VSS 2018-05-19

ShapeToolbox: Creating 3D models for vision research

Toni P. Saarela
Department of Psychology, University of Helsinki

3D models in vision research

3D models are used as stimuli in vision research in, e.g.:

- Shape perception
- Material perception

Advanced software packages exist for model creation, but they

- Might not have scripting capabilities
- Might not provide fine parametric control over shape
- Might not be free

`ShapeToolbox` for `Matlab` & `GNU Octave`

- Provides a handful of simple "base shapes" that can be perturbed by adding noise, bumps, dents, sinusoids, etc.
- Save as Wavefront obj-files
- Free and open source

`github.com/saarela/ShapeToolbox`

Basic usage

- Create a sphere with default sinusoidal modulation, store the model in structure `m`, and view it:

```
m = objMakeSine('sphere')
objView(m)
```

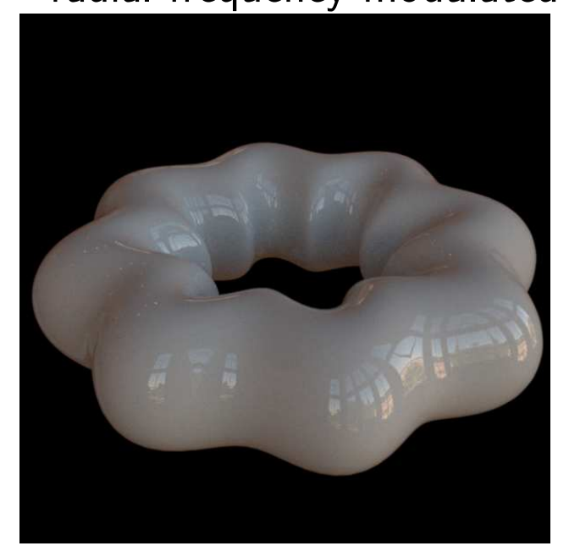
- A torus with custom noisy modulation, save in a file `torus.obj`:

```
m = objMakeNoise('torus',[4 1 60 10 .15],'torus')
```

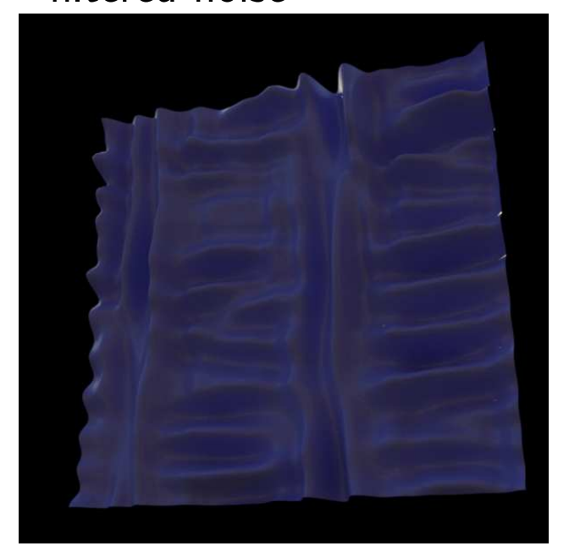
- Several options available for customizing the model, given as name-value pairs to `objMake*`-functions

Example shapes and perturbations

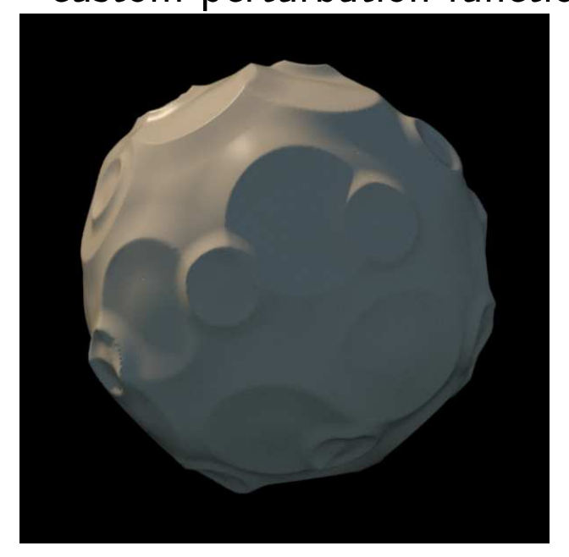
`objMakePlain('torus',...)`
- radial frequency modulated



