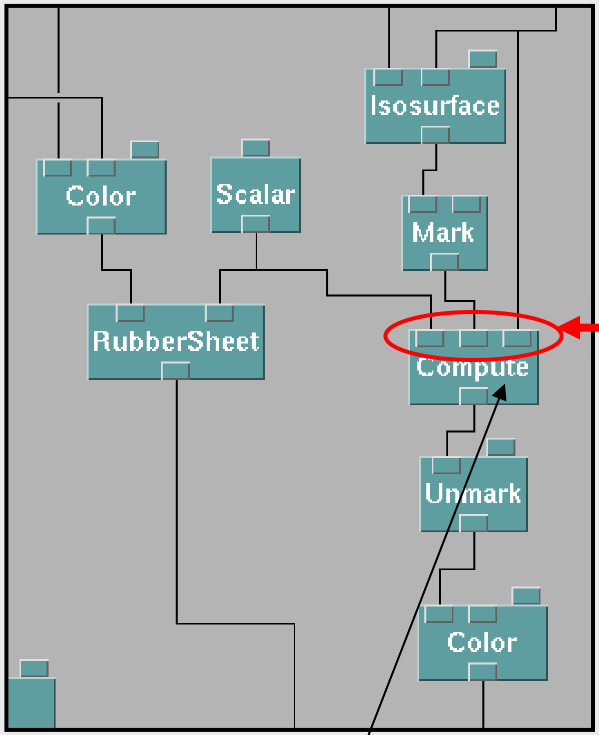
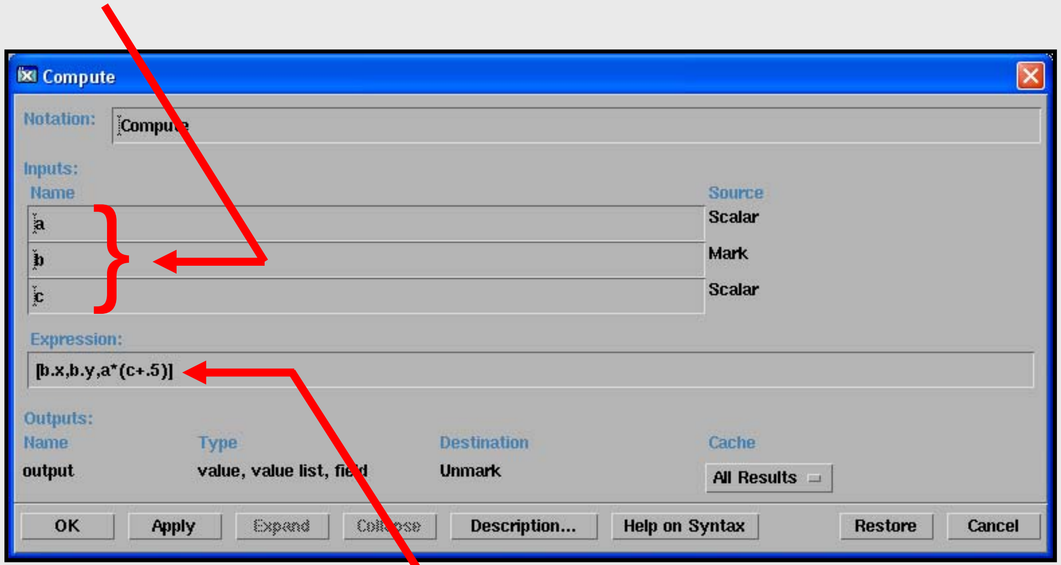
The Compute Module

Does arithmetic on the point-by-point Data component of a field, and outputs the modified field



Transformation

The 3 (in this case) inputs



Notation: Compute

Inputs:

Name	Source
a	Scalar
b	Mark
c	Scalar

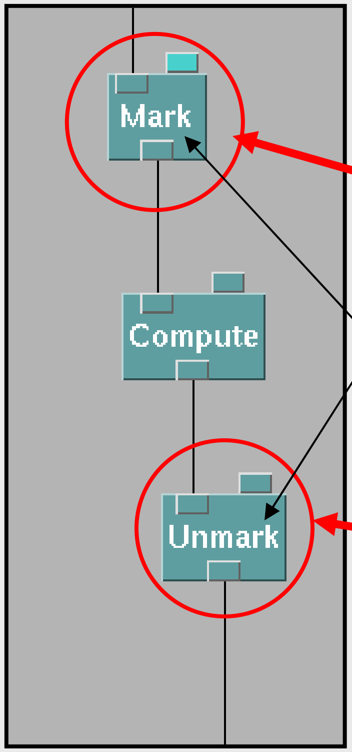
Expression: [b.x,b.y,a*(c+.5)]

Outputs:

Name	Type	Destination	Cache
output	value, value list, field	Unmark	All Results

The output expression, in this case, a 3-vector with a newly-created Z value

The Compute Module

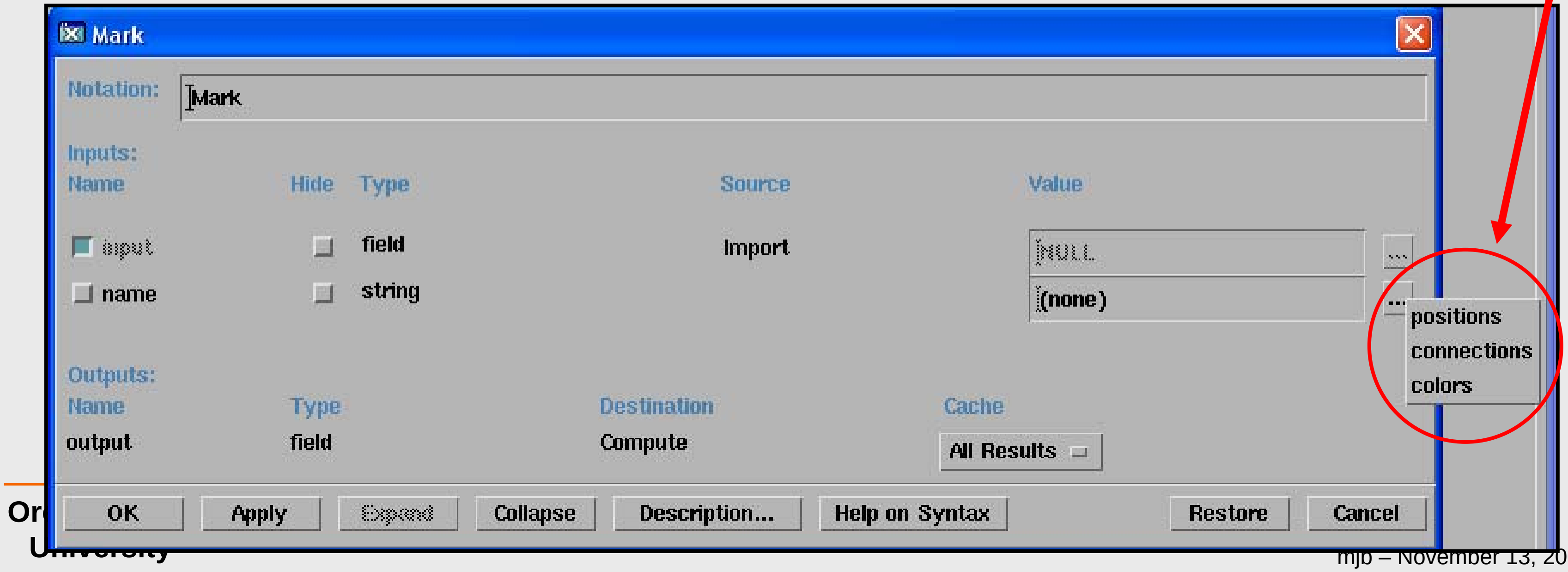


Structuring

Does arithmetic on the point-by-point Data component of a field, and outputs the modified field. But, what if you want to do arithmetic on a different component?

