

RGG User's Guide

Kitware, Inc.

June 26, 2015

Contents

1	Introduction	5
2	Installation	7
2.1	Windows	8
2.2	Mac	8
2.3	Linux	8
3	A Tour	9
3.1	Conceptual Overview And Terminology	10
3.1.1	Cores	10
3.1.2	Assemblies	10
3.1.3	Same As Assembly	10
3.1.4	Pins	10
3.1.5	Ducts	10
3.1.6	Materials	10
3.1.7	Meshes	10
3.2	Basic User Interface Overview	11
3.2.1	Main Window Layout	11
3.2.2	Input Panel	11
3.2.3	Properties Panel	12
3.2.4	3D View	13
3.2.5	2D View	14
3.2.6	Mesh View	14
4	Building Rectangular Cores	15
4.0.7	Creating an Assembly Duct	16
4.0.8	Creating a Fuel Pin	17
4.0.9	Adding Fuel Pins to the Assembly	18
5	Building Hexagonal Cores	19
5.1	The types of Hexagonal Cores	20
5.2	Simple Hexagonal Flat Core	20
5.3	Configuring Duct	20
5.4	Adding a Pin	21
5.4.1	Creating a Pin	21
5.4.2	Configuring the Pin	21
5.5	Populating assembly with our pin	22
5.6	Populating core with our assembly	23
6	Meshing	25
6.1	File Formats	26
6.2	What Makes A Good Mesh	26

6.3	RGG Preferences	28
6.4	Generating a Mesh	28
6.5	Displaying the Mesh	29
6.5.1	Views of the Mesh	29
	Main Index	31
	Interface Index	32

Chapter 1

Introduction

Welcome to the Reactor Geometry Generator User's Guide. Reactor Geometry Generator (RGG) is a program designed to aid you in modeling and meshing hexagonal and rectilinear reactor cores. RGG uses Qt and VTK to produce an intuitive user interface. By integrating a 3D view of the reactor with the meshing tools and combining them into one user interface, RGG streamlines the task of preparing a simulation mesh and enables real-time feedback that reduces accidental mistakes that waste hours of meshing. RGG also interfaces with MeshKit tools to consolidate the meshing process, meaning that going from model to mesh is as easy as a button click.

This guide is designed to acquaint you with RGG's interface and to give you the knowledge and skills to pilot RGG successfully. This book is organized in a concept/example manner, meaning that we cover concepts and the way that RGG thinks before we use those same ideas in an example. Towards the end, we present a reference section that fully explains both the function of particular menus, buttons, and panes and what menu, button, or pane you must use to accomplish your objective.

Chapter 2

Installation



Figure 2.1: Beginning the installation process.

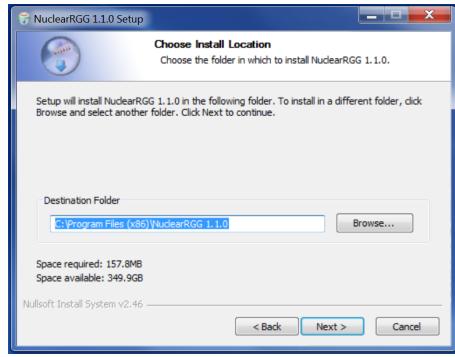


Figure 2.2: Selecting the install location.

RGG installers can be obtained at <http://www.computationalmodelbuilder.org/download/> for all three major operating systems: Windows, Mac OS X, and Linux. Detailed installation guides are detailed below by platform.

2.1 Windows

A security warning may come up and ask for permission to allow the installer to make changes to your computer. Be sure to click **Yes** to allow the installation process to continue. This may require authentication by your computer's administrator.

A window like Figure 2.1 should come up.

Click **Next** to continue and agree to the license displayed. On the next screen (shown below), either use the default install location or select one of your own (Figure 2.2. Click **Next**.

On the next window (shown below), select whether or not you'd like a Start Menu Folder to be created for RGG. If you would not link a Start Menu Folder to be created, select the **Do not create shortcuts** box. Otherwise, confirm the Start Menu Folder name you'd like to use (Figure 2.3. Click **Install** to have the installer extract the binaries.

A window should come up to confirm that RGG has been successfully installed on your computer. At this point, you can click **Finish** to close this wizard.

2.2 Mac

We use a standard drag and drop installer on Mac. Simply mount the .dmg file and drag the RGG app into the Applications folder alias.

2.3 Linux

We provided a self-contained tar.gz file. To install, one extracts the folder and places its contents. In order for the executable to run, the bin and lib file must be at the same directory level.

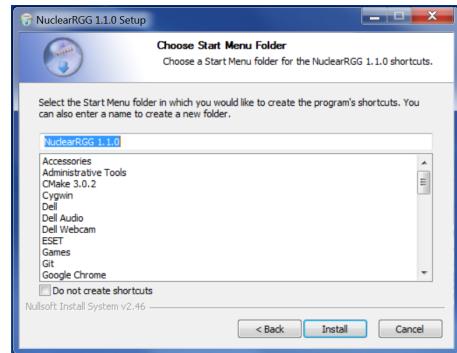


Figure 2.3: Configuring Start Menu Folders.

Chapter 3

A Tour

3.1 Conceptual Overview And Terminology

There are several concepts associated with RGG that must be understood before the user interface or workflow will seem sensible.

3.1.1 Cores

The highest-level unit that RGG constructs is the core itself. Cores can have either a rectilinear or hexagonal geometry to them. This helps to define their lattice, or the grid on to which assemblies can be placed. Cores are made up of assemblies arranged in a particular manner on the core lattice. These cores can be translated into a mesh by MeshKit.

3.1.2 Assemblies

As stated above, assemblies comprise cores. Assemblies specify an arrangement of pins and ducts on their own lattice. Think of assemblies as a configuration or specification of pins and ducts.

3.1.3 Same As Assembly

There are times where the user want to have an assembly with different material set and Neumann set start ids. This is accomplished by a Same As Assembly, which links the desired assembly with the slightly different parameters.

3.1.4 Pins

Pins are cylinders or frustums that model fuel pins, control rods, and the like. They have a name, a label, a material or set of materials, and other diverse properties.

3.1.5 Ducts

Ducts are what surround the fuel pins – in combination, they define the material composition of the space between pins in the lattice cells.

3.1.6 Materials

Materials describe physical properties of a pin or duct. Materials are assigned to pins and ducts in order to produce a mesh that can be used to accurately perform simulations.

3.1.7 Meshes

A mesh is a tetrahedral (triangular pyramid) or hexahedral (rectangular prism) representation of the core at a level of granularity sufficient for accurate simulation. In RGG, they are produced by MeshKit. This is the end product of RGG; the mesh is then fed into some other analytical software to perform the requested simulation.

Now that we've covered some high-level concepts, we're ready to look at an overview of the user interface.

3.2 Basic User Interface Overview

3.2.1 Main Window Layout

All user interaction takes place within the main window. The main window is comprised of several movable widgets that can moved around or floating, depending on what best facilitates the work flow. The main movable widgets are `input`, `properties panel`, `3D view`, `2D view`, and `Mesh view`. At start `3D view`, `2D view`, and `Mesh view` are combined together in the visualization area, selectable by tab, but they can be moved. There are two non-movable widgets. The first non-movable widget is the `toolbar`, located above the `input` and has icons for often used actions and the z-scaling controls. The second non-movable widget is the `menu`, located above the `toolbar`. Refer to Figure 3.1 to see these pictorially.