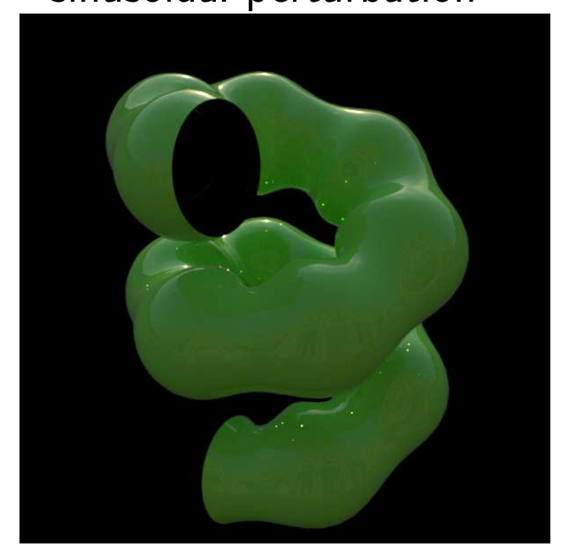
`objMakeNoise('plane',...)`
- filtered noise



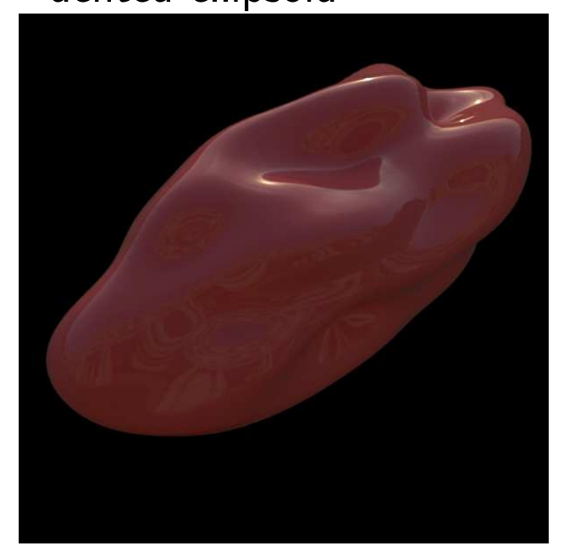
`objMakeCustom('sphere',...)`
- custom perturbation function



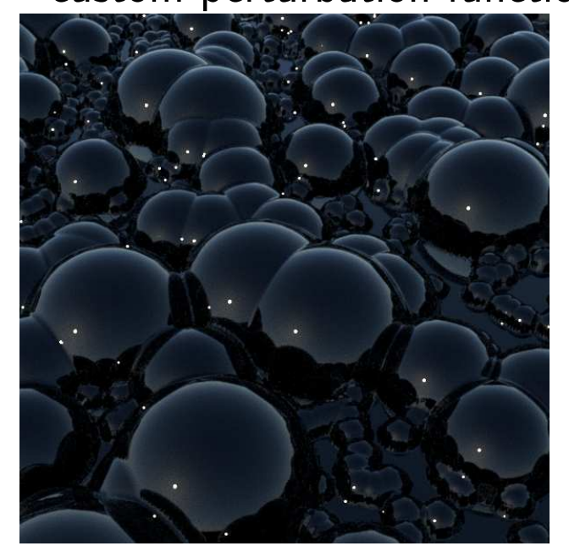
`objMakeSine('worm',...)`
- sinusoidal perturbation



