Catalogues: Basic Tutorial

What are *Catalogues*?

Catalogues are objects used to keep track of the tomograms involved in subtomogram averaging experiments.

They provide a framework to construct models and pick particles, facilitating the retrieval of information and the importing of work carried on third party software.

In this tutorial

We will create a synthetic set of tomograms, and show how they can be integrated into a *catalogue* for visualization.

Along with the tomograms, metadata files (in) will be also created. These metadata files code positions and angles of particles in the tomograms, and the tutorial command will generate different formats. We will see how to import them.

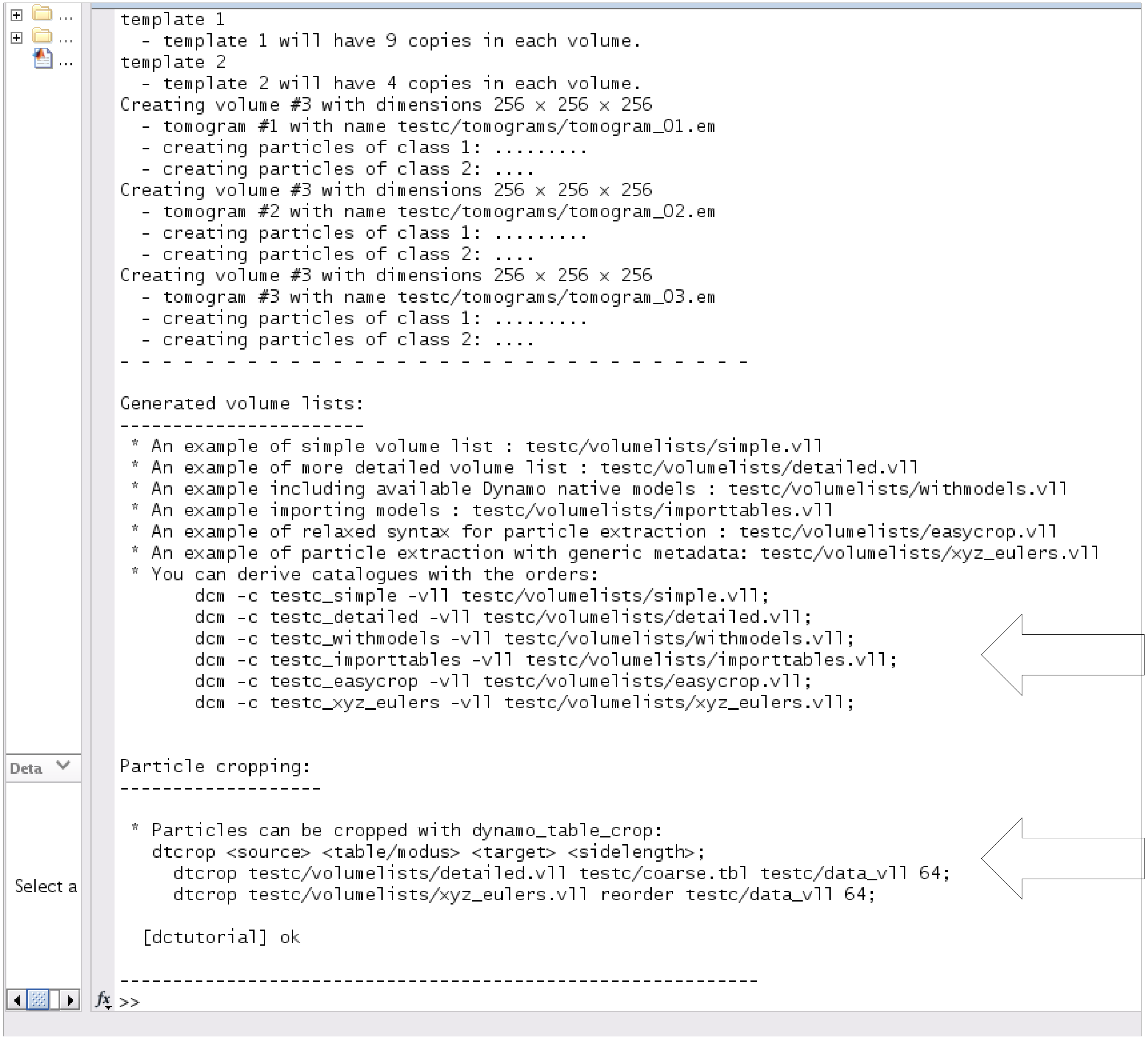
Also, we will see how tomographic properties (defocus, fourier sampling description,etc) can be input into the catalogued information.

So, type in your Matlab or Standalone Dynamo console:

```
>> dctutorial testct -n 3 -tc [9,4]
```

- * This generates 3 volumes, each one with 2 templates (one with 9, the other with 4 copies)
- * Use `help dctutorial` for further options.

Lots of things happen... Actually the tutorial program already suggest us things that we could do in order to get familiar with the functionalities of the tutorial tools....



```
template 1
  - template 1 will have 9 copies in each volume.
template 2
  - template 2 will have 4 copies in each volume.
Creating volume #3 with dimensions 256 x 256 x 256
  - tomogram #1 with name testc/tomograms/tomogram_01.em
  - creating particles of class 1: .........
  - creating particles of class 2: ....
Creating volume #3 with dimensions 256 x 256 x 256
  - tomogram #2 with name testc/tomograms/tomogram_02.em
  - creating particles of class 1: .........
  - creating particles of class 2: ....
Creating volume #3 with dimensions 256 x 256 x 256
  - tomogram #3 with name testc/tomograms/tomogram_03.em
  - creating particles of class 1: .........
  - creating particles of class 2: ....
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Generated volume lists:
-----------------------
 * An example of simple volume list : testc/volumelists/simple.vll
 * An example of more detailed volume list : testc/volumelists/detailed.vll
 * An example including available Dynamo native models : testc/volumelists/withmodels.vll
 * An example importing models : testc/volumelists/importtables.vll
 * An example of relaxed syntax for particle extraction : testc/volumelists/easycrop.vll
 * An example of particle extraction with generic metadata: testc/volumelists/xyz_eulers.vll
 * You can derive catalogues with the orders:
       dcm -c testc_simple -vll testc/volumelists/simple.vll;
       dcm -c testc_detailed -vll testc/volumelists/detailed.vll;
       dcm -c testc_withmodels -vll testc/volumelists/withmodels.vll;
       dcm -c testc_importtables -vll testc/volumelists/importtables.vll;
       dcm -c testc_easycrop -vll testc/volumelists/easycrop.vll;
       dcm -c testc_xyz_eulers -vll testc/volumelists/xyz_eulers.vll;


Particle cropping:
-------------------

 * Particles can be cropped with dynamo_table_crop:
   dtcrop <source> <table/modus> <target> <sidelength>;
     dtcrop testc/volumelists/detailed.vll testc/coarse.tbl testc/data_vll 64;
     dtcrop testc/volumelists/xyz_eulers.vll reorder testc/data_vll 64;

  [dctutorial] ok

------------------------------------------------------------
>>
```

Examples of orders to construct catalogues onto the generated tomograms

Examples of orders to crop particles from the generated without using catalogues

... but we just ignore them and take a look onto the generated data set:

Remember that we did not generate a catalogue, but just some set of data to play with/
The contents of this folder have nothing to do with the internal structure of a catalogue folder.
... and anyway, the idea is that you should *not* need to know the catalogue folder structure to browse in it!

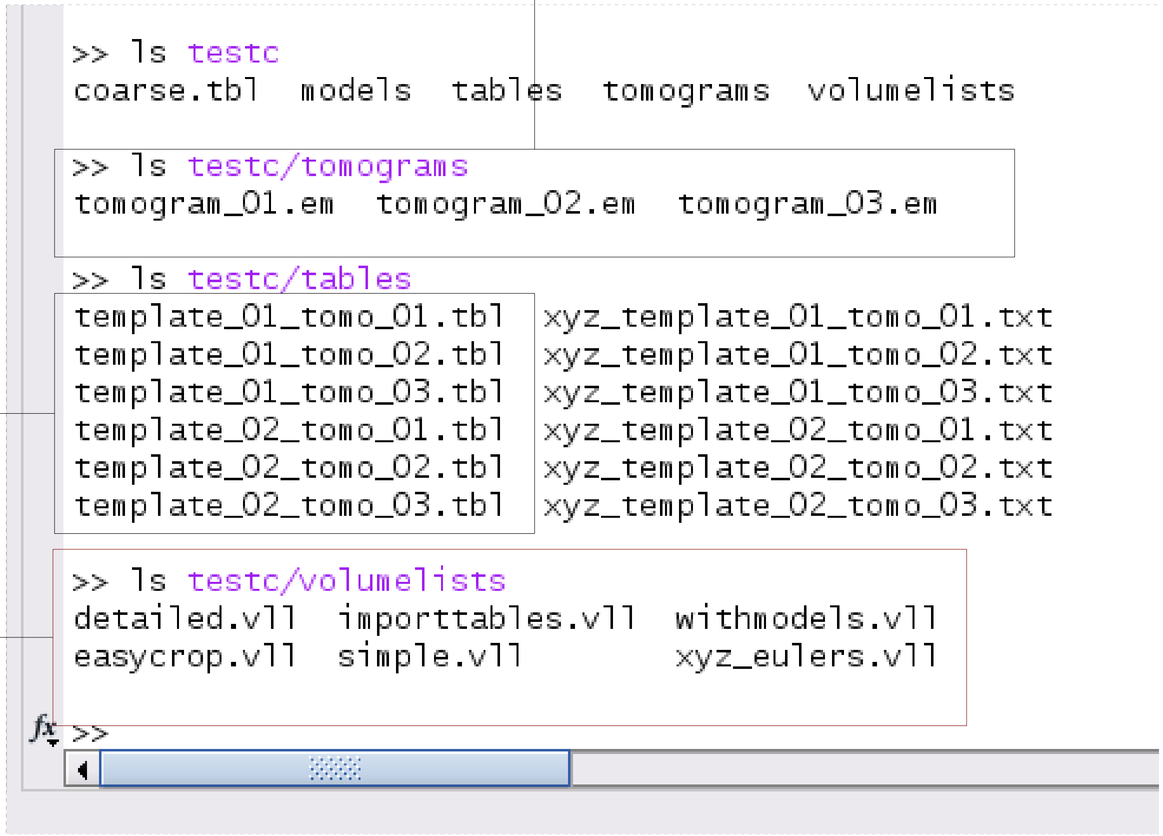
Those are just some small tomograms (256 pixels of sidelength)

These are Dynamo tables, which code the position of the different particles in the differerent tomograms using the Dynamo table format.
This is a simple text format, you can use dthelp (dynamo_table_help) or dtinfo to know more.

Volume lists

They are the easiest way to create catalogues, and also a good way to extract afterwards information from them.