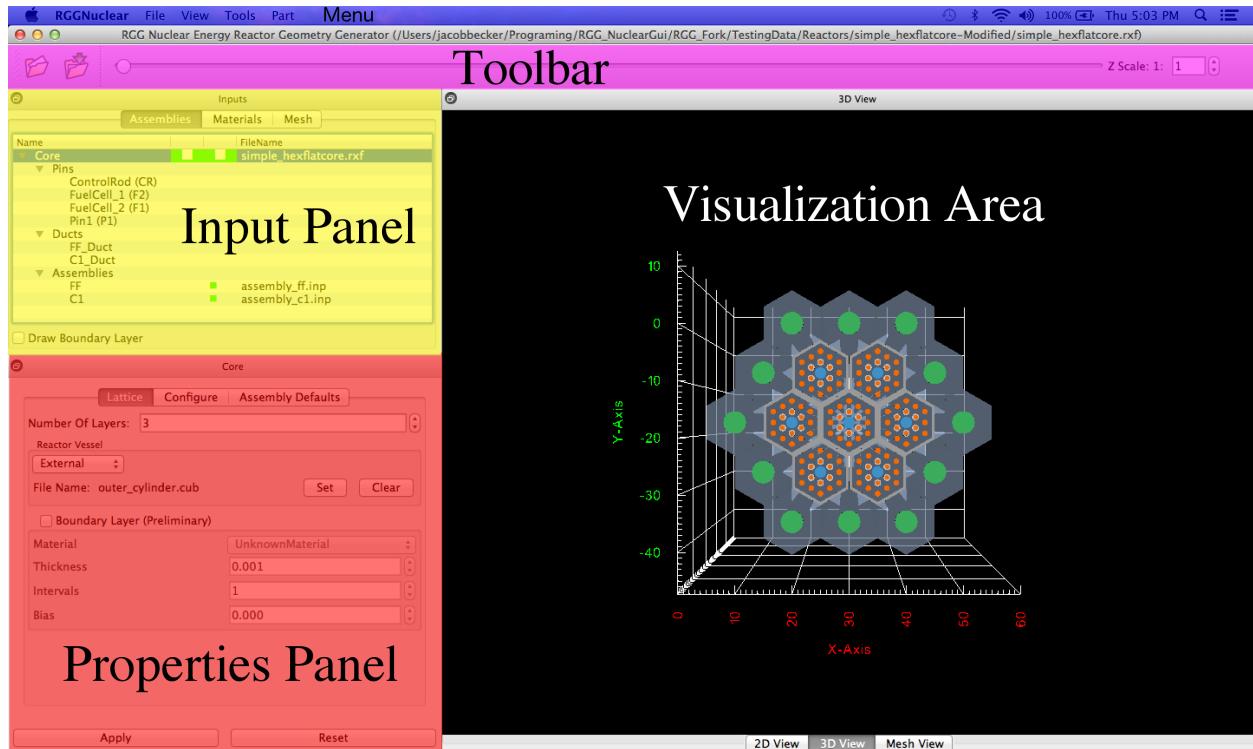


Figure 3.1: Layout of the main window.

3.2.2 Input Panel

The `input` panel has three tabs: the `assemblies` tab, the `materials` tab, and the `mesh` tab.

Assemblies Tab

The `assemblies` tab shows the hierachal nature of the core, assemblies, pins and ducts. Recall that a core is composed of assemblies which are placed in the core's lattice, and that these assemblies are in turn composed of pins and ducts that are placed in the assembly lattice. When

Note you can check to see if the core or an assembly has had changes made to it since the last save. A `pencil` icon (as shown in Figure 3.2) indicates that edits have been made since the file was last saved. A `green box` icon indicates that the file is up to date.

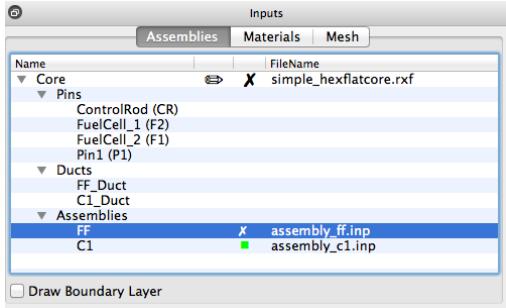


Figure 3.2: The assemblies tab shows the core, assembly, pin, and duct hierarchy.

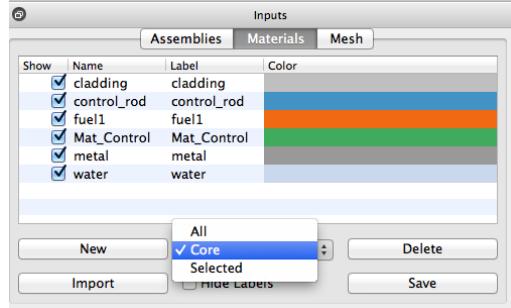


Figure 3.3: The materials tab shows available materials.

You can also see the status of mesh files here. An **x** icon (as shown, to the right of the **pencil** icon) indicates that mesh files have not been generated (or haven't been recreated since edits to the associated core or assembly), while **green box** icons indicate that they are up to date.

Materials Tab

The **materials** tab (Figure 3.3) details the list of available materials, their associated colors, and whether or not they are viewable. You can use this tab to create, remove, import, and export materials, as well as edit the labels and colors of materials and toggle whether or not they are shown in the **3D view** or **Mesh view**. You can also filter the list based on materials used in the core or currently displayed in either the **3D view** or **Mesh view**.

Mesh Tab

The **mesh** tab (Figure 3.4) allows you to control the viewing of the mesh. It is only available when a mesh is loaded. It has the options to control what shows up in the **Mesh view**, including the ability to show volumes, boundaries, surfaces, Neumann sets, Dirichlet sets, Material sets, show mesh edges, and to colorize the different volumes. More information on this tab is given in Section 6.5.

3.2.3 Properties Panel

The **properties panel** changes depending on the current state of the input panel's **assemblies** tab.

- If you're clicking on a core, it will display three tabs: the **lattice** tab, the **configure** tab, and the **assembly defaults** tab.
- If you're clicking on an assembly, it will display the **lattice** tab and **configure** tab. Note that the assembly has a different **lattice** tab and **configure** tab from the core.
- If you're clicking on a duct, it will display the **duct** tab.
- If you're clicking on a pin, it will display the **pin** tab.

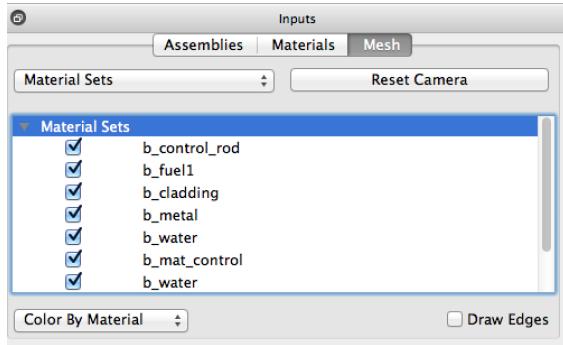
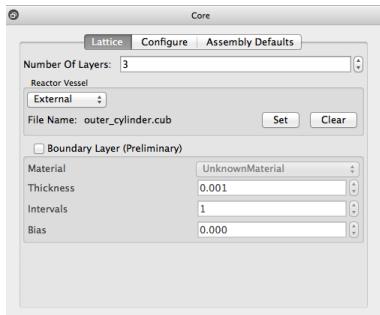
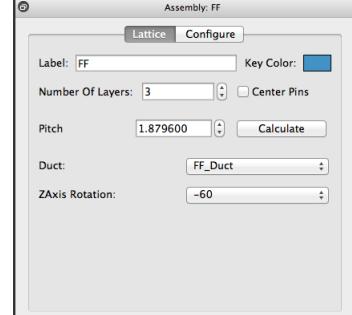


Figure 3.4: Used to control the mesh



(a) Core Lattice Tab.



(b) The Assembly Lattice Tab.

Figure 3.5: The two versions of the Lattice Tab.

Lattice Tab

The **lattice tab** controls the number of layers (in a hexagonal core) or the dimensions (in a rectilinear core) to get the desired lattice geometry. For cores, it also controls the **Reactor Vessel** and **Boundary Layers**. For assemblies, there is a checkbox to allow for auto-centering the pins. Also for assemblies, this tab controls the 2D **view** color and label. There is also controls for assembly rotation and the **duct** used. See Figure 3.5.

Configure Tab

This tab allows you to customize other meshing parameters from MeshKit files which are not currently provided an interface in the GUI. See Chapter 6 for information on how to mesh your core.

Assembly Defaults Tab

The **assembly defaults tab** allows you to specify general meshing information. You can set whether the mesh you want to generate will be tetrahedral or hexahedral, and also change the default dimensions of the ducts comprising your core. Changing these values will propagate all the way down to every component, so you can change it all in one place instead of adjusting the height of every duct in every assembly.

