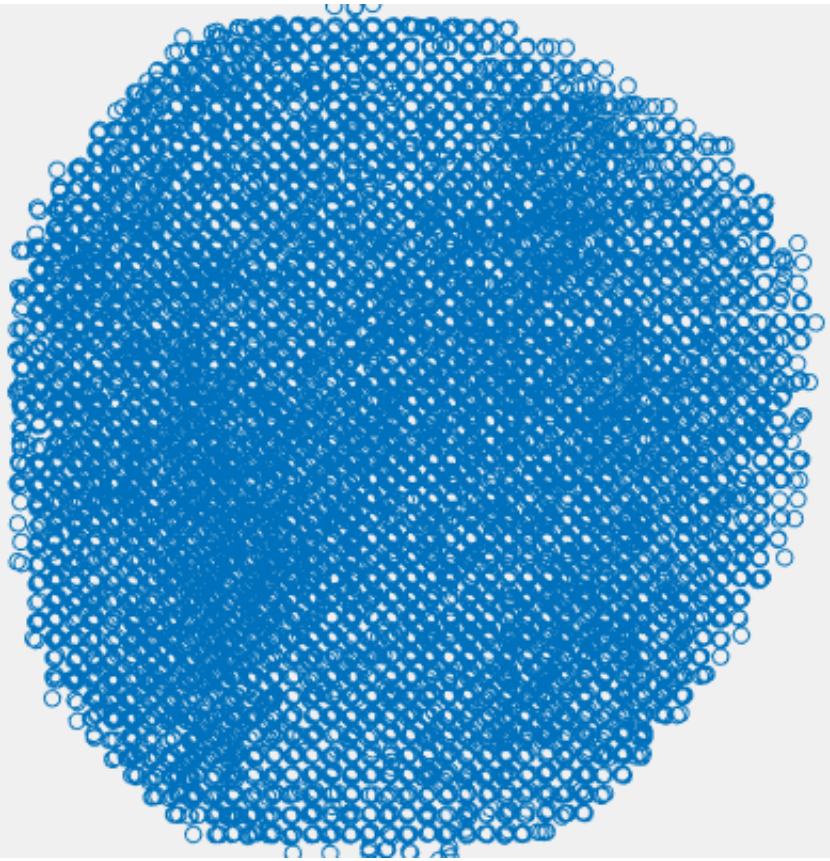
Common Visualization Task – 3D Atoms



This “axes” command will stretch the axes to fill the field of view.

[x_origin y_origin x_width y_width] – all values normalized from 0 to 1.

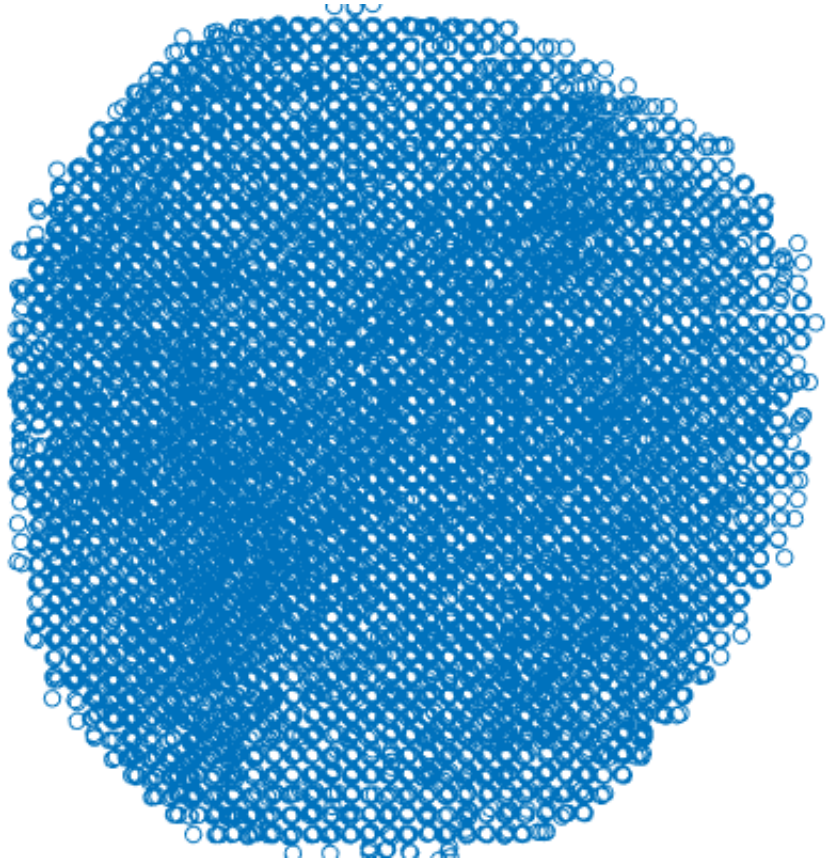
Common Visualization Task – 3D Atoms



axis equal off

Set the aspect ratio of the 3 directions to be equivalent, hide the axes objects.

Common Visualization Task – 3D Atoms



```
set(gcf,'color','w');
```

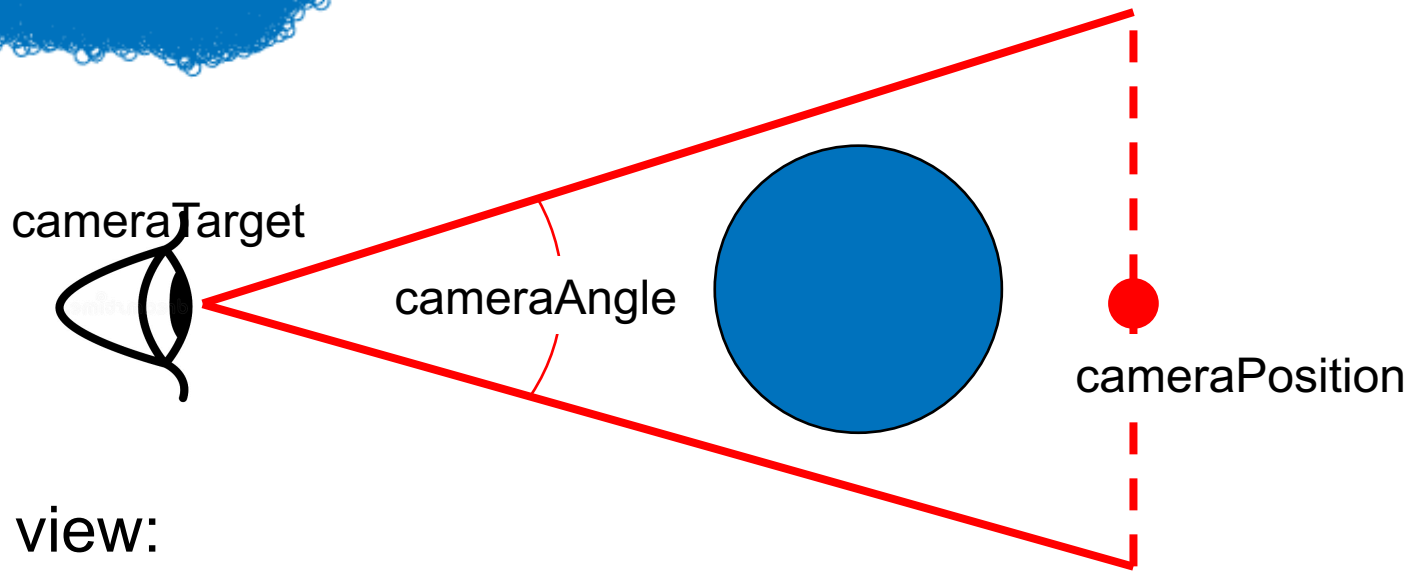
or equivalently

```
set(gcf,'color’,[1 1 1]);
```

“gcf” = get current figure handle, set its color to white (red = 1, green = 1, blue = 1)

Common Visualization Task – 3D Atoms

```
camtarget(cameraTarget)
campos(cameraTarget …
                + cameraPosition);
camva(cameraAngle)
camproj('perspective')
```

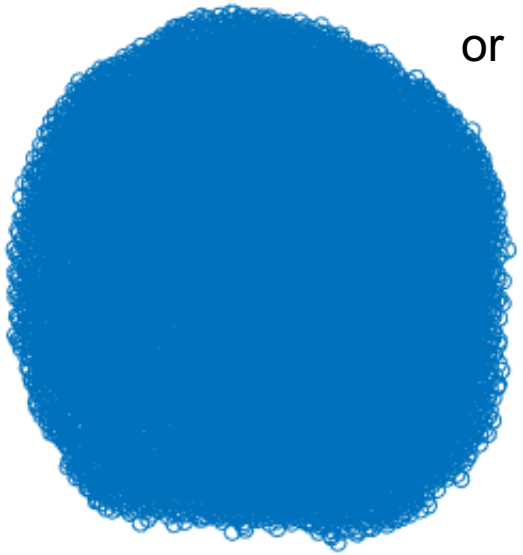


Set up a rendering camera for a 3D view:

Common Visualization Task – 3D Atoms

```
for a0 = 1:3
        atomPos(:,a0) = atomPos(:,a0) - mean(atomPos(:,a0));
end

or              atomPos = atomPos - mean(atomPos, 1);
```

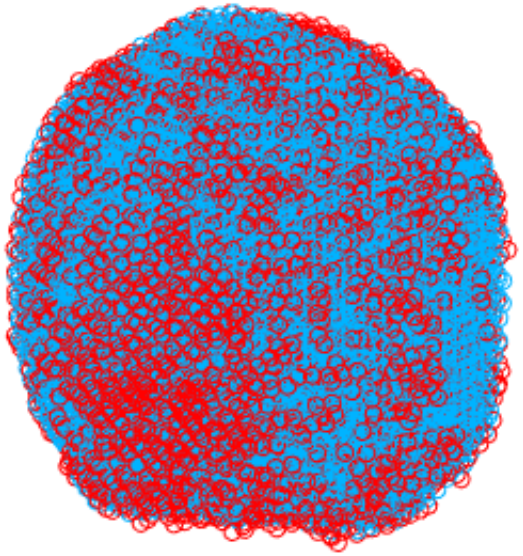


Center the group of atoms on (x,y,z) = (0,0,0) by subtracting the mean position.

Common Visualization Task – 3D Atoms

Pt

Fe



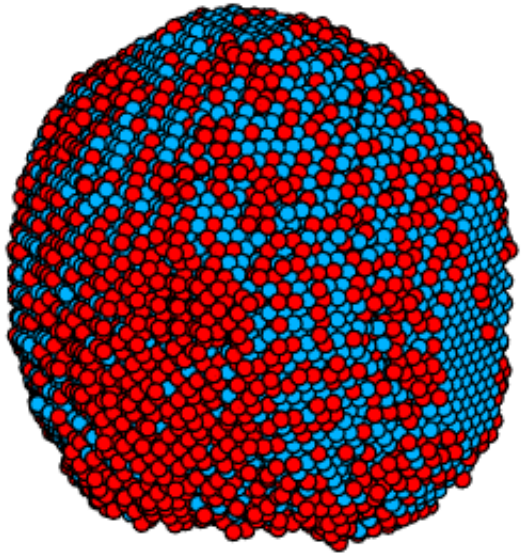
```
scatter3( ...
    atomPos(:,1),...
    atomPos(:,2),...
    atomPos(:,3),...
    ones(numberAtoms,1)*atomSize,...
    atomRGB);
```

Give a unique color (and marker size) to each atom, in order to identify Fe and Pt sites.

Common Visualization Task – 3D Atoms

Pt

Fe

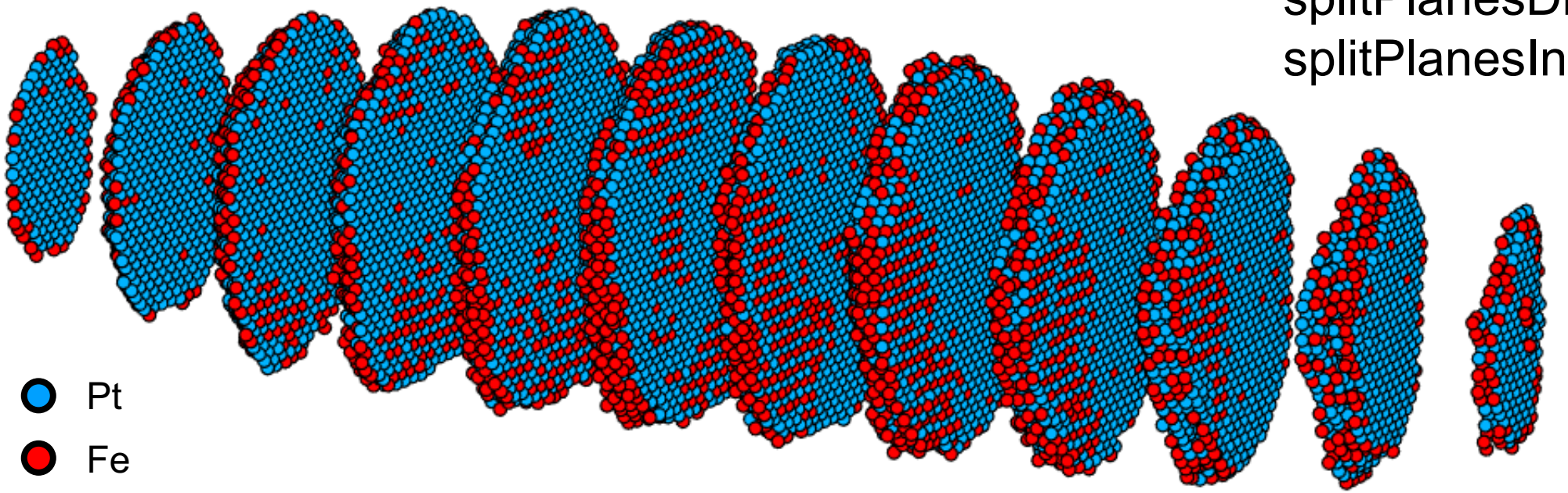


```
scatter3( ...
    atomPos(:,1),...
    atomPos(:,2),...
    atomPos(:,3),...
    ones(numberAtoms,1)*atomSize,...
    atomRGB,...
    'filled','marker',’o’,…
'linewidth',atomLinewidth,...
    'markeredgecolor',atomColorEdges);
```

Update the marker appearance, filled circles, black edges, etc.

