

---

## emPolygonizer Version 1.50

coded by Eric Mootz  
June–August 2009

[www.mootzoid.com](http://www.mootzoid.com)

---

Last modification of this documentation: August 5<sup>th</sup> 2009

### Chapter 1: Introduction

This plug-in for Softimage|XSI is a custom operator for creating polygon meshes. It polygonizes scalar fields, a method that is also called “Metaballs”, “Marching Cubes”, etc.

On my website (<http://www.mootzoid.com/XsiCorner.html>) you can:

- Take a look at some animations that were made using this plug-in.
- Download the demo version.
- Purchase the full version.
- Download demo, example and tutorial scenes.

This documentation has the following chapters:

- Chapter 1: Introduction
- Chapter 2: Installation
- Chapter 3: Demo Version Restrictions
- Chapter 4: Tutorials
- Chapter 5: The Custom Operator's Parameters
- Chapter 6: The Custom Parameter Sets / Properties
- Chapter 7: Vertex Colors
- Chapter 8: Tips and Tricks, Troubleshooting
- Chapter 9: Limitations and Remarks
- Chapter 10: Version History
- Chapter 11: Thanks

If you feel that something is not well explained (or not explained at all) then please write me a short e-mail ([info@mootzoid.com](mailto:info@mootzoid.com)). I will then try to update the documentation and/or make a demo scene as quickly as possible.

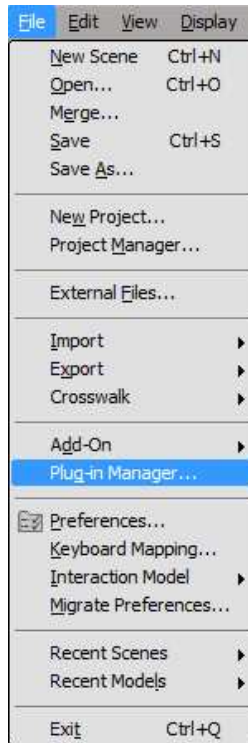
#### *Feedback:*

please let me know what you think is buggy, missing or “not okay” in this plug-in, so that it can be fixed or implemented in the next release.

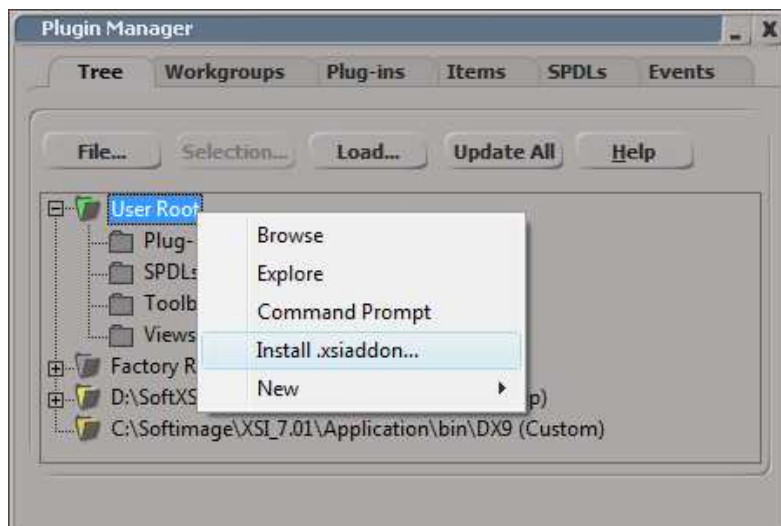
## Chapter 2: Installation

This plug-in comes as a so called “Add-On”, here's how to install it:

1. Open the “Plug-in Manager”. It is located under the “File” Menu:  
“File → Plug-in Manager”



2. Remove/uninstall any previous version of this plug-in. This is really important, because if Softimage|XSI finds two times the same plug-in then one (maybe even both) might not work.
3. To install the plug-in into the user directory simply right-click onto the folder “User Root” and choose “Install .xsiaddon...”:



4. A browser dialogue will be displayed: go to where you copied the .xsiaddon file, select it and click on "OK". The Add-on is automatically installed.
5. Close Softimage|XSI. The plug-in is now installed and ready to be used.  
For more information concerning Add-ons and how to install / uninstall them please check the chapter "Working with Add-ons" in the XSI Guides.

### **Chapter 3: Demo Version Restrictions**

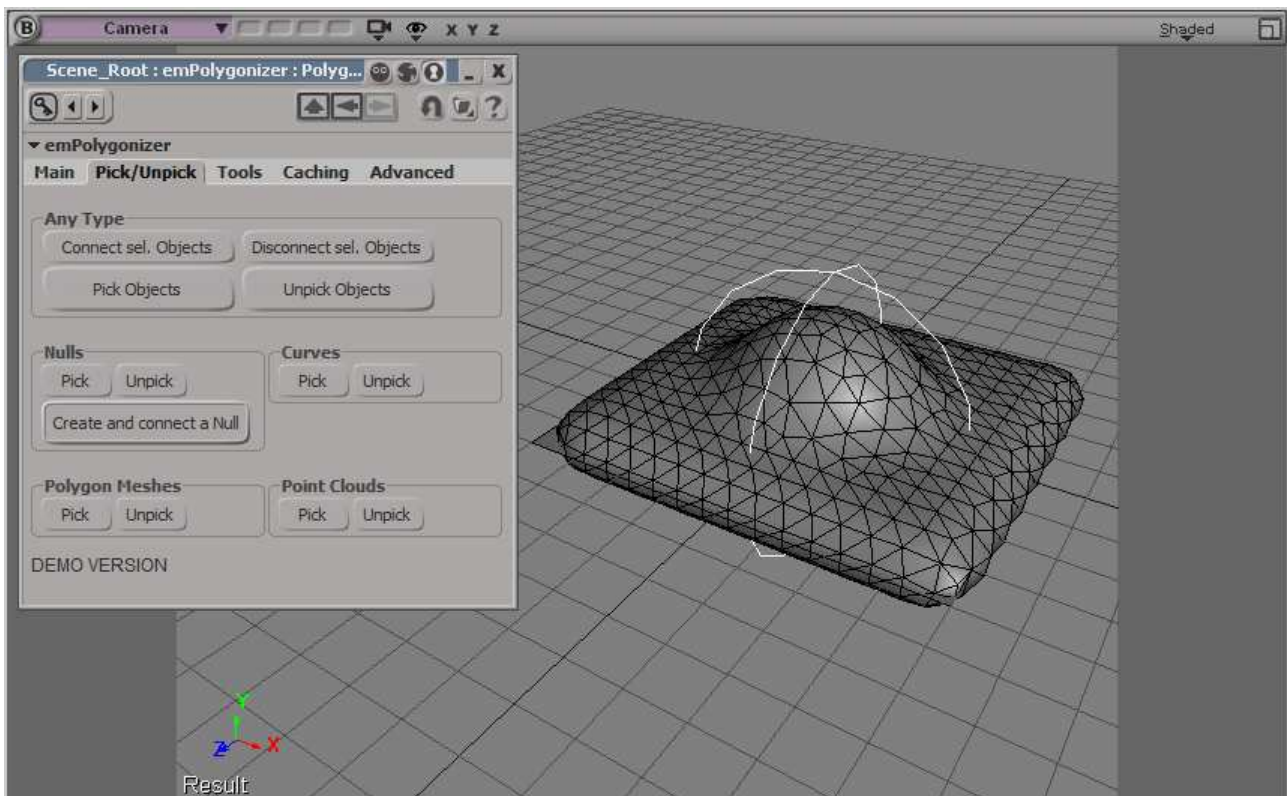
If you are using the full version of emPolygonizer then you can skip this chapter. If not, please read the following list of the demo version's restrictions:

- 1) There is a built in memory limit of 32 Megabytes, thus limiting the level of detail and the number of polygons for the output meshes.
- 2) The operator only works between frame 1 and frame 250.
- 3) The text "DEMO VERSION" is contained in the custom operator's property page.
- 4) Nearly all parameters of the tab "Advanced" of the operator's property page are hidden.