Duct Tab

The **duct tab** exposes the different configuration options for a duct piece. You can change the position and dimensional settings towards the top, the duct pitch, and the duct material (with associated normalized thickness, if desired).

Pin Tab

The **pin tab** allows you to change the key color of the pin, specify the pin material(s), and create, remove, or edit pin pieces.

3.2.4 3D View

The **3D view** (Figure 3.6) displays a representation of the current component. You can use the following controls to interact with it:

- To rotate, click with the left mouse button.
- To pan, either shift-click with the left mouse button or use the middle mouse button.
- To zoom, click with the right mouse button.

Similar to the **properties panel**, the **3D view** changes depending on what is selected in the assemblies tab of the input panel.

- If you're clicking on a core, it will display the entire core.
- If you're clicking on an assembly or a duct, it will display the assembly.
- If you're clicking on a pin, it will display the pin.

3.2.5 2D View

The **2D view** (Figure 3.7) provides a 2d schematic of the core or assembly depending which is selected. Each assembly or pin represented here will be filled with its corresponding key color.

You can edit this information by moving your mouse over the cell you want to edit and then right-click on each cell to select the desired assembly or pin. In the pop-up menu, there are options to **Replace All With** and **Fill Ring With**. **Replace All With** replaces every cell that is same component with the desired assembly or pin. **Fill Ring With** fills every cell in the ring with the desired assembly or pin. Alternatively, you can drag and drop an existing component onto the cell.

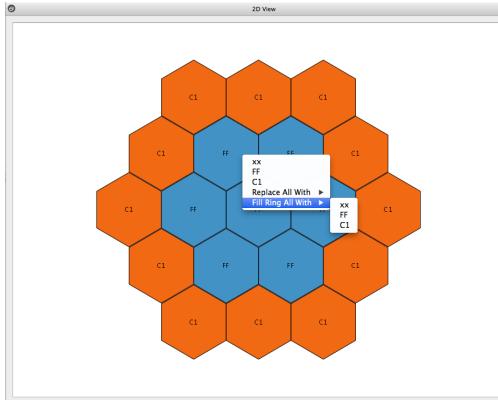


Figure 3.7: The 2D View

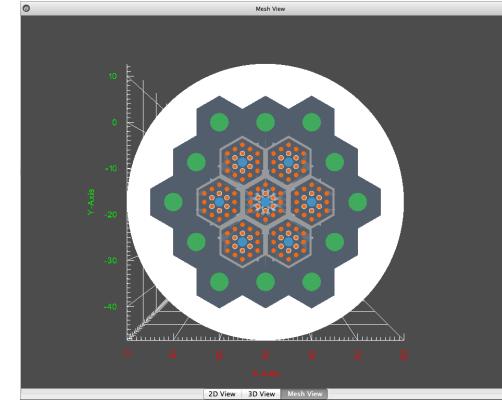


Figure 3.8: The Mesh View

3.2.6 Mesh View

The **Mesh view** (Figure 3.8) is only visible if a mesh has been loaded. The Mesh Tab controls what is displayed here.

Chapter 4

Building Rectangular Cores

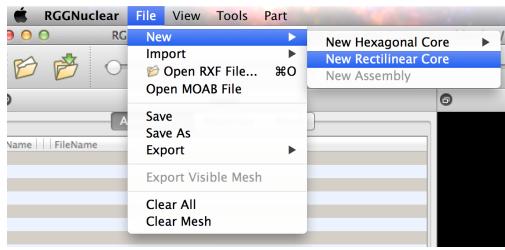


Figure 4.1: Select the option to create a new rectilinear assembly duct.

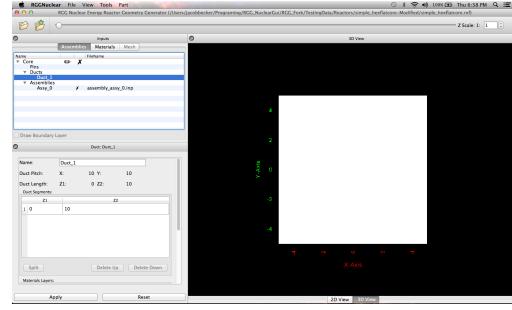


Figure 4.2: The initial rectangular core.

4.0.7 Creating an Assembly Duct

First, create a new rectilinear core, and add a new duct to the assembly by following Figure 4.1.

This results in a core with a single assembly with a duct of unknown material shown in Figure 4.2.

Now, to adjust the size of the duct, select the "Assembly Defaults" tab in Core's Properties Panel. In this example, we are going to set the "Duct Thickness" to 15 (Figure 4.3).

Next adjust the material of the Duct. In the Inputs Panel, select the Duct, in this case Duct_1. Then, in the Duct's Properties Panel, select the first Duct Segment. This populates Material Layers with the Duct Segments materials. Now, select water in the material drop down box drop (Figure 4.4a). Then press Apply. The result should look like Figure 4.4b.

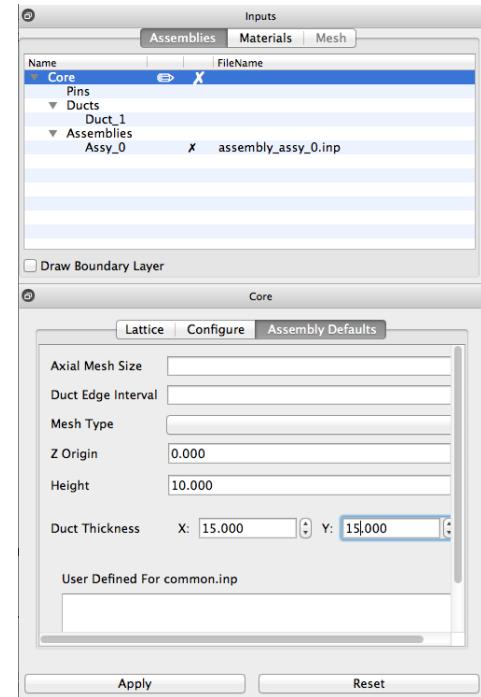
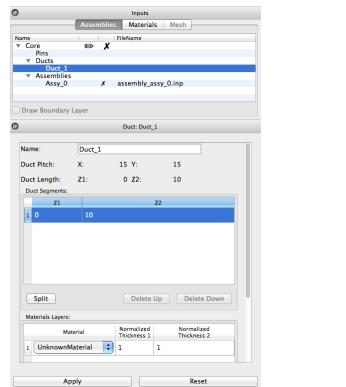
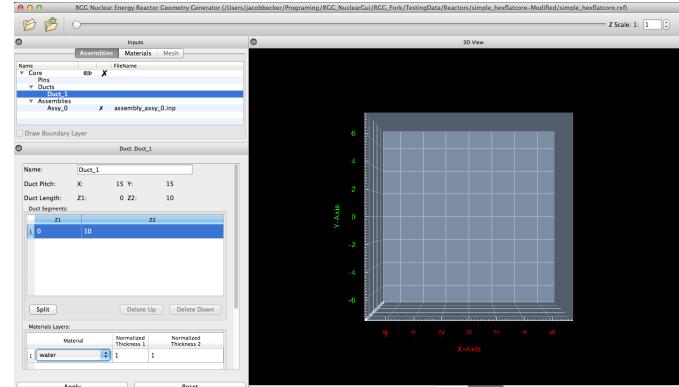


Figure 4.3: Setting the global duct parameters.



(a) Setting material of the duct.



(b) The final product.

Figure 4.4: Finished water duct.

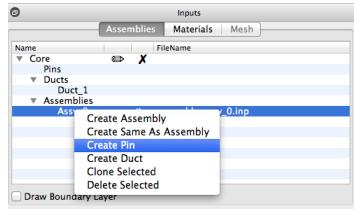


Figure 4.5: Create pin option.

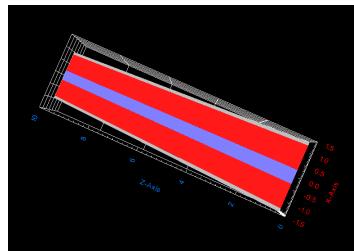


Figure 4.6: Final fuel pin product.