Common Visualization Task – 3D Atoms

splitPlanesBunch = 2;
splitPlanesDistance = 70;
splitPlanesIndex = 2;

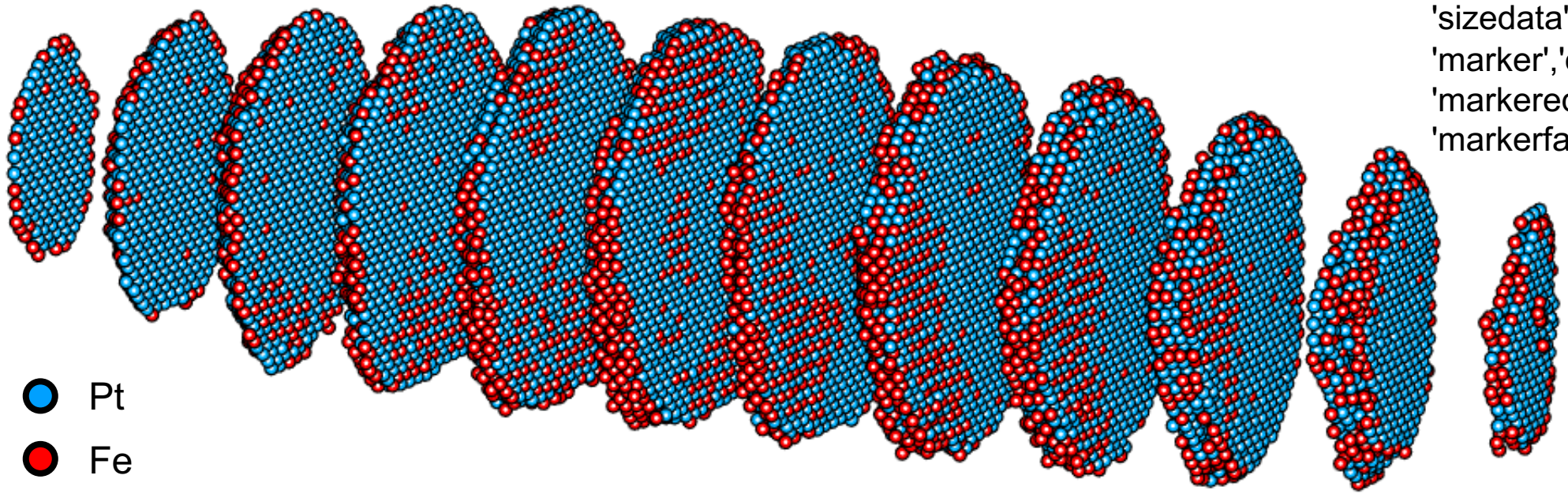


Now we're moving beyond simple scatter plots – atoms are "bunched" up into sets of 4 atomic planes, and then each plane is moved 70 plotting units apart from the neighboring planes. This operation performed along dimension 2 (y axis).

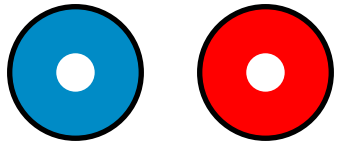
This plotting allows us to see the rich internal structure of this experiment – note the alternating Fe-Pt atomic planes.

Common Visualization Task – 3D Atoms

```
scatter3( ...
        atomPos(:,1),...
        atomPos(:,2),...
        atomPos(:,3),...
        'sizedata',atomSizeTint,...
        'marker','o',...
        'markeredgecolor','none',...
        'markerfacecolor',atomColorTint);
```

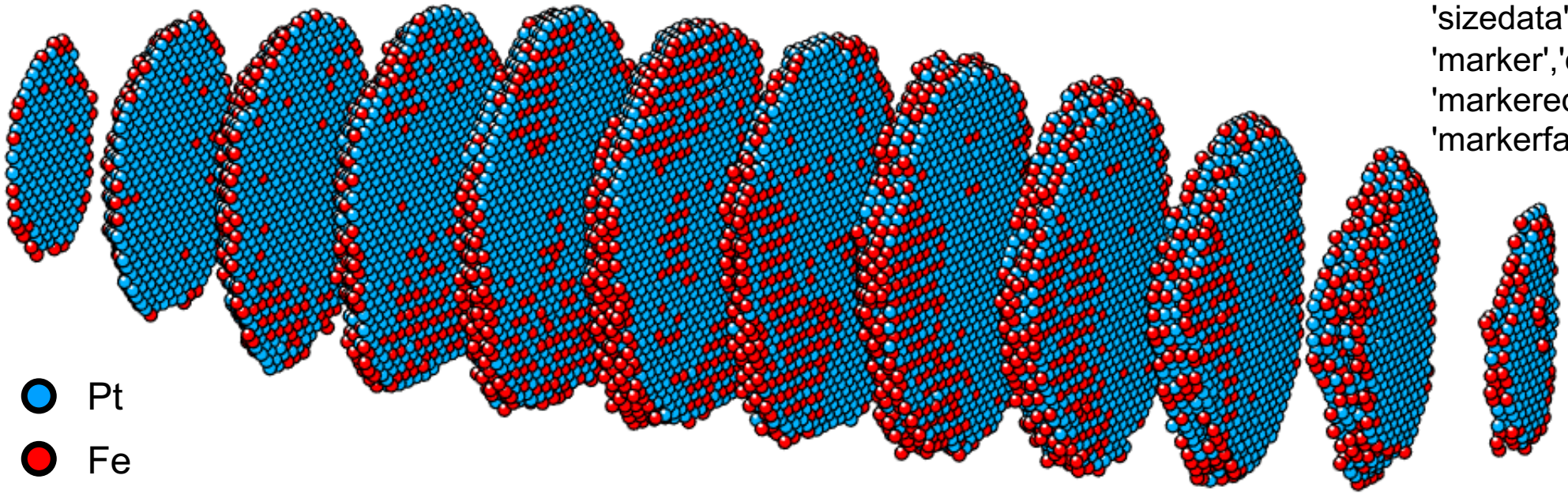


Let’s make a fake specular reflection spot:

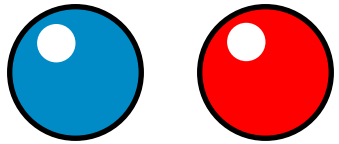


Common Visualization Task – 3D Atoms

```
scatter3( ...
        atomPos(:,1) + shiftTint(1),...
        atomPos(:,2) + shiftTint(2),...
        atomPos(:,3) + shiftTint(3),...
        'sizedata',atomSizeTint,...
        'marker','o',...
        'markeredgecolor','none',...
        'markerfacecolor',atomColorTint);
```

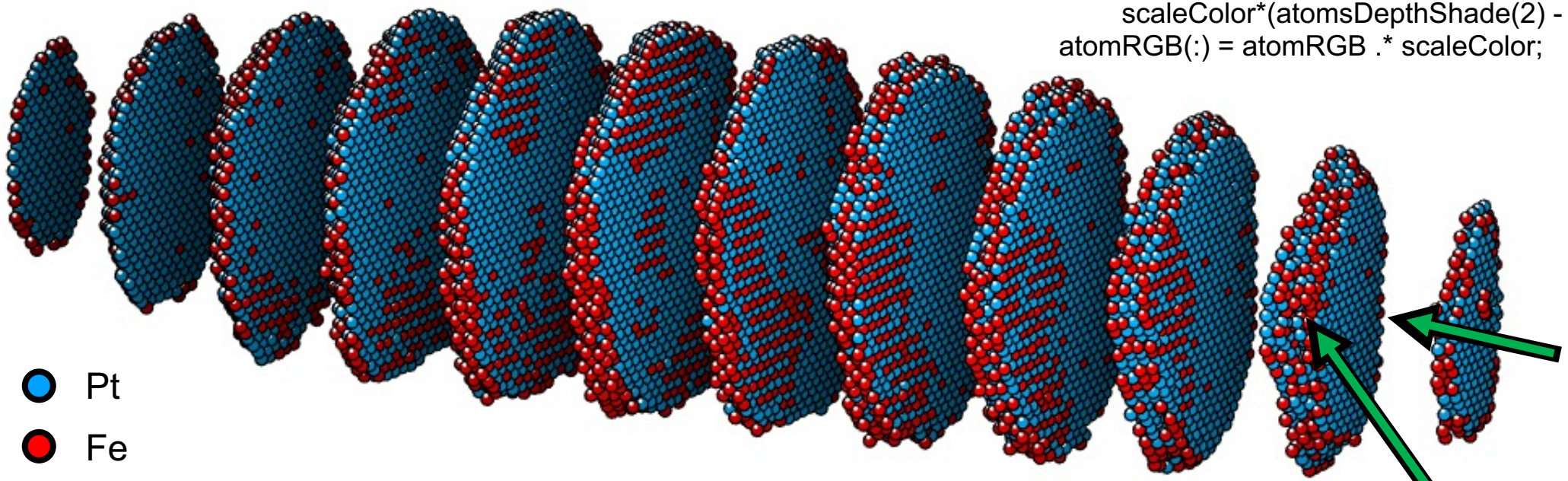


Let’s make a fake specular reflection spot:



Common Visualization Task – 3D Atoms

```
scaleColor = distCamera - min(distCamera);
scaleColor(:) = scaleColor / max(scaleColor);
scaleColor(:) = 3*scaleColor(:).^2 - 2*scaleColor(:).^3;
scaleColor(:) = scaleColor.^atomsDepthPower;
scaleColor = atomsDepthShade(1) + ...
      scaleColor*(atomsDepthShade(2) - atomsDepthShade(1));
atomRGB(:) = atomRGB .* scaleColor;
```



Pt

Fe

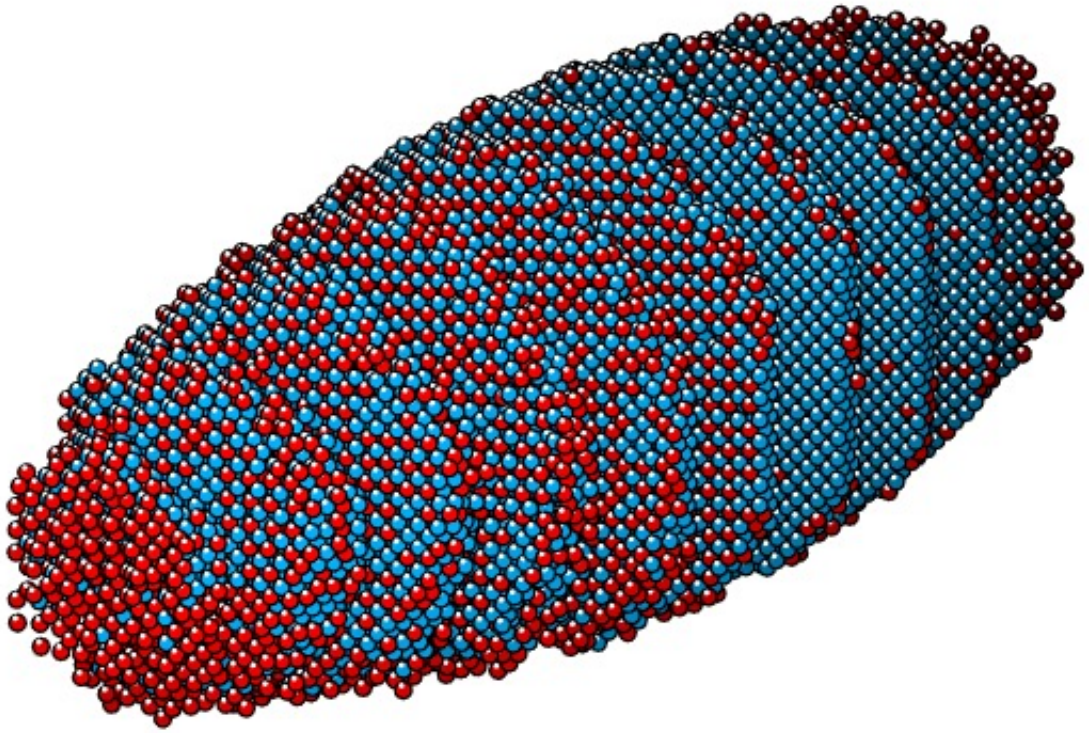
Sites far from the camera are shaded to a darker color.

Sites closest to the camera are left alone.

