# Mesh Generation – RectGenerator

This is a simple mesh generation tool for creating a user defined mesh compatible with the PHASES software. With this tool the user can define a series of connected rectangles from left to right. These rectangles can then define the boundary of a mesh to be generated.

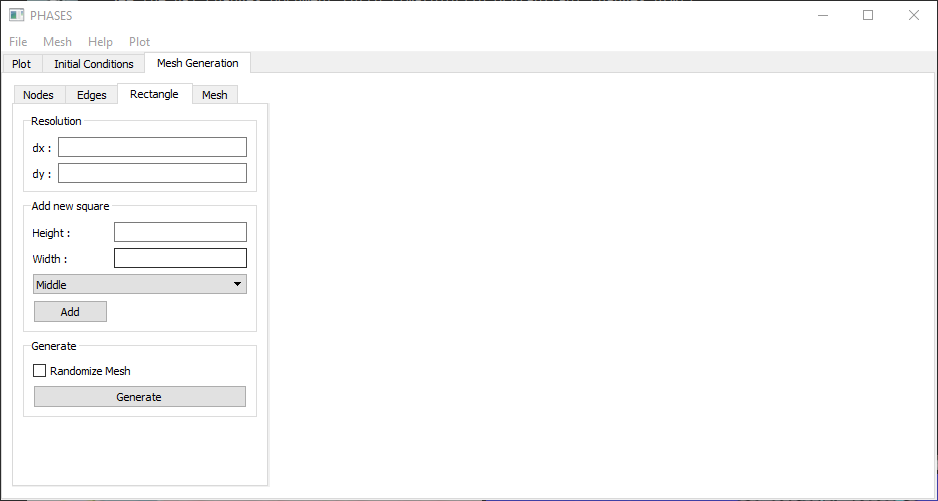


Figure 1-RectGenerator Widget

This mesh generator creates a mesh where each mesh element is rectangular and equal in size. The first step a user takes is to define the dx and dy variables which define the width and the height of each element respectively. Then rectangles are added one by one. Each new rectangle is added to the right of the previous rectangle, and can be aligned in three different ways. ‘Middle’ alignment will align the rectangles along the vertical center line. ‘Top’ alignment will align the top edges of the rectangles. ‘Bottom’ alignment will align the bottom edges. This system should be capable of producing a variety of mesh boundaries, as long as they involve right angles. Some images showing the different options in the program are shown on the following pages. Once the mesh is generated the boundary conditions and initial conditions can be defined in the mesh editor. Simulation parameters also need to be defined in the panel on the left side of the window.