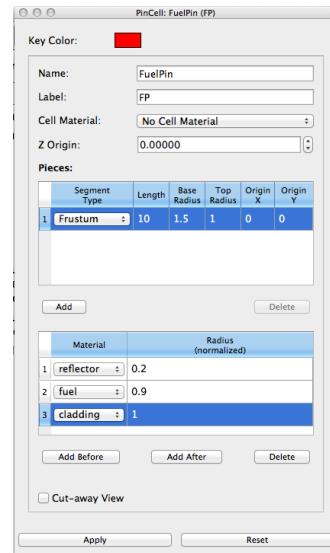


Figure 4.7: Fuel pin parameters.



Common Errors

When finishing editing properties, make sure to hit apply before switch to another component in the Assemblies tab.

4.0.8 Creating a Fuel Pin

Right-click on the assembly and choose “Create Pin” (Figure 4.5).

Changing the Parameters

Choose the dimensions of the pin to correspond to the dimensions of your duct. In our case we have used a length of 10, corresponding to the height of the duct.

Changing the key color and label is recommended, because later this will make the fuel pin more recognizable while organizing the assembly.

Using the “Add Before” button, you can create a fuel pin with different layers of materials. Refer to Figure 4.7 for the parameters used for our example, and Figure 4.6 to view the final product.

Note also the option to toggle the fuel pin’s shape between a cylinder and a frustum. The “Cutaway view” helps to view how the different materials are distributed within the fuel pin.

Control rods are used to control nuclear fission. They are made out of materials which can easily absorb neutrons without undergoing fission themselves such as certain isotopes of boron, silver, or cadmium.

This is created using largely the same process as creating the fuel pin. You can use the “control rod” material. Once again, it’s beneficial to change the key color to green.

Refer to Figure 4.8 for the final product.

4.0.9 Adding Fuel Pins to the Assembly

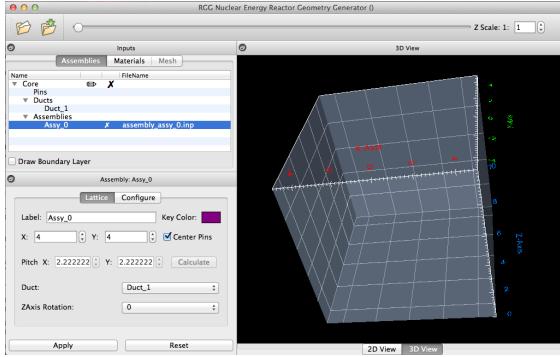


Figure 4.9: Empty duct in the assemblies view.

Now we want to add the pins we've created back to the assembly. Click back on "Assy_0" in the assemblies view in order to see the assembly layout. You should once again be able to see the water duct you created, and it should still be empty. Compare your view with Figure 4.9.

Right click on each square and choose which pin to assign to it. For the fuel rod, we want it to fill the center ring, so we select fuel in **Fill Ring With**. Alternatively, you can drag and drop pins already placed onto new squares.

In the end you should end up with something like 4.11.

Remember to click "Apply" in order to see your changes in the render window, or save them before navigating to another tab or pin.

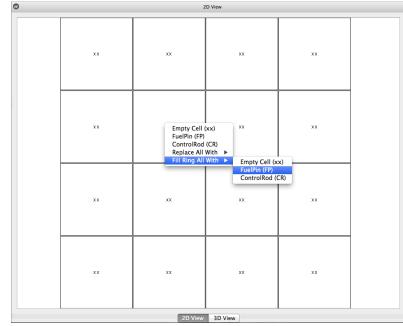


Figure 4.10: Right-click to organize the assembly.

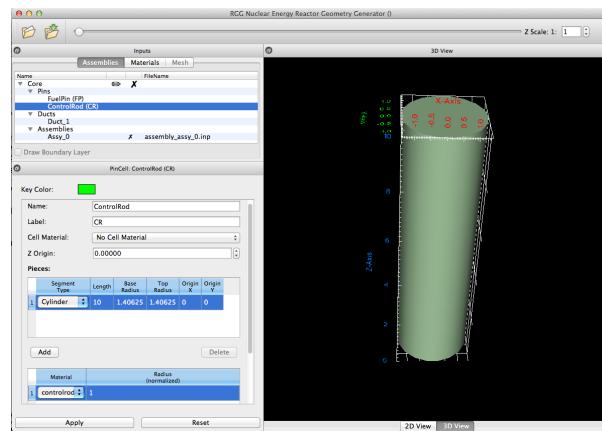


Figure 4.8: Final control product.

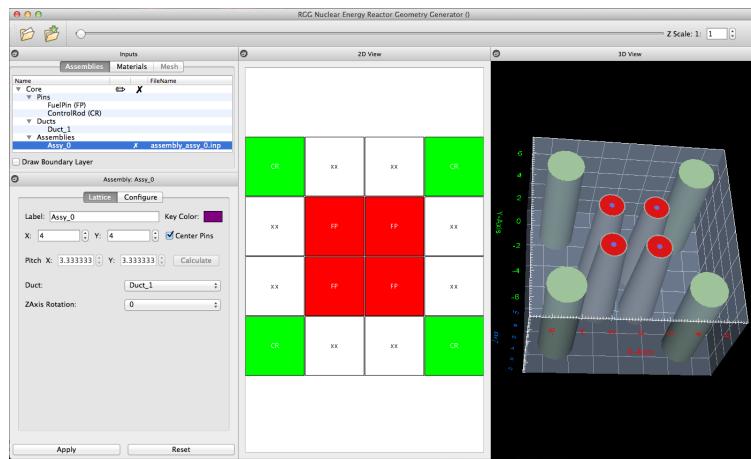


Figure 4.11: Final assembly.

Chapter 5

Building Hexagonal Cores

5.1 The types of Hexagonal Cores

RGG has built-in support for a variety of hexagonal-based cores, including full hexagonal cores, $\frac{1}{6}$ hexagonal flat cores, $\frac{1}{6}$ hexagonal vertex cores, and $\frac{1}{12}$ hexagonal cores. In this example we will construct a full hexagonal core.

5.2 Simple Hexagonal Flat Core

First, we'll need to create a new hexagonal core.

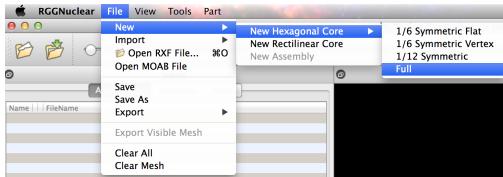


Figure 5.1: Select the option to create a new hexagonal core.

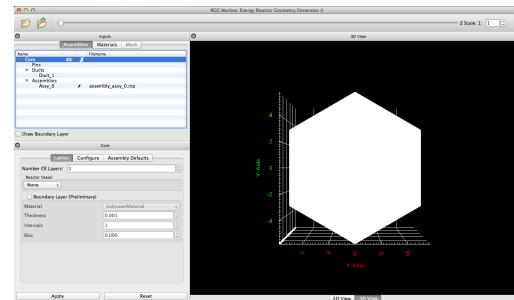


Figure 5.2: The initial view after created a new hex core.

The two panels on the left of the application will change along with the 2D and 3D views. We'll now see that the inputs panel contains an item named “Core” with a subitem “Assy_0.” The core panel should show a hexagon labeled “Assy_0.” The 2D and 3D view will contain the initial core. Confirm that your application looks similar to what's shown in Figure 5.2.

Next, we'll specify how many layers we'd like our lattice to have. In the core panel, either type in the number of desired layers or use the buttons to the right of the field to increment and decrement the number. The panel should look something like what's displayed in Figure 5.3.

Be sure to click the apply button below to see apply changes (Figure 5.4).