The *Mark* module renames the Data component to something temporary, and renames a component you select to “Data”. *Compute* then acts on this component.

The *Unmark* module changes the component names back to what they were originally.



Notation: Mark

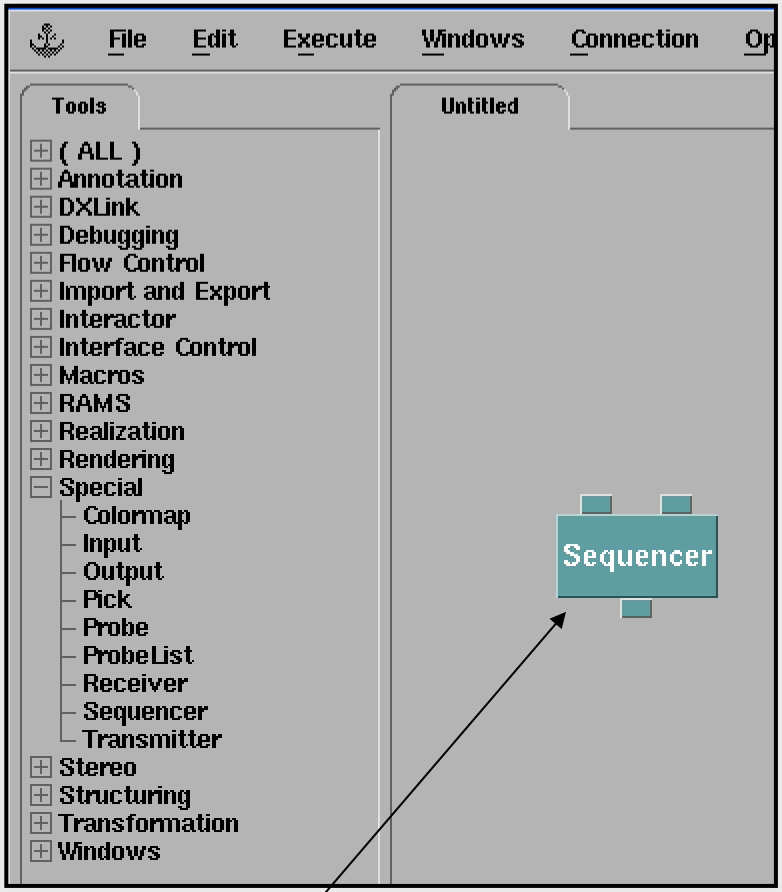
Inputs:

Name	Hide	Type	Source	Value
☒ input	□	field	Import	NULL
□ name	□	string		(none)

Outputs:

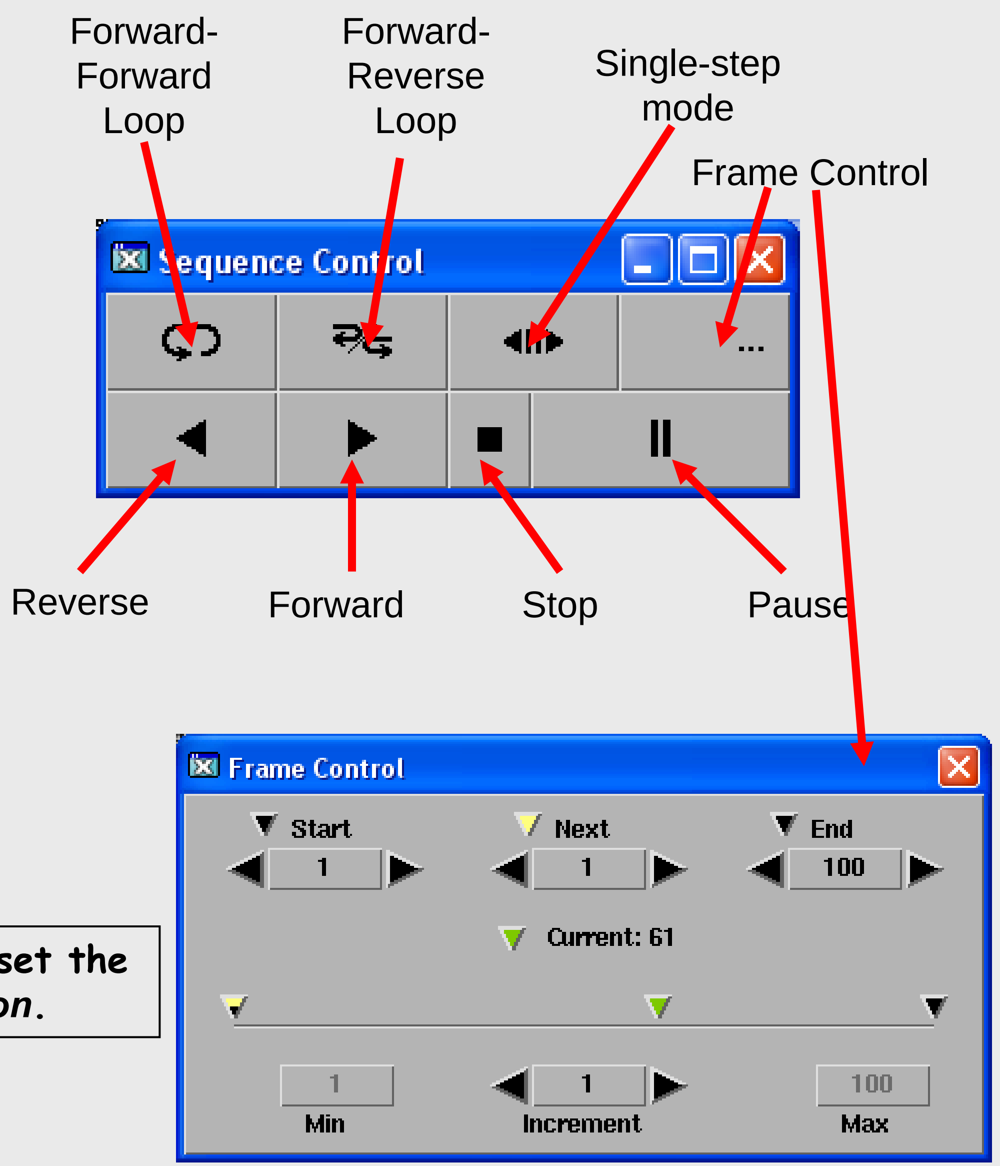
Name	Type	Destination	Cache
output	field	Compute	All Results

Animation:
The Sequencer Module



Special

Sequencer outputs a series of integers. You set the minimum and maximum using *Edit→Configuration*.