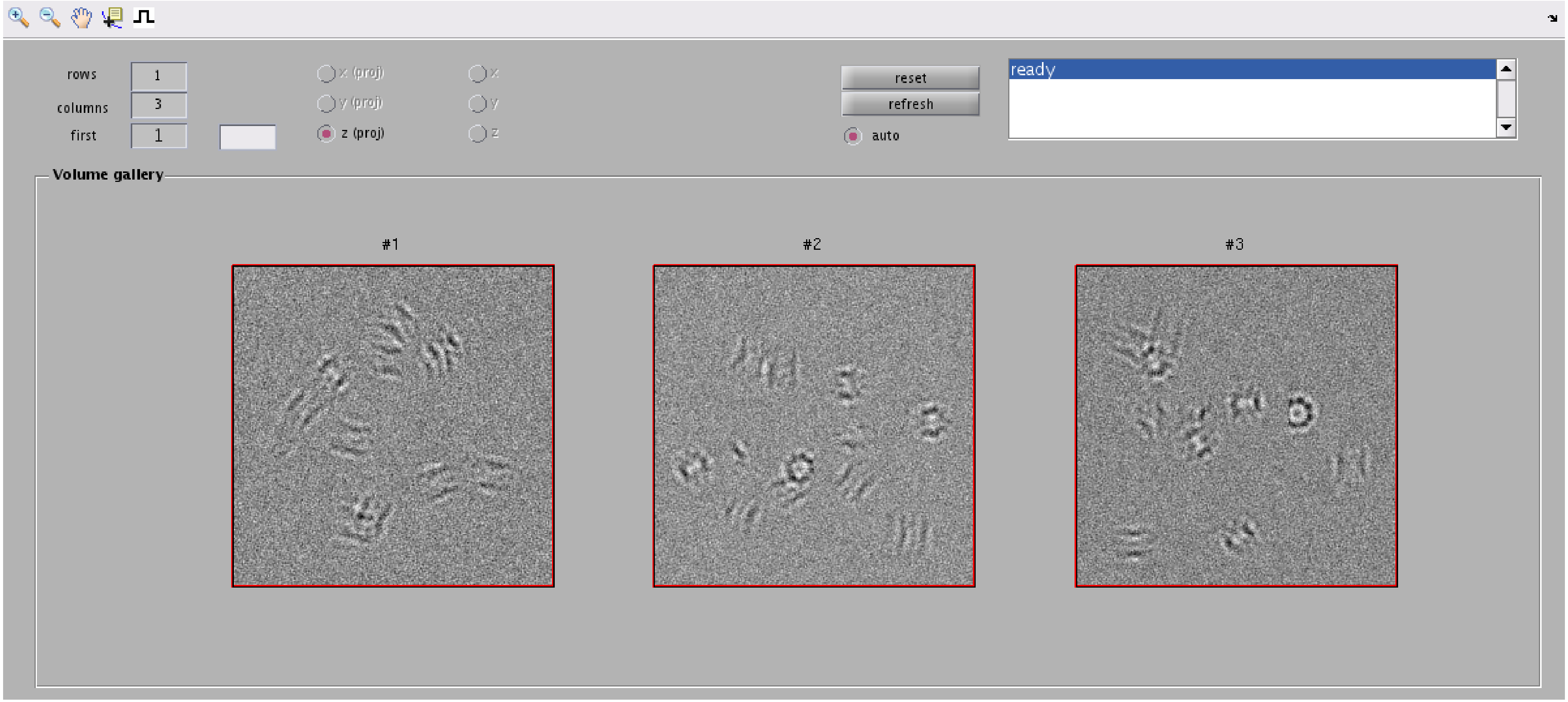
In this tutorial, they simulate how an user would proceed to start to organize his or her data to input it into a catalogue.



```
>> ls testc
coarse.tbl   models   tables   tomograms   volumelists

>> ls testc/tomograms
tomogram_01.em   tomogram_02.em   tomogram_03.em

>> ls testc/tables
template_01_tomo_01.tbl   xyz_template_01_tomo_01.txt
template_01_tomo_02.tbl   xyz_template_01_tomo_02.txt
template_01_tomo_03.tbl   xyz_template_01_tomo_03.txt
template_02_tomo_01.tbl   xyz_template_02_tomo_01.txt
template_02_tomo_02.tbl   xyz_template_02_tomo_02.txt
template_02_tomo_03.tbl   xyz_template_02_tomo_03.txt

>> ls testc/volumelists
detailed.vll   importtables.vll   withmodels.vll
easycrop.vll   simple.vll         xyz_eulers.vll

>>
```

But first, let us start just with a quick glance at the tomogram contents, without any cataloguing yet...

```
>> dpanelview files testc/tomograms/tomogram_*.em -otf on
```



rows 1

columns 3

first 1

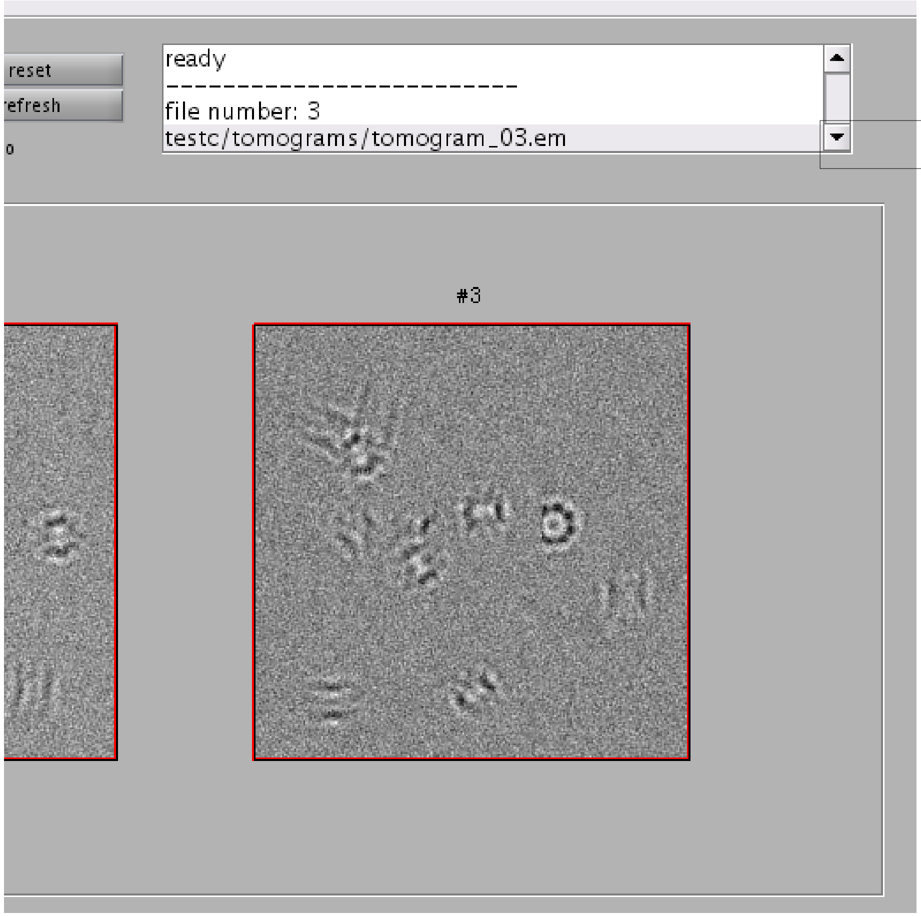
○ x (proj) ○ y (proj) ● z (proj)

○ x ○ y ○ z

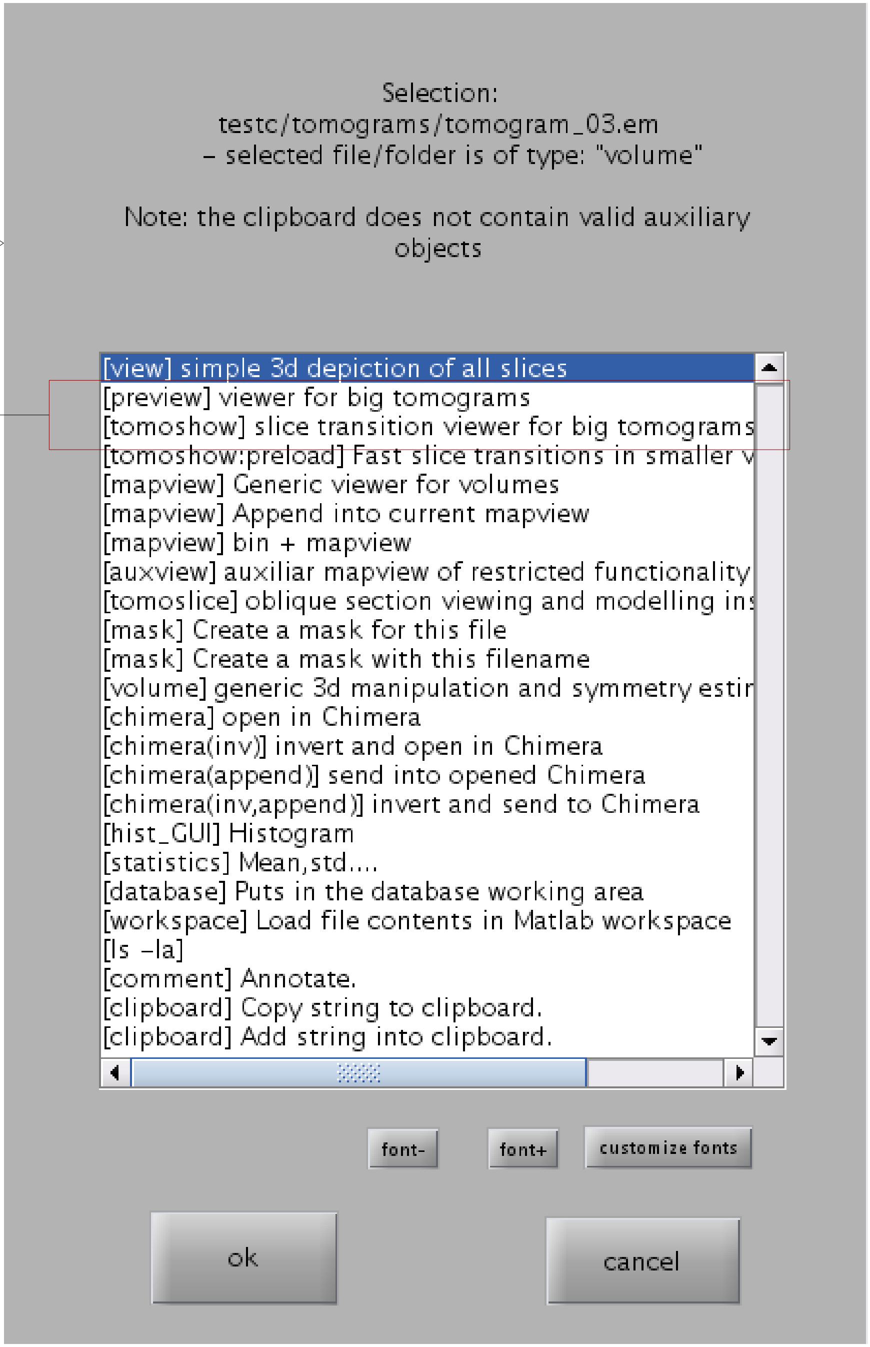
● auto

ready

* This shows for each file (in the regular expression) the projection of all the slices
* They are computed on the fly (parameter “-otf”) to avoid crowding the memory. These tomograms are unrealistically small and this would not be necessary, but in the general case it will.
* You can click on an axis to get the filename in the information window: there you can right click (or CTRL-click in Mac) to get a menu of actions with more options.



```
ready
---------------------------
file number: 3
testc/tomograms/tomogram_03.em
```



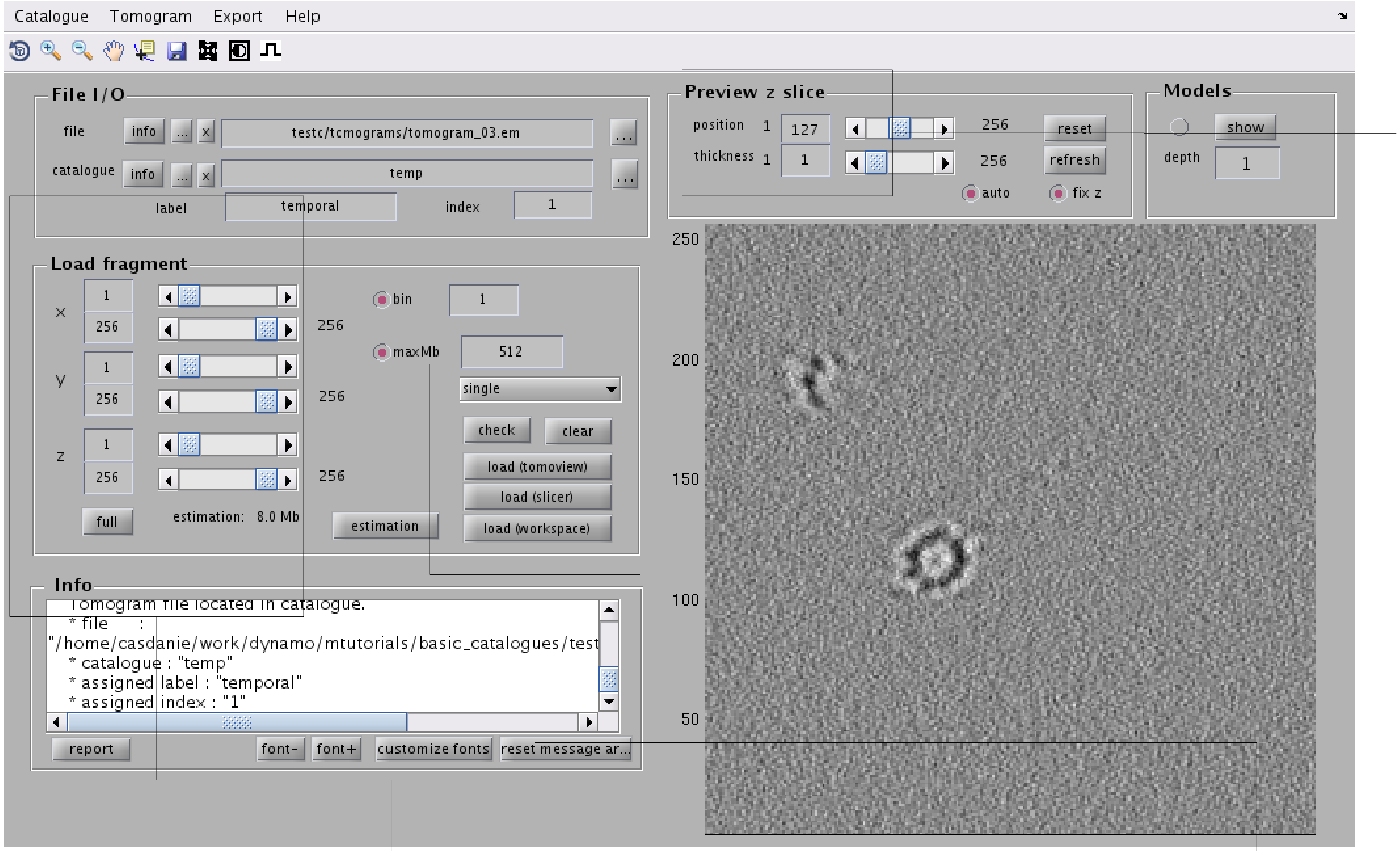
Selection:
testc/tomograms/tomogram_03.em
- selected file/folder is of type: "volume"