`objMakeBump('ellipsoid',...)`
- dented ellipsoid



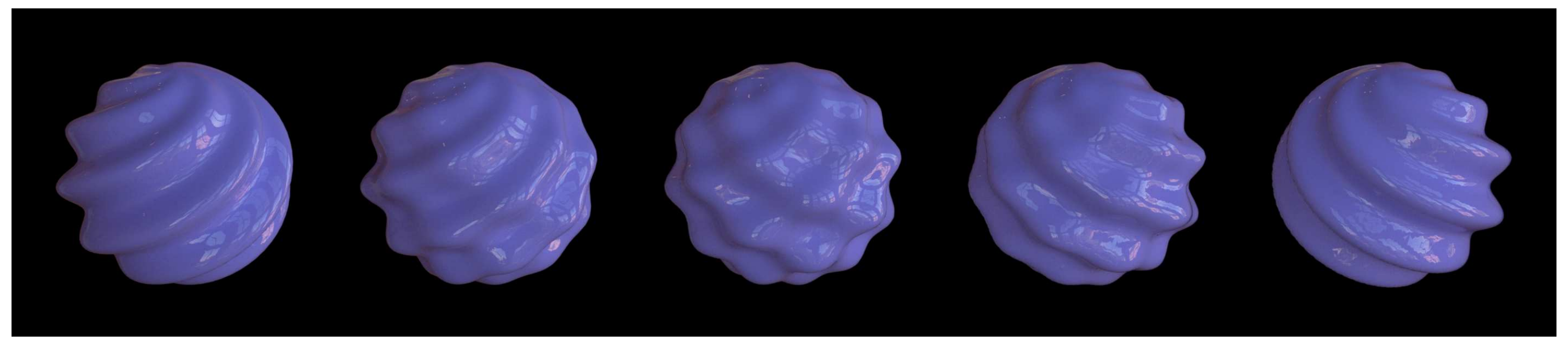
`objMakeCustom('plane',...)`
- custom perturbation function



Parametric series

- Change perturbation amplitude to create a series of shapes:

```
a = 0:.025:.1;
for ii = 1:length(a)
  m{ii} = objMakeSine('sphere',...
                      [12 -60 0 a(ii); 12 60 0 .1-a(ii)],...
                      [2 90 90 1]);
end
```



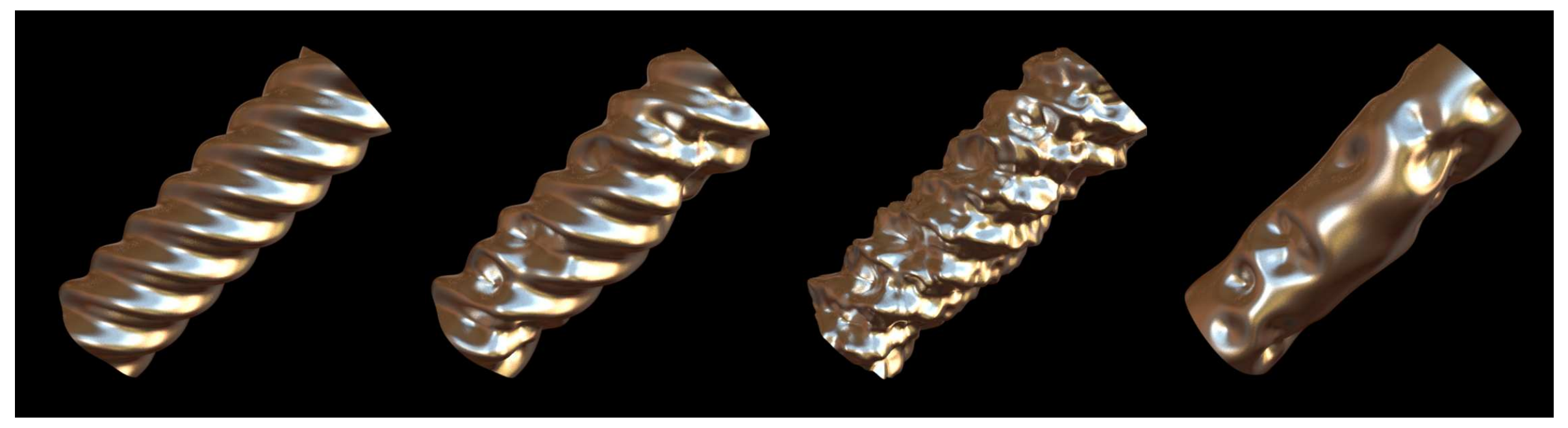
Chaining perturbations

- Combine several types of perturbation in a single model
- Model structure given as input to another `objMake*`-function:

```
m = objMakeSine('cylinder',[8 60 0 .1]);        % sine, left panel
m = objMakeBump(m,[50 pi/8 .1; 50 pi/16 -.2]);  % add bumps & dents
m = objMakeNoise(m,[16 1 0 Inf .1]);            % add noise
```

- "Turn off" some of the perturbations using `objSet`:

```
m = objSet(m,'use_perturbation',[0 1 0]); % sine & noise off, right panel
```

