Structure Generator and Mesher

Informative session

Saurabh Sant, Dr. sc. ETH

saurabh64sant@gmail.com

SemiVi LLC

Zollikon, Switzerland.

May 10, 2025

Overview

Table of Contents

Salient features - structure generator

The structure generator reads in the size and shape of the device from the *config* file.
Salient features of the structure generator are –

- Creates a 2D or 3D device composed of many (possibly intersecting) *regions* of shapes –
  - Axis-aligned rectangle or cuboid,
  - 2D Polygon or 3D polyhedron from vertices,
  - Special shapes – 2D: ellipse — 3D: ellipsoid, cylinder, cone.
- Performs the required shape modifications –
  - 2D or 3D Boolean operations – add, clip, subtract shapes
  - 2D or 3D Shape transformations – translate, rotate, scale, mirror shapes
  - 2D or 3D Shape rounding, chamfering, offsetting, stretching
  - 2D Shape sweep to create prisms, pyramids, pipes, revolutions
- Doping definitions of analytic profiles such as Gaussian, exponential, linear, or user-defined can be set.
- Specifies mesh refinements globally or regions-wise or in specific areas.
- Python interface enables convenient scripting and integration.
- Import *GDS* layout file to create masks for doping, etching, or deposition.

Salient features - mesher

The mesher tool creates the specified type of mesh using the mesh refinements set by the user in the config file. Following mesh are supported.

- FEM-mesh: A triangular (2D devices) or tetrahedral (3D devices) finite-elements based mesh. It is used for electrical drift-diffusion transport simulations, and for dopant diffusion process simulations.
- Tensor-mesh: A rectangular (2D devices) or cubic (3D devices) grid based mesh. It is used for optical FDTD simulations, BPM simulations. It is also used for electrical QT simulations with NEGF formalism.
- *Triangle*[1] mesh: External 2D mesher,
- *TQMesh*[2] mesh: External 2D mesher,
- *TetGen*[3] mesh: External 3D mesher,
- *CGAL*[4] mesh: External 3D mesher.

[1] Jonathan Richard Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator* . Lecture Notes in Computer Science, pages 203-222, Springer-Verlag, Berlin, May 1996

[2] Jonathan Richard Shewchuk, *TQMesh*

[3] Si, Hang (2015): *TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator*, ACM Transactions on Mathematical Software 41, S. 1–36.

[4] P. Alliez, *et al*: *CGAL User and Reference Manual*, CGAL Editorial Board, 5.6.1

Table of Contents

Config file for structure generation

```
Device: {
  Name = "Diode";
  MeshType = "FEM";
  Simulation = "DD";
}

RefWin*RefSi: {
  Position: ([-1.0, -0.2, 0.], [1., 0.2, 0.]);
  Shape = "Rectangle";
}

RefWin*RefCathode: {
  Position: ([0.999, -0.25, 0.], [1.001, 0.25, 0.]);
  Shape = "Rectangle";
}

RefWin*RefPDoping: {
  Position: ([-1.001, -0.25, 0.], [0., 0.25, 0.]);
  Shape = "Rectangle";
}

RefWin*RefNDoping: {
  Position: ([0.0, -0.25, 0.], [1.001, 0.25, 0.]);
  Shape = "Rectangle";
}

Region*RegSi: {
  RefWin = ["RefSi"];
  Material = "Silicon";
  //MaxEdgeLength = 0.05;
}
```

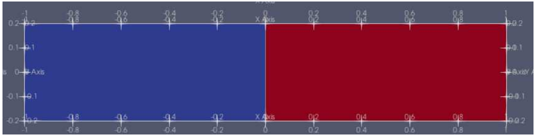


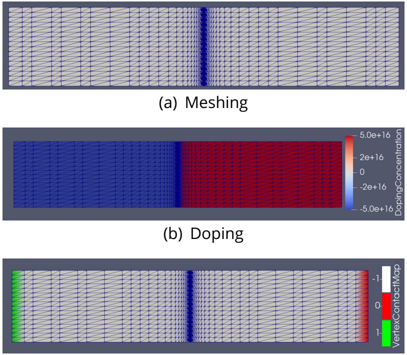
Figure: Device structure

- `Device`: selects the mesh-type and intent of simulation.
- `RefWin`: Shapes which can be used for a variety of purposes, by linking them to `Region`, `Contact`, `Doping`, etc.
- `Region`: Set `RefWin` list and the `Material`.
- `Material`: New material defined by adding a `Material.cfg` file in the working directory.