Note: the clipboard does not contain valid auxiliary objects

```
[view] simple 3d depiction of all slices
[preview] viewer for big tomograms
[tomoshow] slice transition viewer for big tomograms
[tomoshow:preload] Fast slice transitions in smaller v
[mapview] Generic viewer for volumes
[mapview] Append into current mapview
[mapview] bin + mapview
[auxview] auxiliar mapview of restricted functionality
[tomoslice] oblique section viewing and modelling ins
[mask] Create a mask for this file
[mask] Create a mask with this filename
[volume] generic 3d manipulation and symmetry estir
[chimera] open in Chimera
[chimera(inv)] invert and open in Chimera
[chimera(append)] send into opened Chimera
[chimera(inv,append)] invert and send to Chimera
[hist_GUI] Histogram
[statistics] Mean,std....
[database] Puts in the database working area
[workspace] Load file contents in Matlab workspace
[ls -la]
[comment] Annotate.
[clipboard] Copy string to clipboard.
[clipboard] Add string into clipboard.
```

Some of the options are adequate for viewing tomograms of big size.

For instance this is the preview window, which acts as a Preload Tool for other windows, in order to control beforehand which areas of the tomogram will be loaded onto memory.



file: testc/tomograms/tomogram_03.em
catalogue: temp
label: temporal   index: 1

x: 1, 256 (256)
y: 1, 256 (256)
z: 1, 256 (256)
● bin: 1
● maxMb: 512
single
estimation: 8.0 Mb

```
Tomogram file located in catalogue.
   * file     :
"/home/casdanie/work/dynamo/mtutorials/basic_catalogues/test
   * catalogue : "temp"
   * assigned label : "temporal"
   * assigned index : "1"
```

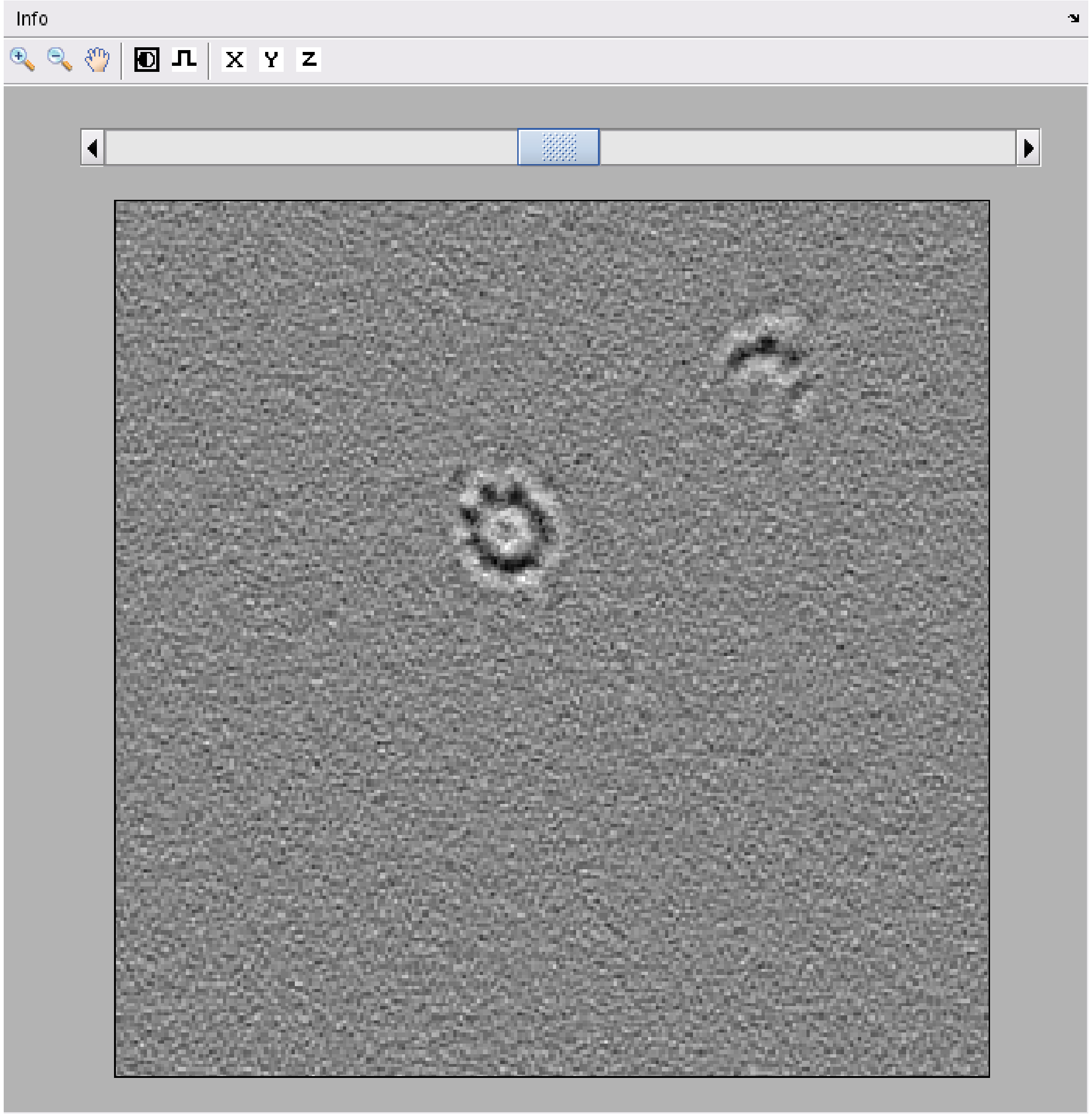
position: 1, 127 (256)
thickness: 1, 1 (256)
● auto   ● fix z

○ depth: 1

selects the height of the visualized z-slice

Other graphical windows

Selects the boundaries of subvolume of interest, which can be then loaded into some other window for more precise visualization and/or modelling tomoshow more is indicated to depict fast transitions of slices across the volume



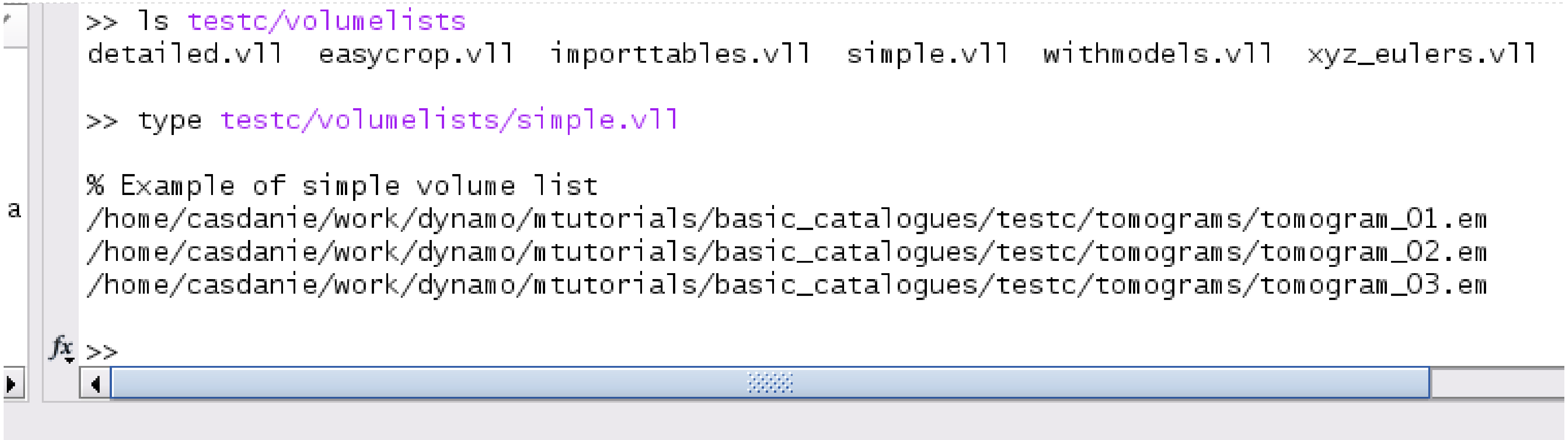
In the “offline” version (with preload of the volume to memory), transitions are equally fast in x y and z.

You can try it in this tutorial, but bear in mind that this can cause problems in larger tomograms/

In the “on the fly” version slices are read in the moment of depiction: everything is thus slower, specially for x and y views, where two subsequent slices do not lie sequentially in the hard disk.

Now that we know how the volumes look like, we come back to the catalogue creation.

We start with one the .vll files that were generated through the tutorial: simple.vll



```
>> ls testc/volumelists
detailed.vll  easycrop.vll  importtables.vll  simple.vll  withmodels.vll  xyz_eulers.vll

>> type testc/volumelists/simple.vll

% Example of simple volume list
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_01.em
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_02.em
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_03.em

>>
```

This simple vll file rcorresnponds be the most basic way an user can employ to keep track of their tomograms: just writing a text file with the name of all the files of interest.

Now, we create an actual catalogue out of this information.

Just by copying and pasting from the screen one the actions suggested by the tutorial:

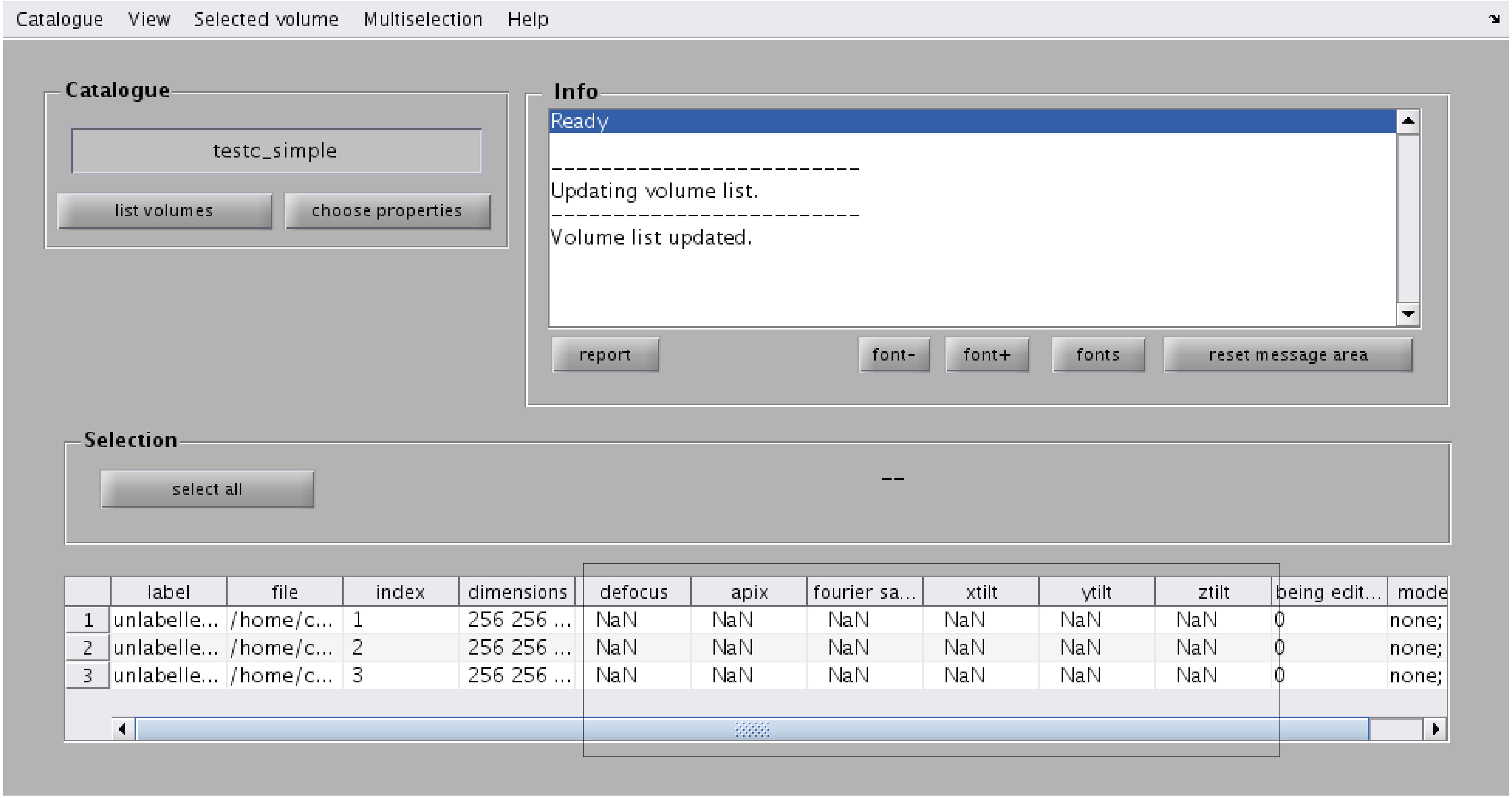
```
>> dcm -c testc_simple -vll  testc/volumelists/simple.vll;
```