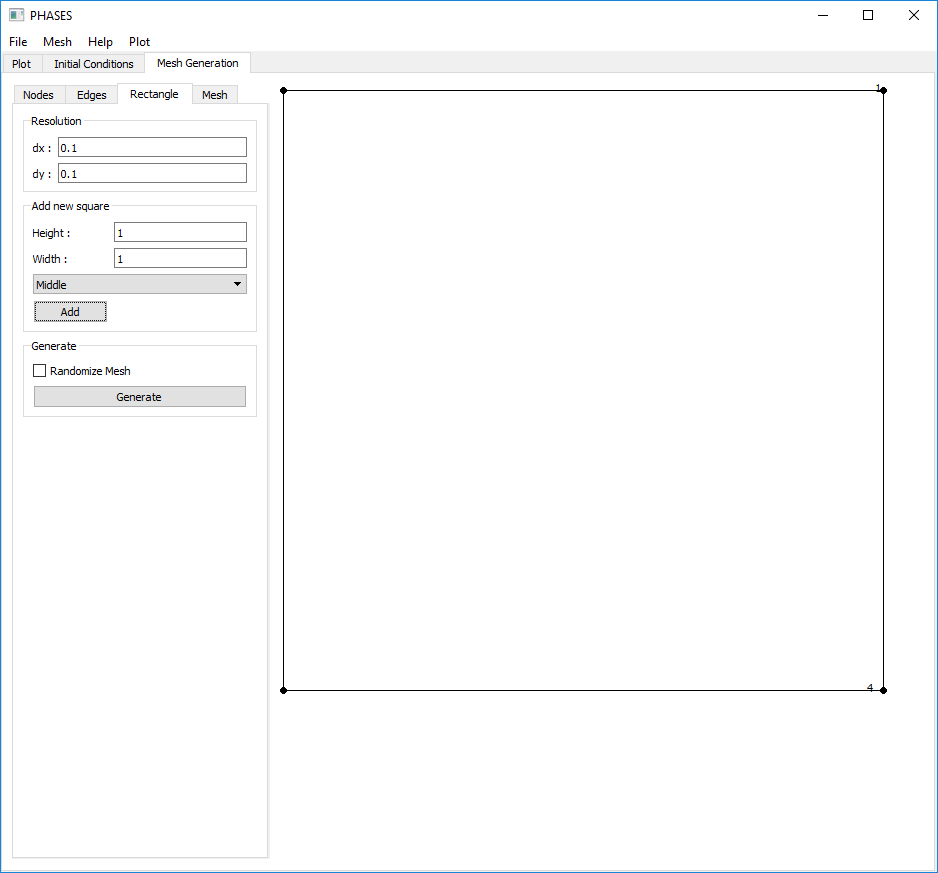


Figure 2 - First Rectangle Added

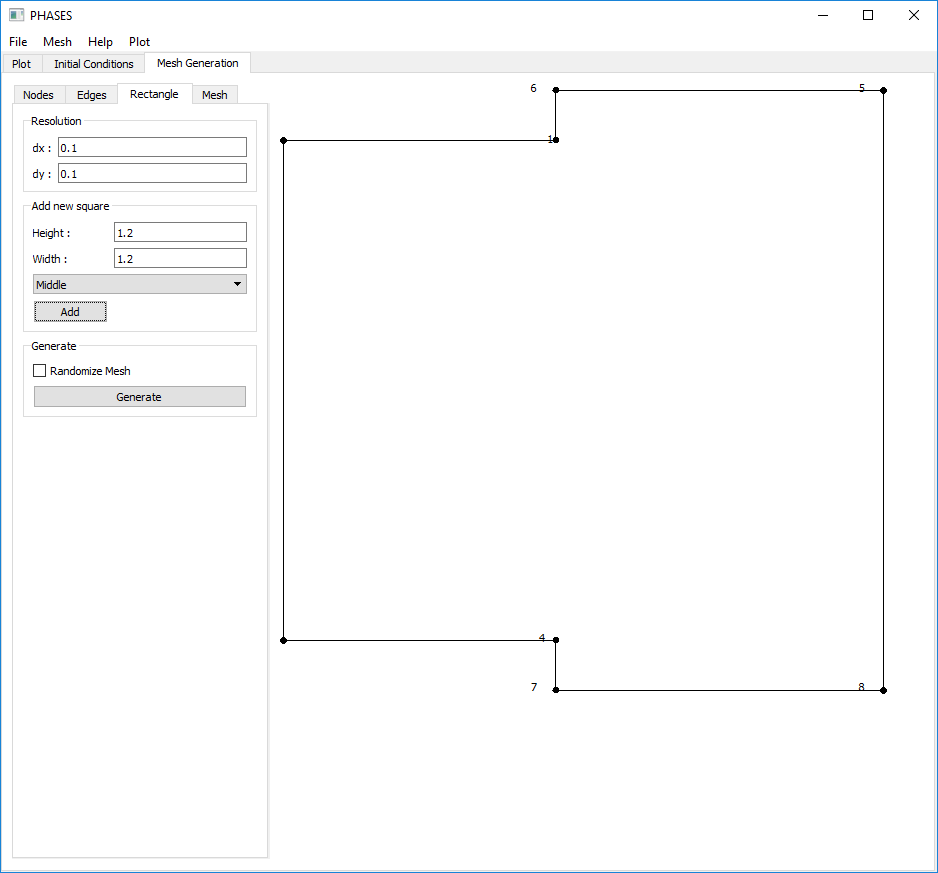


Figure 3 - Rectangle with Middle Alignment Added

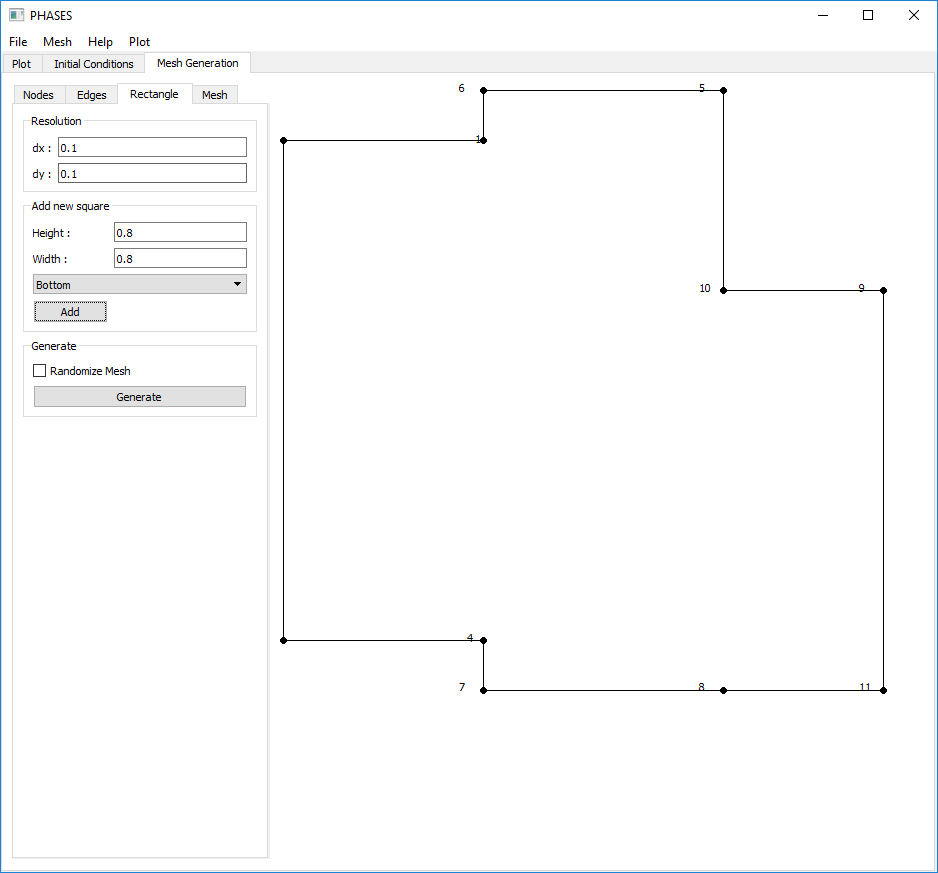


Figure 4 - Rectangle with Bottom Alignment Added

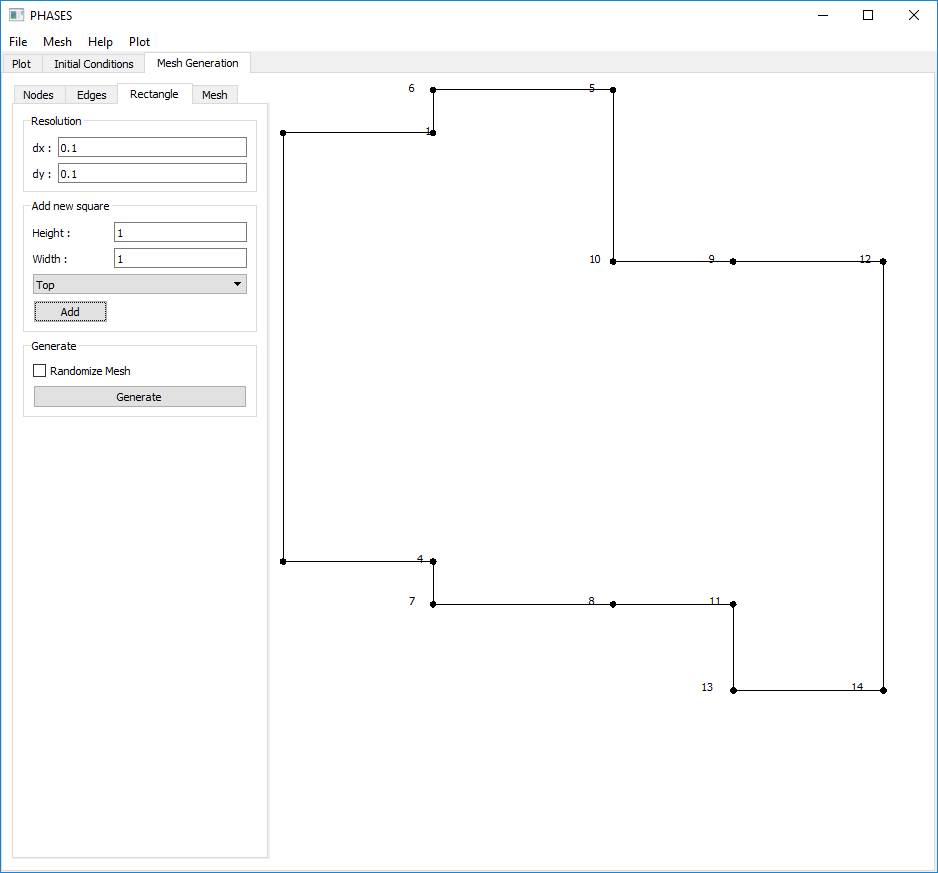