Config file for structure generation

```
MeshDef*MDefSiTop: {
  RefWin = "RefAll";
  X: {cen = 0., minspacing = 0.0001,
      maxspacing = 0.05, incr = 1.3 }
  Y: {cen = 0., minspacing = 0.025,
      maxspacing = 0.025, incr = 1. }
}

DopingDef*Pdoping: {
  RefWin = "RefPDoping";
  Type =  "Constant";
  Concentration = 5E16;
  Dopant = "Boron";
}

...

Contact*Cathode: {
  RefWin = ["RefCathode"]
}

Contact*Anode: {
  RefWin = ["RefAnode"]
}
```

- `MeshDef`: Meshing definition followed in the specified `RefWin`.
  - `cen`: center of the refined mesh.
  - `minspacing`: minimum spacing at the `cen`.
  - `maxspacing`: maximum spacing in the `RefWin`.
  - `incr`: rate of increment of mesh spacing.



(a) Meshing

(b) Doping

(c) Contacts

Figure: Mesh, doping, and contacts.

- `DopingDef`: Doping of the vertices in the `RefWin`.
- `Contact`: Vertices in the specified `RefWin` are set as contact vertices.

Set the mesh-type in `Device` as follows –

```
Device: {
  MeshType = "FEM"; // "Tensor" or "TetMesh"
}
```

Internal mesh engines –

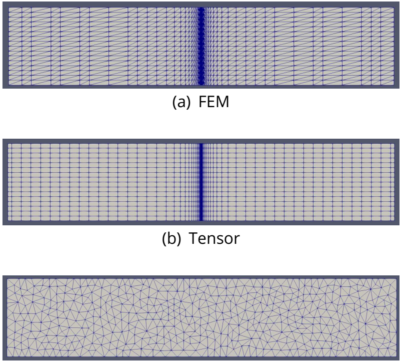
- FEM-mesh: A triangular (2D devices) or tetrahedral (3D devices) finite-elements based mesh.
- Tensor-mesh: A rectangular (2D devices) or cubic (3D devices) grid based mesh.

Note: Internal mesh engines read `MeshDef` section for mesh-spacing specification.

External mesh engines –

- *Triangle* mesh: An external mesh engine which meshes 2D device with triangular finite-elements.
- *TetGen* mesh: An external mesh engine which meshes 3D device with tetrahedral finite-elements.

Note: Mesh-spacing in the given region is set by `MaxEdgeLength` in `Region` section.



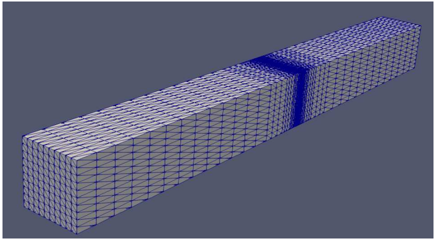
(a) FEM

(b) Tensor

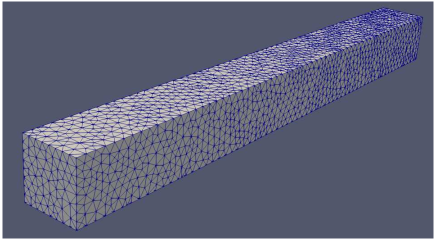
(c) TetMesh

Figure: Mesh of a 2D diode generated by running the diode config file with MeshType of (a) FEM, (b) Tensor, (c) TetMesh.