`dcm is` the short name for the `dynamo_catalogue_manager` command. Here, we let Dynamo parse the contents of the vll file to create a new catalogue in disk (named `testc_simple`), which we can then open simply by:

```
>> dcm -c my_catalogue;
```

we can start to explore the contents of the catalogue by opening it:

```
>> dcm -c my_catalogue;
```



testc_simple

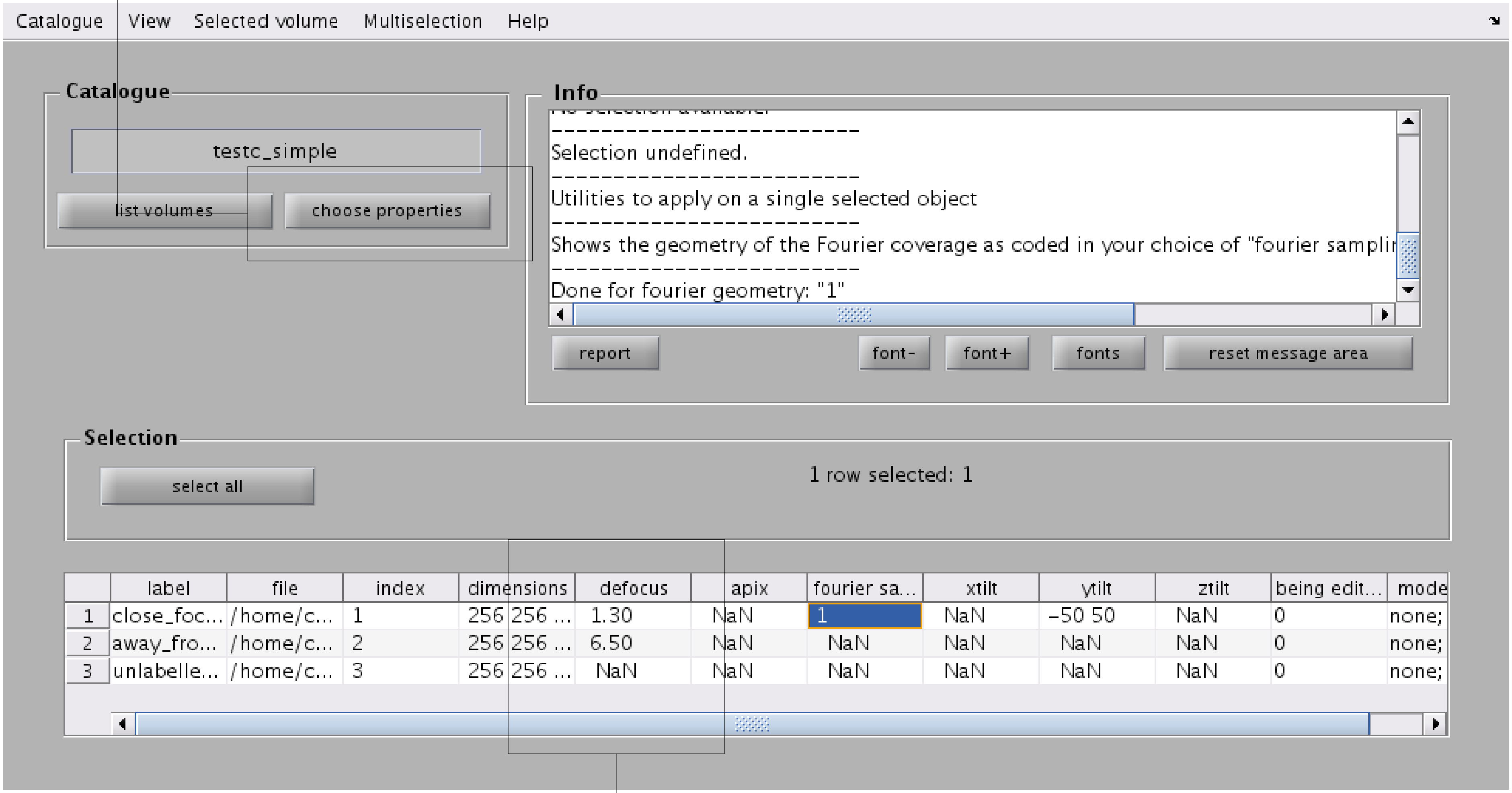
```
Ready

---------------------------
Updating volume list.
---------------------------
Volume list updated.
```

--

	label	file	index	dimensions	defocus	apix	fourier sa...	xtilt	ytilt	ztilt	being edit...	mode
1	unlabelle...	/home/c...	1	256 256 ...	NaN	NaN	NaN	NaN	NaN	NaN	0	none;
2	unlabelle...	/home/c...	2	256 256 ...	NaN	NaN	NaN	NaN	NaN	NaN	0	none;
3	unlabelle...	/home/c...	3	256 256 ...	NaN	NaN	NaN	NaN	NaN	NaN	0	none;

It has recognized the files and assigned indices, but no tomographic metadata is there yet.
Any information not contained in the tomograms themselves has been initialized to default values.

You can choose which properties attached to each tomogram are visualized for edition here



```
--------------------------
Selection undefined.
--------------------------
Utilities to apply on a single selected object
--------------------------
Shows the geometry of the Fourier coverage as coded in your choice of "fourier sampli
--------------------------
Done for fourier geometry: "1"
```

1 row selected: 1

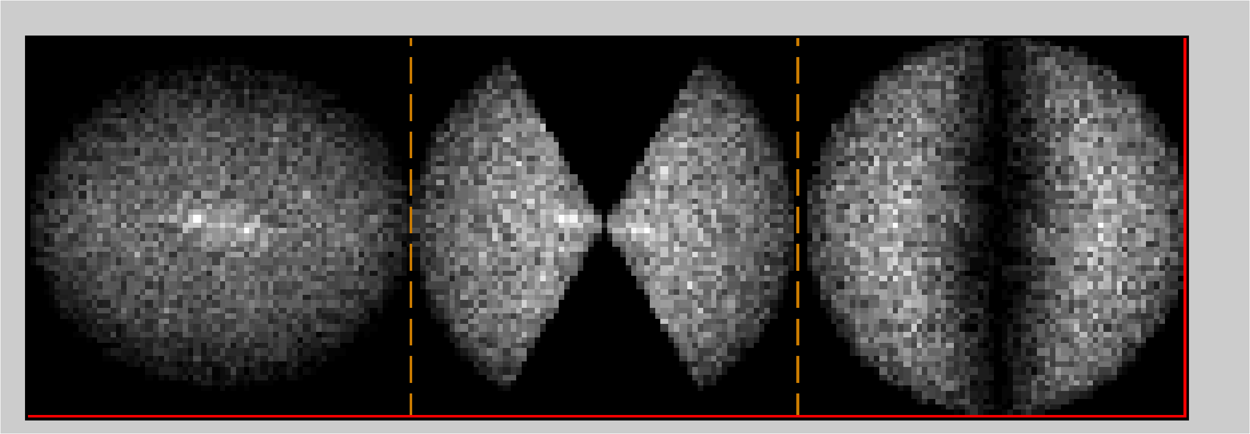
	label	file	index	dimensions	defocus	apix	fourier sa...	xtilt	ytilt	ztilt	being edit...	mode
1	close_foc...	/home/c...	1	256 256 ...	1.30	NaN	1	NaN	-50 50	NaN	0	none;
2	away_fro...	/home/c...	2	256 256 ...	6.50	NaN	NaN	NaN	NaN	NaN	0	none;
3	unlabelle...	/home/c...	3	256 256 ...	NaN	NaN	NaN	NaN	NaN	NaN	0	none;

If you input your values for the properties here, they will get stored into the catalogue under edition.

They can get transmitted to any objects derived from the catalogue.

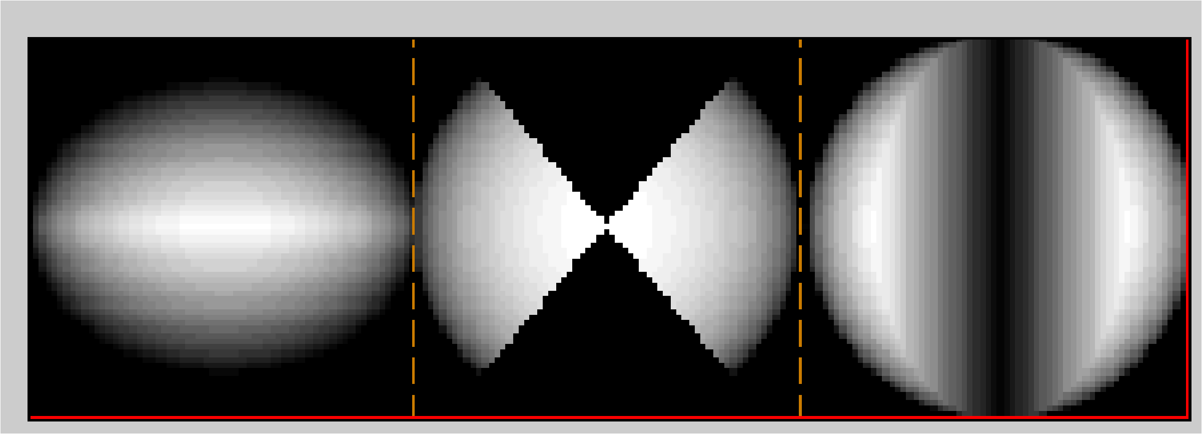
Besides the creation of the database and assignment of properties, the front GUI links the volumes to some basic utilities for visualization (tomoshow in Dynamo or the different windows of Imod, if you have it installed in your system).

The tool that analizes a sample of the tomogram to check the apparent wissing wedge geometry is useful mainly for formatting purposes, as it allows to check if the missing wedge in the tomogram:



is correctly described by the missing wedge that we are introducing as descriptor.

in the example we had indicated a fsampling code of “1” (which means: beam along z, tilt around y).
'An idealized missing wedge looks like this:

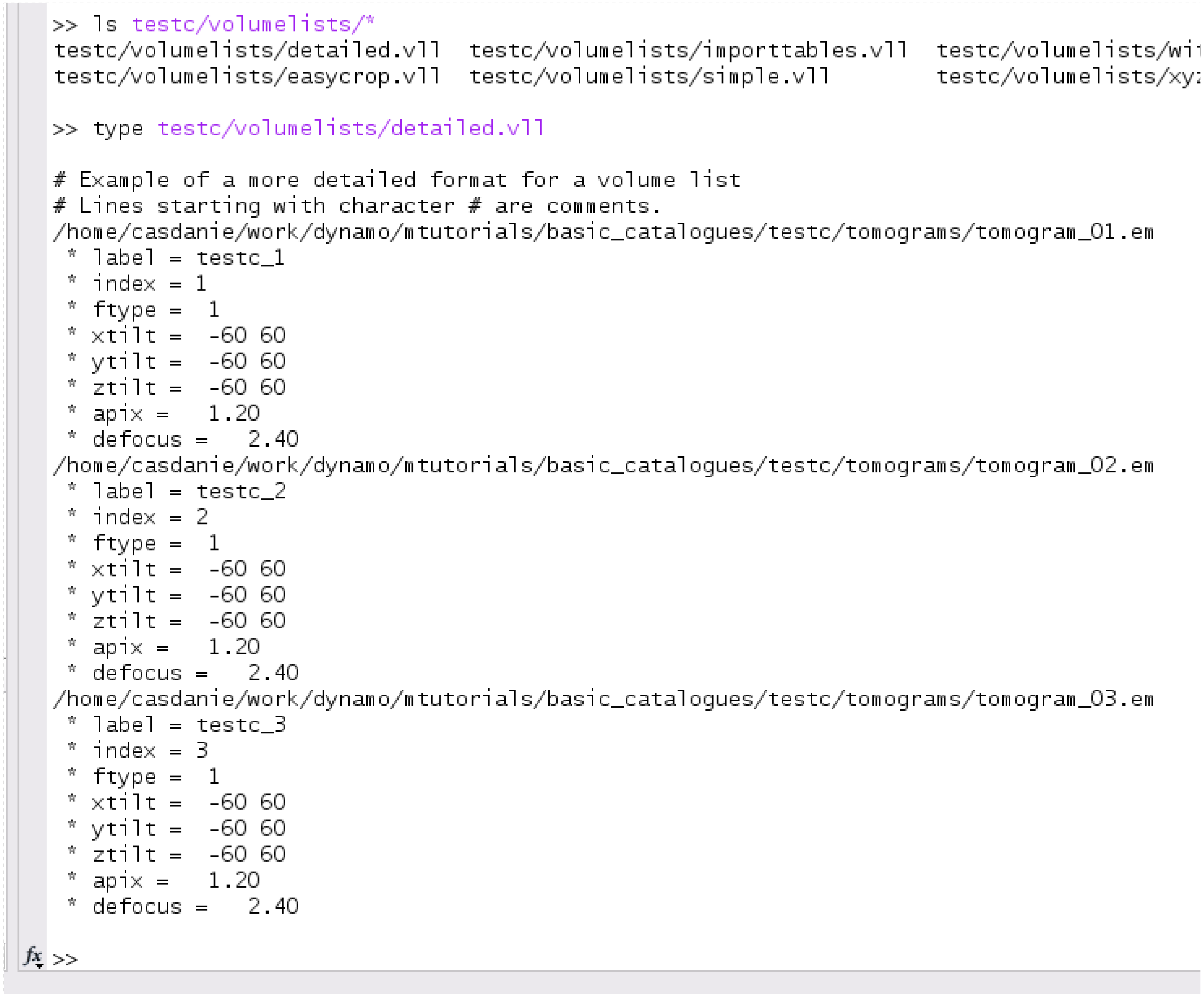


Passing parameters with the volume list

Volume lists can have a more detailed syntax. Let us check one the sample vlls created by the tutorial:

All lines between tomogram names are orders or attributes of the tomogram.

Lines starting with “*” are assignations:
* ftype = 1
assigns the value “ftype” to the property ftype of the previous tomogram.



```
>> ls testc/volumelists/*
testc/volumelists/detailed.vll  testc/volumelists/importtables.vll  testc/volumelists/wi
testc/volumelists/easycrop.vll  testc/volumelists/simple.vll        testc/volumelists/xy

>> type testc/volumelists/detailed.vll

# Example of a more detailed format for a volume list
# Lines starting with character # are comments.
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_01.em
 * label = testc_1
 * index = 1
 * ftype =  1
 * xtilt =  -60 60
 * ytilt =  -60 60
 * ztilt =  -60 60
 * apix =   1.20
 * defocus =   2.40
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_02.em
 * label = testc_2
 * index = 2
 * ftype =  1
 * xtilt =  -60 60
 * ytilt =  -60 60
 * ztilt =  -60 60
 * apix =   1.20
 * defocus =   2.40
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_03.em
 * label = testc_3
 * index = 3
 * ftype =  1
 * xtilt =  -60 60
 * ytilt =  -60 60
 * ztilt =  -60 60
 * apix =   1.20
 * defocus =   2.40

>>
```

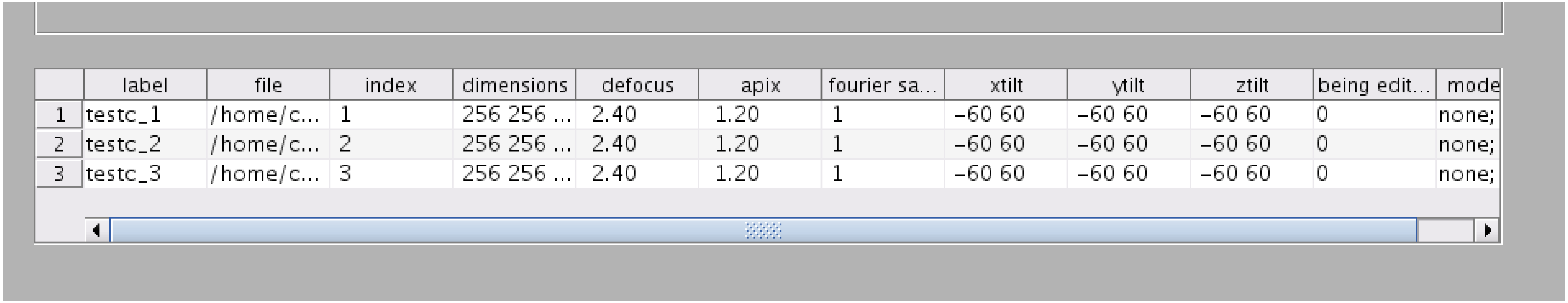
We can operate on this table like before, using:

```
>> dcm -c  detailed -vll testc/volumelists/detailed.vll;
>> dcm -c  detailed;
```

or simply

```
>> dgui testc/volumelists/detailed.vll;
```

(the default *Dynamo* actionon a.vll file is parsing it for catalogue creation and opening it)

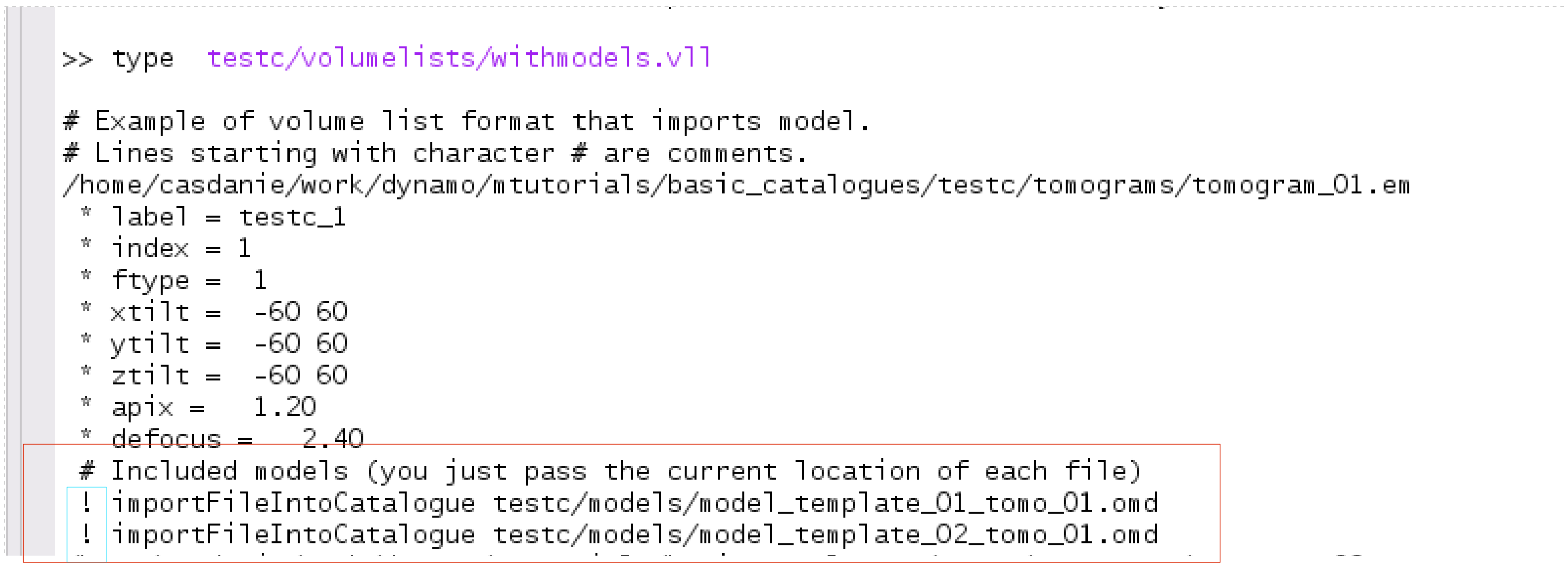


	label	file	index	dimensions	defocus	apix	fourier sa...	xtilt	ytilt	ztilt	being edit...	mode
1	testc_1	/home/c...	1	256 256 ...	2.40	1.20	1	-60 60	-60 60	-60 60	0	none;
2	testc_2	/home/c...	2	256 256 ...	2.40	1.20	1	-60 60	-60 60	-60 60	0	none;
3	testc_3	/home/c...	3	256 256 ...	2.40	1.20	1	-60 60	-60 60	-60 60	0	none;

When we open the catalogue, we see that the values of the parameters in the volume list file have been written into the catalogue.

Importing models: the “!” command

But catalogues are mostly intended to keep track of *models* and objects defined inside the tomograms. Let us take a look into a volume list that informs the created catalogue about some model files that need to be linked into the tomograms.



```
>> type  testc/volumelists/withmodels.vll

# Example of volume list format that imports model.
# Lines starting with character # are comments.
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_01.em
 * label = testc_1
 * index = 1
 * ftype =  1
 * xtilt =  -60 60
 * ytilt =  -60 60
 * ztilt =  -60 60
 * apix =   1.20
 * defocus =   2.40
 # Included models (you just pass the current location of each file)
 ! importFileIntoCatalogue testc/models/model_template_01_tomo_01.omd
 ! importFileIntoCatalogue testc/models/model_template_02_tomo_01.omd
```

The format here is the same, but includes lines that start with “!”. Those are commands imparted to Dynamo during creation of the catalogue. import FileIntoCatalogue just tells Dynamo that file in the same line contains metadata for the corresponding tomogram (the file at the top of the block).

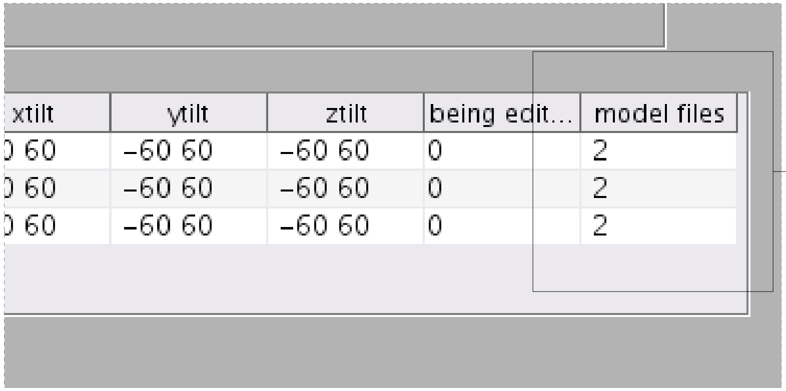
In this case, the metadata are Dynamo models, but you can pass different formats. More information in:

```
>> help cvolume.importFileIntoCatalogue
```

File names are arbitrary, do not need any special convention.

In any case, if you create a catalogue for this vll and open it:

```
>> dgui tersc/volumelists/withmodels.vll
```

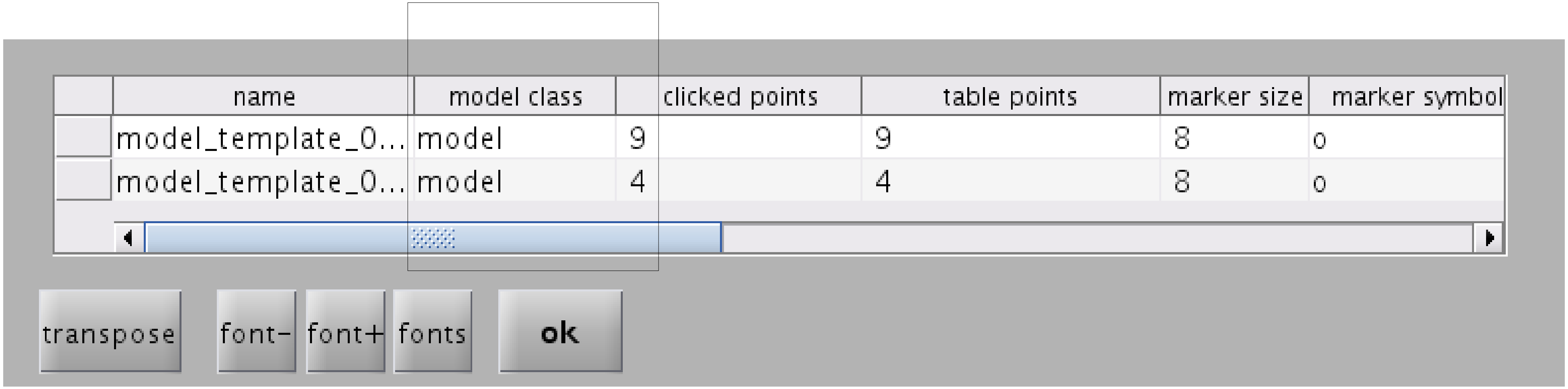


xtilt	ytilt	ztilt	being edit...	model files
0 60	-60 60	-60 60	0	2
0 60	-60 60	-60 60	0	2
0 60	-60 60	-60 60	0	2

you'll see that the two files that you asked for in each volume are being seen by the catalogue.