Forward-Forward Loop

Forward-Reverse Loop

Single-step mode

Frame Control

Reverse

Forward

Stop

Pause

Start: 1

Next: 1

End: 100

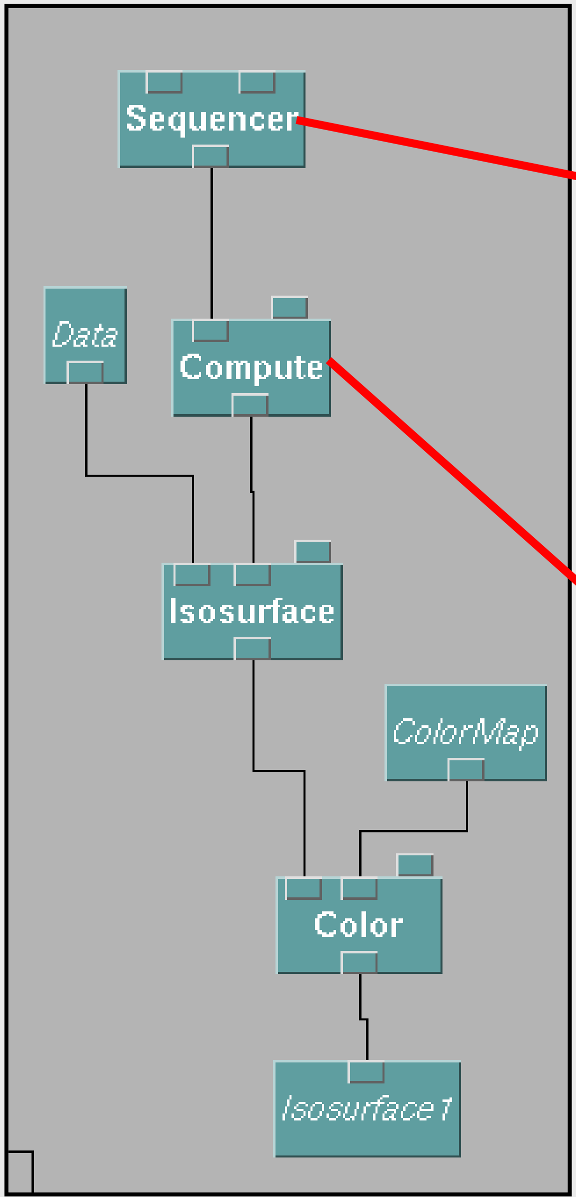
Current: 61

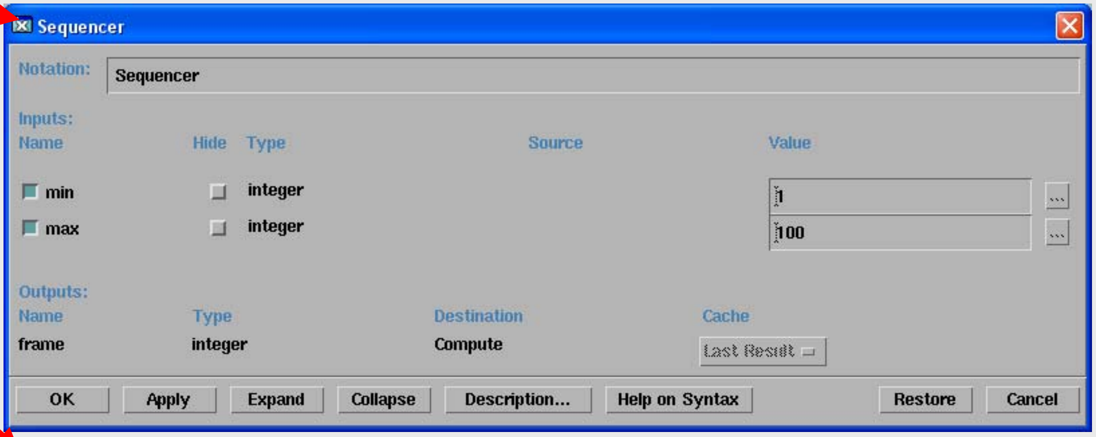
Min: 1

Increment: 1

Max: 100

The Sequencer Module: Usually Used with the Compute Module to turn the Integer into an Animation Parameter





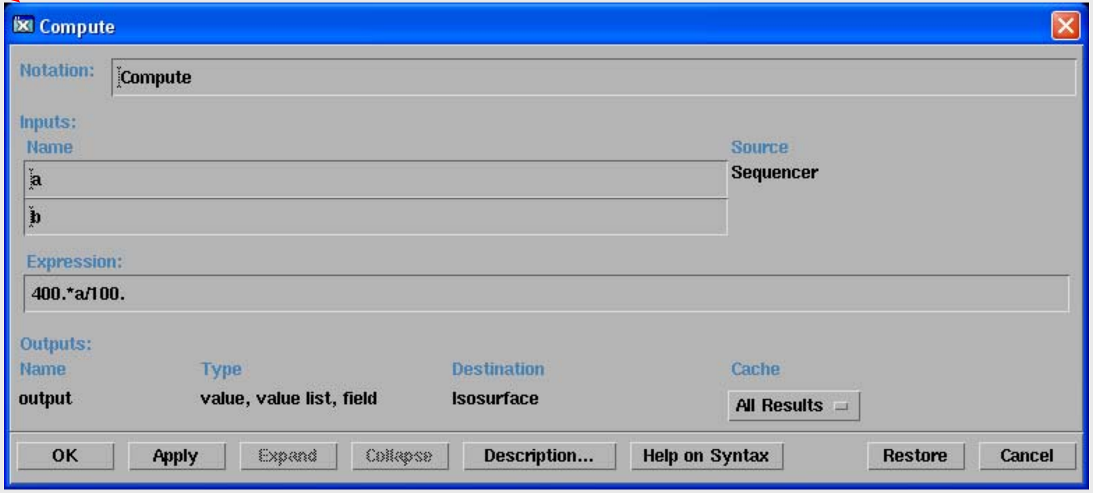
Notation: Sequencer

Inputs:

Name	Hide	Type	Source	Value
☑ min	□	integer		1
☑ max	□	integer		100

Outputs:

Name	Type	Destination	Cache
frame	integer	Compute	Last Result



Notation: Compute

Inputs:

Name	Source
a	Sequencer
b	

Expression:

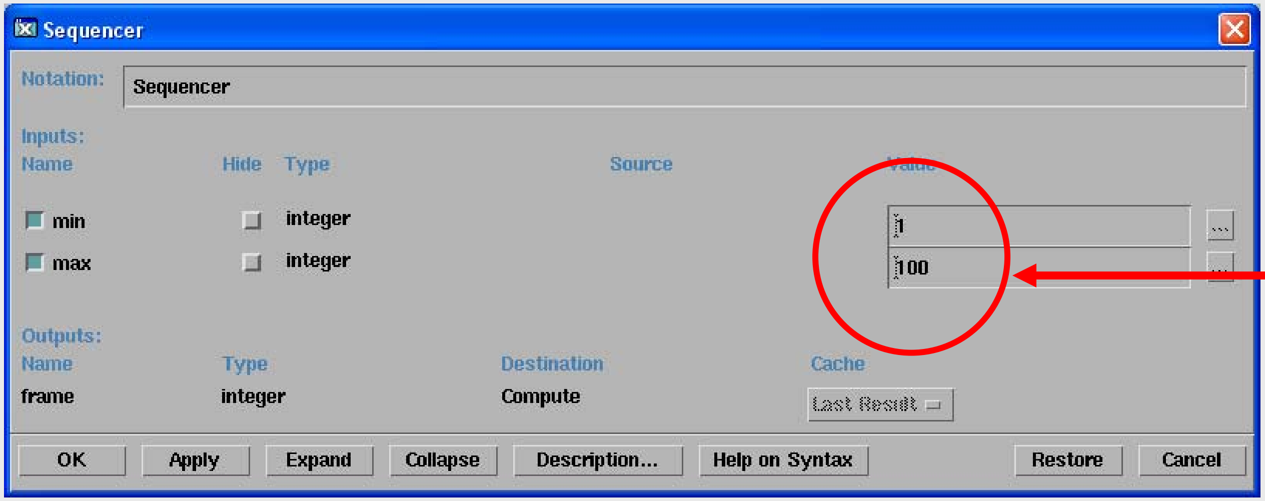
400.*a/100.