## Chapter 4: Tutorials

### Tutorial 1: grid and sphere

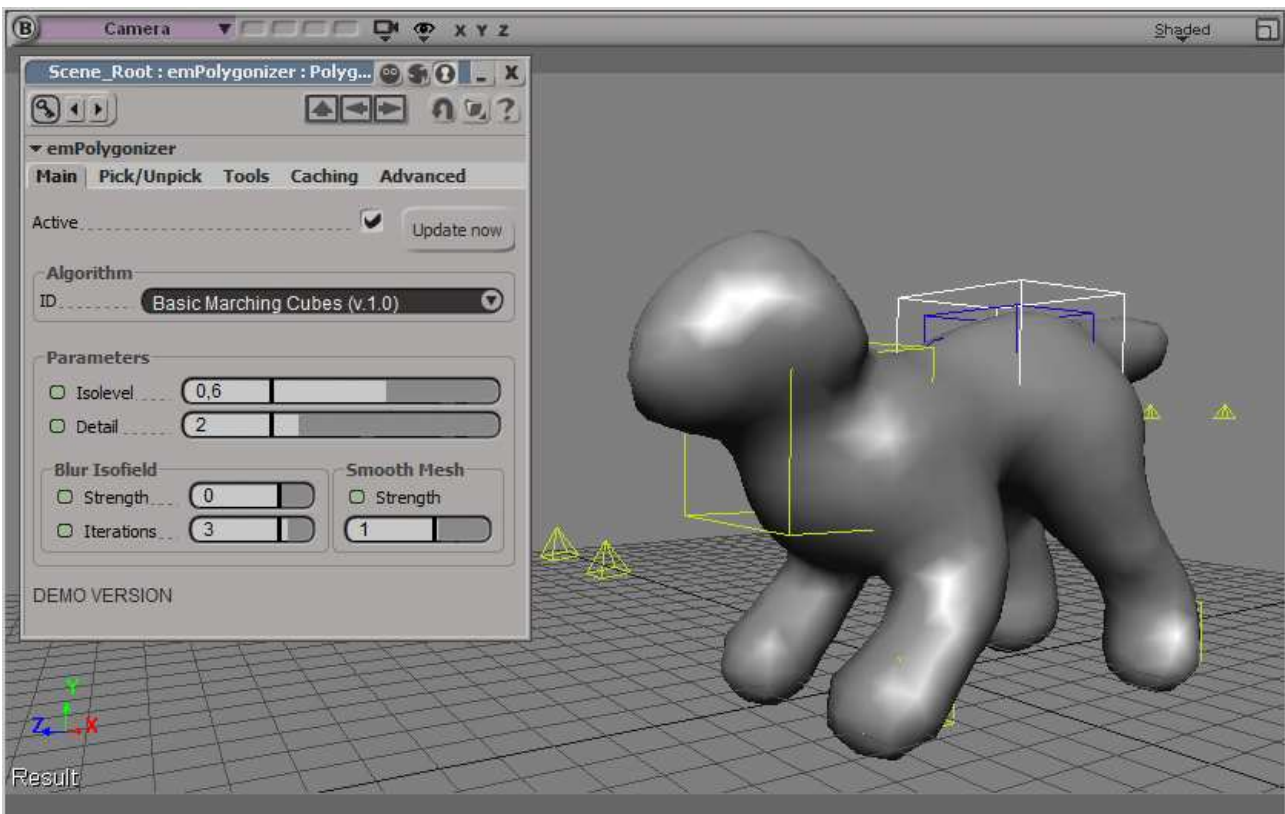
1. Start with a new scene.
2. Create a polygon mesh grid.
3. Now, with the grid still selected, create an emPolygonizer mesh:  
**Model → Create → Poly.Mesh → emPolygonizer-Metaballs**
4. The emPolygonizer's property page should pop up and a mesh surrounding the grid should be visible.
5. In the tab “Pick/Unpick” of emPolygonizer's property page click on “**Create and connect a Null**”: a null is created and immediately connected to the emPolygonizer mesh. If you set the display of the camera view to “shaded” with “Wireframe on Shaded” active then you should now have something that looks like this:



6. Translate and/or scale the null, maybe add a few more nulls by clicking on the “**Create and connect a Null**” button again. You might also play a little with the parameters of the custom parameter sets of the nulls and the polymesh grid, i.e. the radius and falloff.  
TIP: Try a negative Isofactor (i.e. “-5”) for one of the nulls and move it into the grid!

## Tutorial 2: Blobby Metadog

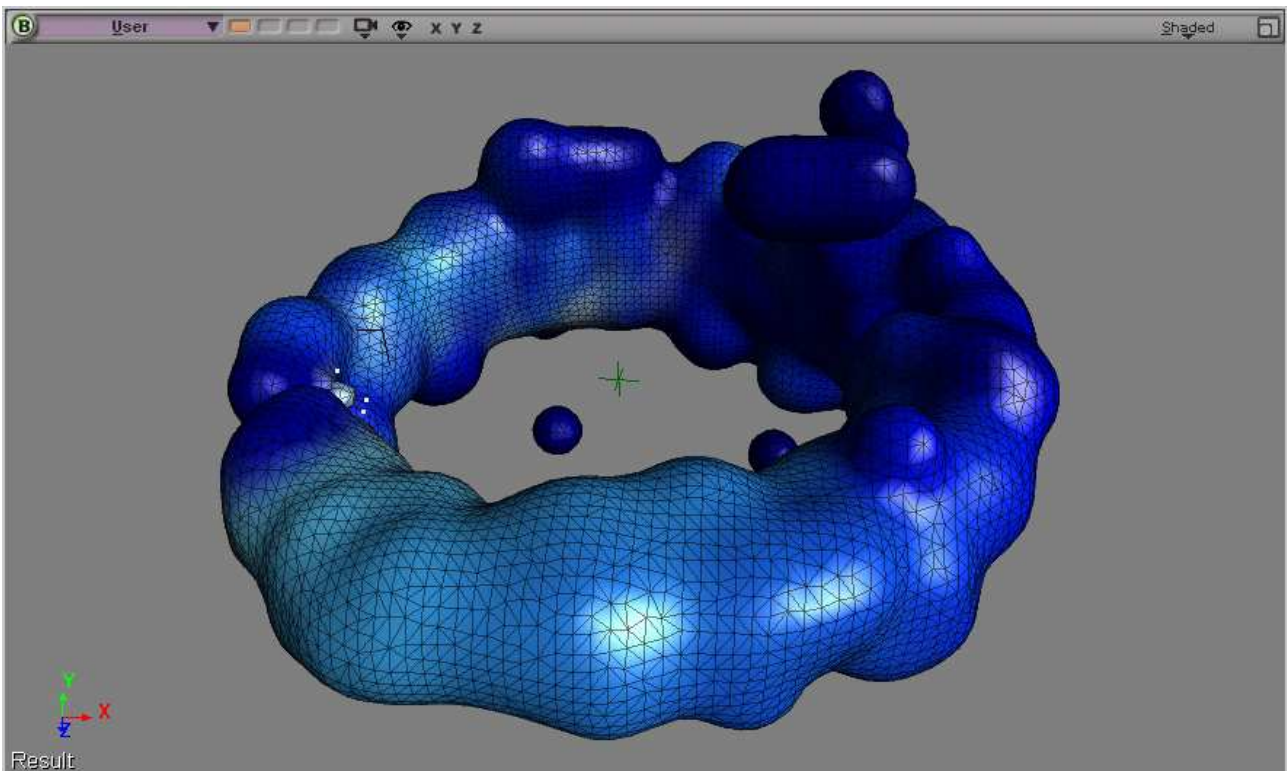
1. Start with a new scene.
2. Get the dog character (Model → Primitive → Character → Dog).
3. Deselect everything, close all property pages that popped up, set the camera view's display to “shaded” and select the dog's main polygon mesh.
4. With the mesh still selected, create an emPolygonizer mesh:  
**Model → Create → Poly.Mesh → emPolygonizer–Metaballs**
5. Select one of the two controls (the big yellow implicit cubes at the shoulders and the rear) and move them around. You should have something that looks like this:



6. Open the property page of emPolygonizer and try some different values for the parameter “Detail”, as well as for the blur parameters “Strength” and “Iterations”.  
Or add a few nulls via “**Create and connect a Null**” in the “**Pick/Unpick**” tab.

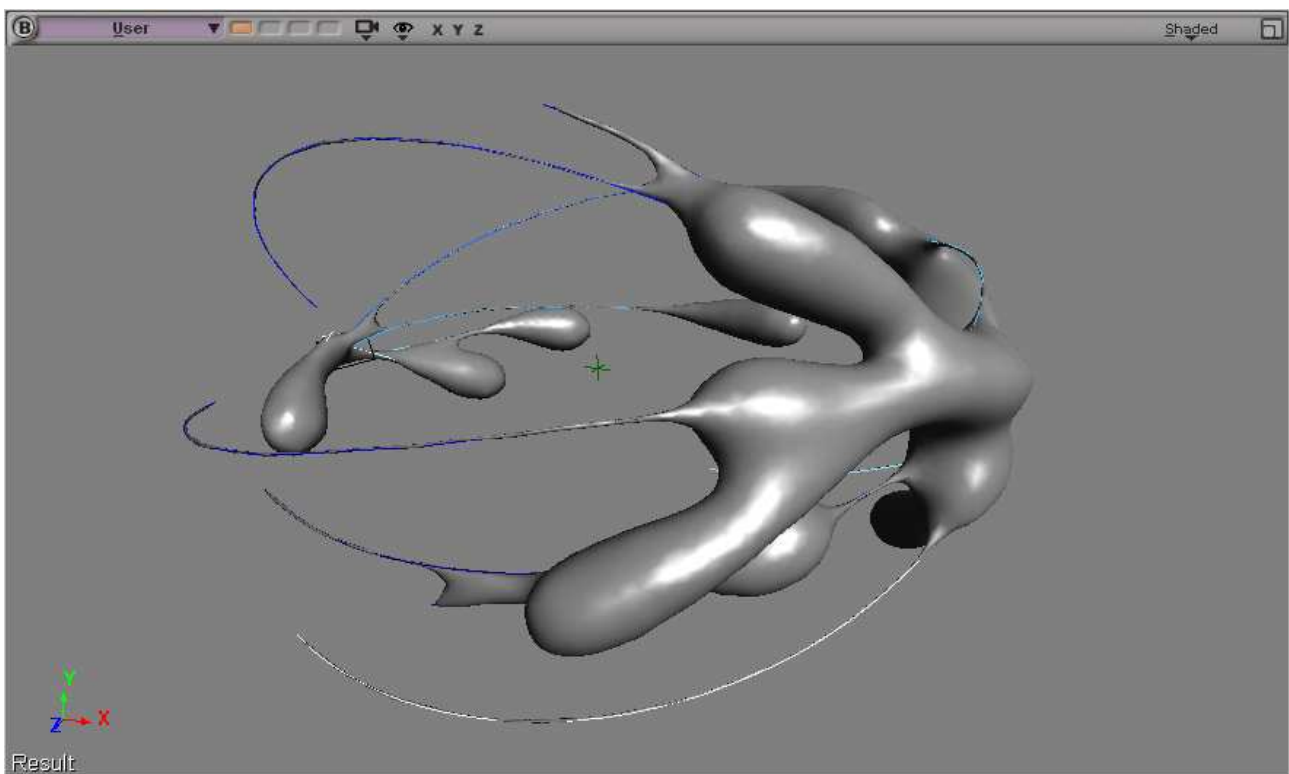
### Tutorial 3: What about a Point Cloud?

1. Load the scene “emPolygonize\_1.50\_Tutorial\_ParticlesOrbit\_100perSecond.scn”, which is included in the demo scenes.
2. Play back the scene just to see what it does. 100 particles are emitted per second and orbit around a null.
3. Select the point cloud and create an emPolygonizer mesh:  
**Model → Create → Poly.Mesh → emPolygonizer-Metaballs**
4. Try some different values for the parameter “Detail”, as well as for the blur parameters “Strength” and “Iterations”, and try some extreme values for the “Blur Mesh Strength” (>25).
5. You can also activate the vertex colors (the particle color is then used as vertex color) by clicking on the “**Recreate**” button in the “Advanced” tab. You will get something like this:



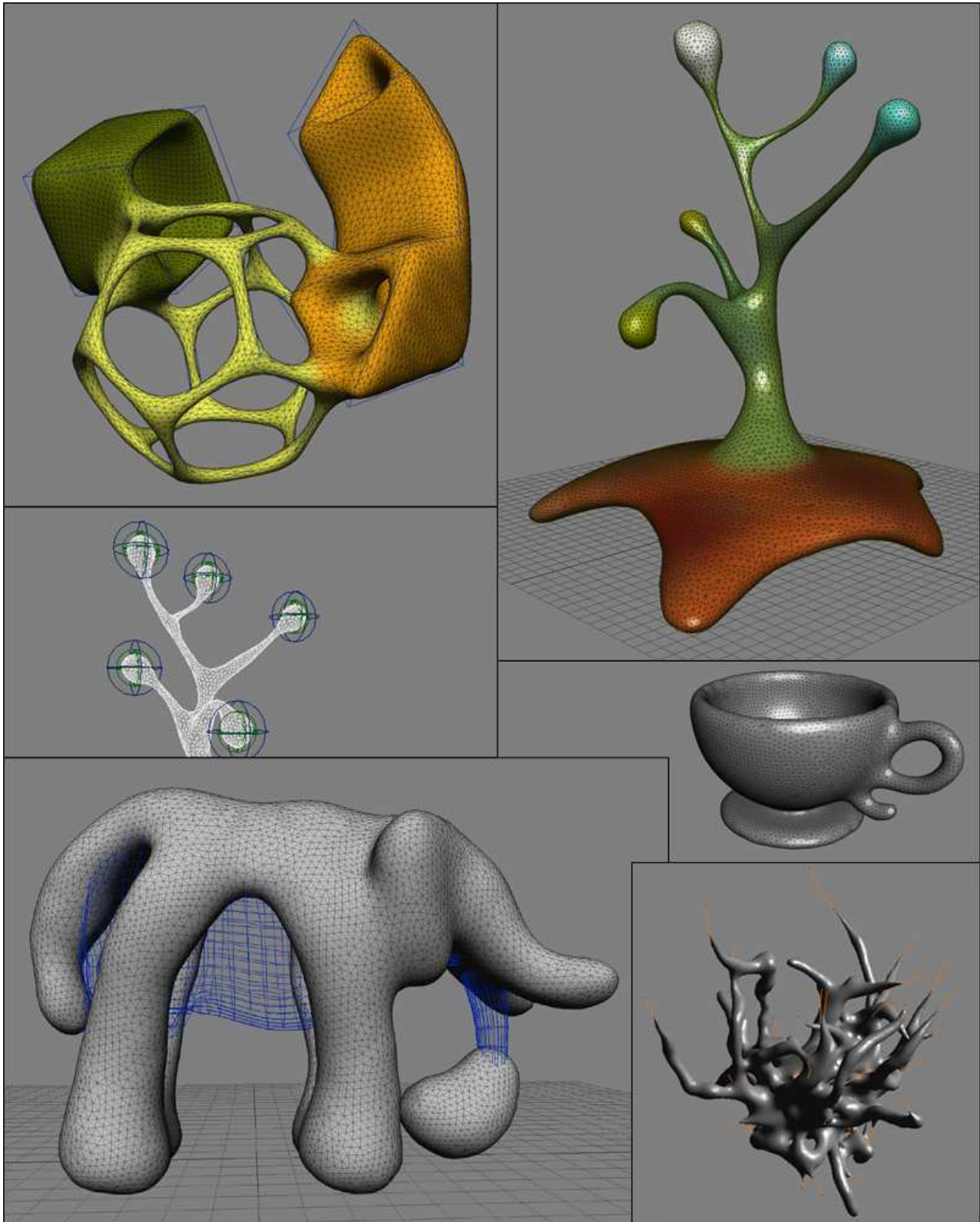
## Tutorial 4: Particle Strands

1. Load the scene “emPolygonize\_1.50\_Tutorial\_Particles\_Strands.scn”, which is included in the demo scenes.
2. Play back the scene just to see what it does. 2 particles with a strand trail are emitted per second and orbit around a null.
3. Select the point cloud and create an emPolygonizer mesh:  
**Model → Create → Poly.Mesh → emPolygonizer-Metaballs**
4. Try some different values for the parameter “**Smooth Mesh Strength**”, i.e. 10 or 20. You should get something like this:



Tutorial 5: Take a look at the “inAction” demo scenes and the example scenes

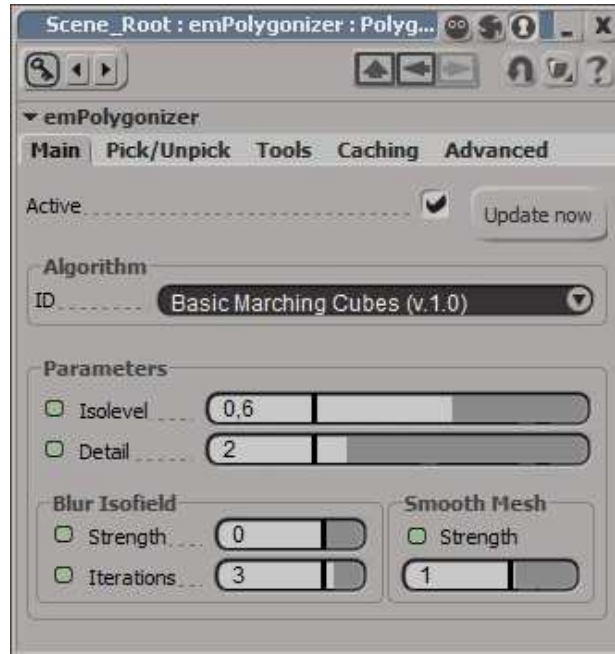
1. There are several scenes called “emPolygonizer\_inAction\_\*” / “emPolygonizer\_Example\_\*”.
2. Load the scenes and play them back (not all are animated though) to see some of the effects that can be created, here a few screenshots from the example scenes:



## Chapter 5: The Custom Operator's Parameters

Here is a brief description of each button and parameter of the custom operator's property page.

The tab “Main”



1. Parameter “Active” and button “Update now”  
Enables/disables the operator.  
If “Active” is unchecked you can force the operator to update the mesh by clicking on the “Update now” button.
2. Parameter “Algorithm”  
The algorithm that shall be used to create the mesh. The parameters below change depending on which algorithm is set.
3. Parameter “Isolevel”  
Defines the Isolevel limit at which geometry is generated.
4. Parameter “Detail”  
Defines how much detail is generated. Higher values result in more detail, but require more memory and more calculations.
5. Parameter “Strength” (Blur Isofield)  
Blurs the iso field and helps reducing the typical “blobby” look of basic marching cubes. When generating vertex colors it is recommended to set this value greater than 0. Set this equal 0 to disable iso field blurring.
6. Parameter “Iterations” (Blur Isofield)  
Amount of blur iterations. Use higher values with high detail values.

7. Parameter “Strength” (Smooth Mesh)

Smooths the final mesh via a so-called “Laplacian Smoothing”, a classic and fast algorithm.

Use high values (i.e. 15 or higher) to considerably reduces the blobbyness of the final mesh.

The tab “Pick/Unpick”



8. Button “Connect sel. Objects” (Any Type)

All currently selected objects are connected to the operator. Objects that are already connected or that are not supported (i.e. Nurbs Surfaces) are skipped.

9. Button “Disconnect sel. Objects” (Any Type)

All currently selected objects are disconnected from the operator. Objects that are not connected are skipped.

10. Button “Pick Objects” (Any Type)

Starts a pick session: pick objects of any supported type to connect them to the operator. Objects that are already connected or that are not supported (i.e. Nurbs Surfaces) are simply ignored.

11. Button “Unpick Objects” (Any Type)

Starts a pick session: pick the objects you want to disconnect from the operator.

12. Button “Pick” (Nulls)

Starts a pick session for nulls only: pick the nulls you want to connect to the operator. When the picking session is over a property page with the custom parameters of the picked nulls is displayed.

13. Button “Unpick” (Nulls)

Starts a pick session for nulls only: pick the nulls you want to disconnect from the operator.

14. Button “Create and connect a Null” (Nulls)

This button creates a null, connects it to the operator and opens its custom parameter set's property page. Furthermore, the null's display options are set in a way to reflect the settings of the parameters “Radius” and “Falloff”.

15. Button “Pick” (Curves)

Starts a pick session for curves only: pick the curves you want to connect to the operator. When the picking session is over a property page with the custom parameters of the picked curves is displayed.

16. Button “Unpick” (Curves)

Starts a pick session for curves only: pick the curves you want to disconnect from the operator.

17. Button “Pick” (Polygon Meshes)

Starts a pick session for polygon meshes only: pick the polygon meshes you want to connect to the operator. When the picking session is over a property page with the custom parameters of the picked polygon meshes is displayed.

18. Button “Unpick” (Polygon Meshes)

Starts a pick session for polygon meshes only: pick the polygon meshes you want to disconnect from the operator.