5.3 Configuring Duct

We now need to configure the duct. In the inputs panel, select the duct. This will change the lower panel to show the materials of the duct control panel. Next select the only segment in the “Duct Segment” table. This will populate the “Material Layers.” For this duct, we'll stay simple and make the material water. Change the material type to water by clicking the drop-down and selecting “water” (Figure 5.5)

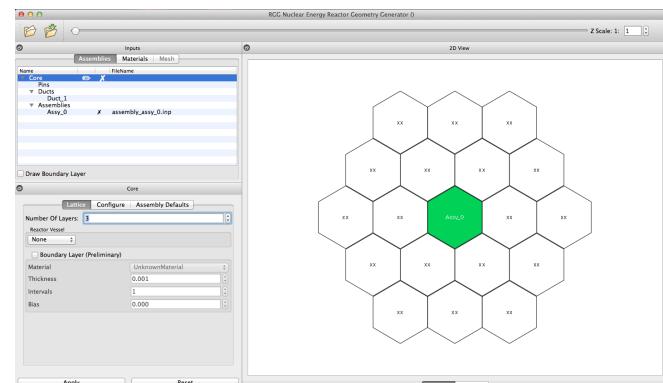


Figure 5.3: Updating the number of layers of the core.



Figure 5.4: The apply button.

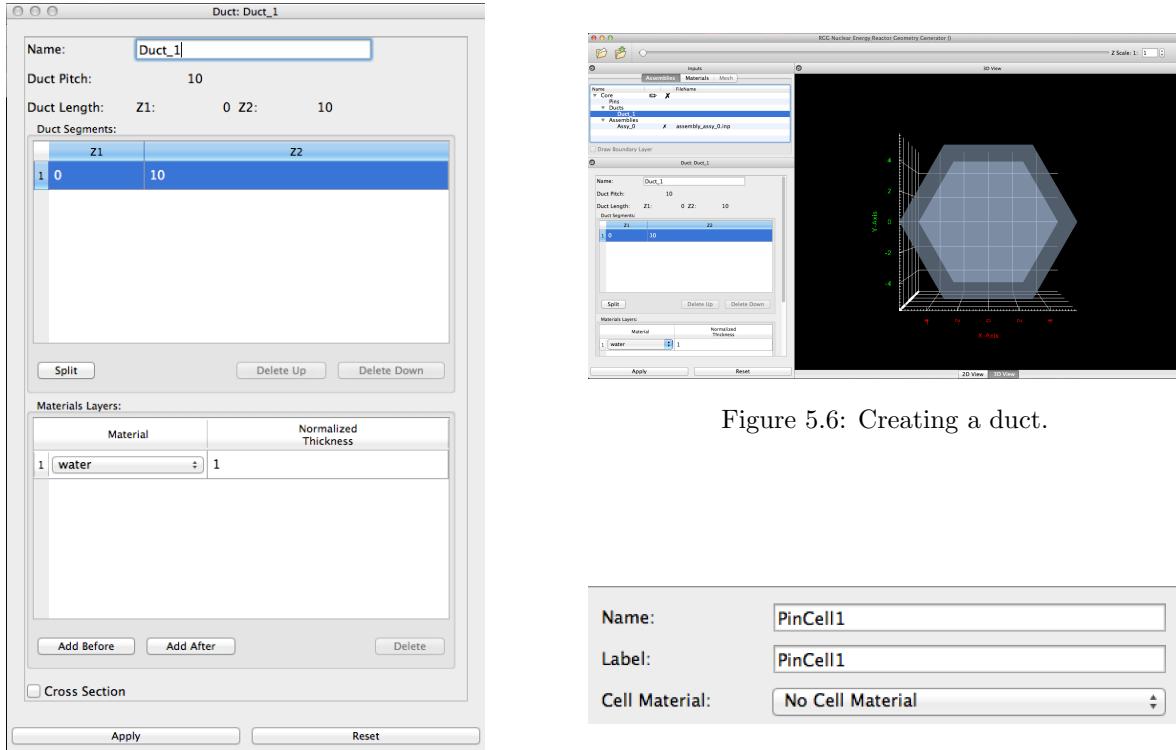


Figure 5.5: Selecting water as our material.

Your screen should now look like Figure 5.6.

5.4 Adding a Pin

5.4.1 Creating a Pin

To create a pin in the assembly, right click on the name of the assembly and select “Create Pin” (Figure 5.7).

5.4.2 Configuring the Pin

You can edit the name, label, and cell material of the pin. Refer to Figure 5.7 to see how to change them. We will leave them to the defaults.

We now need to add pieces of the pin. There are two kinds of pin pieces: frustums and cylinders. On creation of the pin, a cylinder piece is added. This can be changed to frustum using the drop-box.

Then, confirm that our segment type is a cylinder, and that the sum of the length of the segments is equal to the length of the duct (10) (Figure 5.8).

Now, we’re going to modify the material of this cylinder (Figure 5.9). We’ll make this a control rod, so we’ll change the material to match.

Be sure to press apply to ensure your changes go into effect.

We should see a screen like Figure 5.10 the after we’re done.

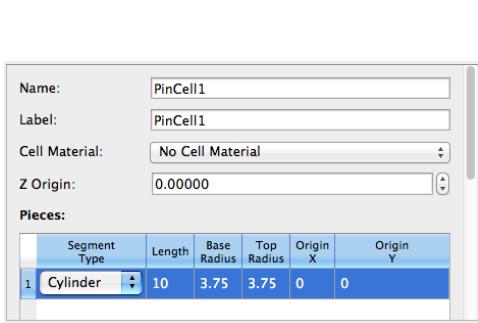


Figure 5.8: Confirming our cylinder's configuration.

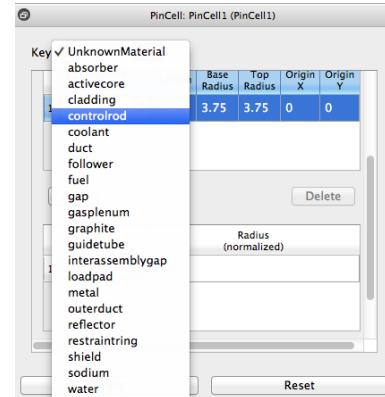


Figure 5.9: Changing the material to be a control rod.

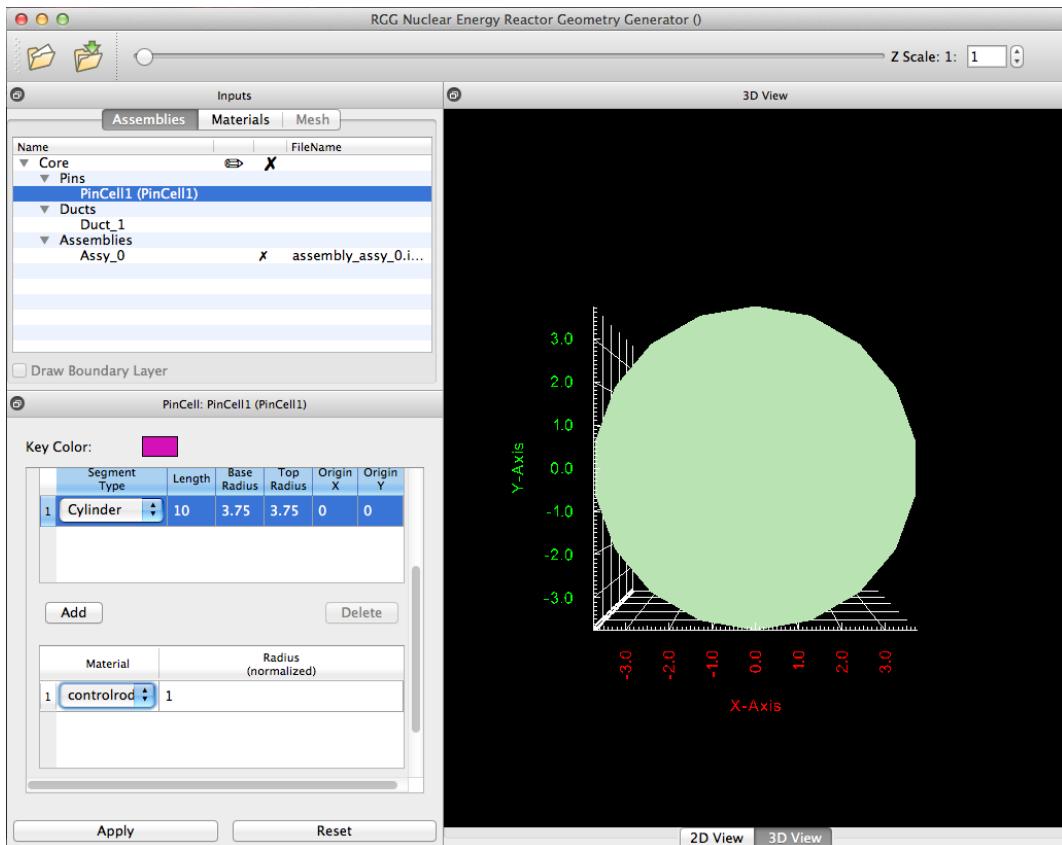


Figure 5.10: After adding the cylinder.