Outputs:

Name	Type	Destination	Cache
output	value, value list, field	Isosurface	All Results

In this case, *Compute* turns an integer into a scalar to be used to animate an isovalue

The Sequencer Module: “Percent Units Strategy”



Sequencer

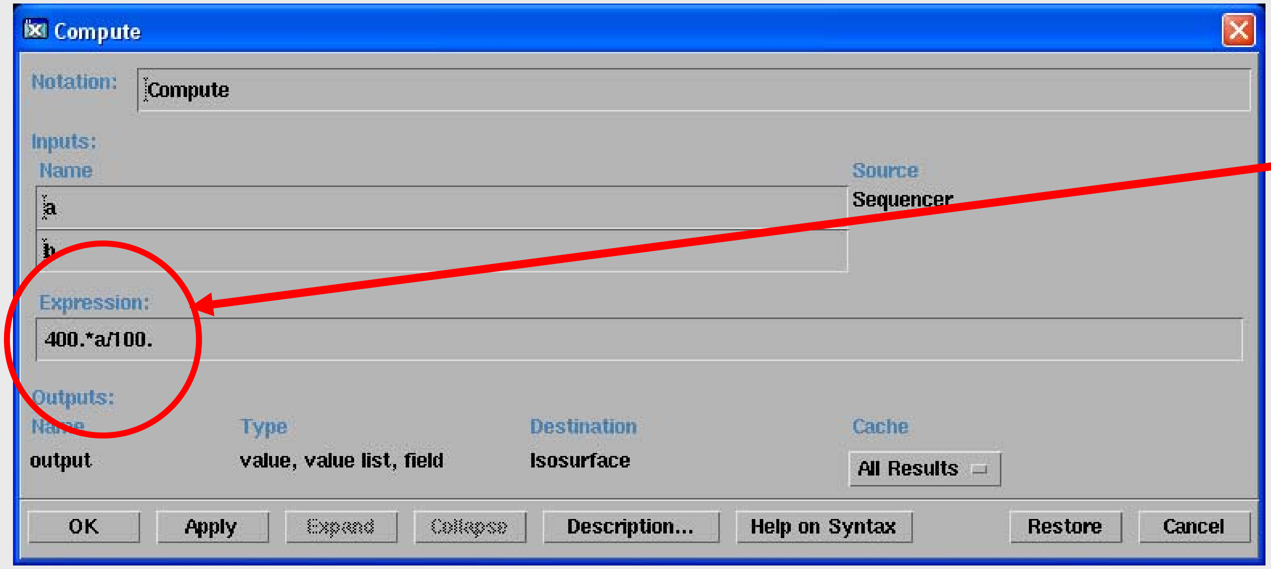
Notation: Sequencer

Inputs:

Name	Hide	Type	Source	Value
■ min	□	integer		1
■ max	□	integer		100

Outputs:

Name	Type	Destination	Cache
frame	integer	Compute	Last Result



Compute

Notation: Compute

Inputs:

Name	Source
a	Sequencer
b	

Expression: 400.*a/100.

Outputs:

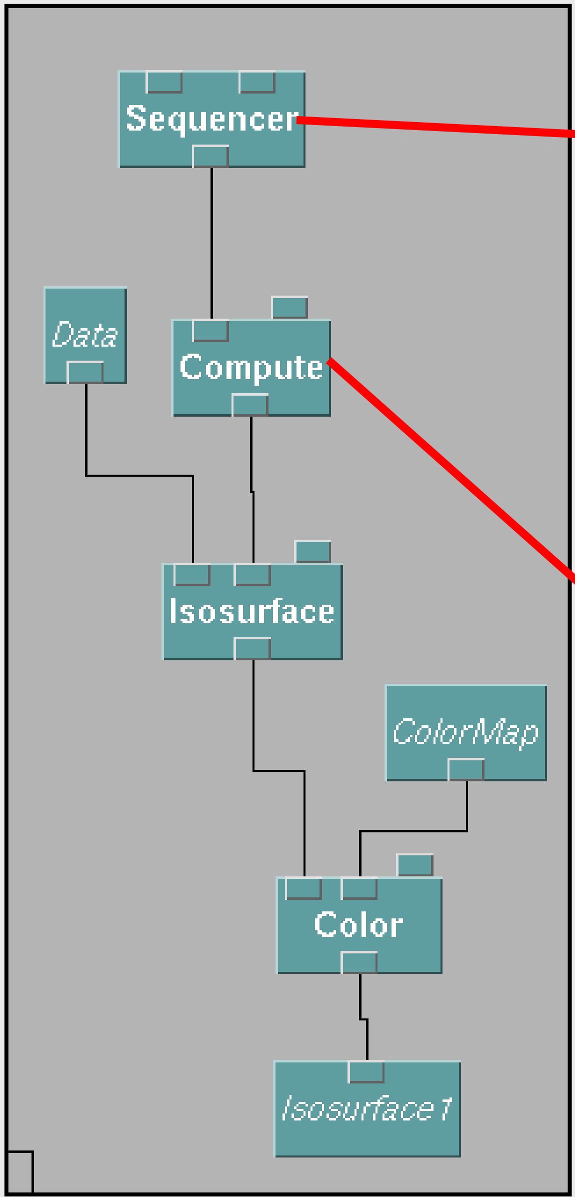
Name	Type	Destination	Cache
output	value, value list, field	Isosurface	All Results

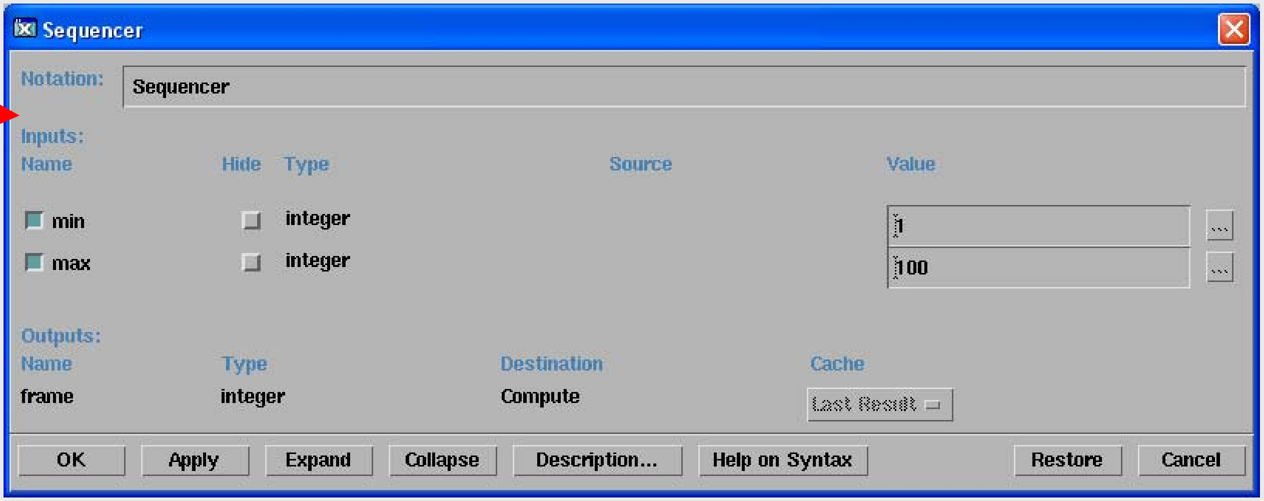
A good Sequencer Strategy: Run the sequence from 1-100 (or 0-100).

Then, base the *Compute* quantity on these “Percent Units”.

Oregon State University

The Sequencer Module: Setting a Scalar Isovalue





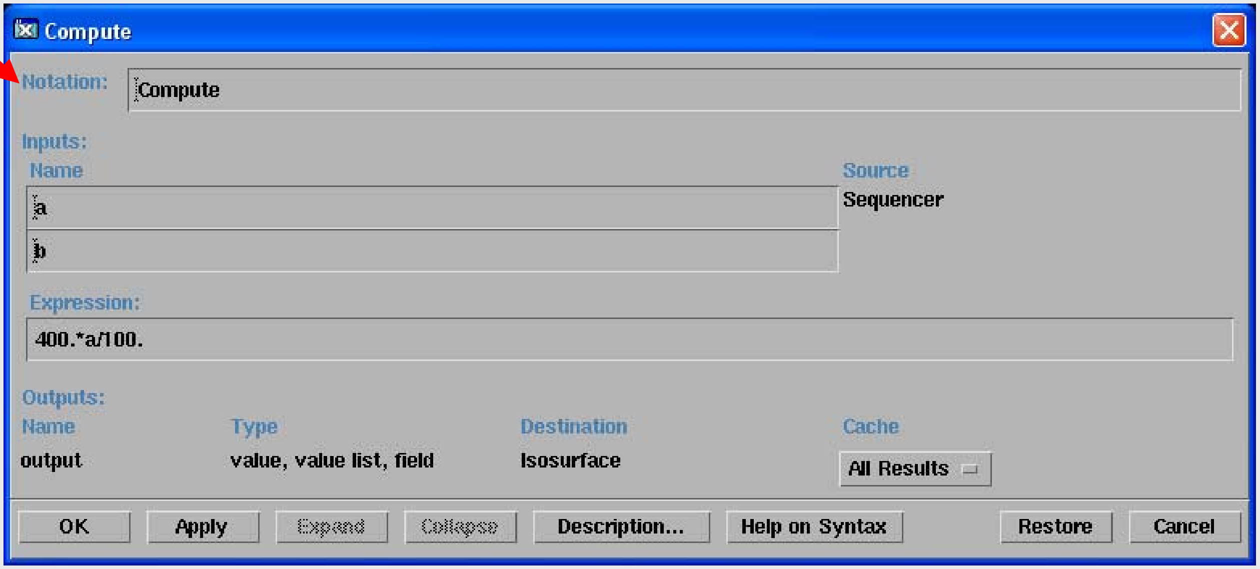
Notation: Sequencer

Inputs:

Name	Hide	Type	Source	Value
■ min	□	integer		1
■ max	□	integer		100

Outputs:

Name	Type	Destination	Cache
frame	integer	Compute	Last Result



Notation: Compute

Inputs:

Name	Source
a	Sequencer
b	

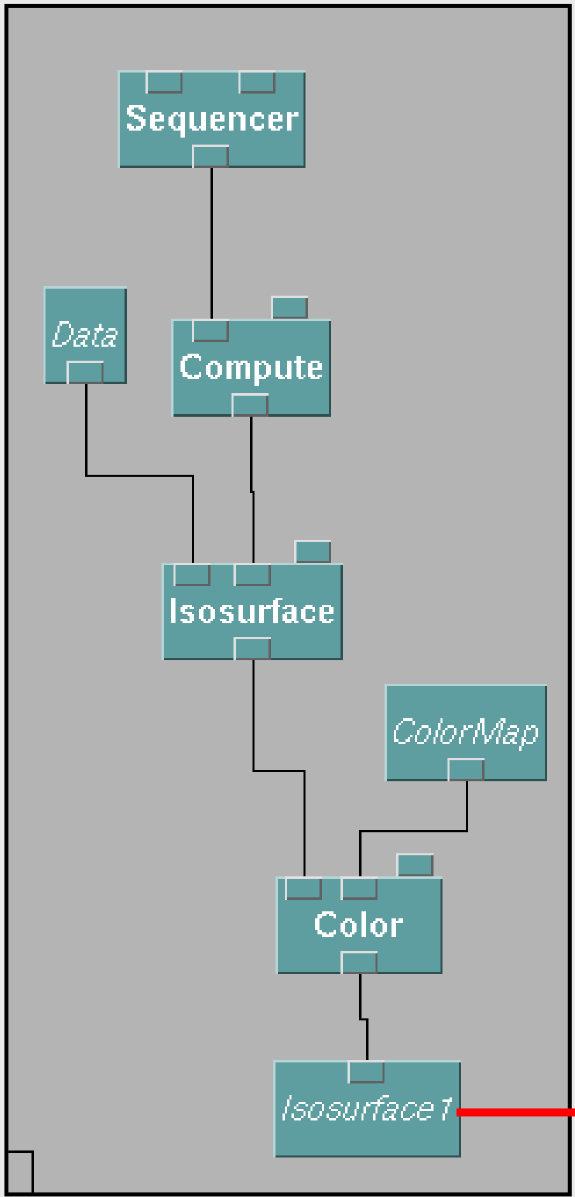
Expression: 400.*a/100.

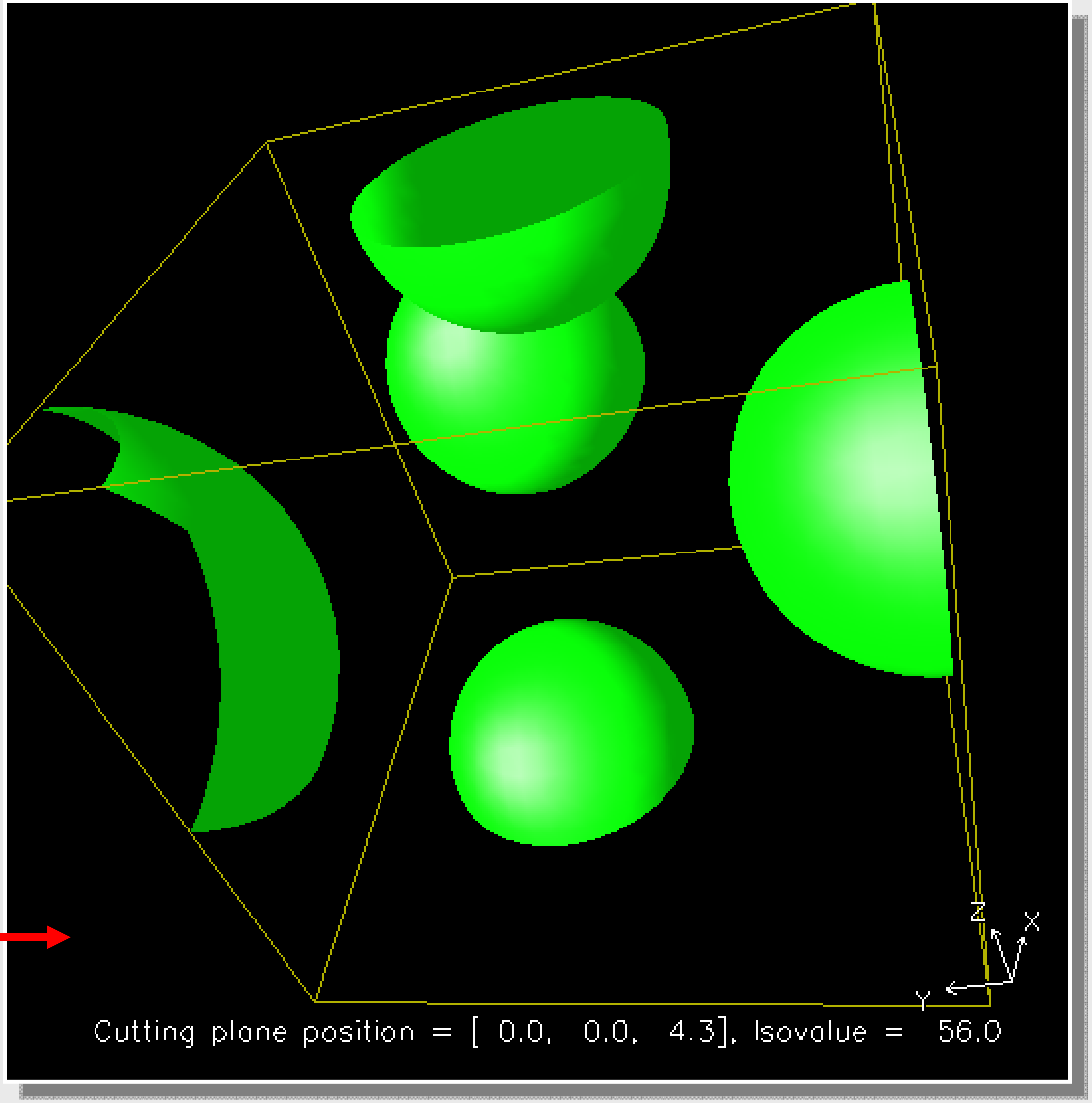
Outputs:

Name	Type	Destination	Cache
output	value, value list, field	Isosurface	All Results

In this case, *Compute* turns an integer into a scalar to be used to animate the isovalue