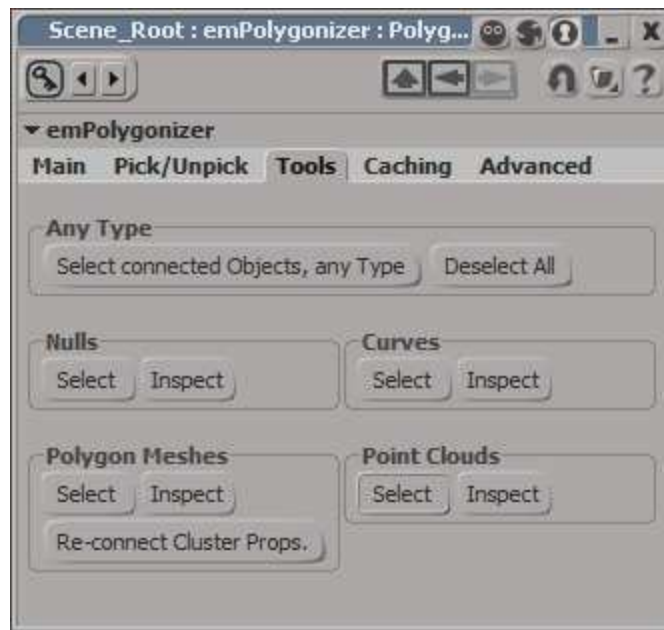
19. Button “Pick” (Point Clouds)

Starts a pick session for point clouds only: pick the point clouds you want to connect to the operator. When the picking session is over a property page with the custom parameters of the picked point clouds is displayed.

20. Button “Unpick” (Point Clouds)

Starts a pick session for point clouds only: pick the point clouds you want to disconnect from the operator.

The tab “Tools”



21. Button “**Select connected Objects, any Type**”  
Selects all objects that are connected to the operator.
22. Button “**Deselect All**”  
Clears the selection.
23. Buttons “**Select**” (Nulls/Curves/Polygon Meshes/Point Cloud)  
Selects all objects of a specific type that are connected to the operator.
24. Button “**Inspect**” (Nulls/Curves/Polygon Meshes/Point Cloud)  
Displays a property page with the custom parameters of all connected objects of a specific type.
25. Button “**Re-connect Cluster Props.**” (Polygon Mesh)  
Re-connects the cluster properties.  
When working with weight maps and/or vertex color maps it sometimes is necessary to re-connect all cluster properties, especially if some clusters or cluster properties were renamed.

## The tab “Caching”

*Note:* Geometry caching is only available for “marching cubes” algorithms.



## 26. Parameter “Action”

Tells the operator how it shall cache the geometry:

- “**No Caching**”: Disables all geometry caching (neither write nor read) => the geometry is always created through simulation.
- “**Simulate and Write**”: Always simulate (=create the mesh) and write it to disk (existing files are overwritten).
- “**Read only**”: always read cached data and never simulate. If no cached data was found then an error occurs.
- “**Read or Simulate**”: read cached data if available, else simulate (without writing data to disk).
- “**Read or Simulate and Write**”: read cached data if available, else simulate and write the data to disk. This is sort of a “Skip rendered Frames” mode.

## 27. Parameter Group “Range / Extrapolation”

The parameters in this group and their functions are self-explanatory and well-known, i.e. image sequences or animation mixer clips have similar parameters.

However it is important to know that the behaviour of the parameter “Action” can vary depending whether a frame range is used or not.

*If the parameter “Use range” is checked then:*

- the actions “**Read or Simulate**” and “**Read or Simulate and Write**” behave exactly like the action “**Read only**”, in other words: nothing is simulated nor written to disk!
- the action “**Simulate and Write**” will *always* simulate the mesh but *only write* the geometry to disk if the simulation frame is within the range defined by “Start Frame” and “End Frame”.

## 28. Parameter Group “File”

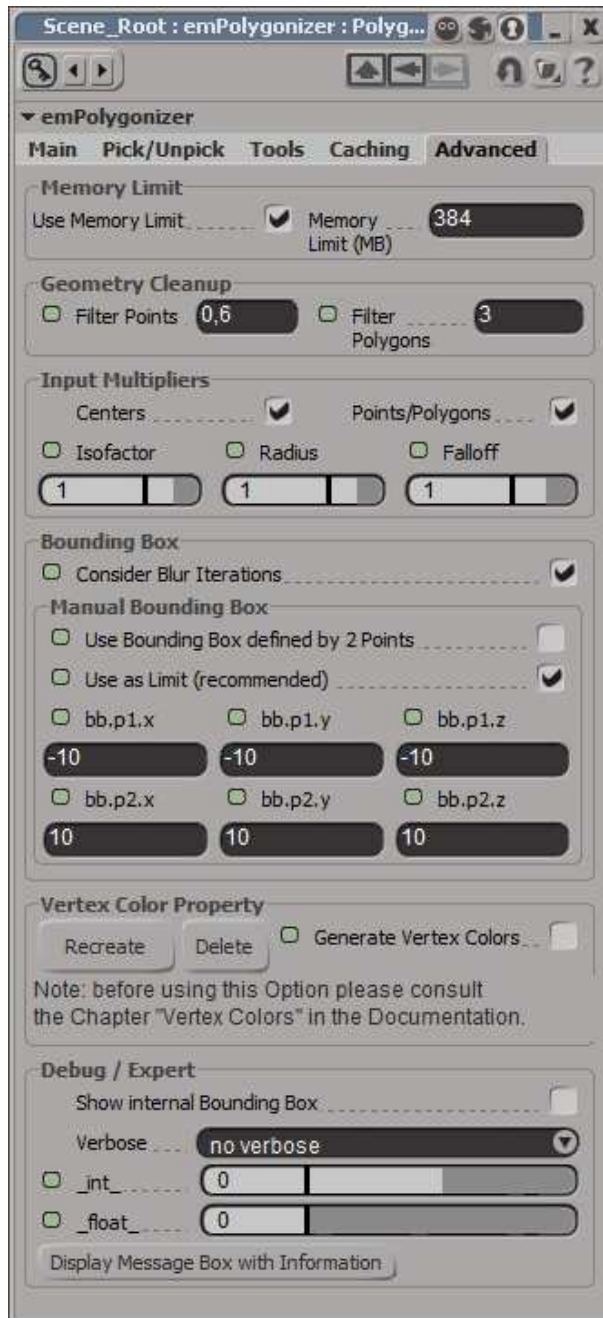
In this parameter group you define the path, file name and file format of the geometry cache files.

Currently there are three supported file formats, each with their own (dis)advantages:

- “**Wavefront (.obj)**”: the Wavefront obj file format is a well-known ASCII file format which is supported by most of the existing 3D packages, making it a good choice if you want to re-use the cached geometry in another 3D software or maybe even edit the files with a text editor.  
The disadvantages of the obj file format are the relatively large file sizes (longer read and write times, especially in a network) as well as the fact that vertex colors are not supported.
- “**Proprietary, ascii (.empff)**”: this proprietary file format is a basic ASCII format, similar to the Wavefront format.  
It has all the disadvantages of the Wavefront format (i.e. file size) and is definitely not supported by any other 3D package.  
Its two main advantages are the fact that it is an ASCII format (and therefore editable) and that this format supports vertex colors.
- “**Proprietary, binary (.empff)**”: this binary proprietary file format is the default for caching geometry. Its file size is the smallest, making it the best choice for distributed rendering, and it supports vertex colors.

The tab “Advanced”

*Note:* most of the following parameters are not visible/accessible in the demo version.



29. Parameters “Use Memory Limit” and “Memory Limit (MB)”

These two parameters let you use and define a maximum amount of memory, in megabytes, that the operator is allowed to use for creating the mesh.

These parameters are there to avoid creating gigantic meshes by mistake, i.e. by setting the level of detail to a huge number. That way, instead of trying to build a mesh with “trillions of polygons” (which would require many, many gigabytes of RAM and probably result in some sort of crash) the operator will simply cancel the operation with a warning/error, i.e. “error: reached memory limit”.