5.5 Populating assembly with our pin

Next, we need to add this pin to our assembly lattice. Make sure you select “Assy_0” in the inputs menu. Click the lattice tab in the lower panel, and right click on the hexagon to bring up a list of available pins. Select our pin (Figure 5.11).

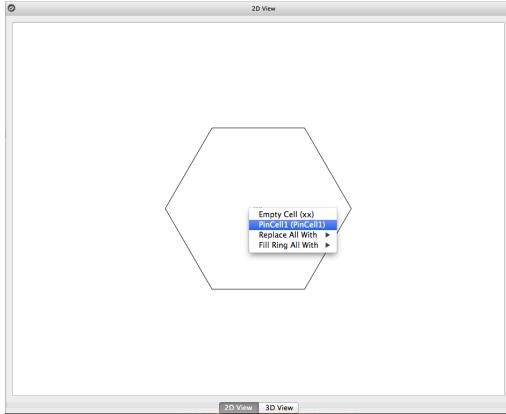


Figure 5.11: Selecting our pin for the assembly.

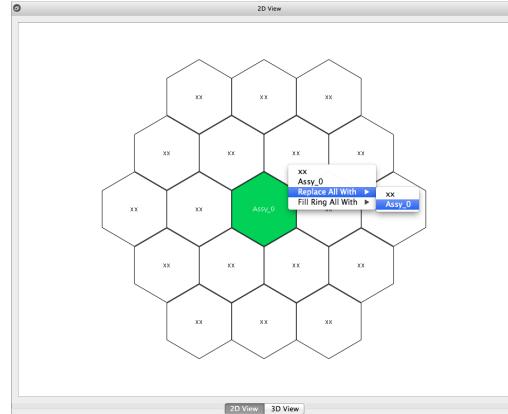


Figure 5.12: Selecting assembly for a cell.

Click the apply button to save your changes.

5.6 Populating core with our assembly

Click on our assembly (“Assy_0”) in the inputs pane. In the lower pane, click on the lattice tab. To add the assembly to any of the cells, right click to bring up a list of available assemblies, and then click on the assembly you’d like (Figure 5.12).

We’ll select Assy_0 because that’s the name of the assembly we’d like. In this case, we want to replace all remaining empty cells with Assy_0, so we use the Replace All With. Make sure to click the apply button to make sure that the changes we’ve made are applied. Your screen should look like Figure 5.13 when you’re done.

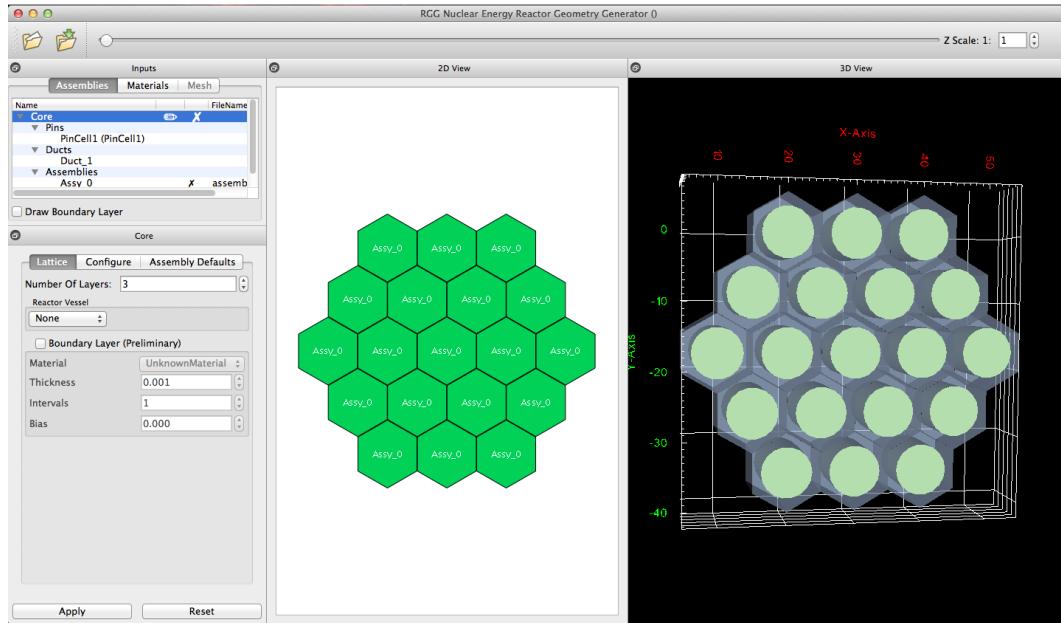


Figure 5.13: Final view.

Congratulations! You’ve made a hexagonal core!

Chapter 6

Meshing

6.1 File Formats

RGG uses its own XML based file format, RXF. It allows to maintain states not stored in Meshkit's INP file format. However, RGG can import INP files through **File → Import → INP File**. RGG can also export INP files by **File → Export → Inp Files**. At the time of mesh generation, the user will be asked for a directory if the INP files have not been exported yet.

6.2 What Makes A Good Mesh

In general, there are four different parameters that drive the meshing process.

We can specify the mesh type – the simple geometric differential we'd like to use to model the core with, like tetrahedrals (triangular pyramids) and hexahedrons (rectangular prisms).



Did you know?



All other parameters equal, it always takes more tetrahedrons to mesh a model than hexahedrons. While this means that it has the potential to be better for analysis than a hexahedral mesh, it also means that it takes more computer resources – like memory or disk space – to store or manipulate a tetrahedral mesh.

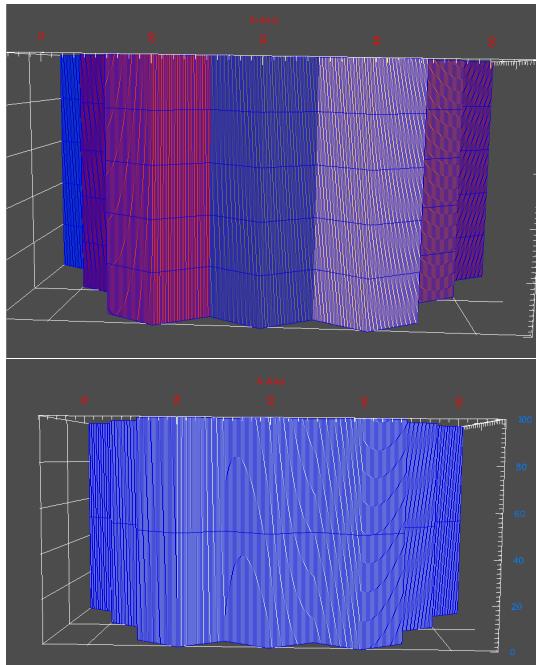


Figure 6.1: Example of a 100 unit tall core with axial mesh size set to 20 and 50 units.

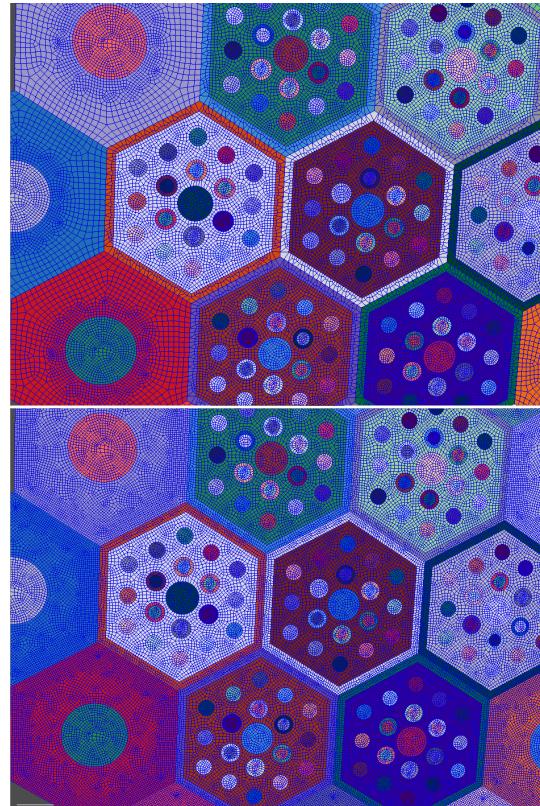


Figure 6.2: Example of a hexagonal core meshes with the edge interval changed. The top one uses an edge interval of 20 units and the lower one uses an edge interval of 50 units.