This is a subtle effect to create a 3D view – “depth cueing”

Common Visualization Task – 3D Atoms

splitPlanesIndex = 1

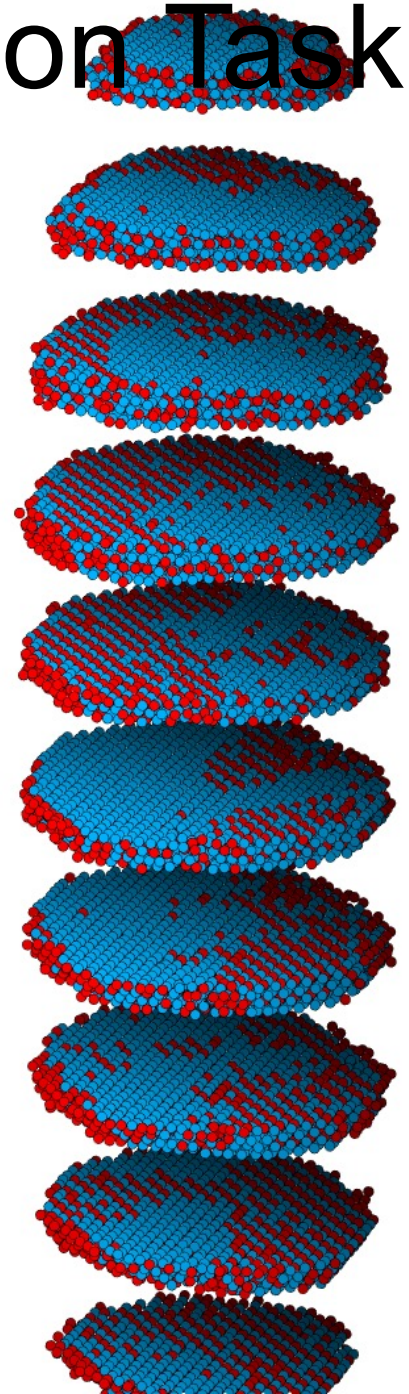


Pt

Fe

Splitting apart the particle along the 1st dimension.

Common Visualization Task – 3D Atoms

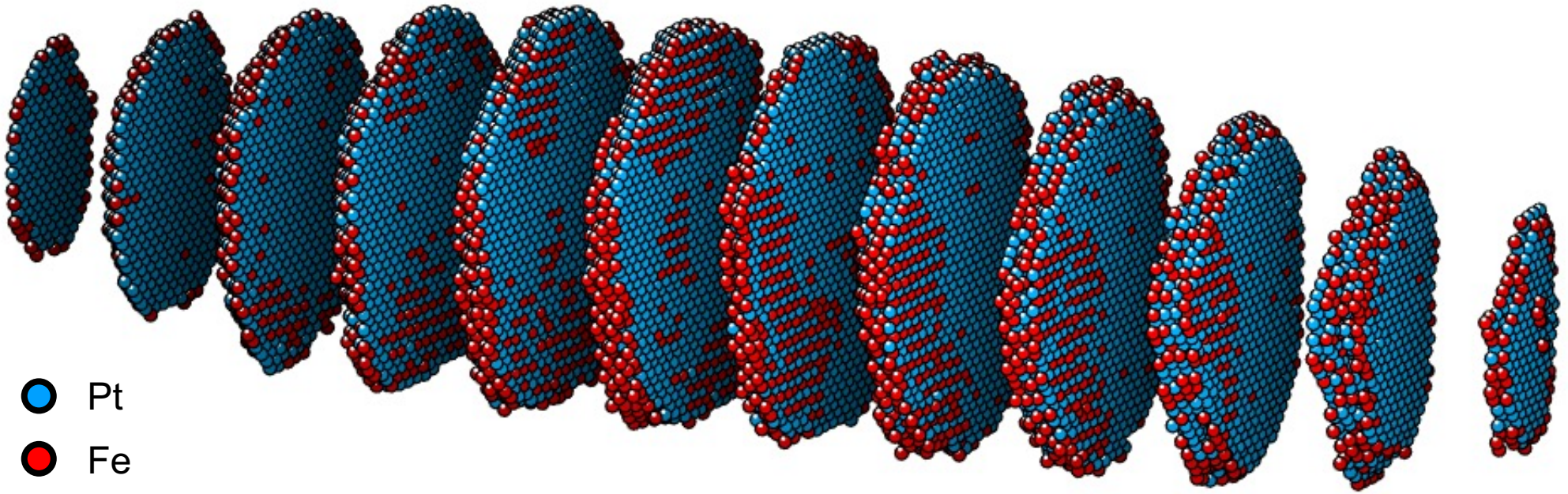


splitPlanesIndex = 3

- Pt
- Fe

Splitting apart the particle along the 3rd dimension.

Common Visualization Task – 3D Atoms

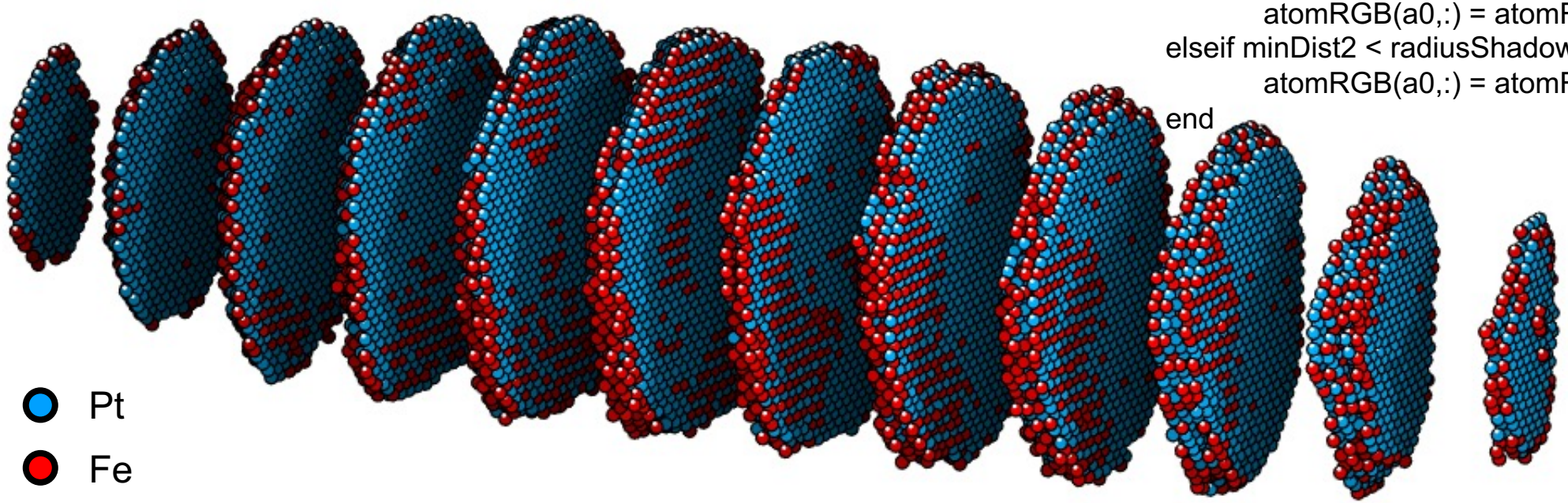


So far we are just manipulating parameters and positions – can we do more?

Common Visualization Task – 3D Atoms

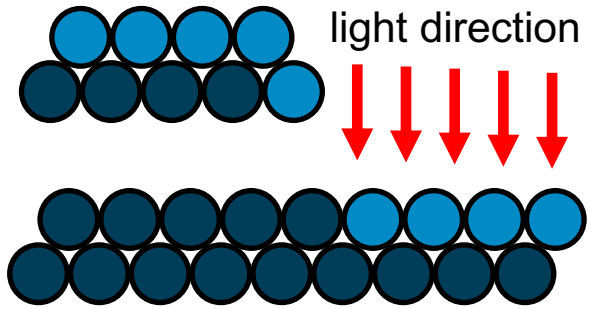
```
distProj2(:) = sum(cross(atomPos - xyz0, vecRep).^2,2);
sub(:) = sum((atomPos - xyz0).*vecShadow,2) > 0;
minDist2 = min(distProj2(sub));

if minDist2 < radiusShadow2(1)
        atomRGB(a0,:) = atomRGB(a0,:) * shadingShadow(1);
elseif minDist2 < radiusShadow2(2)
        atomRGB(a0,:) = atomRGB(a0,:) * shadingShadow(2);
end
```



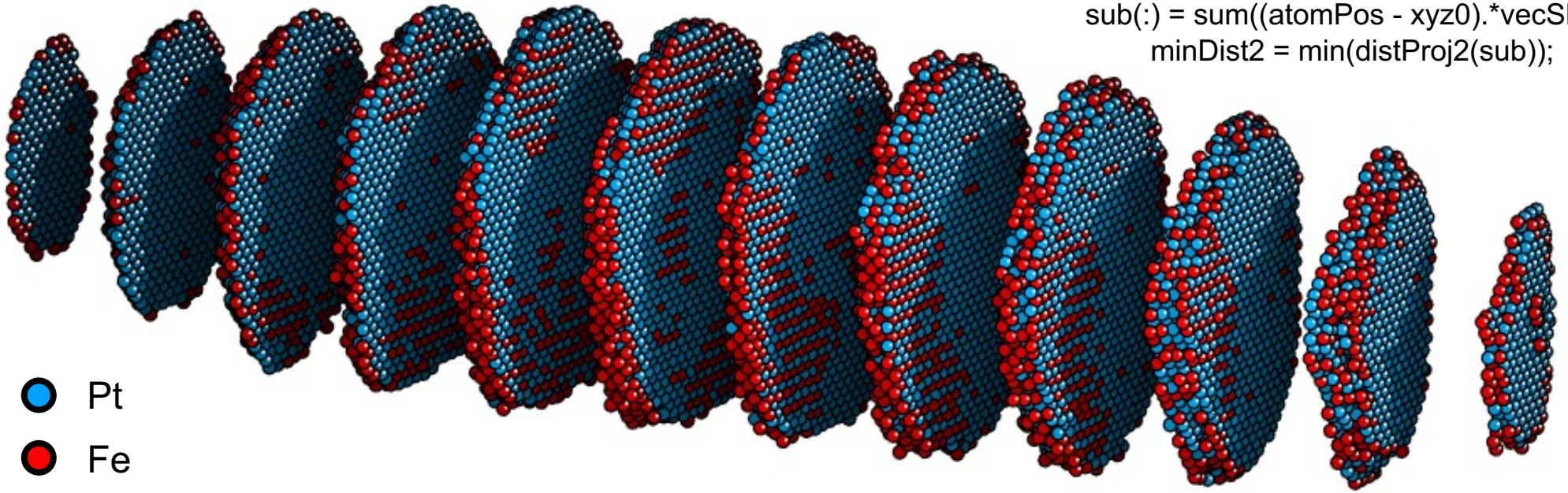
Pt

Fe



Ray traced shadows!

Common Visualization Task – 3D Atoms

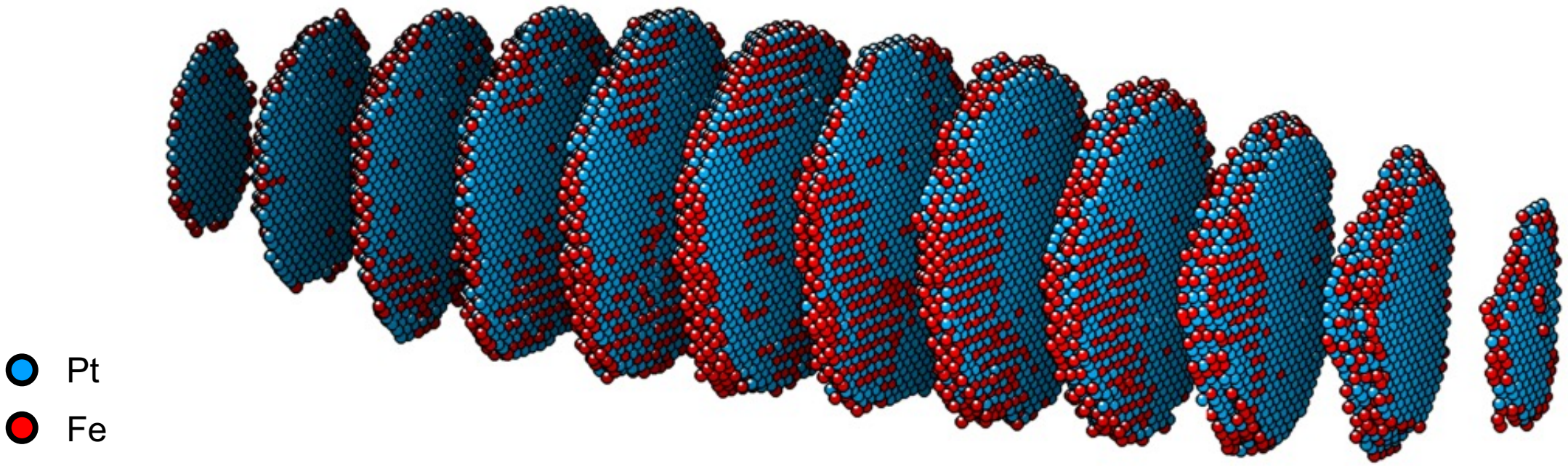



```
distProj2(:) = sum(cross(atomPos - xyz0, vecRep).^2,2);
sub(:) = sum((atomPos - xyz0).*vecShadow,2) > 0;
        minDist2 = min(distProj2(sub));
```

```
if minDist2 < radiusShadow2(1)
            atomRGB(a0,:) = atomRGB(a0,:) * shadingShadow(1);
            subShadowed(a0) = true;
        elseif minDist2 < radiusShadow2(2)
            atomRGB(a0,:) = atomRGB(a0,:) * shadingShadow(2);
            subShadowed(a0) = true;
end
```