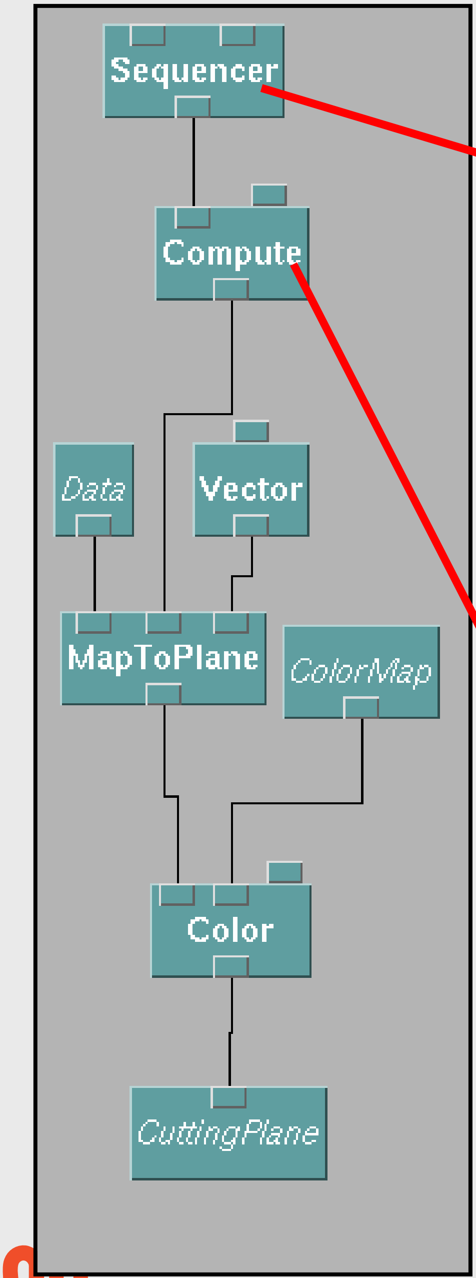
The Sequencer Module: Setting a Scalar Isovalue

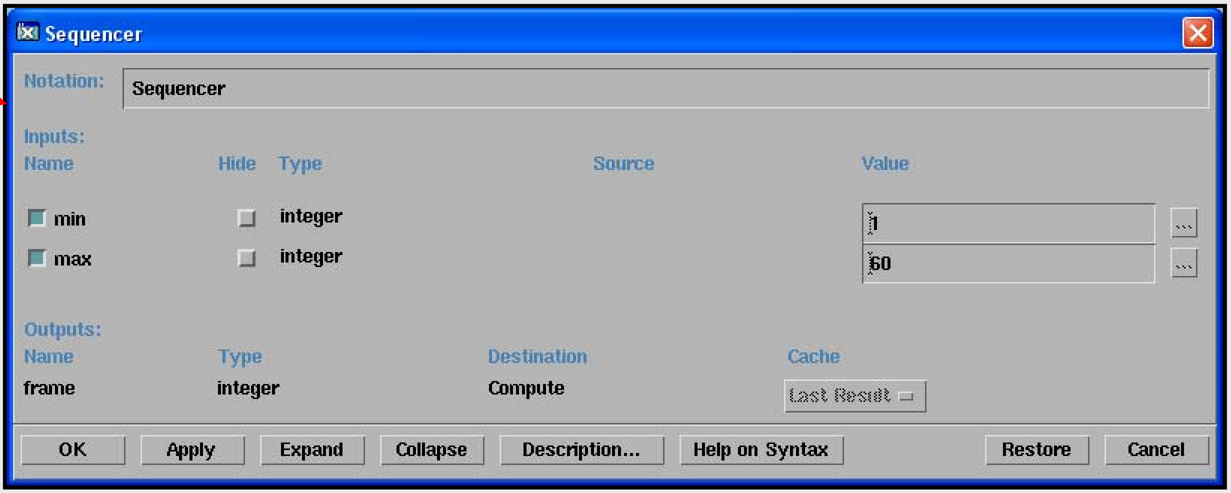




Cutting plane position = [0.0, 0.0, 4.3], Isovalue = 56.0

The Sequencer Module: Setting a Vector to act as a Plane Location




Sequencer

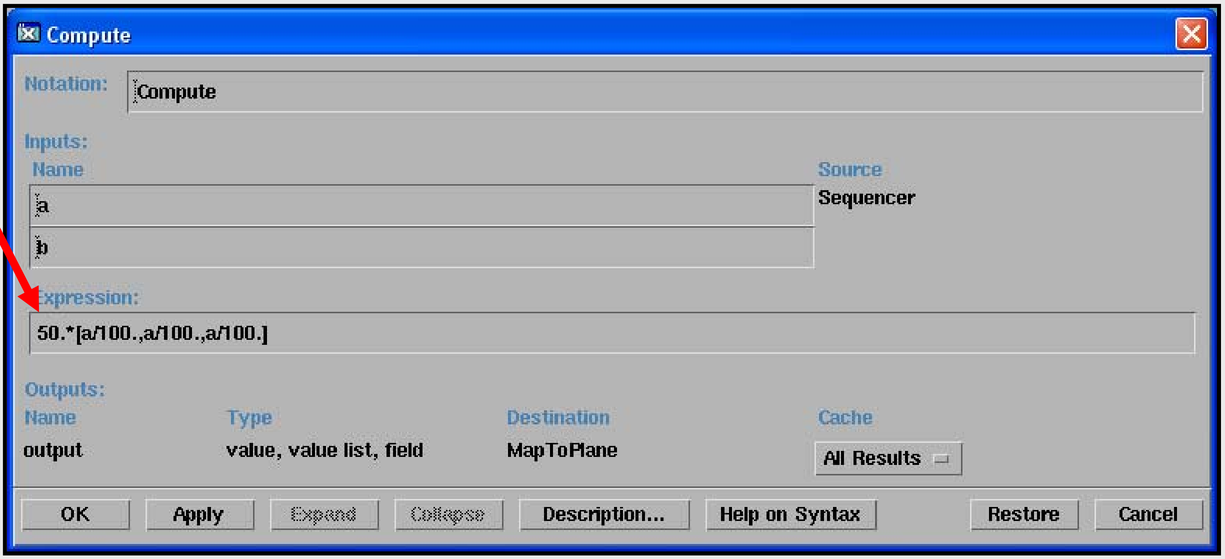
Notation: Sequencer

Inputs:

Name	Hide	Type	Source	Value
■ min	□	integer		1
■ max	□	integer		60

Outputs:

Name	Type	Destination	Cache
frame	integer	Compute	Last Result


Compute

Notation: Compute

Inputs:

Name	Source
a	Sequencer
b	

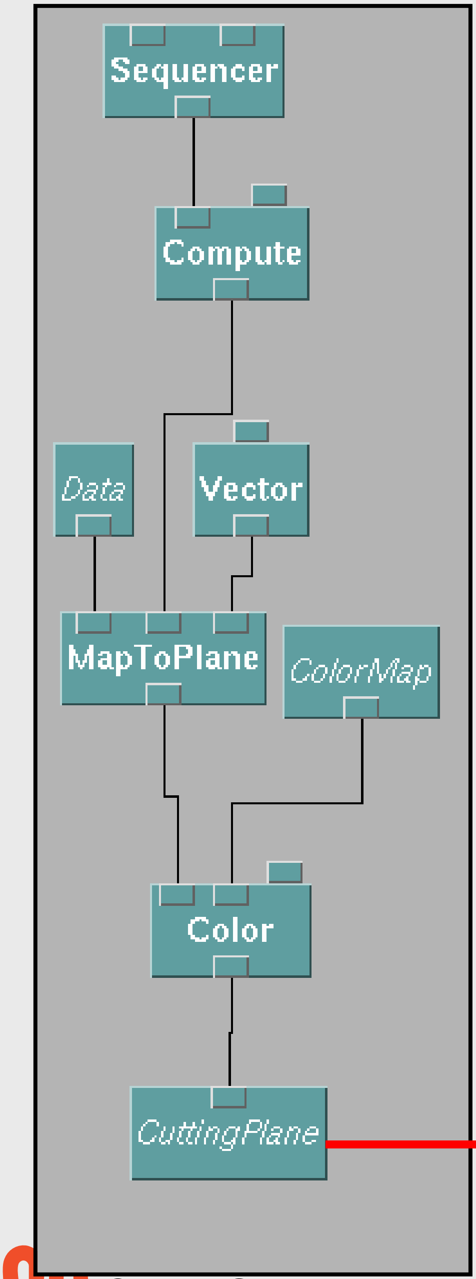
Expression: 50.*[a/100.,a/100.,a/100.]