Additionally, we can control how many layers tall we want the mesh to be. We specify this by setting the axial mesh size, which is the unit length of each mesh layer in the vertical direction. For example, a core that is 100 units high with an axial mesh size of 20 would produce 5 layers down the side of the mesh. If you were to change the axial mesh size to 50, two layers would be seen down the side. As with mesh type, a core must have the same axial mesh size in order to be conforming (Figure 6.1).

We also can specify how many edges we'd like along an edge of a hexagonal or rectilinear pin. We specify this by declaring the edge interval, or the number of mesh edges we'd like per model edge (Figure 6.2).

Lastly, we can control how coarse or fine the mesh is by specifying the radial mesh size, or the length of the square or triangle used in the top-down projection of the 3D mesh (Figure 6.3). In practice, it does not make sense to choose a radial mesh size larger than the size of the smallest feature you would like to preserve in an assembly, as that would mean that that is not captured in the resulting mesh.

Not all meshes are created equal. Meshes used for analysis must be conforming; that is,

- vertices of mesh polygons or polyhedra must only touch other vertices,
- edges must only touch other edges,
- and faces must only contact other mesh faces.

That means that we must ensure that the mesh type, axial mesh size, and edge interval all match in the core. If they are different in each assembly, the resulting mesh may not always be conforming. RGG ensures that they match by only letting you specify them at the core level, in the **assembly defaults tab**.



Figure 6.3: These two assemblies have two different radial mesh sizes. The one on the left has a radial mesh size of 0.3, while the one on the right is 0.1.

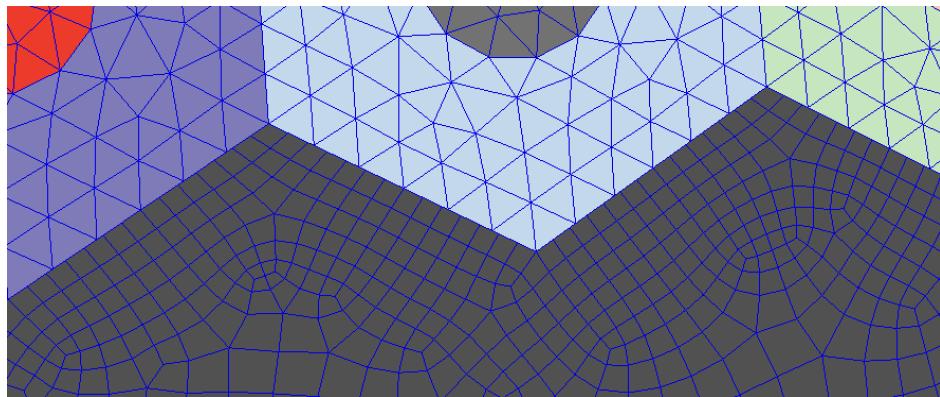


Figure 6.4: A non-conforming mesh. Note how vertices touch edges and edges touch vertices along the border between the blue and the hexes.

However, the radial mesh size is not required to be the same for each assembly. Refer to Figure 6.3; even though two different assemblies side by side have different radial mesh sizes, it is still conforming. You can change the radial mesh size and specify a different level of coarseness or fineness on a per-assembly basis without suffering bad consequences. RGG gives you this ability by making radial mesh size accessible from an assembly's **configure tab**. RGG ships AssyGen and CoreGen from Meshkit 1.2¹. RGG allows you to

¹See <http://trac.mcs.anl.gov/projects/fathom/wiki/MeshKit>

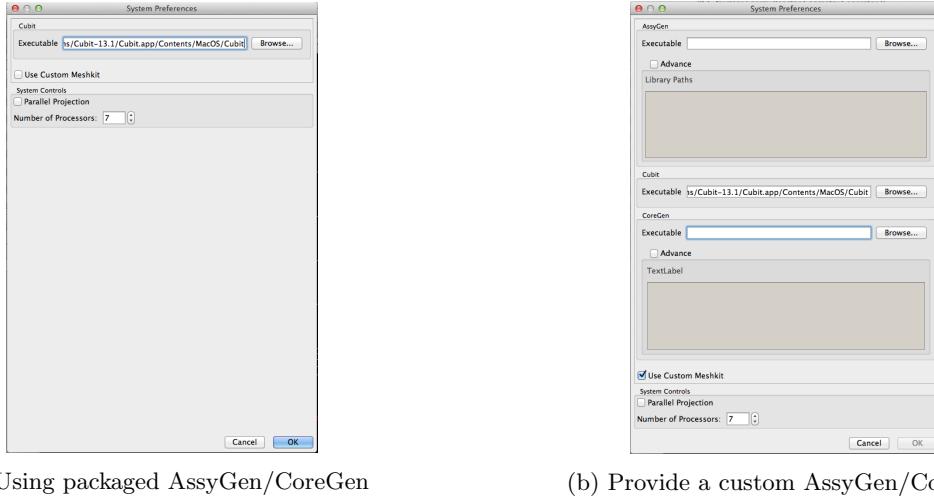


Figure 6.5: Configuration of meshing executable.

provide paths to Cubit².

6.3 RGG Preferences

In order for RGG to mesh an INP file, we utilize AssyGen, CoreGen, and Cubit. On Linux and Mac, we package Assygen and CoreGen with RGG. However, RGG needs an external Cubit for meshing. RGG will look for Cubit 13.1 in the normal location. However, we provide a means to manually set Cubit and also point to a different AssyGen and CoreGen than what is shipped.

Access the `system preferences` window by clicking on the `edit menu` of the `toolbar` on Linux and Windows. On Mac, click on the `preferences item` of the `RGG Nuclear menu`.

This brings up the `system preferences window`. By default, we use the packaged AssyGen and CoreGen, so we only provide a section to set Cubit (Figure 6.5a). By checking “Use Custom Meshkit,” one is provided sections to direct RGG toward the desired AssyGen/CoreGen (Figure 6.5b). To add one, click the `browse button` in the section that corresponds to the desired executable. Locate the executable and click the `open button` to get back to the system preferences window.

Note the `advance checkbox` in the AssyGen and CoreGen sections. If either of these executables require shared libraries in non-standard locations, you can click this checkbox to activate the `library paths text box`. You can specify directories where the shared libraries are, with each line being another location.

We also run multiple AssyGen and Cubits in parallel decrease the wait time for a mesh. The number of parallel jobs is controlled by “Number of Processors.”

6.4 Generating a Mesh

Now that the system preferences have been adjusted to include AssyGen, Cubit, and CoreGen, you can generate a mesh though the `Run MeshKit RGG dialog` accessed through the `tools menu` (Figure 6.6). Note the “x” icon in the third column

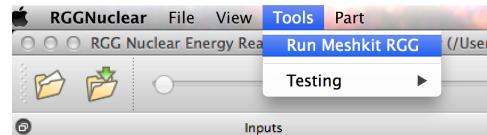


Figure 6.6: Opening the Run MeshKit RGG dialog. The x icons denote that the FF and C1 assemblies need to be meshed.

²See <https://cubit.sandia.gov/>.

of the assemblies and core in the `inputs panel`. This designates that RGG believes that these components need to be meshed.

You will be prompted for an output directory if one has not been set.

The dialog should be populated with the assemblies that need to be re-meshed automatically (Figure 6.7). In the example below, two assemblies need to be meshed, which also means that the core must be remeshed. If “Keep Going on Error” is checked, RGG will process every possible Assembly Files after an error occurs. Any job that depends on the erred job will not be run. If it is not checked, RGG will quite processing after the first error. Checking “Keep Going on Error” is advised if there are a large number of files to be processed.