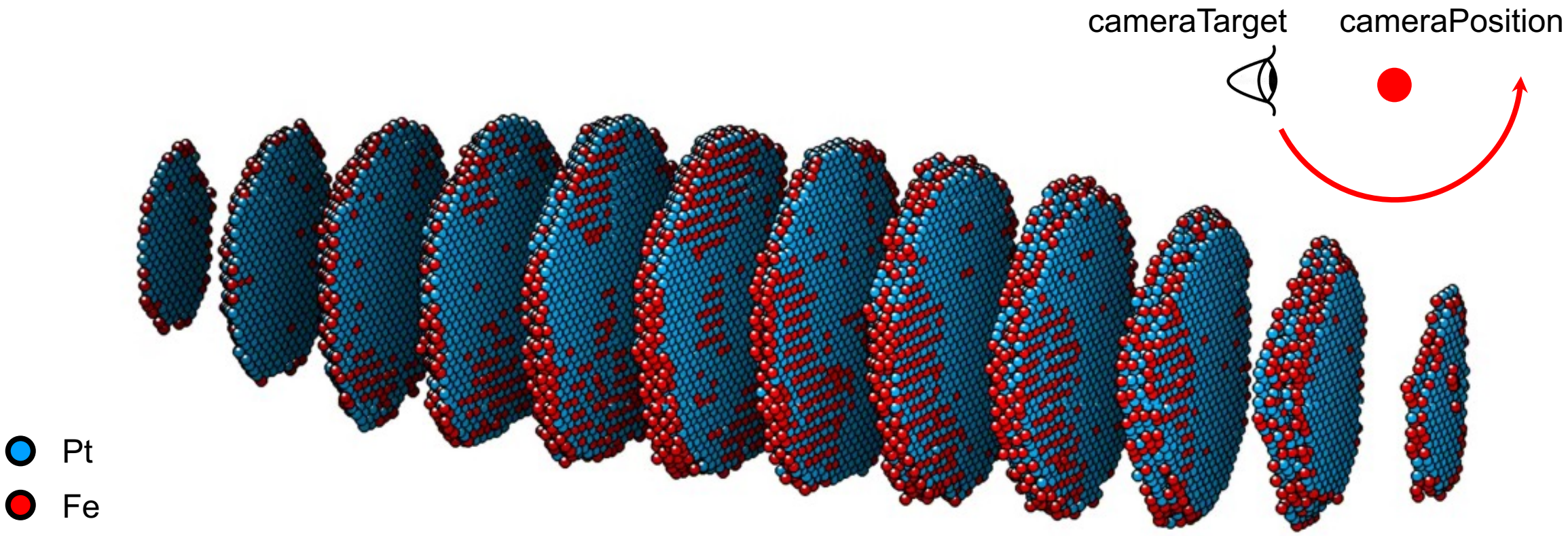
description

Making Movies from Images Using FFMPEG



Each time plotting script is called, move camera, save image.

Making Movies from Images Using FFMPEG



Each time plotting script is called, move camera, save image.

Making Movies from Images Using Code

Step 1 – Modify drawing script to have input variables.

Step 2 – Write a second script to repeatedly call the first script, changing the inputs.

Step 3 – Each time the drawing script is called, render an image, output to file.

Step 4 – Use FFMPEG to stitch / encode images into a movie. Handbrake edit if needed.

Tutorial example, animated 3D atoms plot with animAtoms01a(), animAtoms01b()

Making Movies from Images Using FFMPEG

FFMPEG command line example for low frame rate movies:

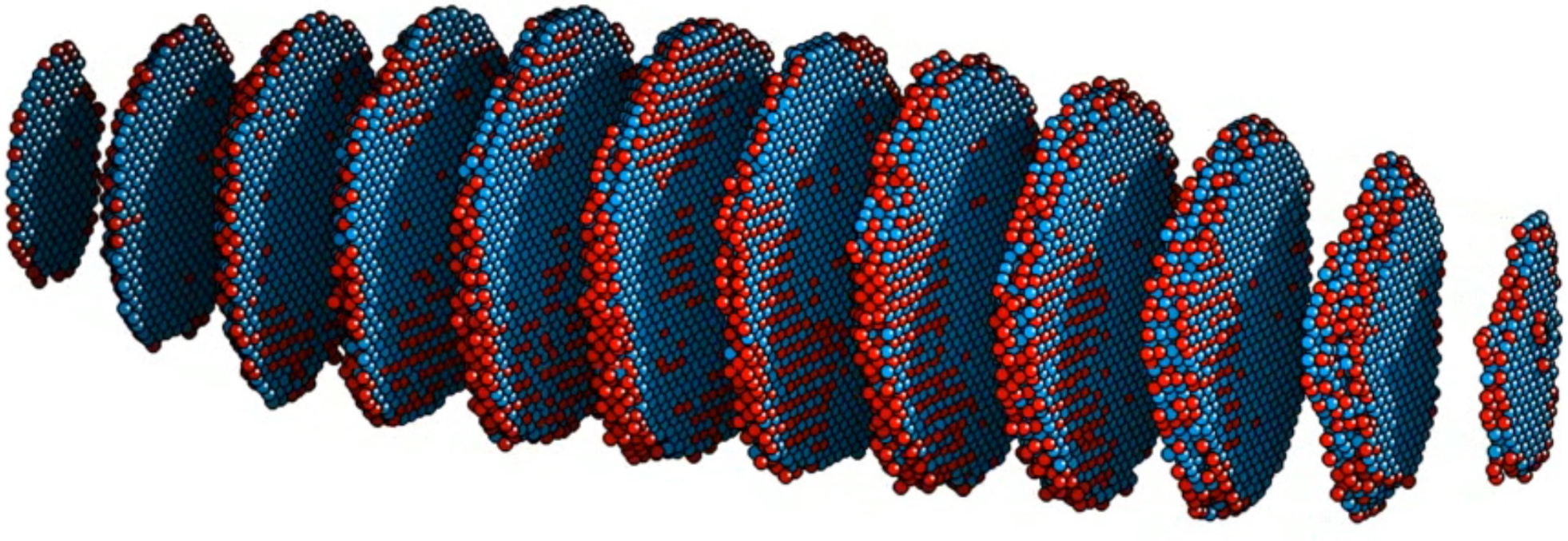
```
ffmpeg
-f image2
-i image%04d.png
-vcodec mpeg4
-qmax 4
-vf setpts=4.0*PTS
movieFileName.mp4
```

FFMPEG command line example for high frame rate movies:

```
ffmpeg
-f image2 -r 60 -i image%04d.png
-c:v libx264 -preset slow
-profile:v high -level:v 4.0
-pix_fmt yuv420p -filter:v fps=fps=60
movieFileName.mp4
```

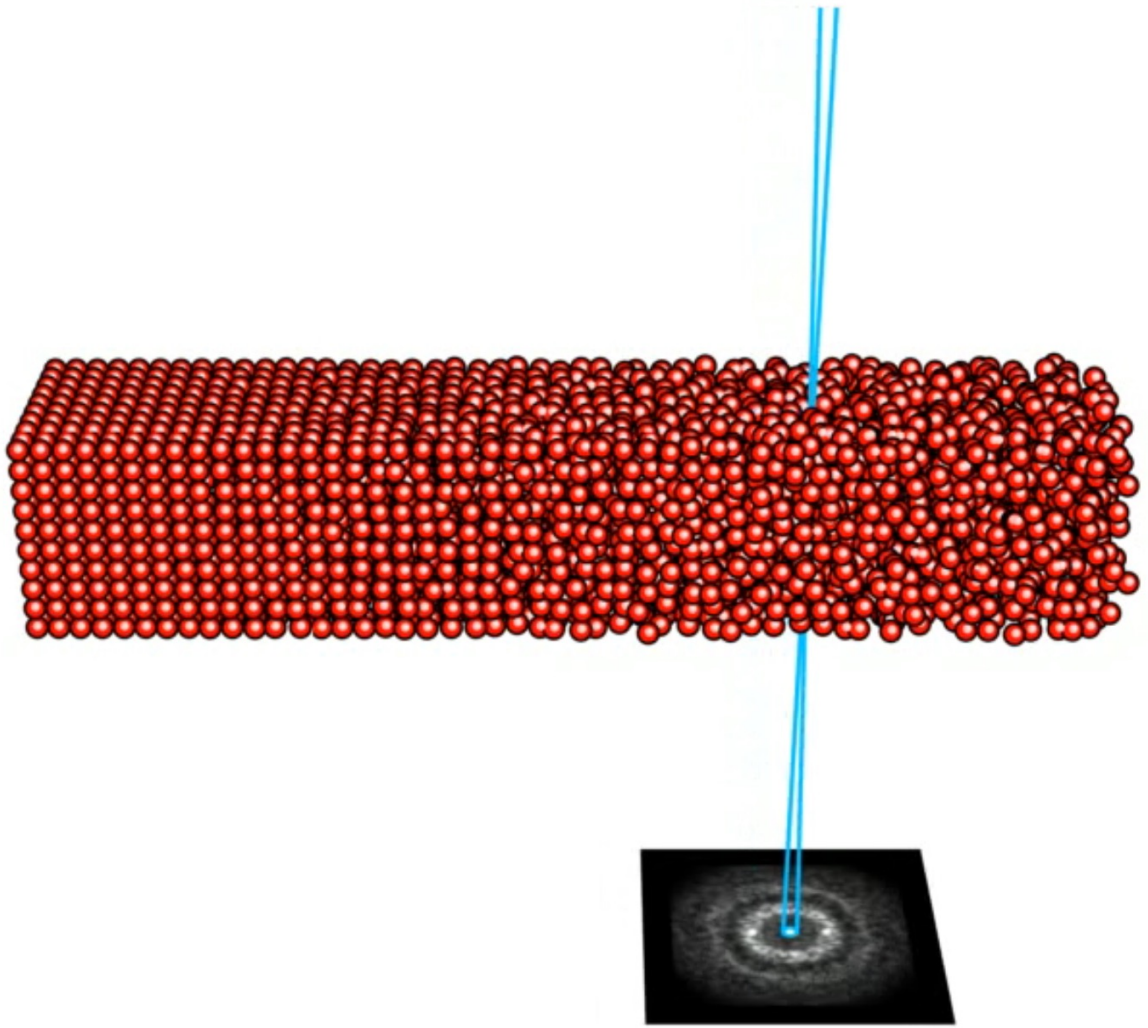
These are just examples – lots of settings to tweak in FFMPEG!

Making Movies from Images Using FFMPEG



Each time plotting script is called, change splitPlanes from 0 to 70, save image.

Examples of Using Movies to Explain Physics

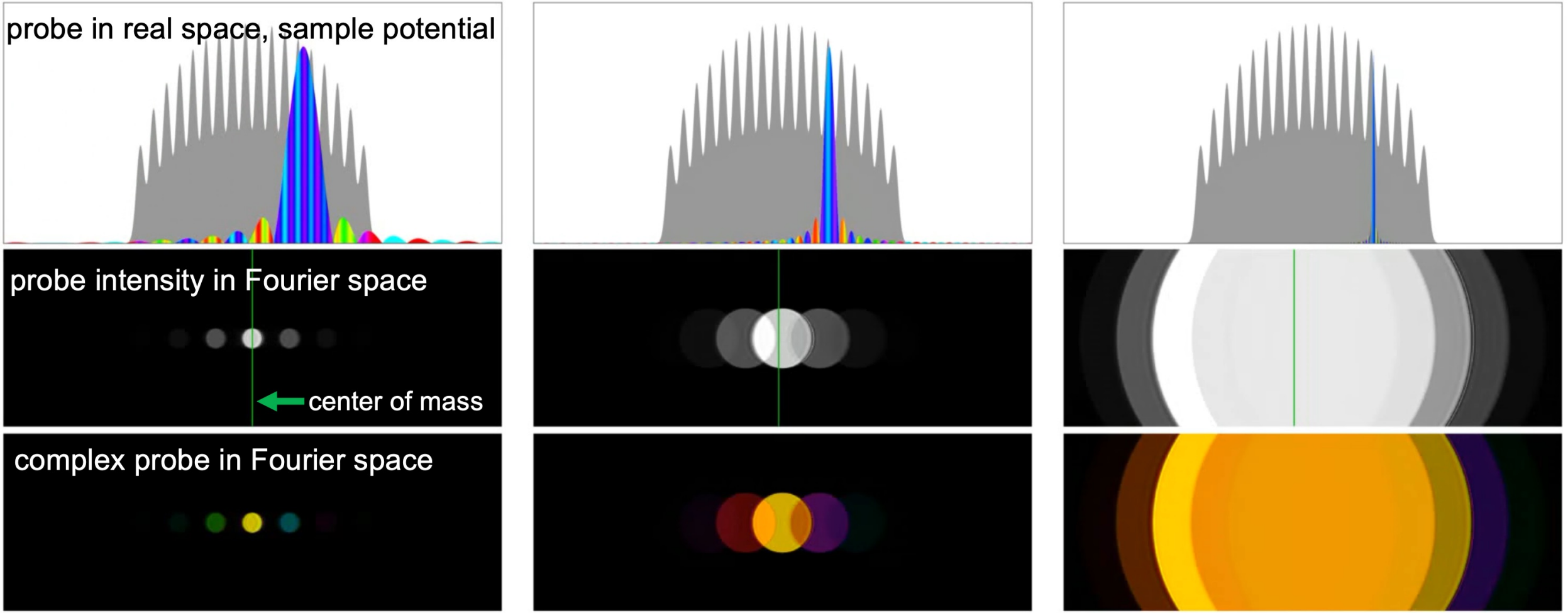


animFEM01(probeSites, rScale)
draws the scene,

animFEM02()
animates the atoms transforming from ordered → disordered,

animFEM03()
animates STEM probe scanning.