Under [Selected Volume] you can also find a tool to open a summary of all the models found for a given tomogram.

We don't care about most parameters right know: A whole lot of them are there just for depiction settings.



	name	model class		clicked points	table points	marker size	marker symbol
	model_template_0...	model	9		9	8	o
	model_template_0...	model	4		4	8	o

... but just to stop for the most important ones:

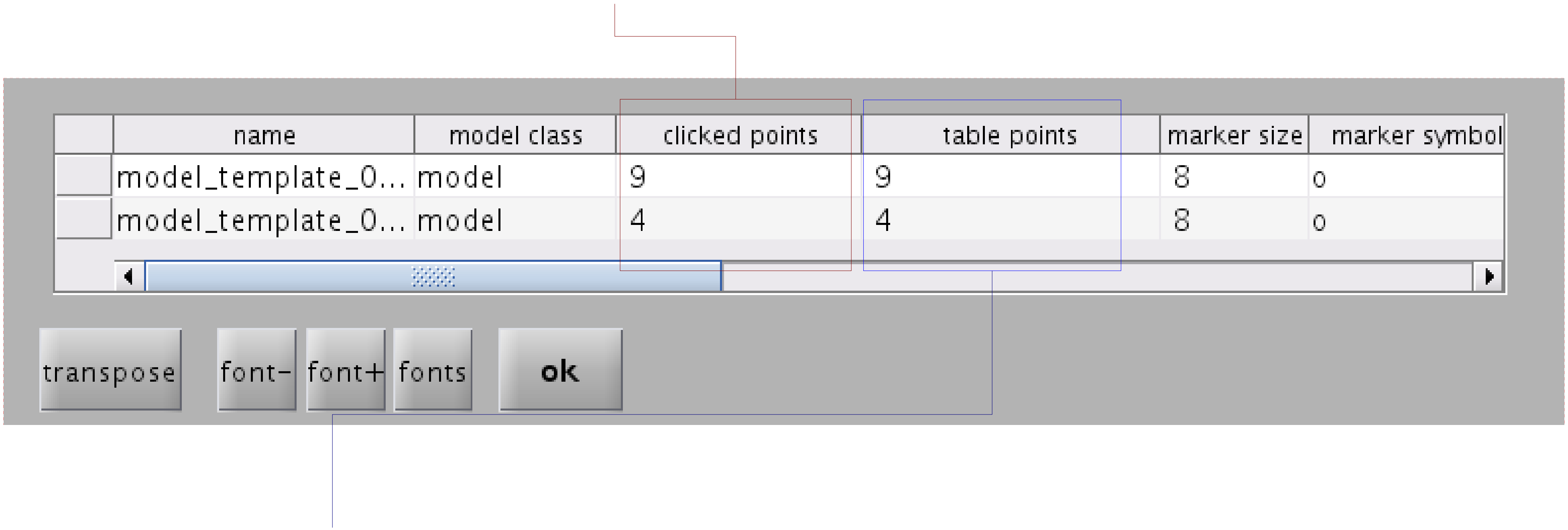
The class of the model in this case is the generic “model”... there some other classes for different tasks and to describe different particle collection geometries.

... and the points!

Current number of CLICKED POINTS

* Points “clicked” by the user.
In this case they were generated by the tutorial and imported by the catalogue.

They are NOT ALWAYS points that mark the center of a particle intended to be cropped!
Sometimes the model requires several steps between the positions clicked on screen and the estimated location of the particles: for instance with particles lying in a membrane you click on visilble boundaries of the membrane, then define a membrane and then generate a table.... but we'll see all of this later....

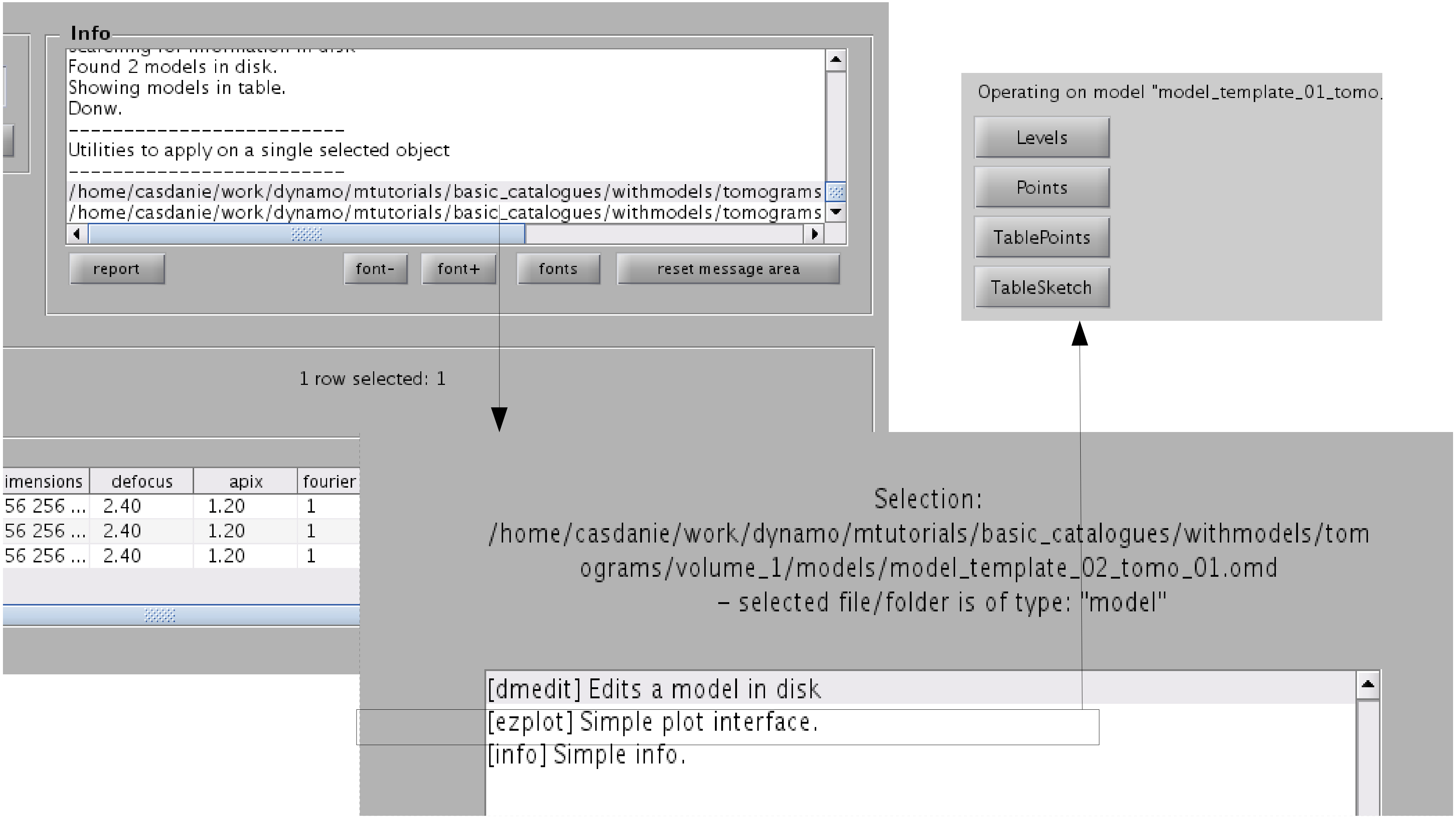


	name	model class	clicked points	table points	marker size	marker symbol
	model_template_0...	model	9	9	8	o
	model_template_0...	model	4	4	8	o

Current numebr of TABLE POINTS

* At some point you'll need models that contain table points: these points are the ones that will appear in “tables” used to crop data our of the tomograms and feed the subtomogram averaging refinement

You can start to get a feeling on how the model works just by listing them in the information window and right clicking on the name of file to get a menu of selected actions associated with a model file.



Info

```
Found 2 models in disk.
Showing models in table.
Donw.
--------------------------
Utilities to apply on a single selected object
--------------------------
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/withmodels/tomograms
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/withmodels/tomograms
```

1 row selected: 1

imensions	defocus	apix	fourier
56 256 ...	2.40	1.20	1
56 256 ...	2.40	1.20	1
56 256 ...	2.40	1.20	1

Selection:
/home/casdanie/work/dynamo/mtutorials/basic_catalogues/withmodels/tom
ograms/volume_1/models/model_template_02_tomo_01.omd
- selected file/folder is of type: "model"

[dmedit] Edits a model in disk
[ezplot] Simple plot interface.
[info] Simple info.

Operating on model "model_template_01_tomo

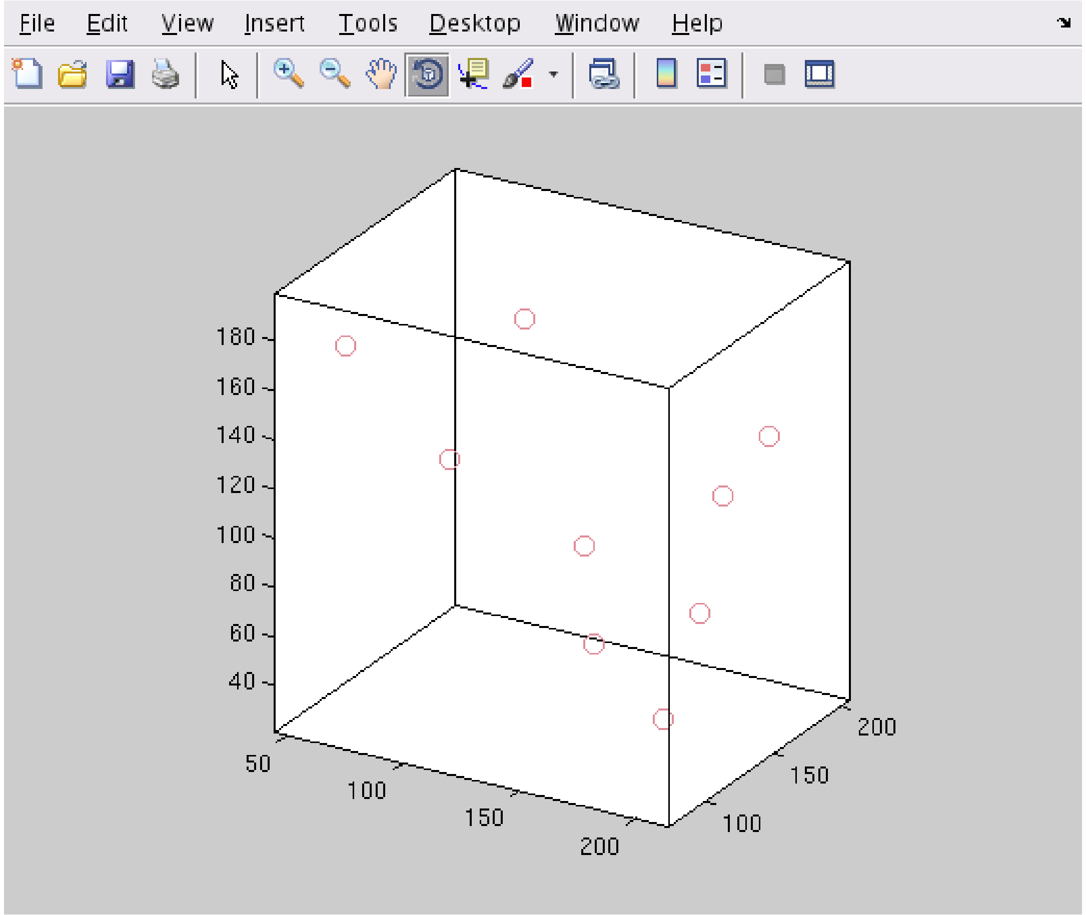
For instance, you can just click on the ezplot utility to get a basic representation of the model points (next screen)

This depiction just plots the model points in three dimensional distribution on a regular MATLAB window.

You can interact with the window in the usual MATLAB way to add annotations, control graphics etc...

But now the question is:

How do we see the models in their context inside the tomograms?



The most immediate way corresponds to the use of dynamo_preview, accessible as Preview/Load Tool in the catalogue manager menu under the [Selected Volume] menu.