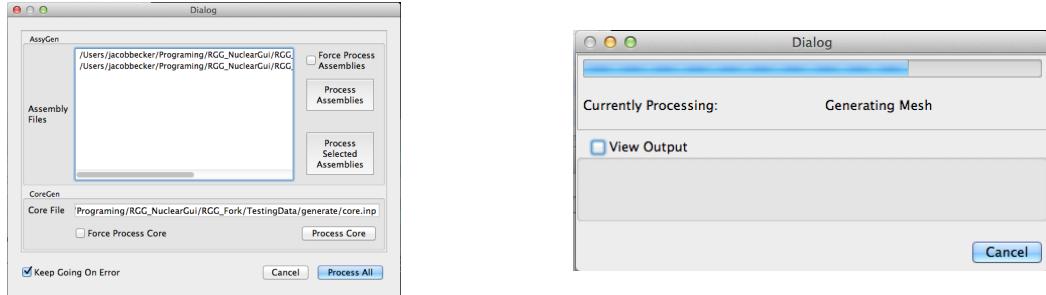


Figure 6.7: The Run MeshKit RGG dialog.

One can force the generation of meshes regardless of their current state by using the `force process assemblies` checkbox and the `force process core` checkbox for an assembly or core, respectively. To mesh both the core and assemblies, check both boxes and use the `process all` button. This is useful in cases where the exporting executables have changed. A window that gives you the status of the processing should come up, similar to the shown in Figure 6.8:

You can see the output of AssyGen, CoreGen, and Cubit by clicking on the `view output` checkbox. To cancel the current export, click the `cancel` button.

To verify that the assemblies and core have been meshed, confirm that the `x` icons seen previously are now `green square` icons, as shown in Figure 6.9.

Note that the mesh needs to be loaded in after MeshKit creates it. During this process, the tools menu is inaccessible.

6.5 Displaying the Mesh

Upon successful creation of a mesh, you either need to load it in manually via the `open MOAB file dialog` accessed from the `open MOAB file item` in the `file menu`, or, if you've just meshed an INP file you had loaded in previously, the mesh will load in automatically.

6.5.1 Views of the Mesh

To control viewing the mesh, be sure to click on the `mesh tab` of the `inputs panel`. RGG allows you to six different options to control the `Mesh view` of the mesh inside a drop down box (Figure 6.11), listed below:

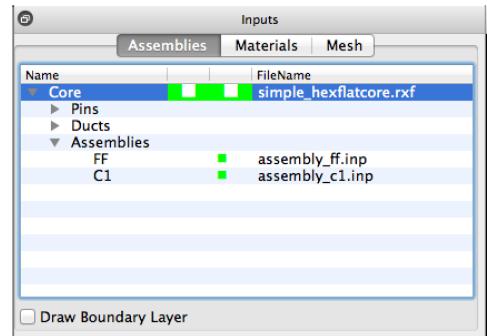


Figure 6.9: Verifying successful meshing.

- Volumes
- Boundary
- Surfaces
- Neumann Sets
- Dirichlet Sets
- Material Sets

More detail on these views is provided below. When available, we allow you to select a subsection for the selected option. When selected, only that part is displayed in the Mesh View. Also, using the checkboxes, one can hide a subsection (Figure 6.12). Additionally, you can check the `show edges` checkbox to view the mesh superimposed on the Mesh View and check the `color` checkbox to colorize the different pieces of the view based on the option you've selected.

Volumes

Clicking the `volumes` option shows all the volumes of the mesh. Checking the color checkbox will colorize all of the distinct volumes in different colors.

Boundary

When the `Boundary` option is selected, RGG displays only the boundary conditions. Checking the color checkbox will colorize all of the distinct volumes in different colors.

Surfaces

When the `surfaces` option is selected, RGG displays only the surfaces of the mesh.

Neumann Sets

Clicking the `Neumann Sets` option will show the natural boundary conditions on sides of domains.

Dirichlet Sets

Clicking the `Dirichlet Sets` option will show the essential boundary conditions on points of domains.

Material Sets

The `material sets` option displays all the volumes of the mesh, but colorizes them on a per material basis. Additionally, you can toggle the visibility of materials by checking and unchecking them in the `materials` tab of the inputs panel.

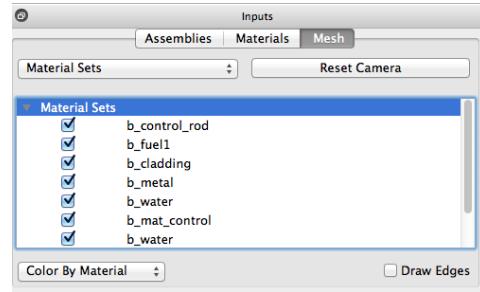


Figure 6.10: Used to control the mesh

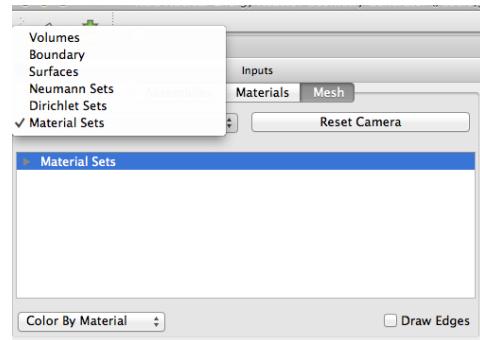


Figure 6.11: The Views

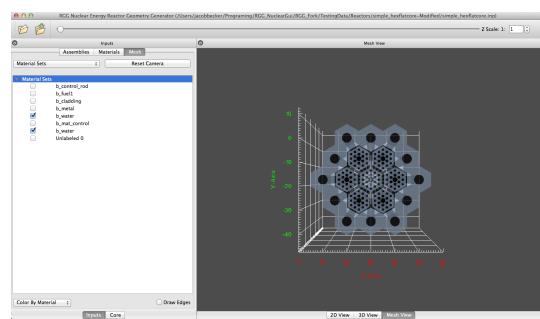


Figure 6.12: All but water subsection are hidden

Main Index

Assemblies, 10
axial mesh size, 26

conforming meshes, 27
Cores, 10

Ducts, 10

edge interval, 27

Linux Installation, 8

Mac Installation, 8

Materials, 10, 12

mesh size

 axial, 26

 radial, 27

mesh type, 26

Meshes, 10

meshes

 conforming, 27

Pins, 10

radial mesh size, 27

SameAsAssembly, 10

Interface Index

- 2D view, 11, 13, 14
3D view, 11–14
- advance checkbox, 28
assemblies tab, 11, 12
Assembly Defaults, 16
assembly defaults tab, 12, 13, 27
- Boundary Layer, 13
Boundary option, 30
browse button, 28
- cancel button, 29
color checkbox, 30
configure tab, 12, 27
Cutaway view, 17
- Dirichlet Sets option, 30
Duct, 16
duct, 13
duct tab, 12, 13
- edit menu, 28
Export, 26
- File, 26
file menu, 29
Fill Ring With, 14, 18
force process assemblies checkbox, 29
force process core checkbox, 29
- green box icon, 11
green box icons, 12
green square icon, 29
- Import, 26
INP, 26
INP File, 26
Inp Files, 26
input panel, 11
Inputs Panel, 16
inputs panel, 29
- lattice tab, 12, 13
library paths text box, 28
- main window, 11
material sets, 30
materials tab, 11, 12, 30
menu, 11
mesh tab, 11, 12, 29
Mesh view, 11, 12, 14, 29
- Neumann Sets option, 30
open button, 28
open MOAB file dialog, 29
open MOAB file item, 29
- pencil icon, 11, 12
pin tab, 12, 13
preferences item, 28
process all button, 29
properties panel, 11, 12, 14
- Reactor Vessel, 13
Replace All With, 14, 23
RGG Nuclear menu, 28
Run MeshKit RGG dialog, 28
RXF, 26
- show edges checkbox, 30
surfaces option, 30
system preferences window, 28
- toolbar, 11, 28
tools menu, 28
- view output checkbox, 29
volumes option, 30
- Windows Installer, 8
- x icon, 12, 28, 29