Figure 5 - Rectangle with Top Alignment added

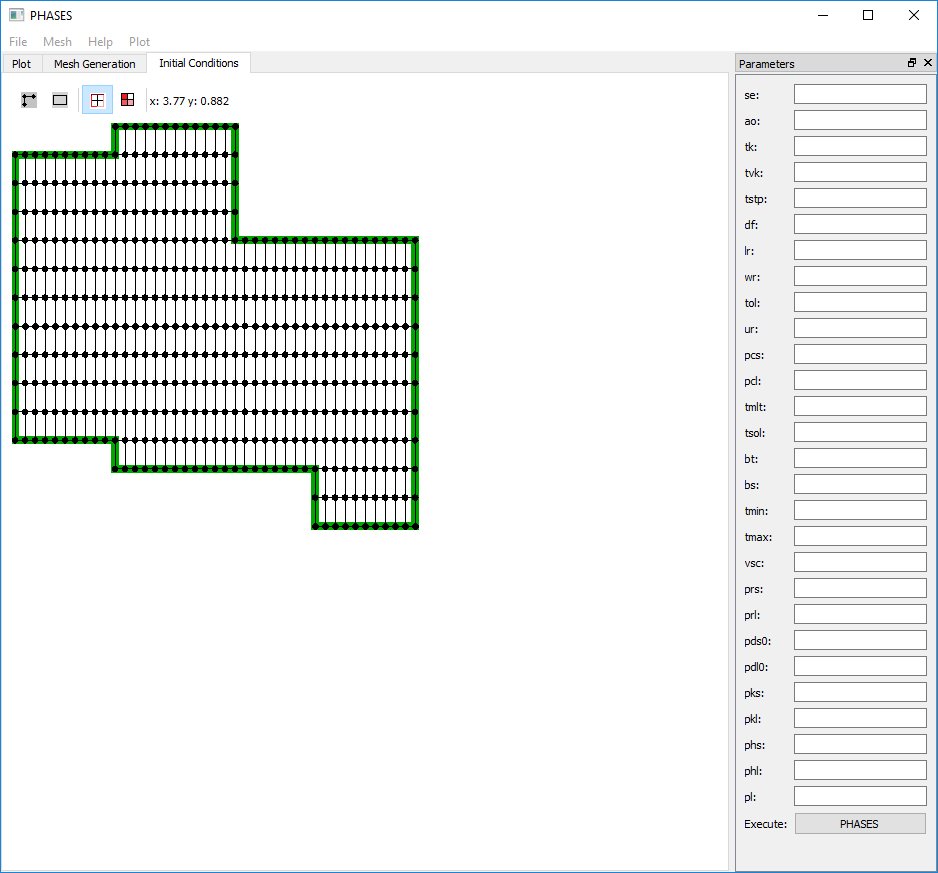


Figure 6 - Generated Mesh

# Randomized Meshes

The mesh generator includes an option to randomize the generated mesh. This can be used to generate a mesh which has elements consisting of irregular quadrilateral shapes. This is done by using the rectangular mesh, and randomly changing the position of each node that isn’t on the boundary of the mesh. The node will randomly be shifted up or down by a value less then a quarter of the element height and left or right by a value less than a quarter of the element width.