Diff. Phase Contrast – Effect of Probe Semiangle

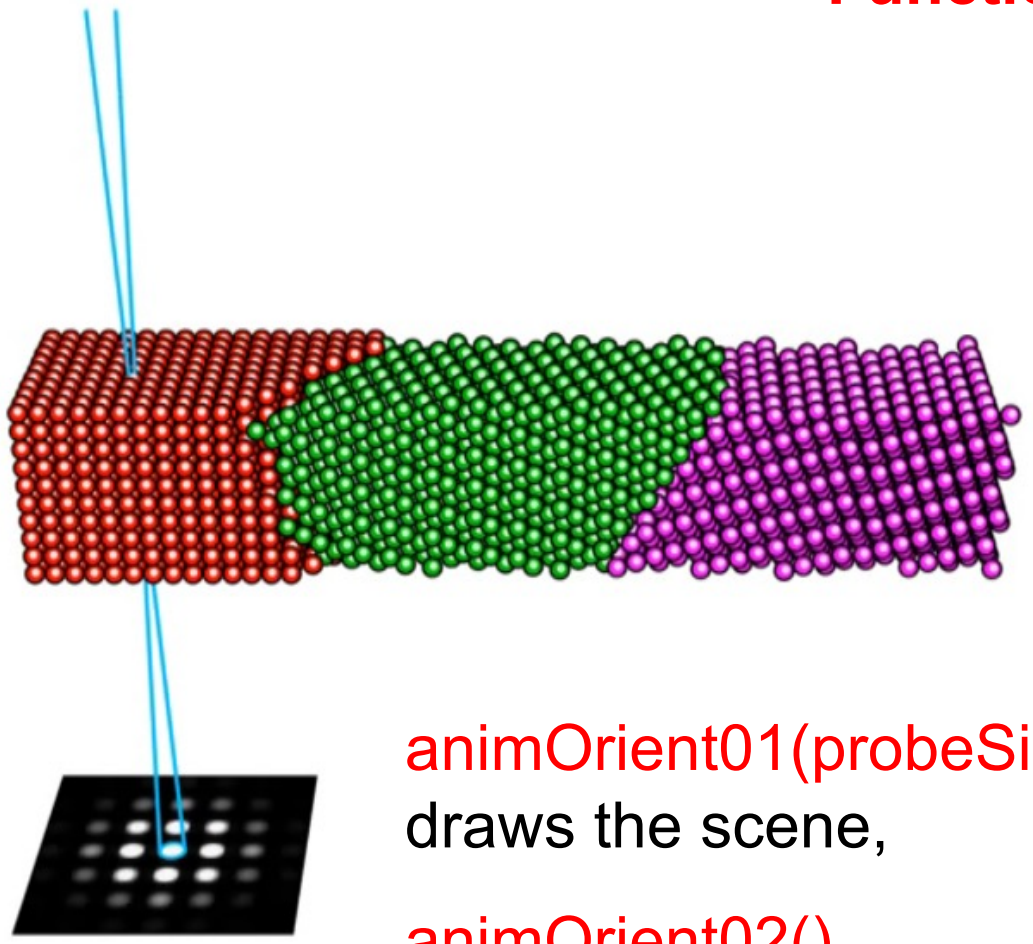


Same sim., plotting and output script for all movies – just change one number!

Try it:

```
>> sDPC = DPC01();
>> DPC02(sDPC);
```

Using a Movie to Explain Diffraction Physics



Functions

animOrient01(probeSites)
draws the scene,

animOrient02()
animates it.

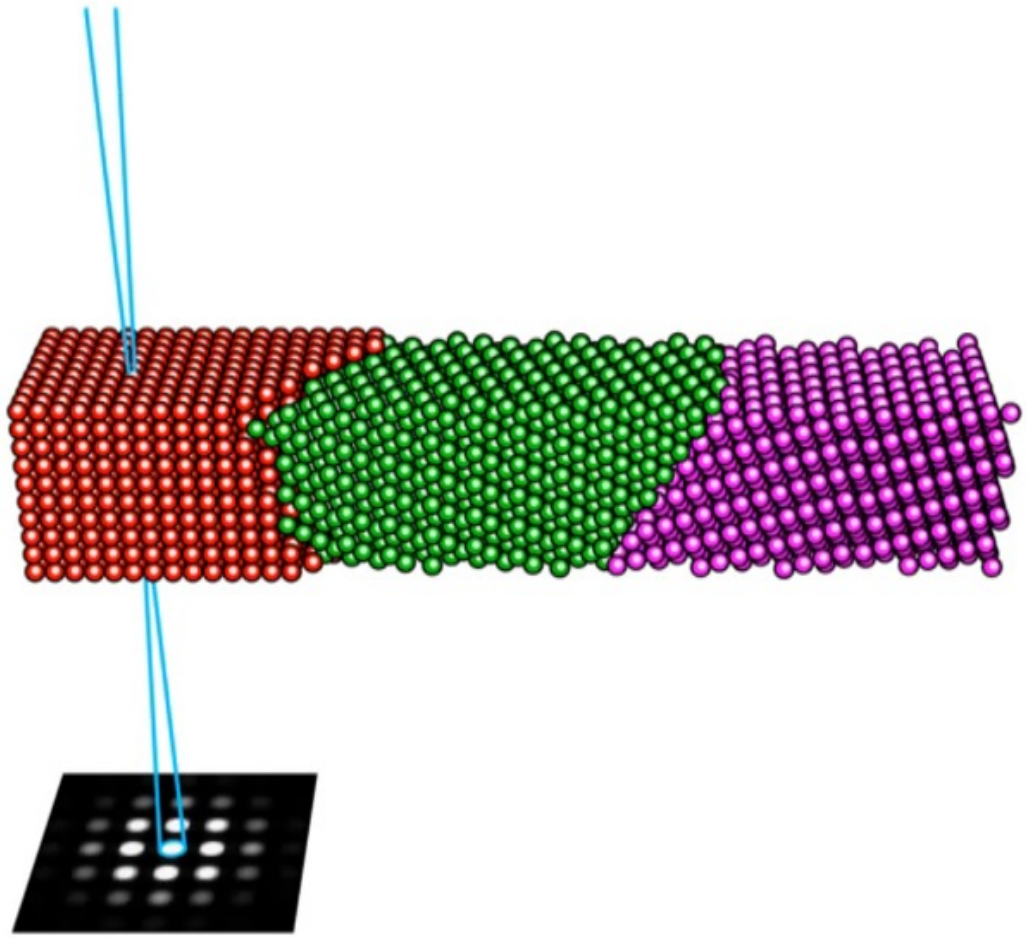
line(x,y,z,…)
Probe is just 2 lines and a circle

scatter3(x,y,z,…)
Three grains tiled & rotated, each with 2 scatter plots for the atoms and tints.

sum(), conv2(), fft2(), …
Images are generated by summing atoms along dim 3, taking the FFT.

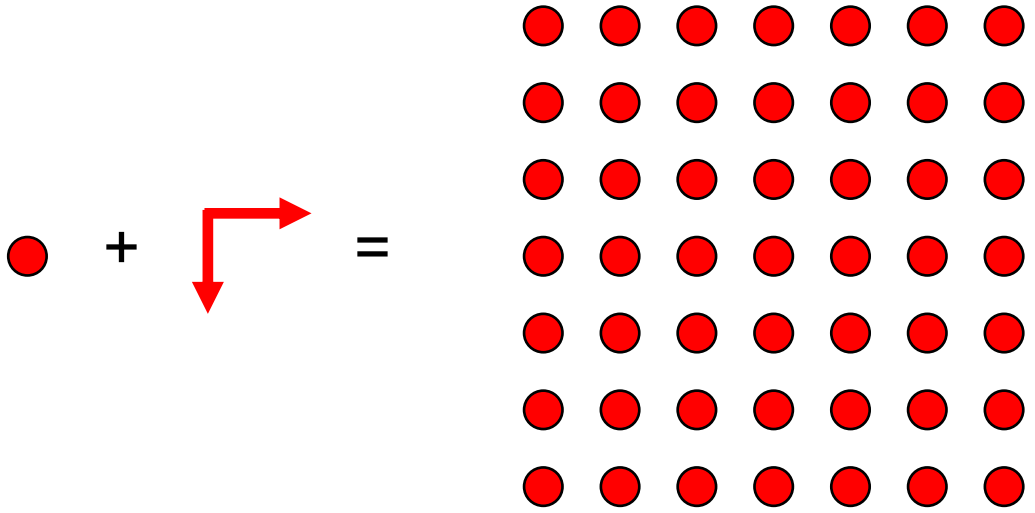
warp(), conv2(), fft2(), …
Easy methods for placing a texture-mapped surface in 3D, modifying images.

Using a Movie to Explain Diffraction Physics

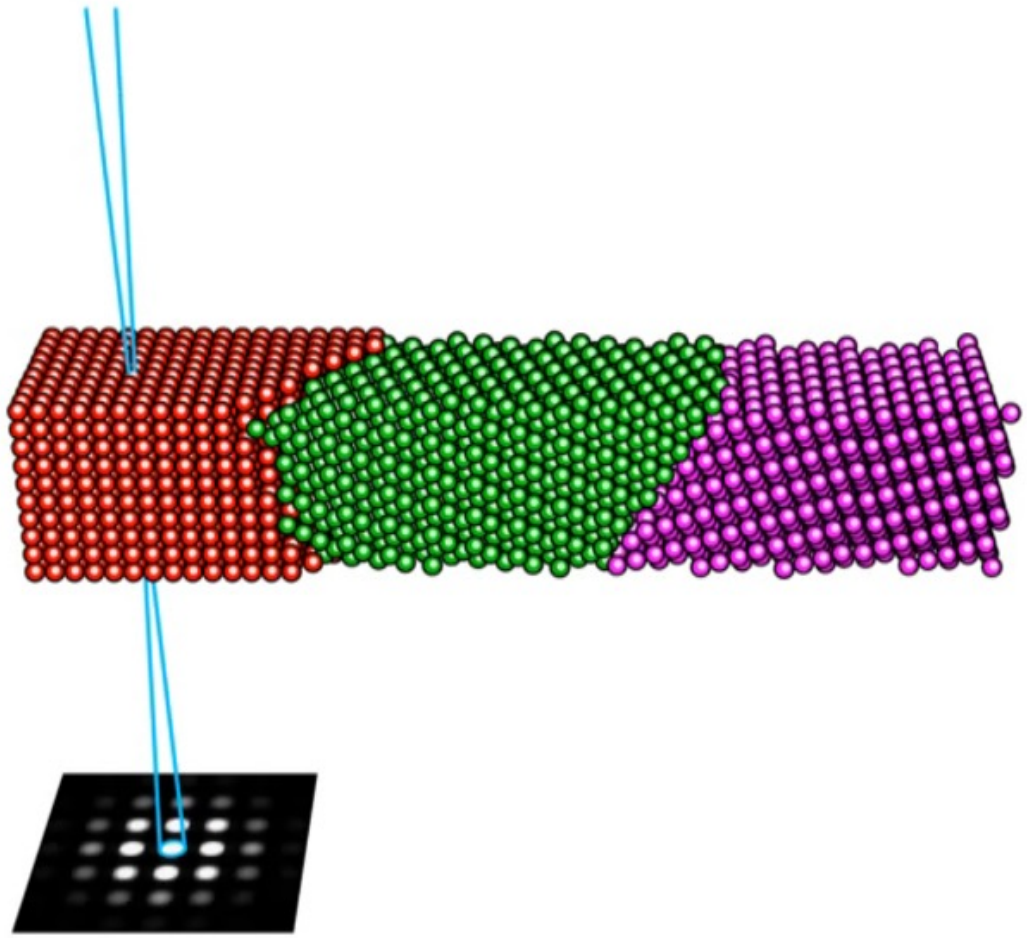


Building a block of atoms

```
numAtomsMax = max(numAtoms);
v = -round(numAtomsMax/2):round(numAtomsMax/2);
[yy, xx, zz] = meshgrid(v, v, v);
p0 = [xx(:) yy(:) zz(:)];
```