30. Parameter “Filter Points” (Geometry Cleanup)

This is similar to the XSI's “Filter Points” operator: points that lay near together are merged together.

Set this equal 0 to disable point filtering and to get that “faceted” look.

This parameter is only used for “marching cubes” and similar algorithms.

31. Parameter “Filter Polygons” (Geometry Cleanup)

Defines the strength of the polygon filter.

This parameter is only used for “marching cubes” and similar algorithms.

Note: This only has an effect if the parameter “Filter Points” is greater than zero.

32. Parameter Group “Input Multipliers”

This set of parameters let's you modify the isofactor/radius/falloff values of all connected objects by multiplying them with a value.

33. Parameters of group “Bounding Box”

This set of parameters let's you define a (global) bounding box in which “marching cubes” and similar algorithms shall operate. If disabled, then emPolygonizer automatically calculates the bounding box.

This parameter is only used for “marching cubes” and similar algorithms.

34. Parameters of group “Vertex Color Property”

More important information concerning vertex colors and some known issues can be found in the chapter “Vertex Colors”.

Here a quick overview:

◆ Button “Recreate”

Creates (or recreates) the vertex color property of the emPolygonizer mesh and checks the parameter “Generate Vertex Colors”.

This button MUST be clicked to have vertex colors, simply checking the parameter “Generate Vertex Colors” might not be enough.

◆ Button “Delete”

Deletes the vertex color property of the emPolygonizer mesh and checks the parameter “Generate Vertex Colors”.

As vertex colors slow down everything you most often just want to take a quick look at the vertex colors and then deactivate them again. Instead of only unchecking the parameter “Generate Vertex Colors” it is recommended to delete the vertex color property using this button → more performance!

◆ Parameter “Generate Vertex Colors”

Instead of directly checking/unchecking this parameter please use the buttons “Recreate” and “Delete” instead.

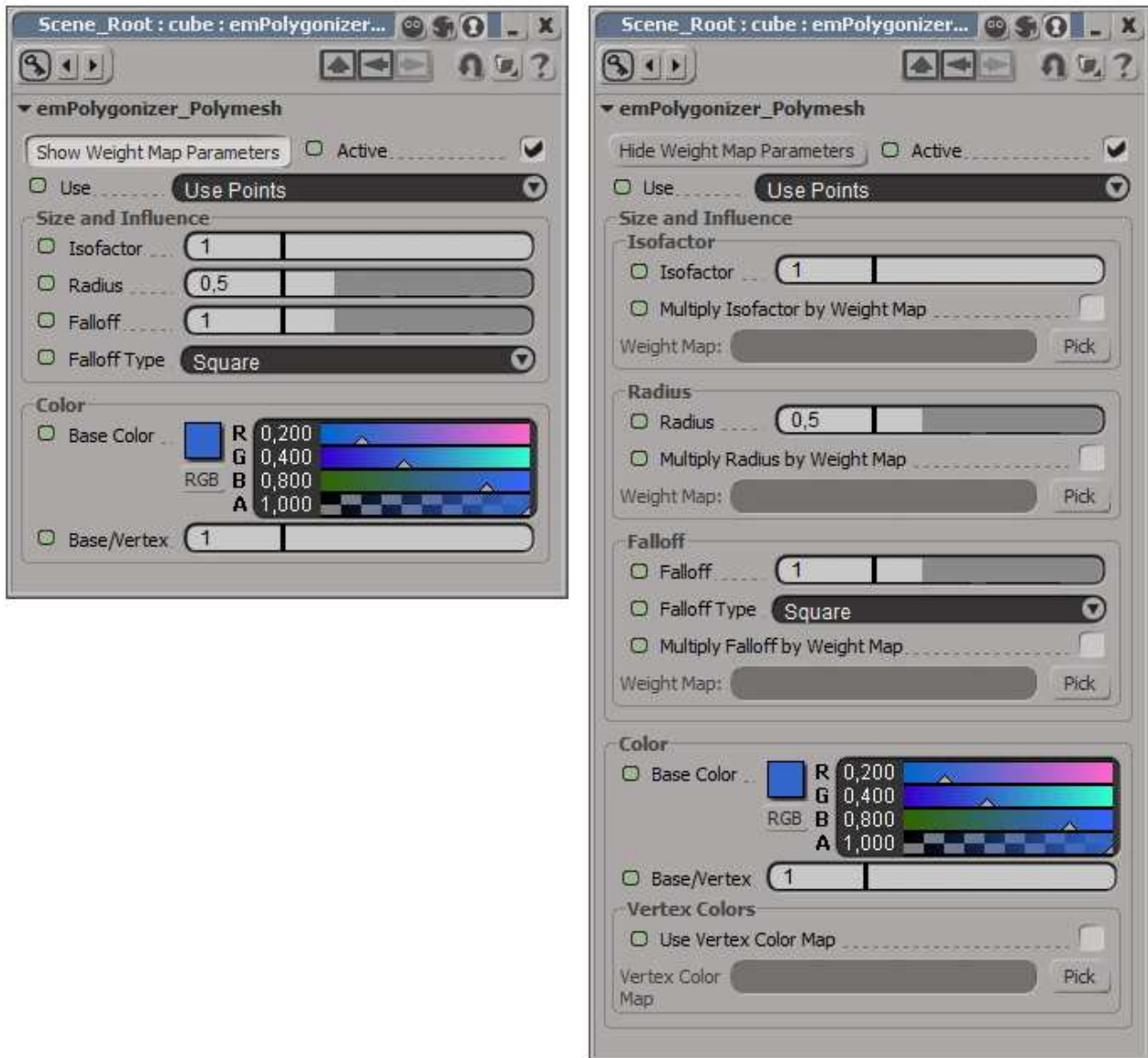
35. Parameters of group “Debug / Expert”

Some stuff for debugging when developing new features. The only interesting parameter is maybe “Verbose”: it let's you define the amount of logging in the history log.

Note: warnings and errors are always logged, even if “Verbose” is turned off.