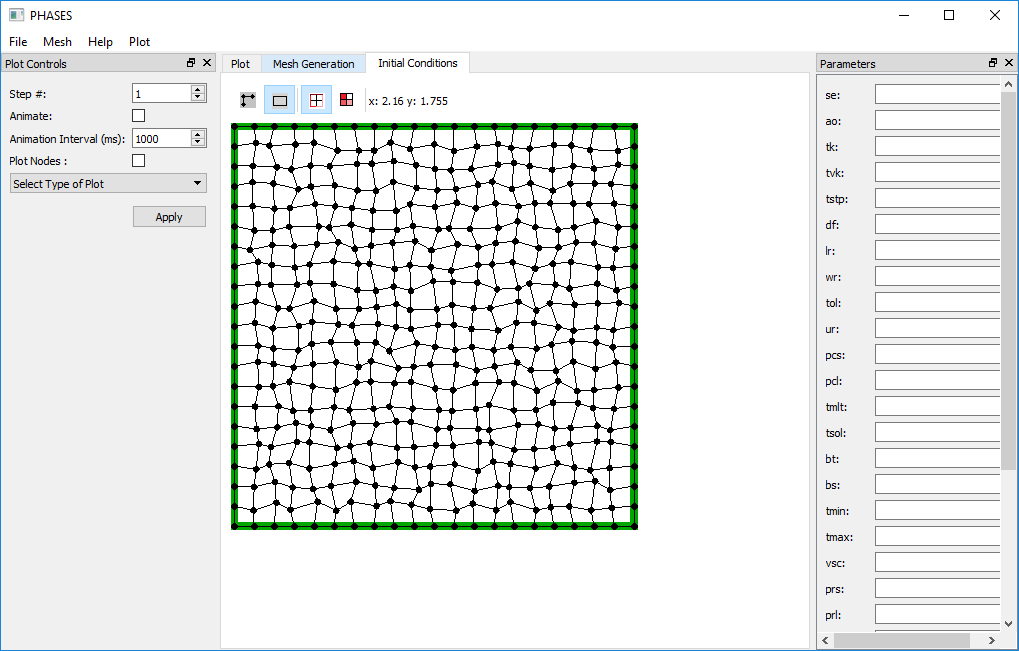


Figure 7 - Example of Randomized Mesh

# Calculating Entropy

Entropy calculation was added to the PHASES simulation. At the end of each simulation, the entropy will be calculated for each node at each timestep. The entropy is calculated as:

The partial derivates are calculated using the gradient function from the numpy library. This method of calculating partial derivatives is only valid when the horizontal distance is constant and the vertical distance is constant between adjacent nodes in the mesh. Therefore this calculation will not be accurate for randomized meshes and a more complicated method will need to be used.

# Other Bug Fixes

* The original version had various issues in the order that different boundaries were sent the PHASES algorithm. For each element the nodes must be sent in an anti-clockwise order. This seems to also apply to the boundaries when an element has two or three boundaries. Top boundaries must be prioritized followed by left, bottom then right boundaries. Previously it was though the orientation did not matter, so boundaries orientation was not labeled. Now the boundary orientation is labeled when a mesh is generated or laded from a file.
* Certain values, such as the u-velocity reference, were calculated upon loading a file. The calculations used other parameters which had been saved to the project files. This lead to different results depending on if a mesh was generated or if a mesh was loaded from a file. These calculations now occur within the PHASES simulation function, so that the origin of a mesh should not affect how simulation parameters are entered.
* Initial conditions for temperature were previously defined on a scale of 0 to 1 based on the *tmin* and *tmax*. This has been changed so that the temperature can be defined as its actual value (in kelvins) to be more user-friendly.

## Mesh.Generator.RectGenerator.py

### Class RectGenerator

#### This class inherits from the Qt class QWidget. It is a widget that can be used to create simple user-defined meshes.[Rectangles](#_Class_Rectangle) **can be added one by one by the user to define a mesh boundary**. **A mesh will then be generated using evenly spaced nodes with a rectangle mesh. These node positions can also be randomized slightly to create elements with irregular quadrilateral shapes.** A Mesh will be emitted after the Generate Button has been pressed**.**

#### Signals

##### meshCreated(Mesh.Mesh)

Emitted whenever an edge is edited, and contains the edge the was edited.

#### Methods

##### addNewRectangle(height, width)

Adds a new [Rectangle](#_Class_Rectangle) to the mesh boundary definition with the specified height and width. The alignment that has been selected is also taken into account.

##### addRectangle (refX, refY, height, height)

Adds a [rectangle](#_Class_Rectangle) with the specified height and width. refX and refY define the location of the bottom left corner.

##### deleteRectEdge(rect,edge)

Deletes the specified edge from rect.

##### generateNodes()

Main mesh generation algorithm. Splits the region enclosed by the boundary into equally sized rectangles. Generates and organizes all nodes, elements and boundaries. Randomizes mesh if the Randomize Mesh checkbox is selected. Emits the meshCreated signal.

### Class Rectangle

Definition of the rectangles used to define a mesh boundary.

#### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Documentation** |
| nodes | list<Mesh.Node> | The four nodes that define the corners of the rectangle | [Class Node](#_Class_Node_(x,y)) |
| Height | Double | Height of the Rectangle | N/A |
| Width | Double | Width of the rectangle | N/A |
| Edges | List<StraightLine> | StraightLines Defining the 4 edges of the rectangle | [Class StraightLine](#_Class_StraightLine) |
| topEdges | List<StraightLine> | Top Edges of the rectangle | [Class StraightLine](#_Class_StraightLine) |
| leftEdges | List<StraightLine> | Left Edges of the rectangle | [Class StraightLine](#_Class_StraightLine) |
| botEdges | List<StraightLine> | Bottom Edges of the rectangle | [Class StraightLine](#_Class_StraightLine) |
| rightEdges | List<StraightLine> | Right Edges of the rectangle | [Class StraightLine](#_Class_StraightLine) |

#### Methods

##### removeEdge(edge)

Removes the specified edge from the Rectangle

##### delete()

Deletes the Rectangle.