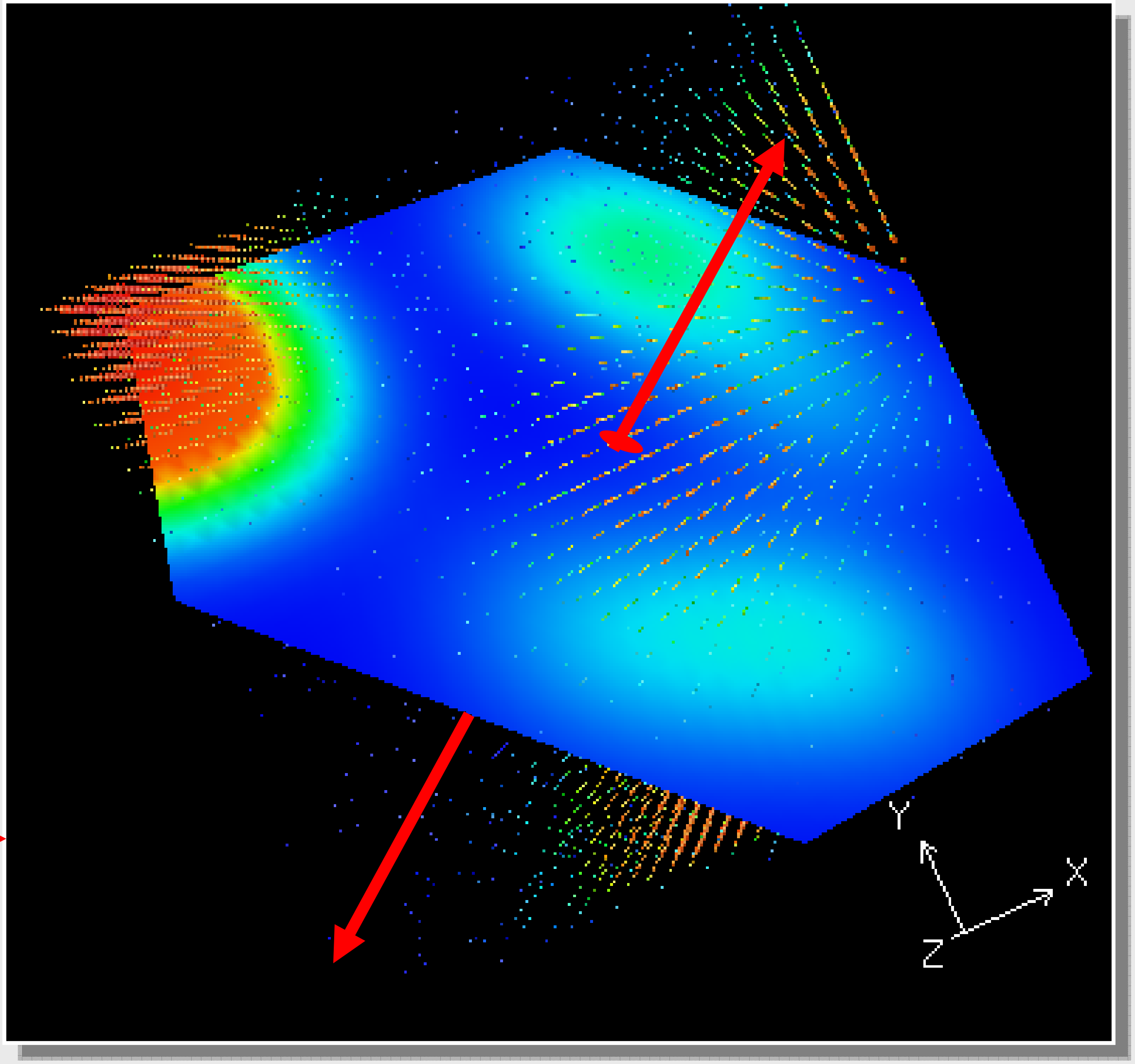
Outputs:

Name	Type	Destination	Cache
output	value, value list, field	MapToPlane	All Results

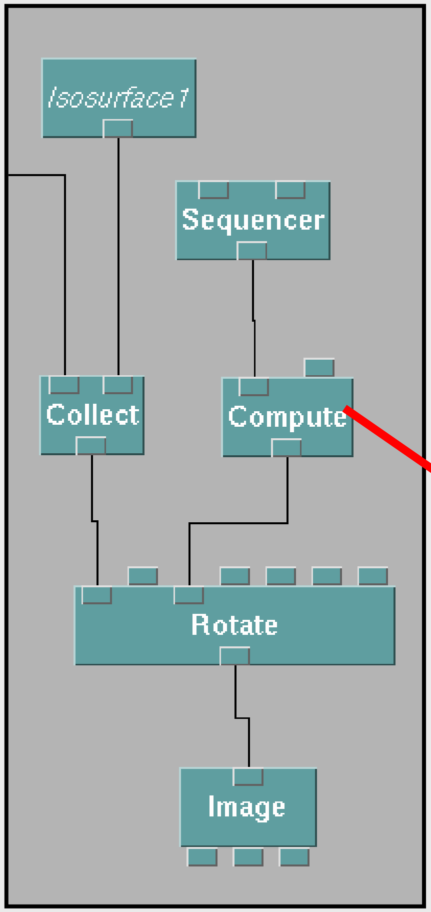
In this case, *Compute* turns an integer into a 3-element vector to be used to animate the position of the cutting plane

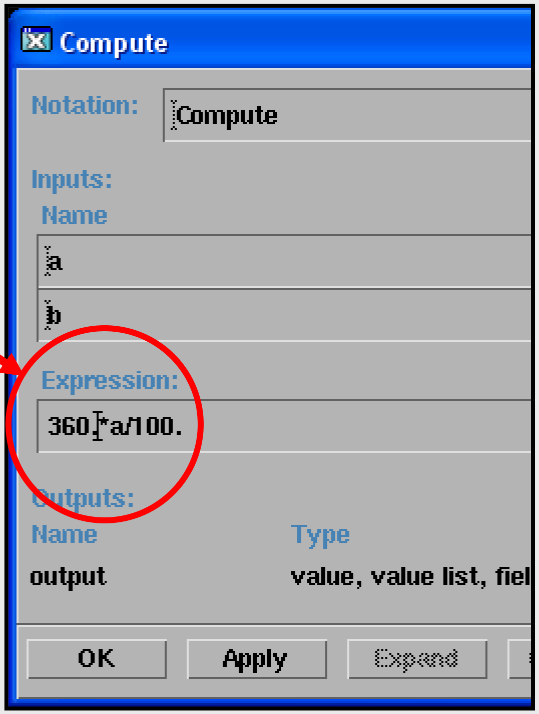
The Sequencer Module: Setting a Vector to act as a Plane Location





The Sequencer Module: Setting a Transformation




Compute

Notation: Compute

Inputs:

Name

a

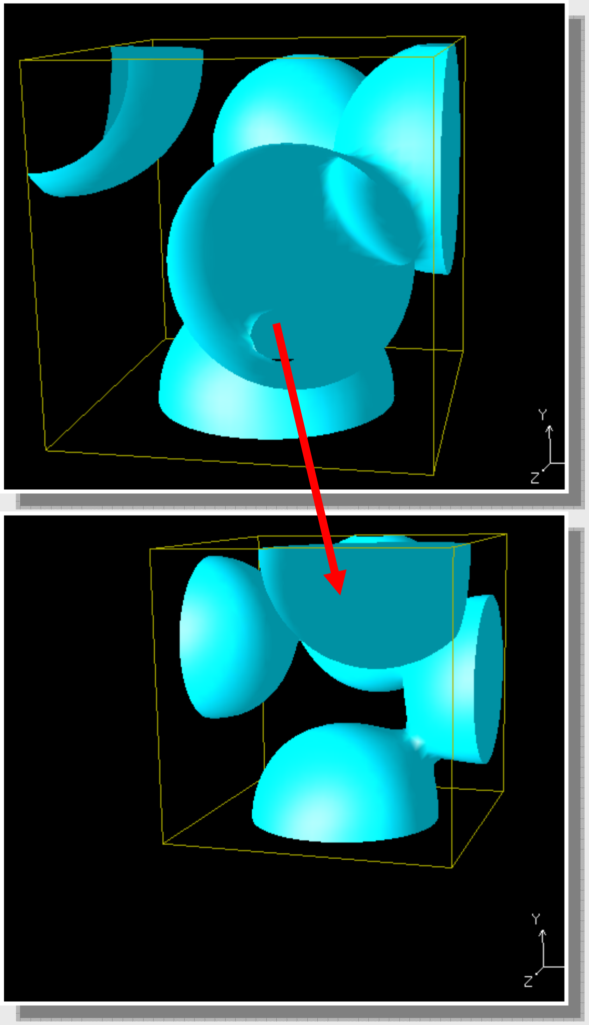
b

Expression:

360*a/100.

Outputs:

Name	Type
output	value, value list, fiel



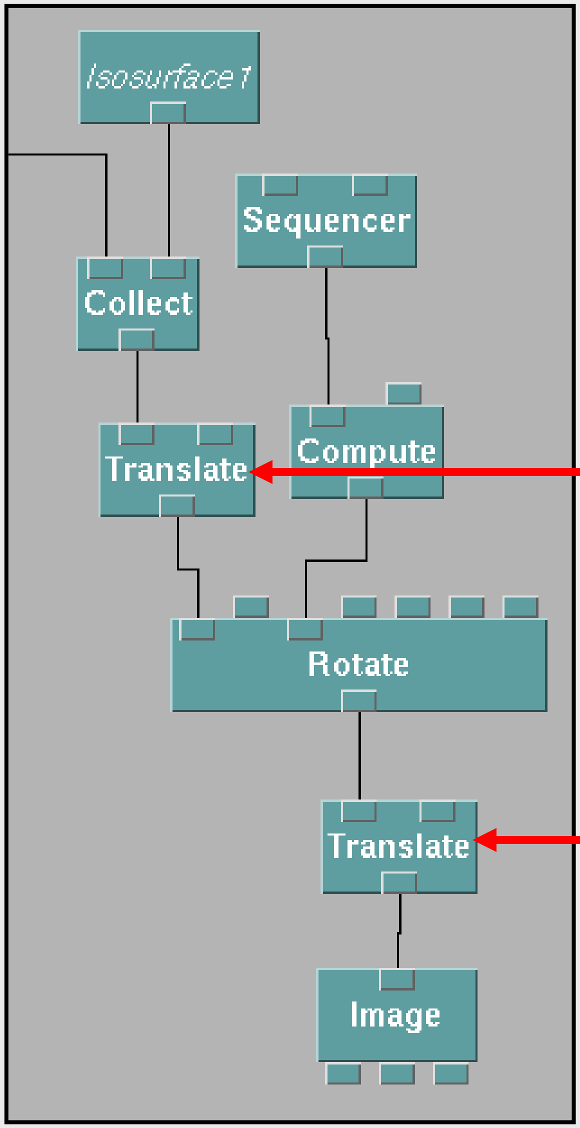
In this case, *Compute* turns an integer into a rotation angle in *degrees*.

Oregon State University

mjb – November 13, 2006

Why Does the Rotation Occur around the Edge of the Cube, not about its Center?

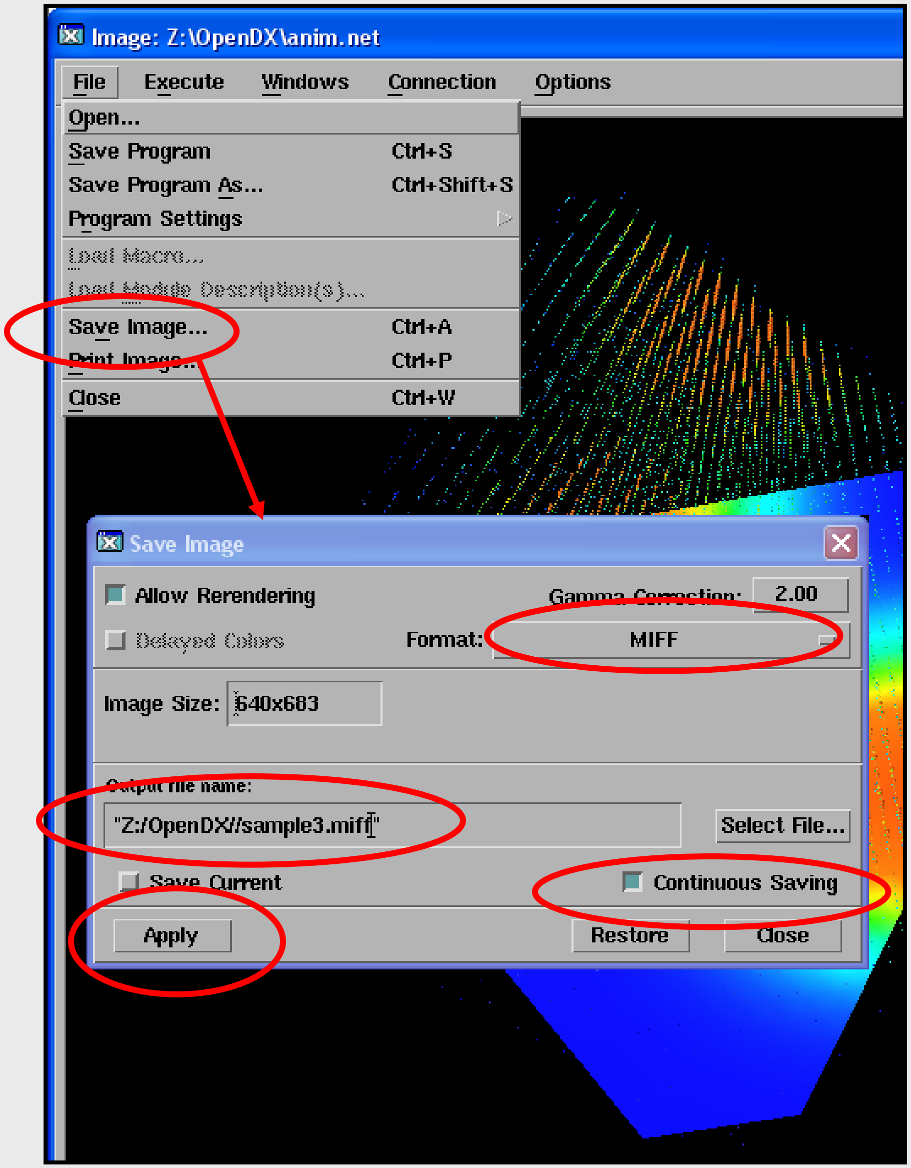
Rotation and Scaling *always occur about the origin*. To change this to the center of the volume, translate the volume to the origin, perform the rotation or scale, and then translate it back.



Translate by [-15,-15,-15]

Translate by [15,15,15]

Writing Out a MIFF Animation File



☑ Allow Rerendering

Gamma Correction: 2.00

□ Delayed Colors

Format: MIFF

Image Size: 640x683

Output file name:

"Z:/OpenDX//sample3.miff"

□ Save Current

☑ Continuous Saving

```
convert -quality 100 sample2.miff sample2.gif
```

Oregon State University