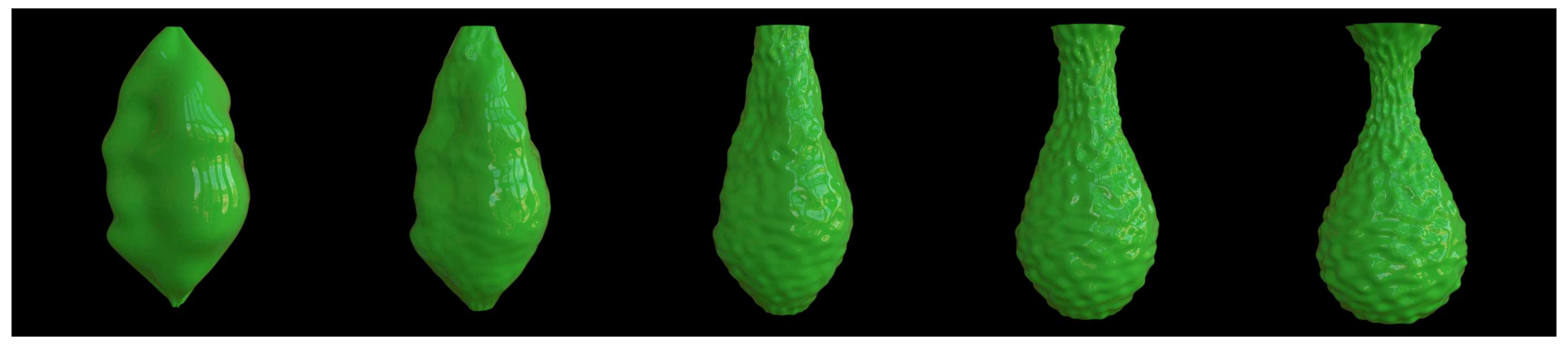


Blending

- Blend two shapes in arbitrary proportions:

```
m1 = objMakeBump('revolution',...)
m2 = objMakeNoise('revolution',...)
w = [0 .25 .5 .75 1];
```

```
for ii = 1:length(w)
  m{ii} = objBlend(m1,m2,w(ii))
end
```



More complex shapes

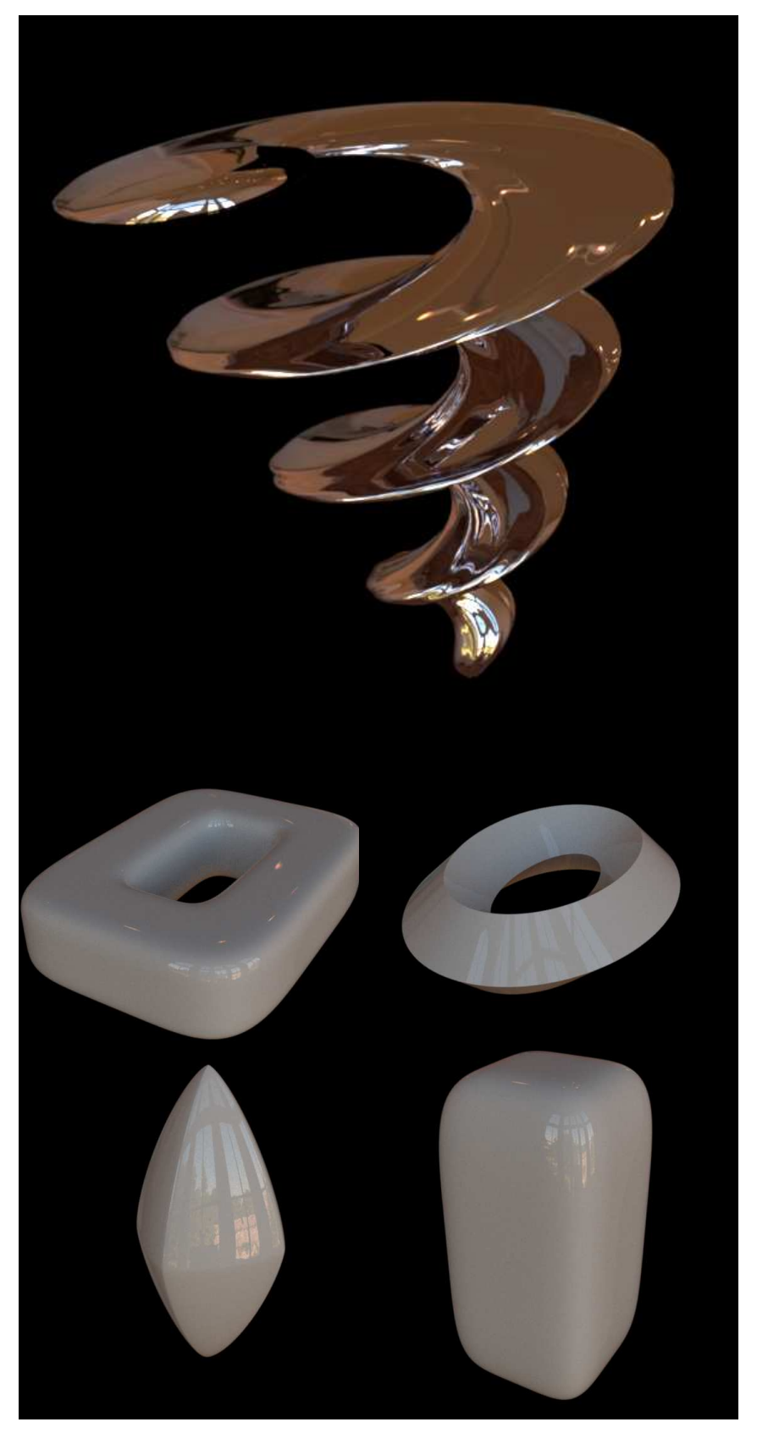
- Surface of revolution, spiral midline:

```
y = linspace(0,1,256);
x = 2*sin(8*pi*y).*y;
z = 2*cos(8*pi*y).*y;

profile = y.^.5;
m = objMakeNoise('revolution',...
                 [8 1 0 30 .03],...
                 'rcurve',profile,...
                 'spinex',x,'spinez',z,...
                 'caps',true);
```