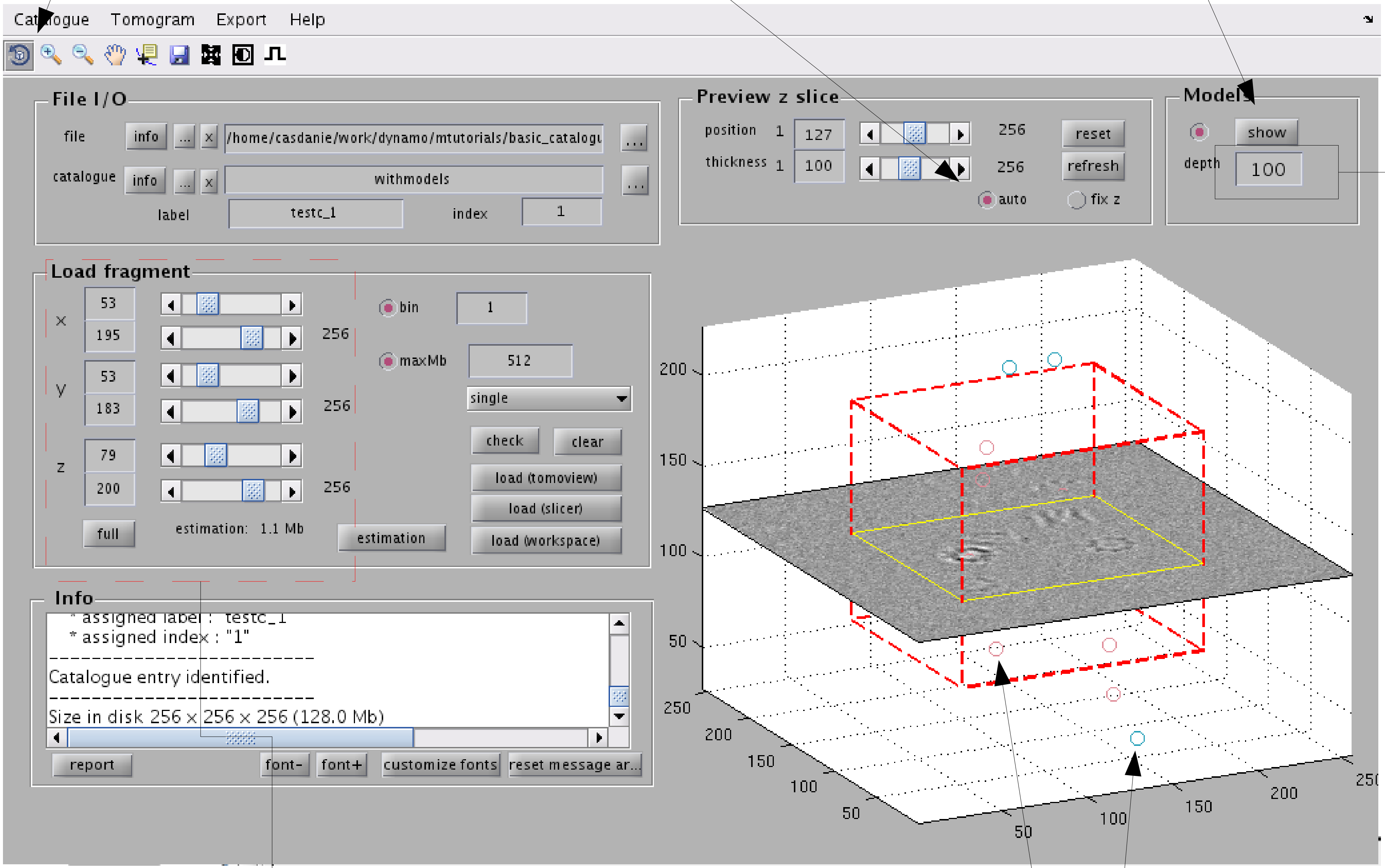
In the slide you'll see a possible representation, with the controls needed to generate it.

rotation tool

allows to use the rotation tool

shows the models attached to this volume

depth of view



file /home/casdanie/work/dynamo/mtutorials/basic_catalogu

catalogue withmodels

label testc_1 index 1

position 1 127 256

thickness 1 100 256

● auto ○ fix z

● depth 100

x	53		
	195		256
y	53		
	183		256
z	79		
	200		256

● bin 1

● maxMb 512

single

estimation: 1.1 Mb

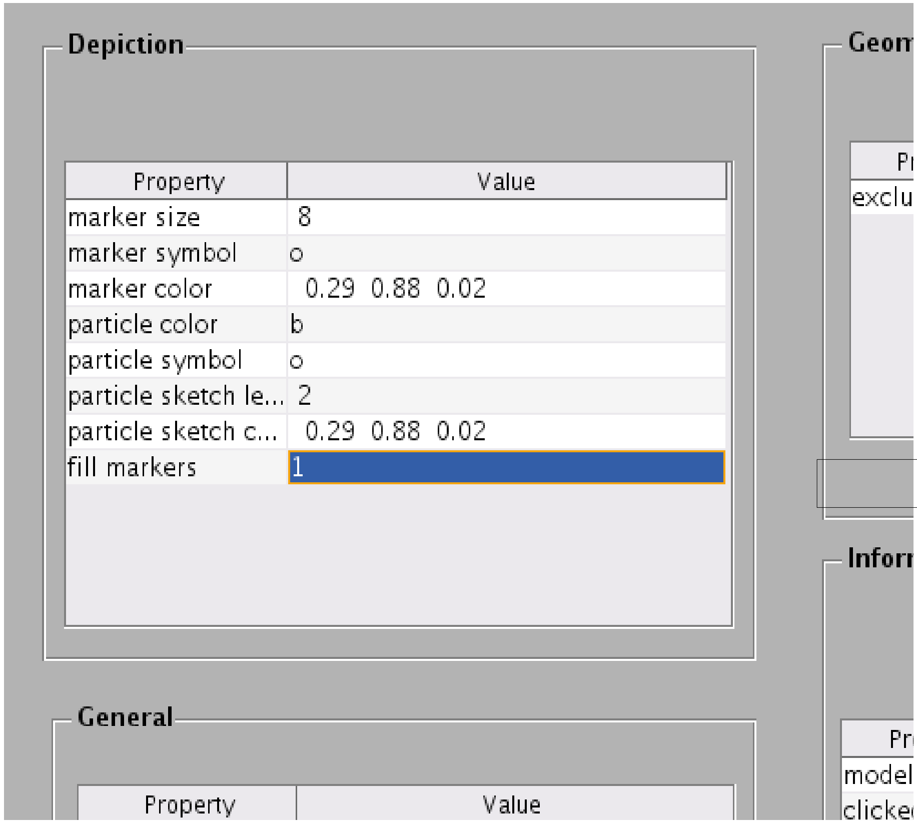
```
  * assigned label :  testc_1
  * assigned index : "1"
-------------------------
Catalogue entry identified.
-------------------------
Size in disk 256 x 256 x 256 (128.0 Mb)
```

Region chosen to be loaded into memory

3d positions of all models found for this volume

You can change the color and other depiction settings of the model using dynamo_model_edit.
you can invoke it cliking on the [tomogram] menu to produce a submenu with, among other options,
the listing of model files found in the catalogue for this volume.

Then, you right-click on the model file of interest to edit its image in disk.
As dynamo_preview reads the model from the disk, clciking the [show] button will update the depiction.

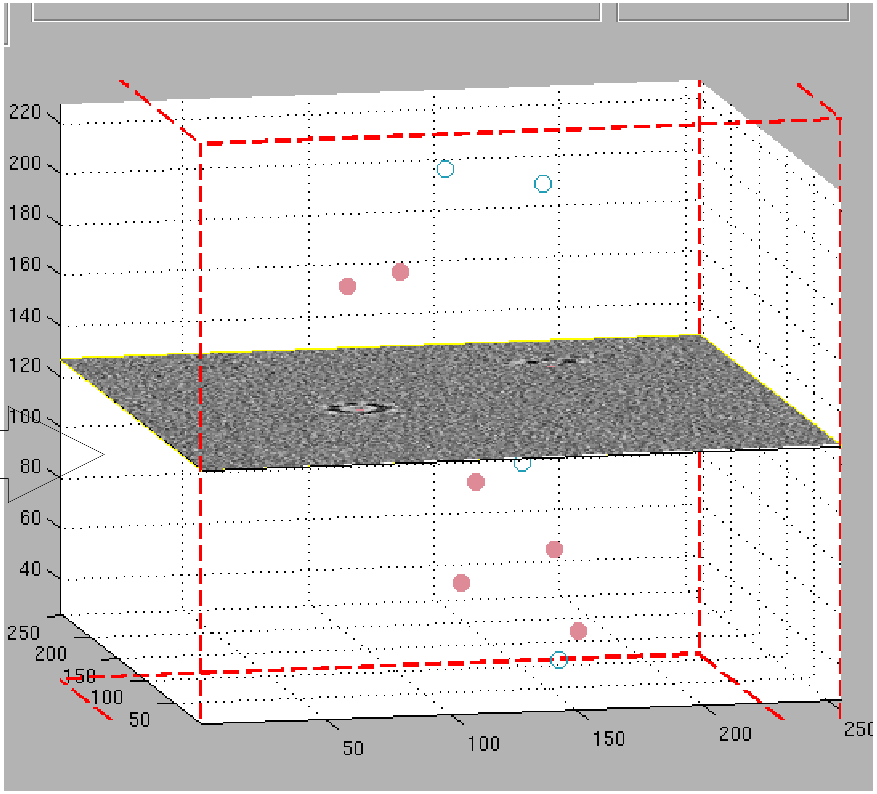


Depiction

Property	Value
marker size	8
marker symbol	o
marker color	0.29 0.88 0.02
particle color	b
particle symbol	o
particle sketch le...	2
particle sketch c...	0.29 0.88 0.02
fill markers	1

General

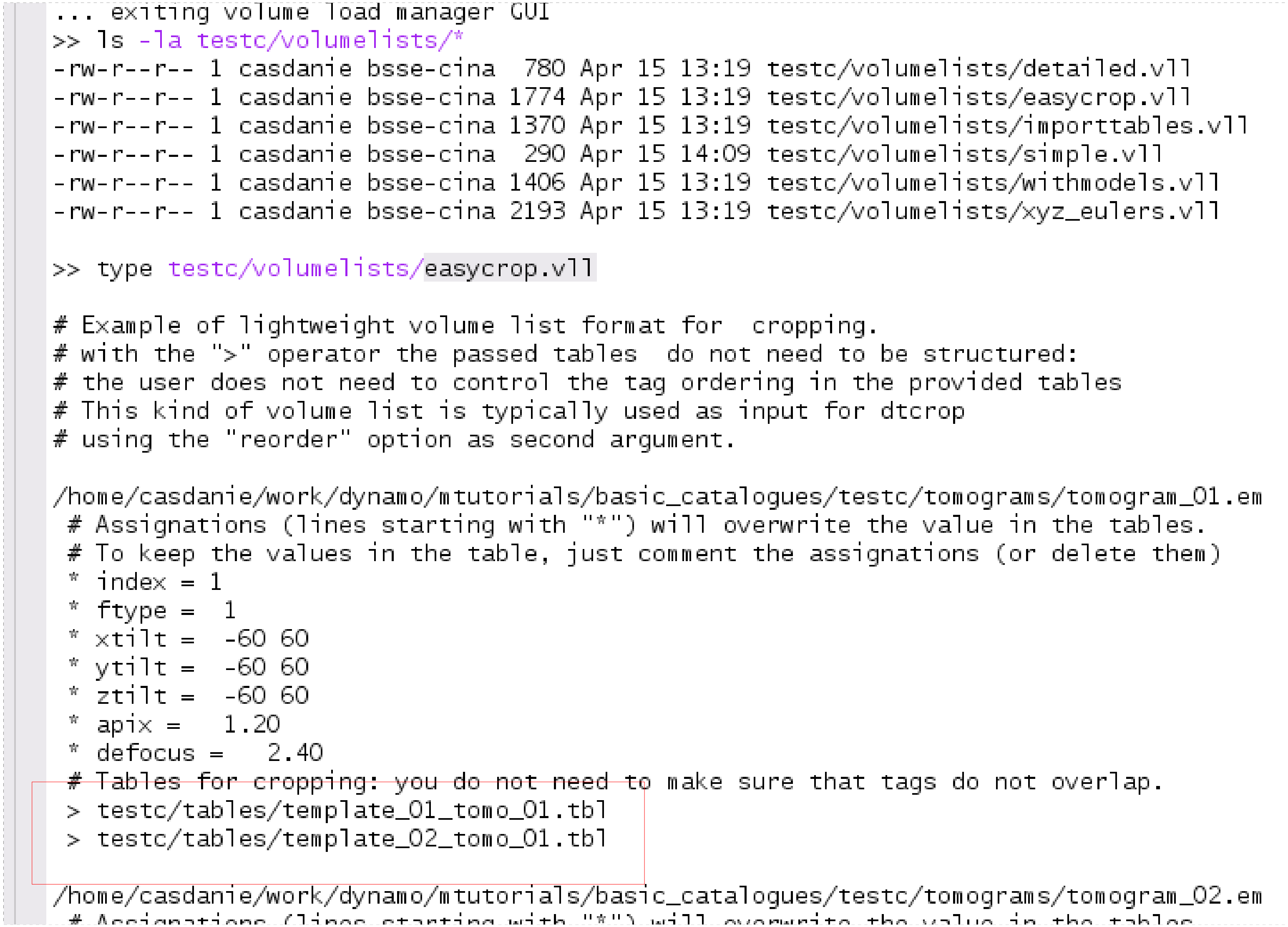
Property	Value



Extracting particles   command “>”

Particle extraction is the ultimate goal of the Catalogue construction. Let us start with some basic techniques.