Using a Movie to Explain Diffraction Physics

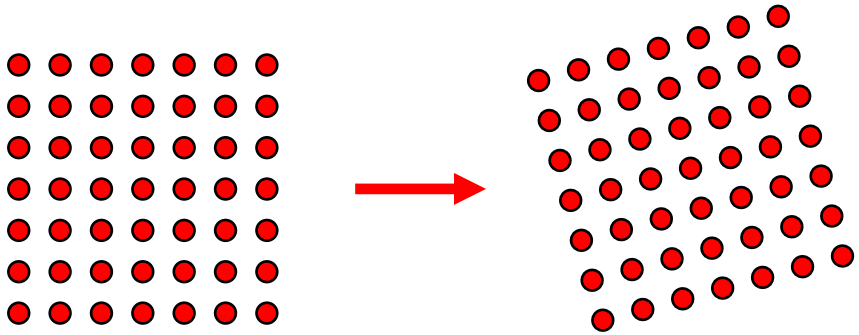


Rotating a block of atoms

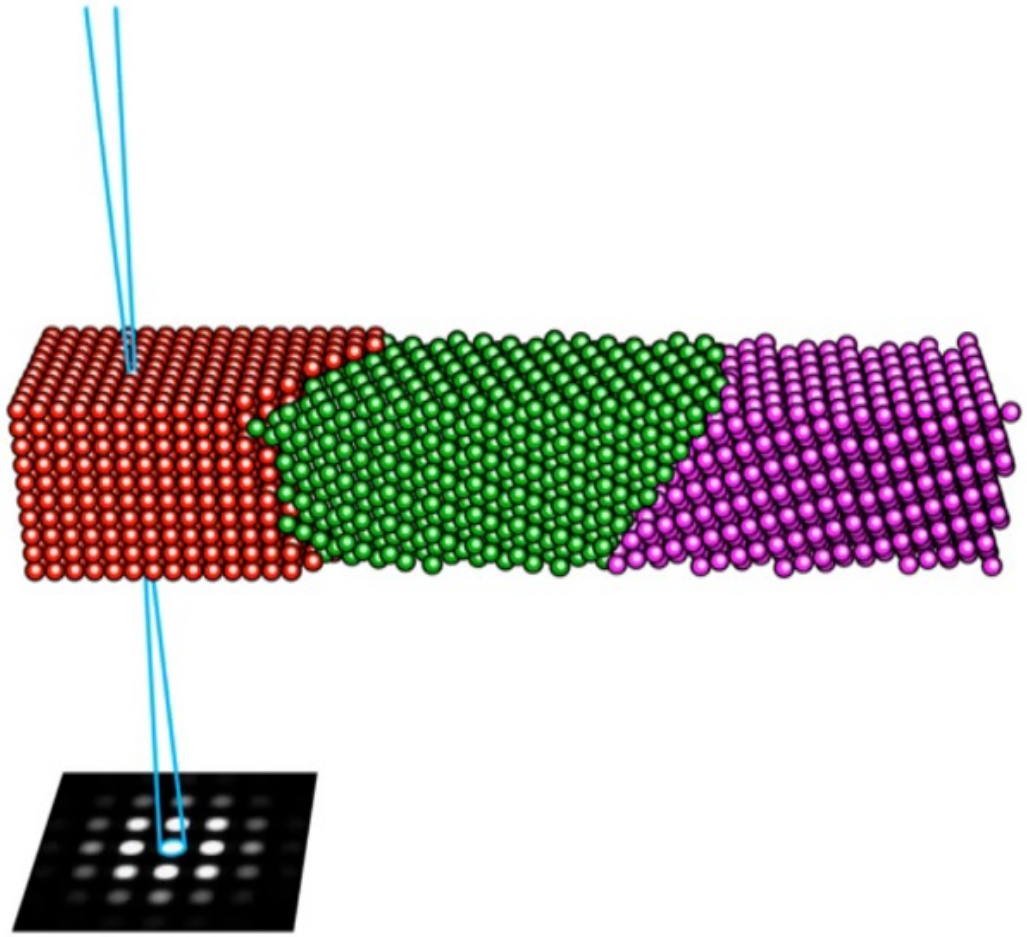
```
t = [45 54 -18.43]*pi/180;
m1 = [cos(t(1)) -sin(t(1)) 0;
    sin(t(1)) cos(t(1)) 0;
    0 0 1];
m2 = [1 0 0;
    0 cos(t(2)) -sin(t(2));
    0 sin(t(2)) cos(t(2))];
m3 = [cos(t(3)) -sin(t(3)) 0;
    sin(t(3)) cos(t(3)) 0;
    0 0 1];
p2 = p2 * m1 * m2 * m3;
```

Euler angle rotations:

Axis	Matrix
z	m1
x	m2
z	m3

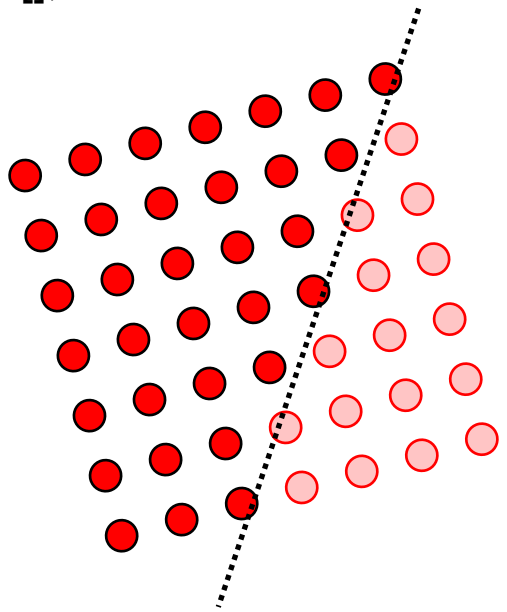


Using a Movie to Explain Diffraction Physics

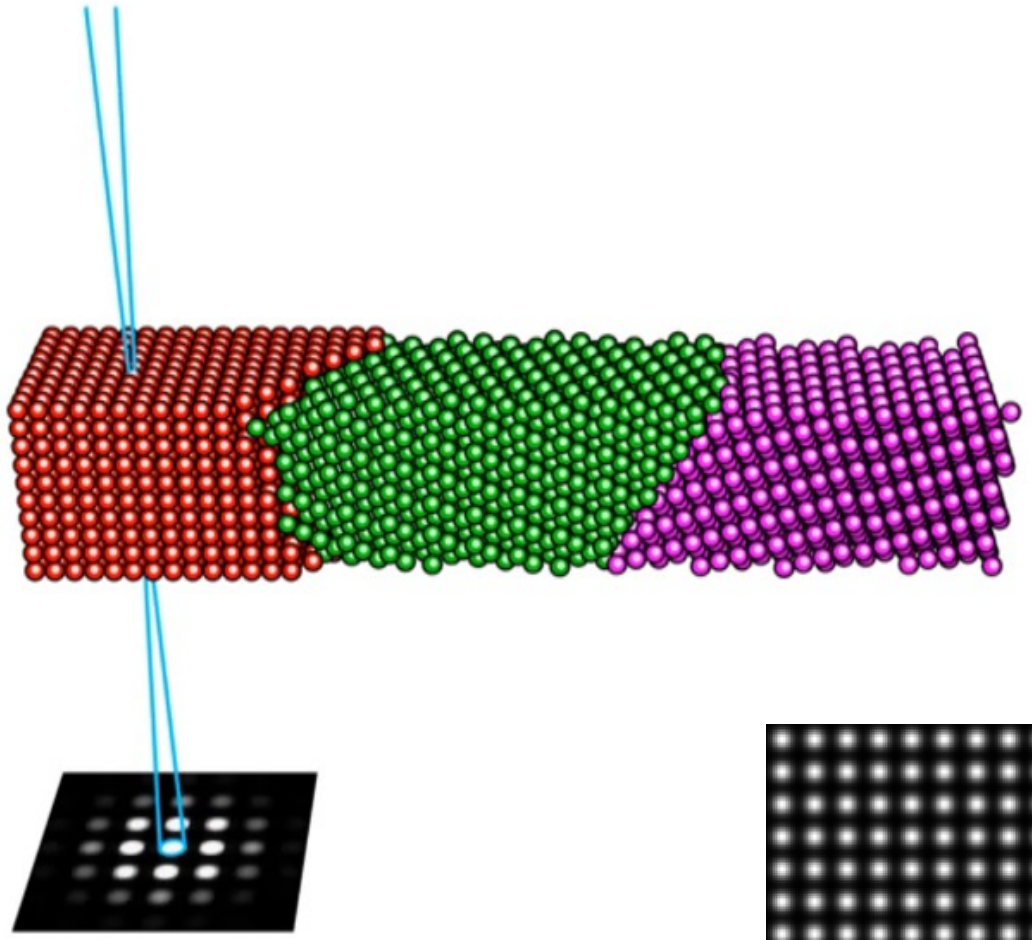


Deleting atoms on one side of a plane:

```
abcd12 = [0.2 1 1.2/2 22];
del = p1(:,1) < 1 | p1(:,1) > cellDim(1) ...
    | p1(:,2) < 1 | p1(:,2) > cellDim(2) ...
    | p1(:,3) < 1 | p1(:,3) > cellDim(3) ...
    | p1(:,1)*abcd12(1) + p1(:,2)*abcd12(2) + ...
      p1(:,3)*abcd12(3) > abcd12(4) - 0.1;
p1(del,:) = [];
```

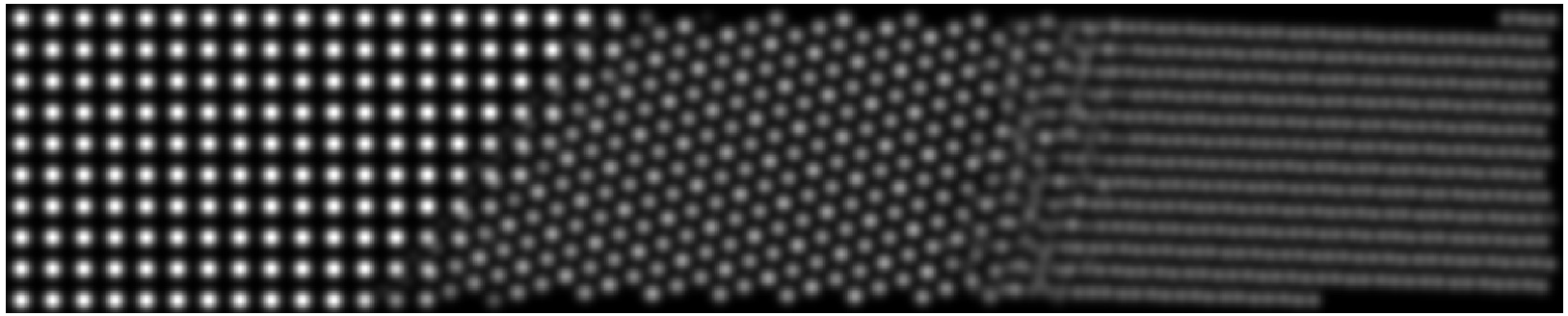


Using a Movie to Explain Diffraction Physics



Create atom image by summing along dim 3, blurring:

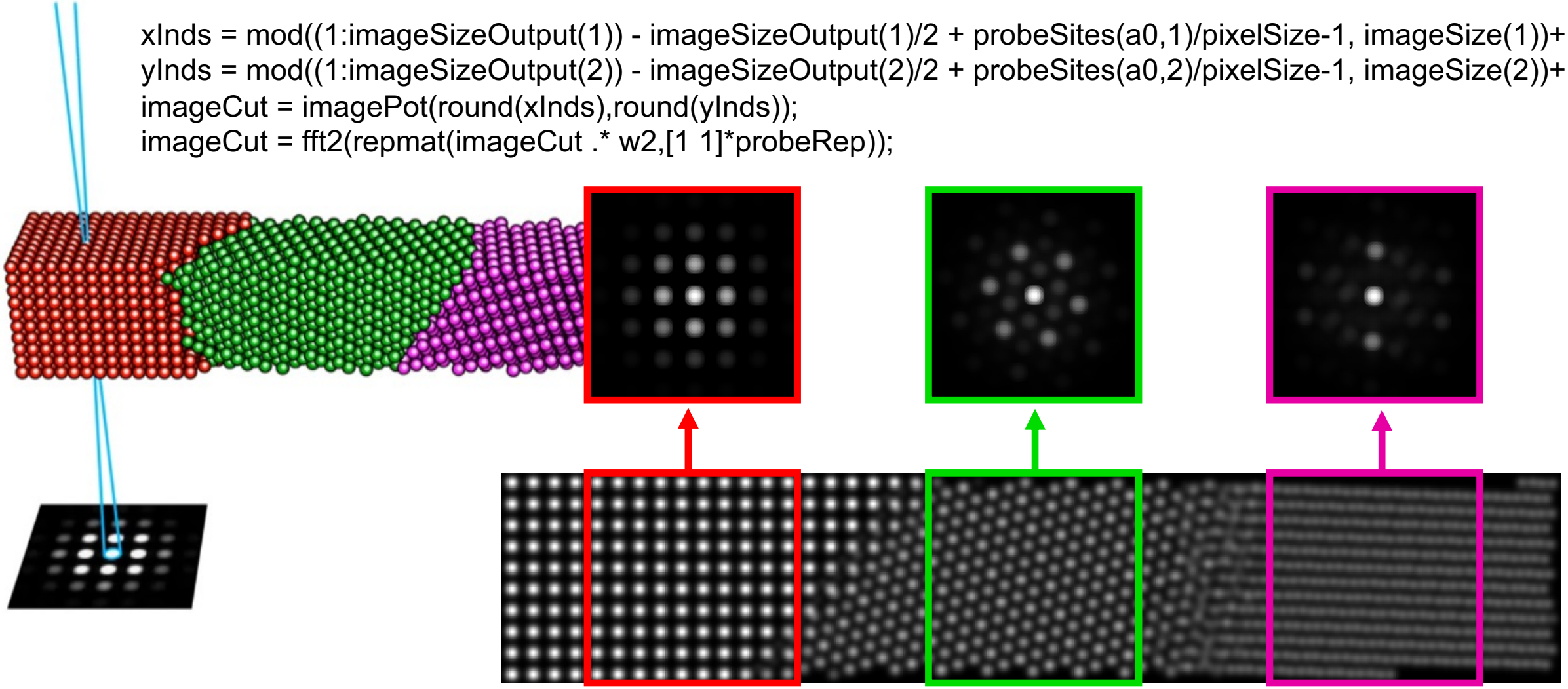
```
k = fspecial('gaussian',2*ceil(4*sigmaAtoms)+1,sigmaAtoms);
imageSize = round(cellDim(1:2) ./ pixelSize);
xInd = (p(:,1) - 0.5) / pixelSize;
yInd = (p(:,2) - 0.5) / pixelSize;
xInd = min(max(round(xInd),1),imageSize(1));
yInd = min(max(round(yInd),1),imageSize(2));
imagePot = accumarray([xInd yInd],...
    ones(length(xInd),1),imageSize);
imagePot(:) = conv2(imagePot,k,'same');
```