## Chapter 6: The Custom Parameter Sets / Properties

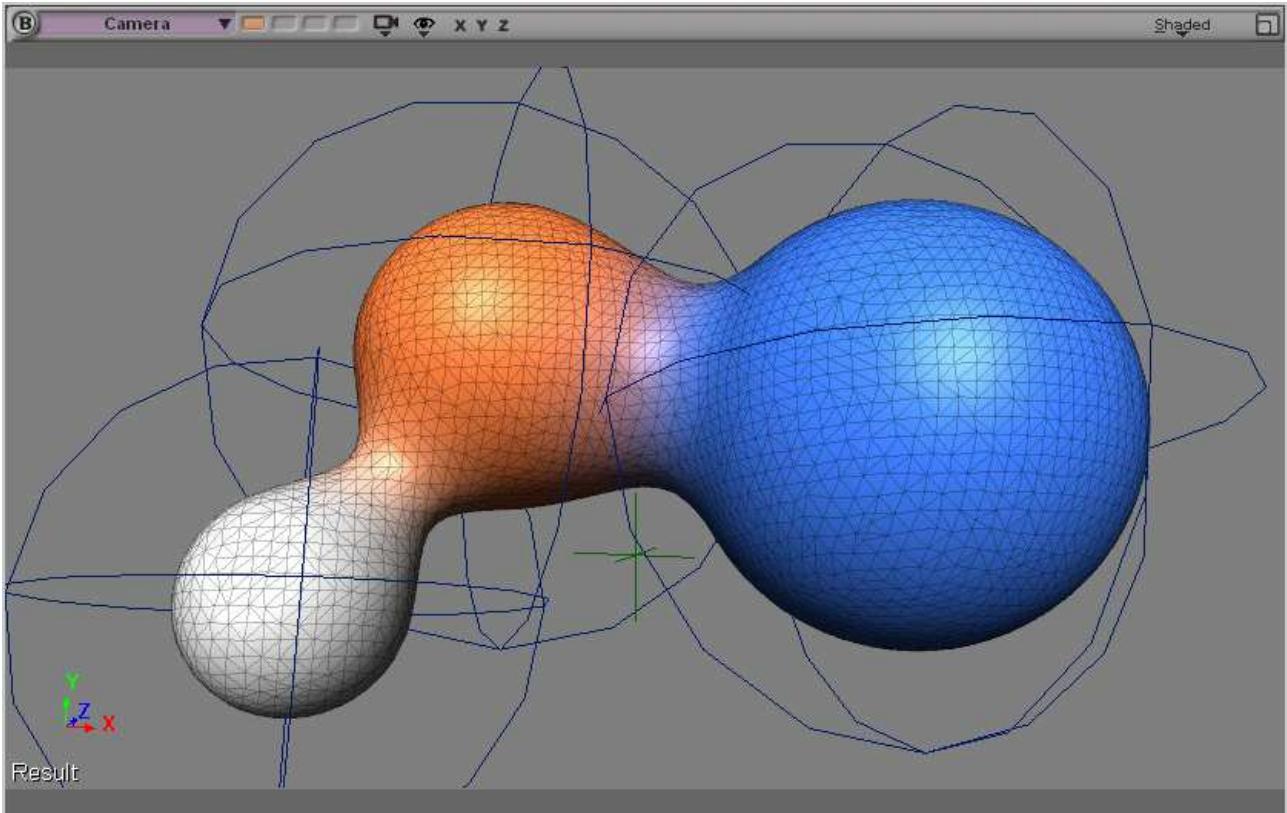
When an object is connected to the emPolygonizer operator it automatically gets a so-called custom property, which basically is a custom parameter set with a little VB script code behind it. Each type of object (null, curve, polygon mesh, point cloud) has its own custom property. The custom properties are attached under the objects and are more or less self-explanatory, except maybe for the one of the polygon meshes:



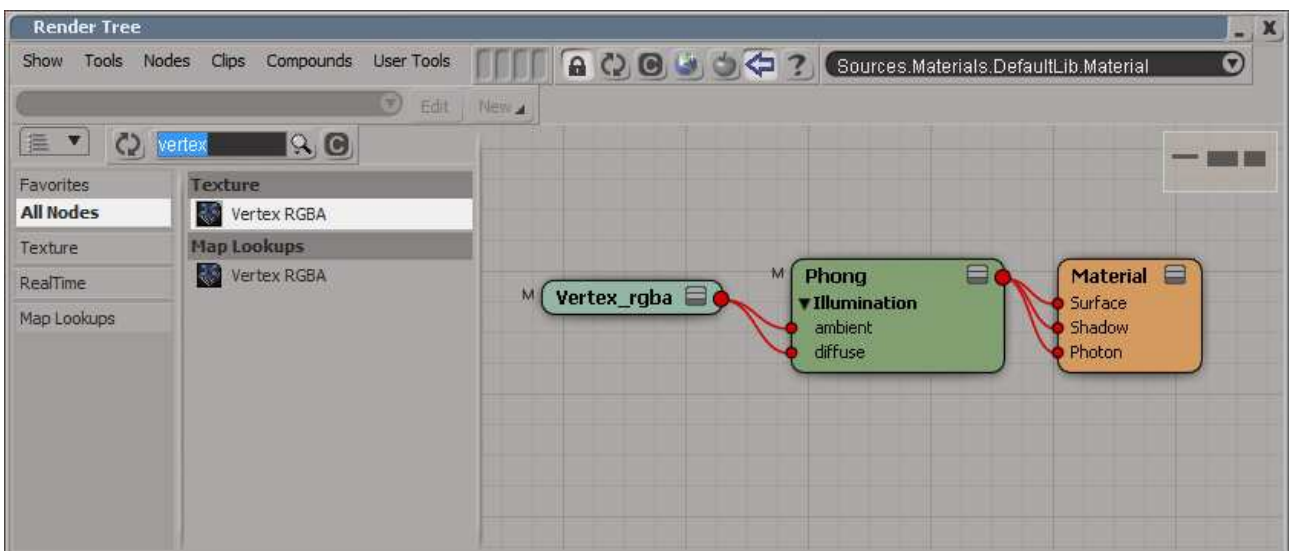
Notice the button “**Show/Hide Weight Map Parameters**” on the top left. By default the property page looks like the left one. By clicking on the “**Show Weight Map Parameters**” button the weight map and color vertices options are displayed. All important parameters can be driven by a weight map.

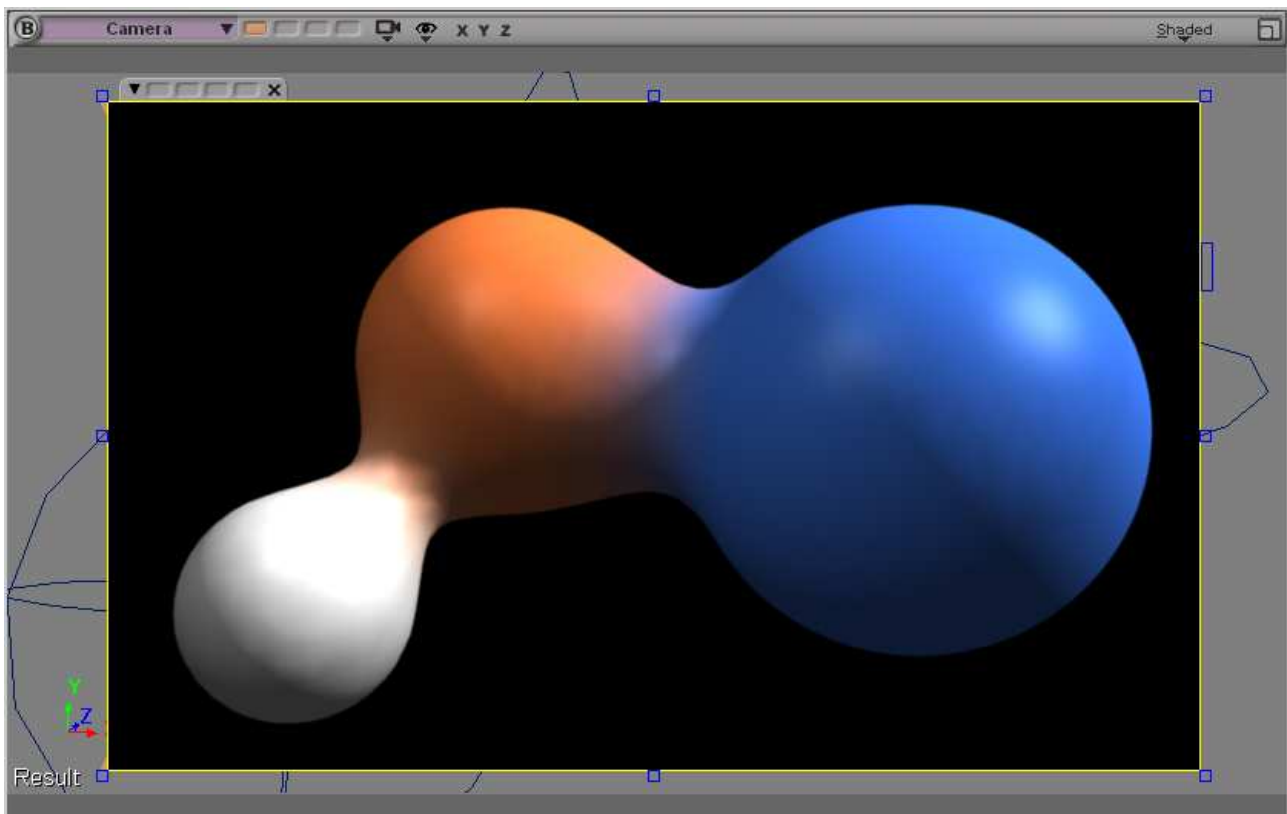
## Chapter 7: Vertex Colors

emPolygonizer can not only generate a mesh, it can also generate vertex colors for that mesh:



These vertex colors can then easily be accessed in a Render Tree via the shader “Vertex RGBA”:





The vertex colors can be set for each connected object separately via their custom property.

### Important known issues with vertex colors

**Issue 1)** Unfortunately you cannot generate vertex colors when rendering a sequence. There seems to be a conflict between mental ray and emPolygonizer concerning the vertex colors. This is very, very annoying, I know. Luckily there is a workaround! It's not very elegant but it works: Instead of rendering a sequence as you would normally do, you render the sequence frame by frame. Use the following little VB Script to render the current pass frame by frame. Simply copy and paste it into XSI's Script Editor and run it (save your scene before!).

```
'
set oPass = GetCurrentPass           ' get current pass object
pass      = oPass.FullName           ' get passes full name
frameStart = GetValue(pass&".FrameStart") ' get start frame
frameEnd   = GetValue(pass&".FrameEnd")   ' get end frame
frameStep  = GetValue(pass&".FrameStep")  ' get step
'
for frm = frameStart to frameEnd step frameStep
    SetValue "PlayControl.Current", frm ' set current frame
    Refresh                                     ' refresh UI
    RenderPasses pass, frm, frm          ' render the current frame
next
'
```

**Issue 2)** When you interactively make modifications to the emPolygonizer mesh, i.e. by moving a connected null via the translation tool, then the vertex colors often are corrupt. This is only a problem in interactive mode, it does NOT occur when you play back the scene. Concerning this problem you can also check the chapter "Tips, Tricks and Trouble Shooting".

## Chapter 8: Tips and Tricks, Troubleshooting

Here are a few tips and tricks as well as some troubleshooting when working with emPolygonizer.

- **Tip: many, many particles?**  
You can use point clouds that have really many particles, emPolygonizer can handle that with no problem. But using many large particles can sometimes take forever and at the same time produce rather “boring” meshes.  
Best is to use many small particles.
- **Troubleshooting: XSI crashes!!**  
That is probably due to the following known issue:  
XSI 7.xx sometimes crashes when "Smooth Mesh Strength" is greater than zero AND the "Geometry Approximation Subdivision Level" is also greater than 0.  
The good news: XSI 8.0 (aka Softimage 2010) doesn't crash.
- **Troubleshooting: I connected a point cloud and get the following weird message:**  
"WARNING: problem with object "PointCloud": ICEAttribute.GetDataArray2D() failed"  
This means that a required ICE attribute could not be found or that its data could not be read. In most cases it is the attribute “color”, especially when using strands.  
Workaround: simply uncheck the parameter “Use Point/Particle Color (if available)” in the point cloud's custom property.
- **Troubleshooting: when I load a scene that was build with an older version of emPolygonizer I don't see the new parameters in the property pages**  
Parameters and their definitions seem to be stored with the XSI scene. The workaround consists of deleting the old emPolygonizer mesh and creating a new one. It also is recommended to delete any old emPolygonizer custom properties.
- **Troubleshooting: when I load a scene that was created with emPolygonizer v. 1.0 the "Use Polygons" or "Use Points and Strands" parameters cannot be used**  
Workaround: right-click on the animation icon (on the left of the parameter "use"), choose "Edit Parameter definition" and set the Maximum Value Range to 100).
- **Troubleshooting: when I load a scene (created with the most recent version of emPolygonizer) I sometimes cannot set the parameter "Use" to "Use Points and Strands" or "Use Polygons and LOD"**  
This is another (annoying) problem with plug-ins and scripted custom operators: you probably loaded an old emPolygonizer scene before loading the new one. When you do that then XSI somehow “remembers” the old custom property definitions instead of using the new ones.  
Workaround 1: right-click on the animation icon (on the left of the parameter "Use"), choose "Edit Parameter definition" and set the Maximum Value Range to 100).  
Workaround 2: restart XSI and *don't* load an old emPolygonizer scene as the first scene.

- **Troubleshooting:** generating vertex colors makes everything slow  
Generating custom cluster properties, i.e. vertex colors or user normals, somehow takes quite some time.
- **Troubleshooting:** the generated vertex colors look bad when I interactively make changes yes, that also is a known issue. Just press the “Update” button in the main tab of the operator's property page to get the colors right.  
Note that the vertex colors *are correctly generated* when you play back the scene!  
Here an extract of XSI's SDK documentation:  
“SI currently does not fully support custom topology operators. The problem is that any cluster or cluster property will not properly update when a topology operator adds or removes points that belong to the cluster. In the worst case XSI may crash...”  
The good news: XSI does NOT crash when you use “generate vertex colors”. Only the interactive update is sometimes corrupted.
- **Troubleshooting:** there are ugly dark spots when generating vertex colors  
the solution: set the blur isofield parameter “Strength” to a value greater than zero.

## Chapter 9: Limitations and Remarks

### Limitations

- When generating vertex colors the undo function sometimes won't work properly, because the operator deletes and re-creates the vertex color property using commands.
- You cannot use “Generate Vertex Colors” when rendering a sequence. There is a workaround, though, please check the chapter “Vertex Colors”.
- There is a problem with gathering the strand color information in XSI 7.xx => the strands get the base color as defined in the emPolygonizer\_PointCloud property of the point cloud.  
Note: Getting the strand color works fine in XSI 8.0.
- XSI 7.xx sometimes crashes when "Smooth Mesh Strength" is greater than zero AND the "geometry approximation subdivision level" is also greater than 0.  
The good news: XSI 8.0 (aka Softimage 2010) doesn't crash.

## Chapter 10: Version History

new in Version 1.500

- The buttons in the tab “Tools” have been re–arranged for a better overview.
- Point Cloud Strands are now supported.
- Curves are now supported.
- Polygon mesh surfaces are now supported.
- A fast “Laplacian Smooth” has been implemented (Parameter “Smooth Mesh Strength”).
- Geometry caching has been implemented.
- New feature “Use two point bounding box as limit”.
- Bug fix: when pressing “go to next frame” (with “generate vertex colors” active) while the render region was still being rendered then the vertex colors were never re–calculated again.
- Better default values for “Geometry Cleanup”:  
“Filter Points” = 0.5 (old default value was 0.6)  
“Filter Polygons” = 5 (old default value was 3)

## Chapter 11: Thanks

Special thanks –as usual– to my buddy Oliver Weingarten who sent me a link about marching cubes saying “it might be a nice thing to have that in XSI” and gave me lots of advice and ideas for this plug–in.