Again, we take a look onto the created volume lists:



```
... exiting volume load manager GUI
>> ls -la testc/volumelists/*
-rw-r--r-- 1 casdanie bsse-cina  780 Apr 15 13:19 testc/volumelists/detailed.vll
-rw-r--r-- 1 casdanie bsse-cina 1774 Apr 15 13:19 testc/volumelists/easycrop.vll
-rw-r--r-- 1 casdanie bsse-cina 1370 Apr 15 13:19 testc/volumelists/importtables.vll
-rw-r--r-- 1 casdanie bsse-cina  290 Apr 15 14:09 testc/volumelists/simple.vll
-rw-r--r-- 1 casdanie bsse-cina 1406 Apr 15 13:19 testc/volumelists/withmodels.vll
-rw-r--r-- 1 casdanie bsse-cina 2193 Apr 15 13:19 testc/volumelists/xyz_eulers.vll

>> type testc/volumelists/easycrop.vll

# Example of lightweight volume list format for  cropping.
# with the ">" operator the passed tables  do not need to be structured:
# the user does not need to control the tag ordering in the provided tables
# This kind of volume list is typically used as input for dtcrop
# using the "reorder" option as second argument.

/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_01.em
 # Assignations (lines starting with "*") will overwrite the value in the tables.
 # To keep the values in the table, just comment the assignations (or delete them)
 * index = 1
 * ftype =  1
 * xtilt =  -60 60
 * ytilt =  -60 60
 * ztilt =  -60 60
 * apix =   1.20
 * defocus =   2.40
 # Tables for cropping: you do not need to make sure that tags do not overlap.
 > testc/tables/template_01_tomo_01.tbl
 > testc/tables/template_02_tomo_01.tbl

/home/casdanie/work/dynamo/mtutorials/basic_catalogues/testc/tomograms/tomogram_02.em
 # Assignations (lines starting with "*") will overwrite the value in the tables.
```

Lines starting with “>” are crop operations.

They convert a volume list into an object that can be used into any of the Dynamo programs that access data sources.

Let us see an example in next slide, where this vll structure is used to access all particles located in different tomograms.

As suggested by the tutorial code, we write:

vll as generic data source *target data folder* sidelength

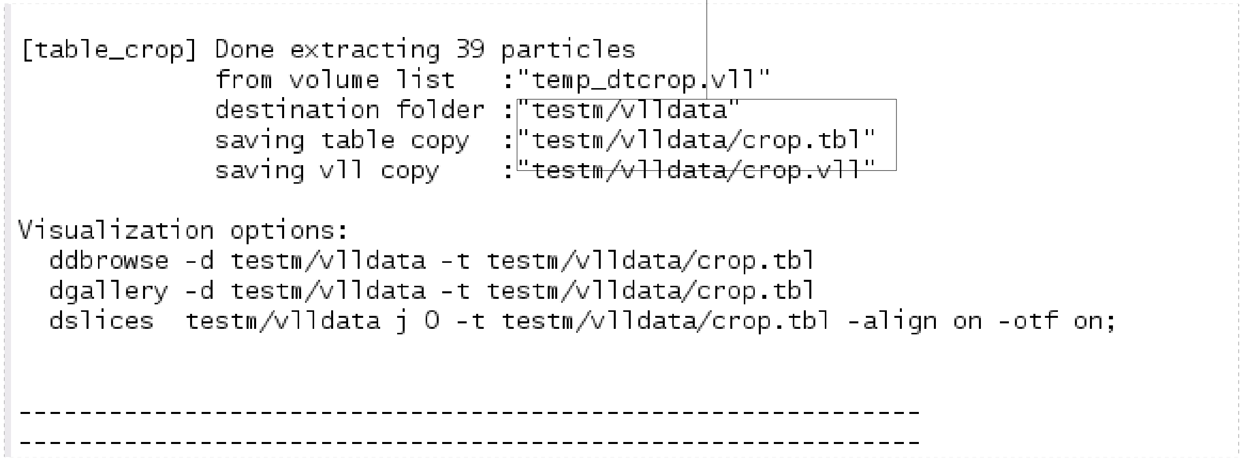
```
>> dtcrop testc/volumelists/easycrop.vll reorder testm/vlldata 40
```

If you are familiar with the syntax of dtcrop, you notice that the second argument, which is normally a table has been replaced with a code word: reorder.

This instructs dtcrop to look inside the passed volume list to look for tables (or models or AV3 motls or some other formats) that belong to a given tomogram, and extract them.

Original tags are not respected (as a “reordeing” takes place), but it a new table is produced that runs on all the particles in the created data set.

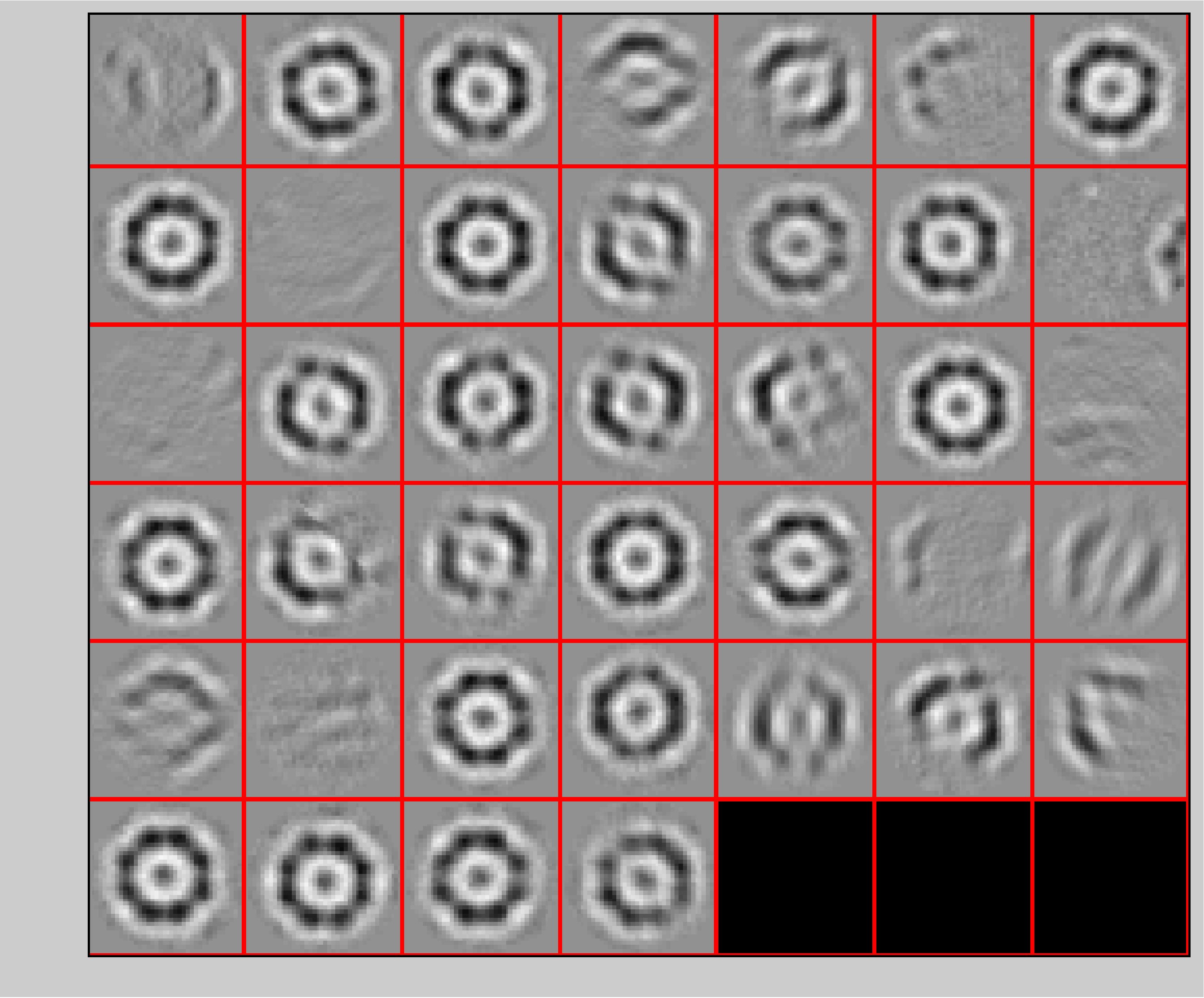
You should the summary of results announced by the code:



```
[table_crop] Done extracting 39 particles
             from volume list   :"temp_dtcrop.vll"
             destination folder :"testm/vlldata"
             saving table copy  :"testm/vlldata/crop.tbl"
             saving vll copy    :"testm/vlldata/crop.vll"

Visualization options:
  ddbrowse -d testm/vlldata -t testm/vlldata/crop.tbl
  dgallery -d testm/vlldata -t testm/vlldata/crop.tbl
  dslices  testm/vlldata j 0 -t testm/vlldata/crop.tbl -align on -otf on;


------------------------------------------------------------
------------------------------------------------------------
```

In other words, you have a data folder and a table that can be used as normally in Dynamo

```
>> dslices  testm/vlldata j 0  -t testm/vlldata/crop.tbl -align on -otf on;
```



COMMAND LINE ACCESS

Command line accesses will be seen in a next tutorial.

We will see how catalogues mix with *data containers* in Dynamo. For instace:

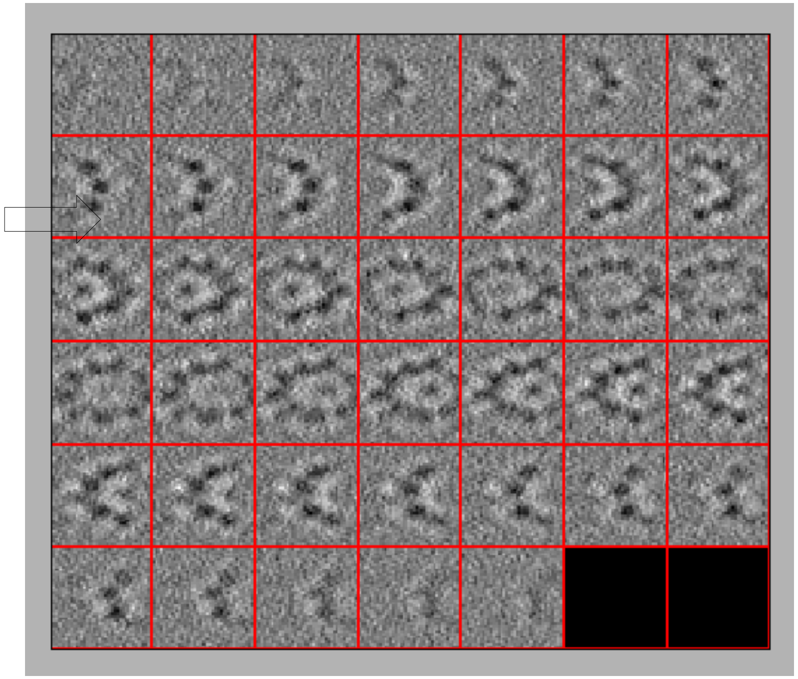
```
>> VLL = dData.new('s','testc/volumelists/easycrop.vll','m','reorder');
```

here, `VLL` is an object that allows accessing data sources of any type. In this case we have use as source (“s”) a volume list file with text, and we have especified that the metadata ('m') needs to be constructed reordering the files indicated inside the volume...

Now we can use the object in a variety of forms (the “methods”) that are attached to the class.

The most basic use would be to extract directly a particle:

```
>> p = getParticle(16,'sidelength',40);dview(p);
```



The notation might seem obscure at the beginning, but it quickly pays of to use these objects when you manipulate particles stemming from different tomograms with different models defined to extract the subtomograms.