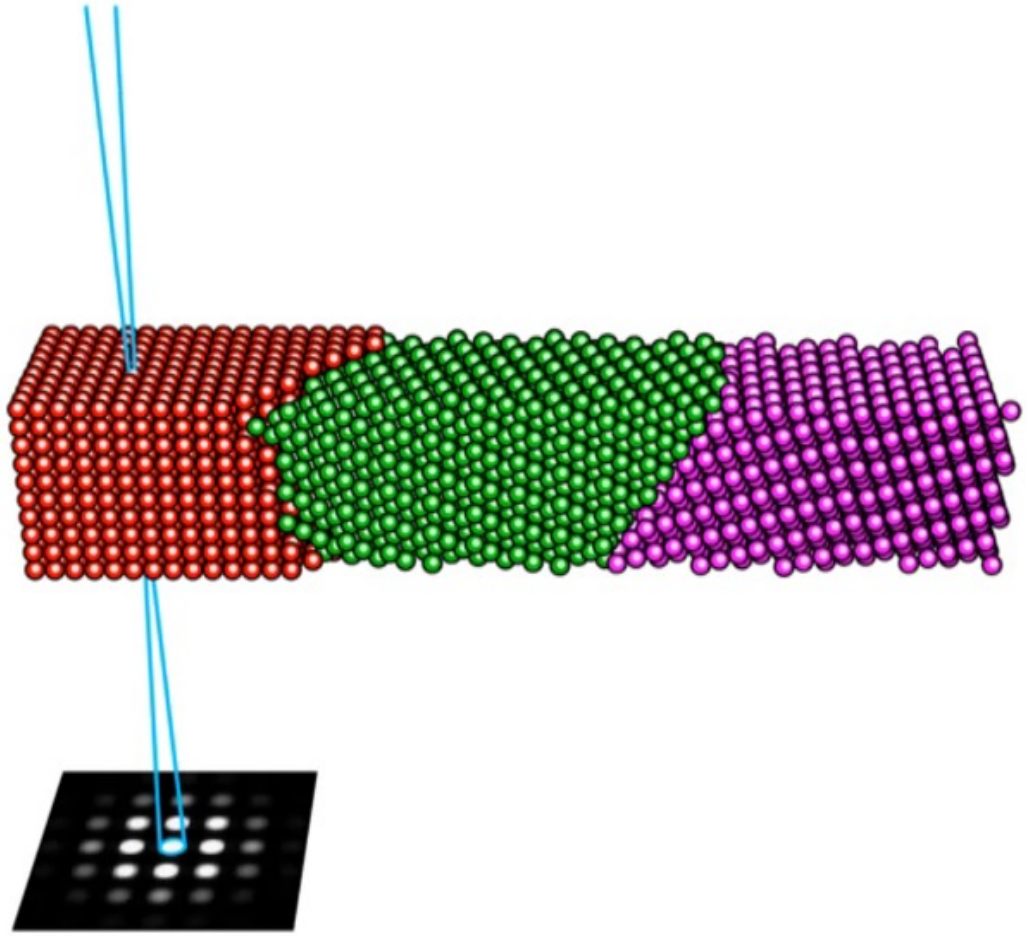
Using a Movie to Explain Diffraction Physics

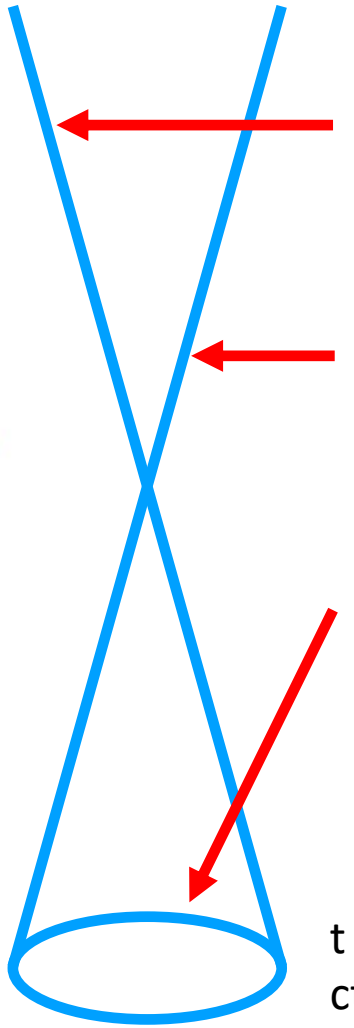
Generate diffraction images:

```
xInds = mod((1:imageSizeOutput(1)) - imageSizeOutput(1)/2 + probeSites(a0,1)/pixelSize-1, imageSize(1))+1;
yInds = mod((1:imageSizeOutput(2)) - imageSizeOutput(2)/2 + probeSites(a0,2)/pixelSize-1, imageSize(2))+1;
imageCut = imagePot(round(xInds),round(yInds));
imageCut = fft2(repmat(imageCut .* w2,[1 1]*probeRep));
```



Using a Movie to Explain Diffraction Physics





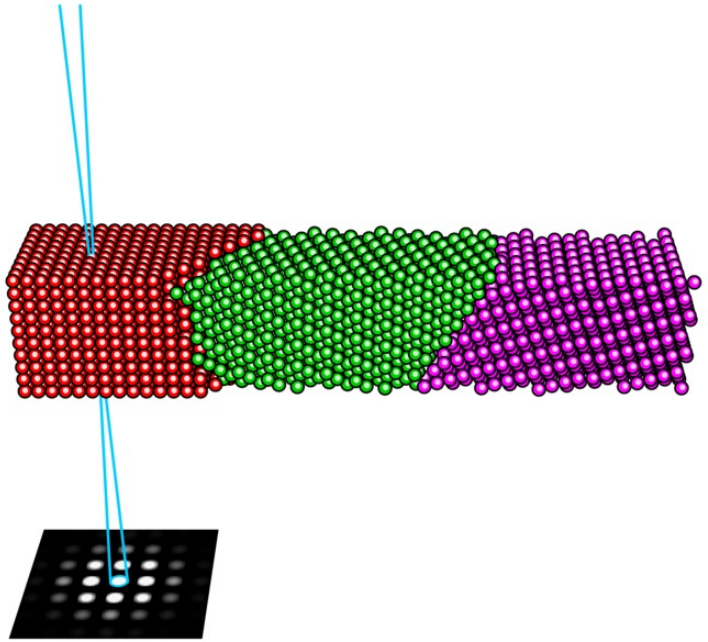
Draw the STEM probe:

```
line([-1 1]*probeRadius+probeSites(a0,2),...
     [0 0]+probeSites(a0,1),...
     probePos([1 3]),...
     'linewidth',linewidthProbes,'color',[0 0.8 1])
line([1 -1]*probeRadius+probeSites(a0,2),...
     [0 0]+probeSites(a0,1),...
     probePos([1 3]),...
     'linewidth',linewidthProbes,'color',[0 0.8 1])
line(ct*probeRadius+probeSites(a0,2),...
     st*probeRadius+probeSites(a0,1),...
     probePos(3)+ct*0,...
     'linewidth',linewidthProbes,'color',[0 0.8 1])
```

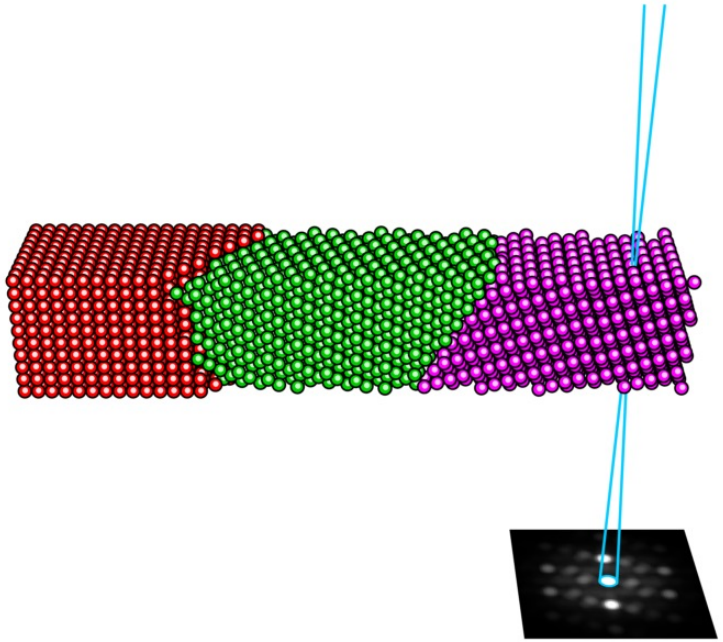
```
t = linspace(0,2*pi,180+1);
ct = cos(t);
st = sin(t);
```

Using a Movie to Explain Diffraction Physics

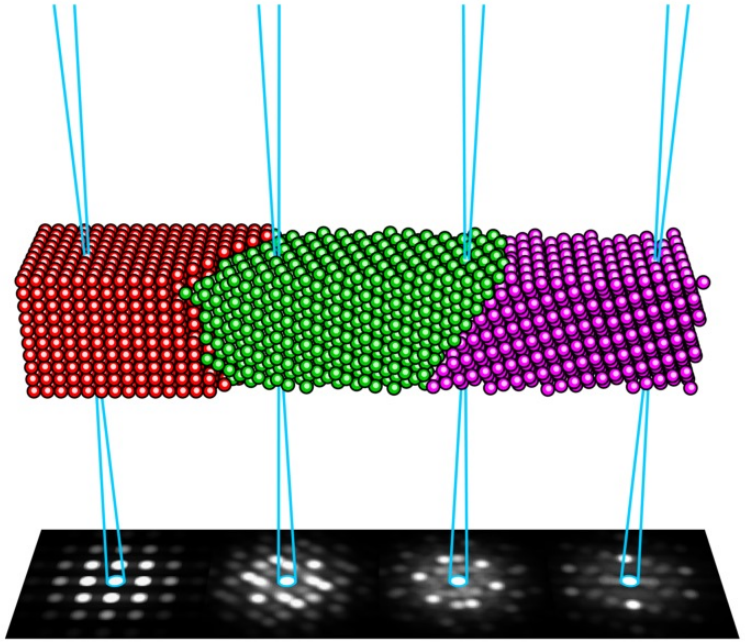
>> animOrient01([5 45]);



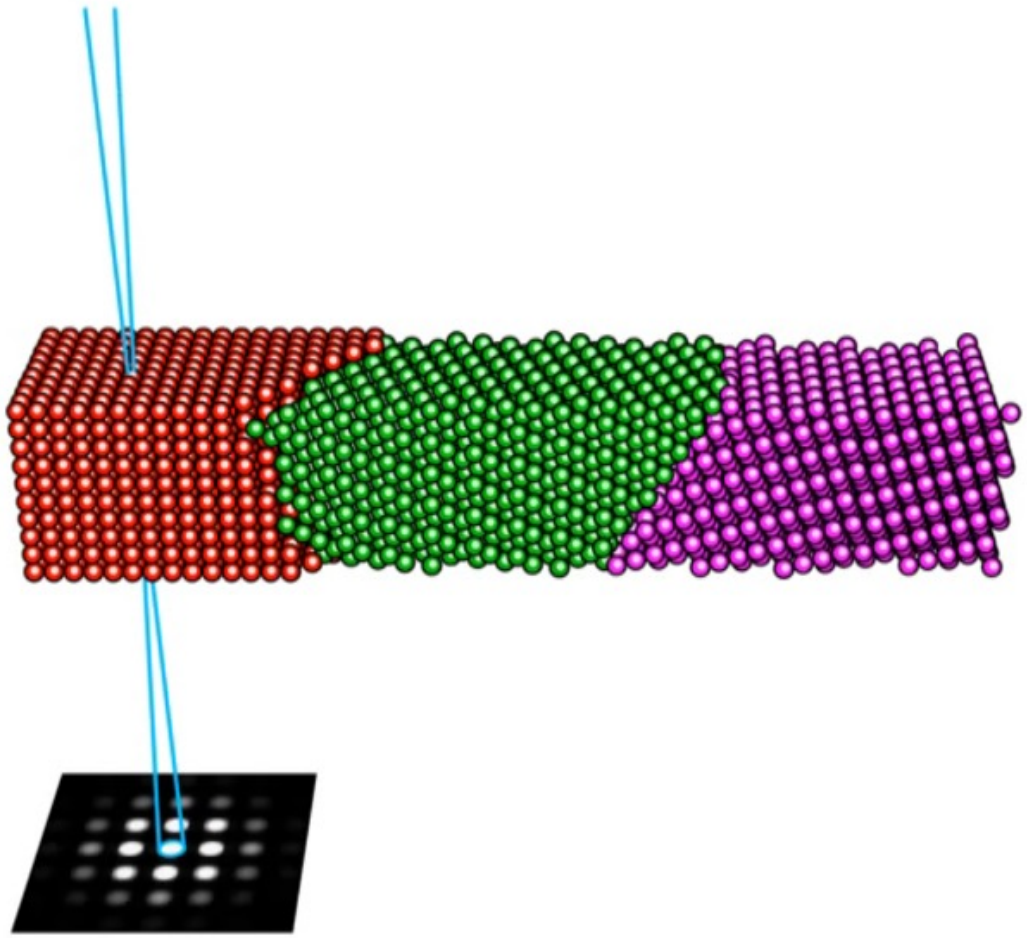
>> animOrient01([5 5]);