- "Super-torus" and "super-ellipsoid":

```
tor = objMakePlain('torus','super',[.5 .5],...
                   'radius',[1 2]);
ell = objMakePlain('ellipsoid','super',[1.5 1.5]);
```



GUIs and other options

- The functions are most useful and versatile when called from your own code, but
- There are two graphical tools that help in designing new shapes:
  - `objDesigner` — GUI for creating models
  - `objBlendGui` — GUI for blending models
- Several other options and tools for, e.g.
  - Vertex groups
  - Surface normals
  - Texture coordinates

Prepare for 3D printing

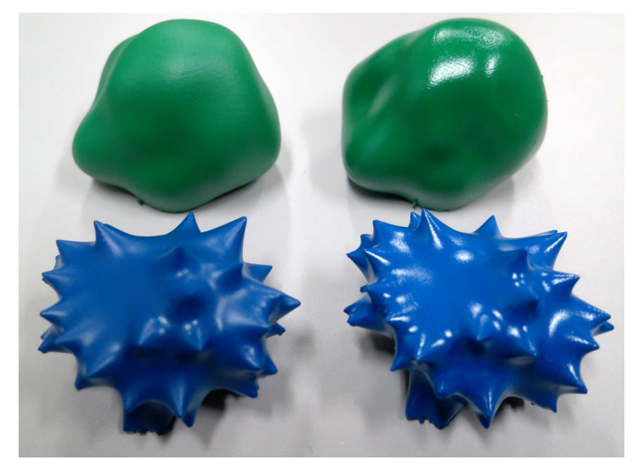
Tools for

- Adding wall thickness
- Scaling the model
- Cutting a piece out (for SLA printing, to avoid a "suction effect" with a closed, hollow shape)

```
% Make a sphere with bumps and dents
m = objMakeBump('sphere',[50 pi/16 .1; 50 pi/16 -.1]);
m = objScale(m,20);       % scale to bigger size
m = objCutSphere(m,16);   % cut a piece out
m = objAddThickness(m,5); % wall thickness
m = objSet(m,'filename','sphere_for_print');
objSave(m);               % save
```

Rendered

3D-printed



All images rendered with Mitsuba renderer: Jakob (2010). Mitsuba renderer. http://www.mitsuba-renderer.org.

toni.saarela@helsinki.fi