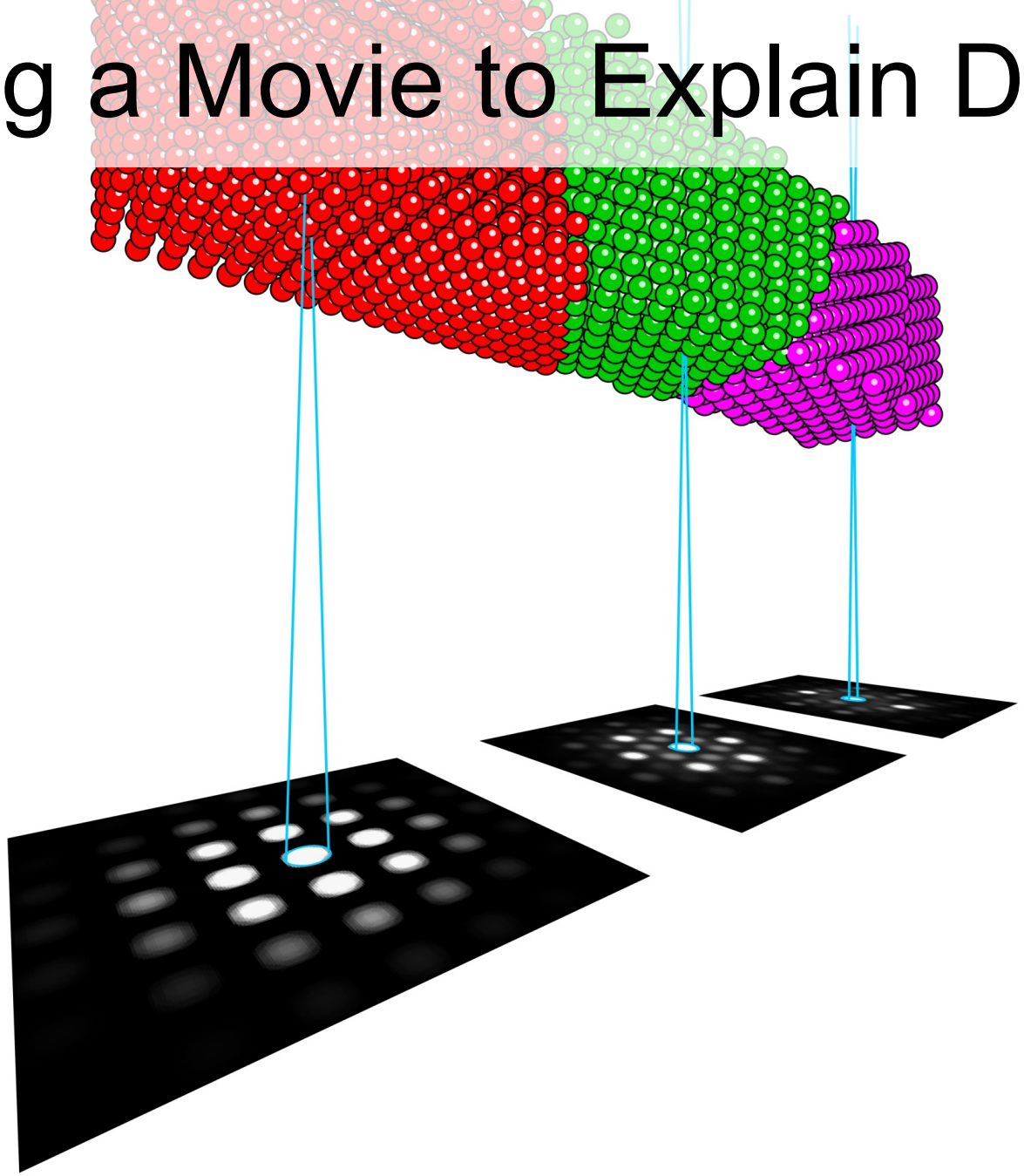
>> animOrient01([5 4; 5 18; 5 32; 5 46]);



Using a Movie to Explain Diffraction Physics



Using a Movie to Explain Diffraction Physics

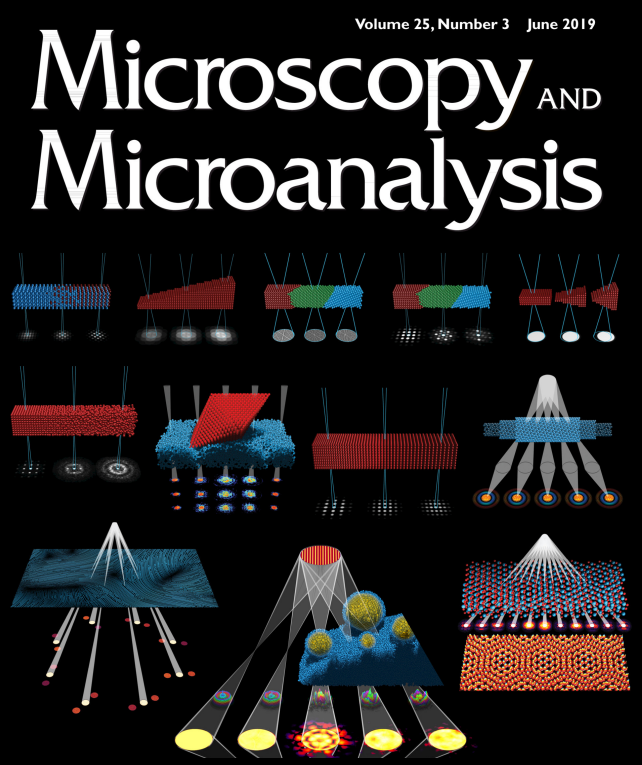


Just play with the camera, plot, image settings until you are happy with the result!

For example:

```
cTar = [0 0 0] + cellDim * 0.5 + [10 5 -15];
cPos = [-2.25 1.75 0]*13;
cAngle = 75;
```

More Info – **clophus@lbl.gov** – ncem-lbl.slack.com

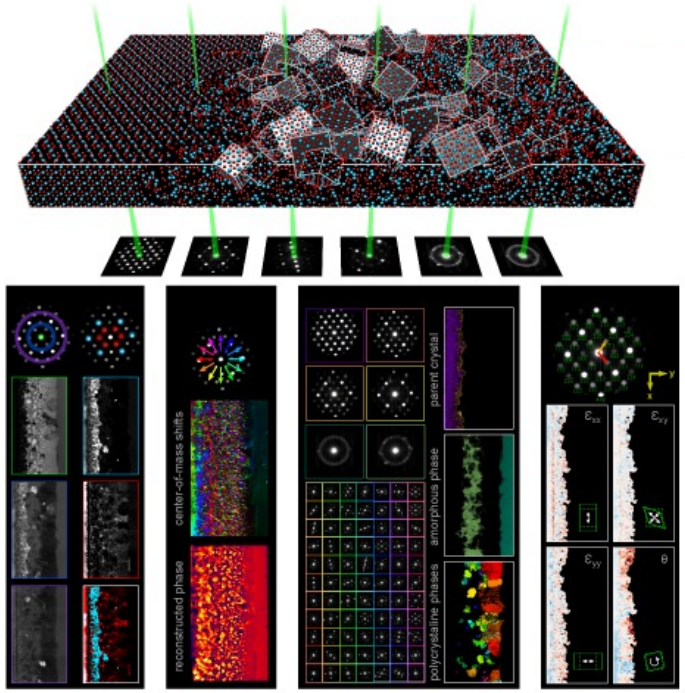


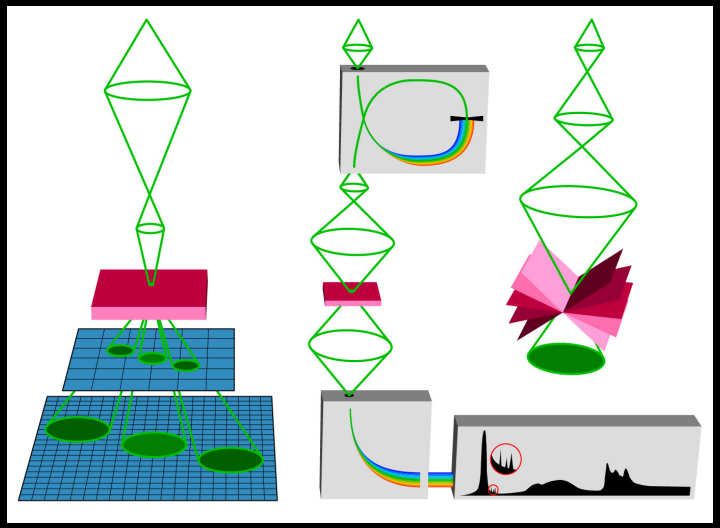
4D-STEM Review

C Ophus, Microscopy and Microanalysis 25, 563 (2019)

4D-STEM analysis: **py4DSTEM**

www.github.com/py4dstem





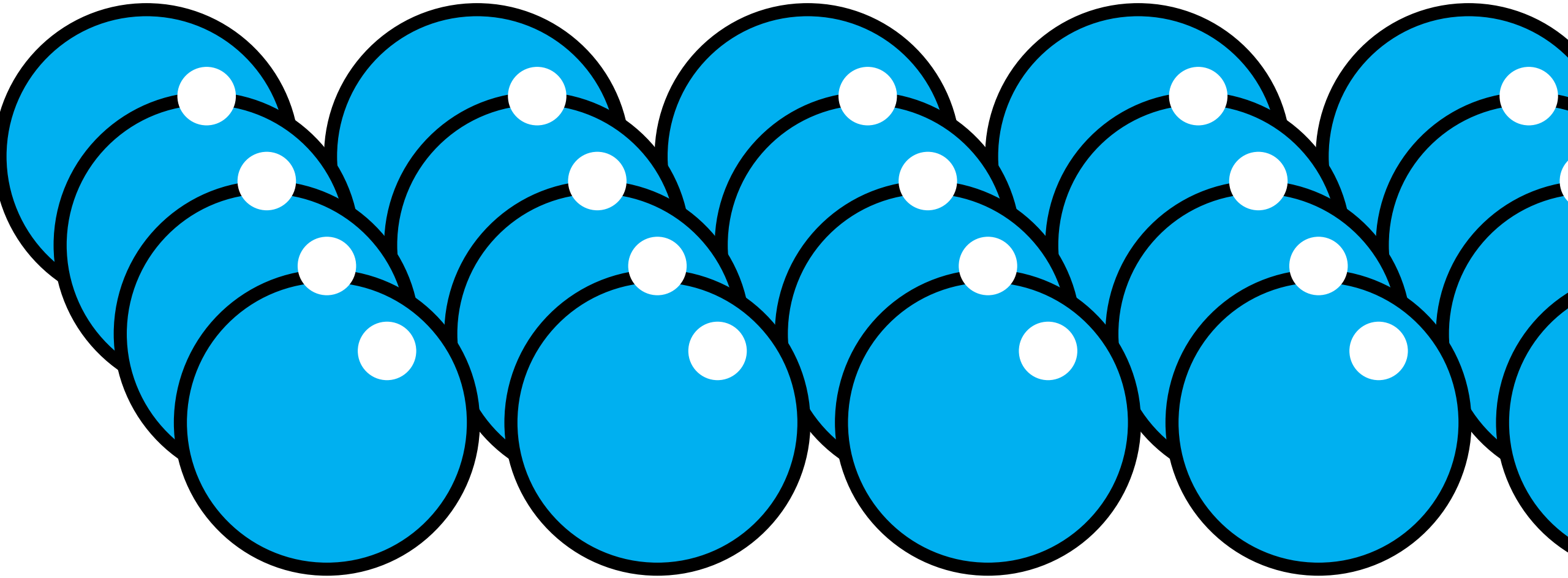
Quantitative STEM Imaging, Diffraction, Spectroscopy and Tomography

Colin Ophus, Annual Review of Materials Research 53, 105 (2023).

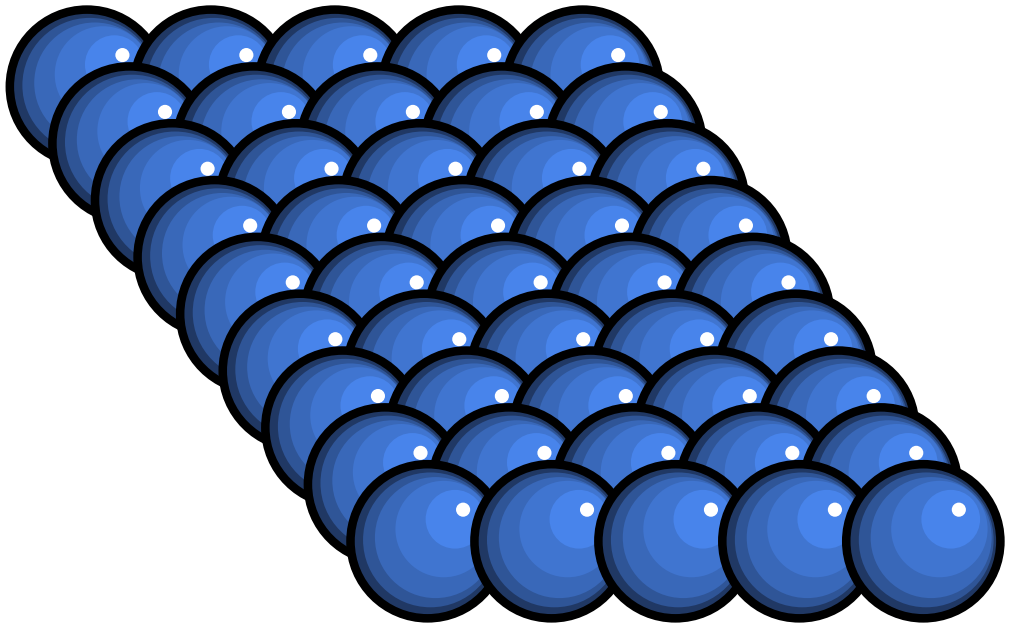
Workshop Tutorials – Python + Google Colab

tinyurl.com / **progviz**

Scratch



Scratch



Scratch