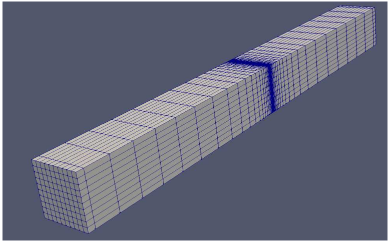
3D Mesh Types



(a) FEM



(b) TetMesh



(c) Tensor

Table of Contents

Python interface for structure generation

```
import device as dev
import numpy as np

# define device object
dio = dev.Device ("Diode", "FEM", "STR", 1)

xmin = -1.; xmax = 1.; xmid = 0.; ymin = -0.2; ymax = 0.2
# define RefWins
dio.setRefWin (Name="RefSi", Shape="Rectangle",
    Position=np.array ([[xmin, ymin, 0.], [xmax, ymax, 0.]]))

dio.setRefWin (Name="RefAnode", Shape="Rectangle",
    Position=np.array ([[xmax-1E-3, ymin-1E-3, 0.],
                        [xmax+1E-3, ymax+1E-3, 0.]]))

dio.setRefWin (Name="RefPDoping", Shape="Rectangle",
    Position=np.array ([[xmin, ymin, 0.], [xmid, ymax, 0.]]))

# define Regions
dio.setRegion (Name="RegSi", RefWin="RefSi", Material="Silicon")

# define Meshing
dio.setMeshDef (Name="MDefSi", RefWin="RefSi",
    Xarg=[0., 1E-4, 0.05, 1.3], Yarg=[0., 0.025, 0.025, 1.])

# define Doping
dio.setDopingDef (Name="Pdoping", RefWin="RefPDoping",
    Type="Constant", Concentration=5E16, DopantSpecies ="Boron")

# define Contacts
dio.setContact (Name="Anode", RefWin="RefAnode")

dio.createDeviceMesh () # create Mesh

dio.saveMeshData () # save Mesh
```

Argument names in *config* file and *python script* are mostly similar. Please check the manual first!

Compare `DopingDef` defined in *config* file vs. in *python* –

```
DopingDef*Pdoping: {
  RefWin = "RefPDoping";
  Type =  "Constant";
  Concentration = 5E16;
  Dopant = "Boron";
}
```

Run the *duly completed* python script as follows –

```
>> python3 diode_str.py
```

- *Equivalency*: Python interface internally runs the software commands.
- *Parametrization*: Script can be reused for a similar structure.
- *seamless integration*: `device` passed as argument to the simulator objects.
- Python libraries (e.g. scikit or optimization libs) can be used in the script.

Table of Contents

Editing RefWins – part 1

Define `RefWin` objects in python as follows.

```
import mesher as m

win1 = tm.refwin("RefCyl1","Polygon",poly,{},{})

win2 = tm.refwin("RefCyl2","Polygon",poly2,{},{})
```

Boolean operations on 2D/3D RefWins –

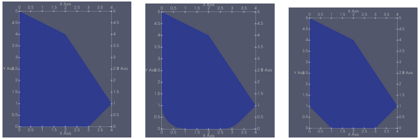
- `win1.add(win2)` : boolean unite
- `win1.subtract(win2)` : boolean subtract
- `win1.clip(win2)` : boolean intersection

Geometric transformations on 2D/3D RefWins –

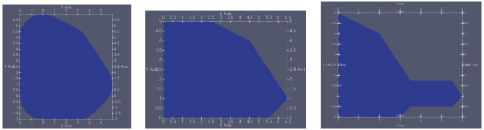
- `win1.translateBy`($\vec{d}$) and $\vec{d}$ = `tm.position(0,0,0)`
- `win1.scaleBy(0.5,`$\vec{c}$)
- `win1.rotateAroundAxis`($\vec{c}$,$\vec{d}$,45°) : point $\vec{c}$ and direction $\vec{d}$ define the axis.
- `win1.mirrorByPlane`($\vec{d}$) : $\vec{d}$ defines plane normal

Editing 2D/3D RefWins –

- `win1.chamferAt([tm.position(0,0,0)],[1])`
- `win1.roundAt([tm.position(0,0,0)],[0.5])`



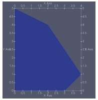
(a) shape of "win1" (b) rounding (c) chamfer



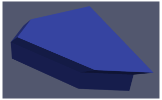
(d) offset (e) stretch (f) stretch at a point

Editing 2D/3D RefWins –

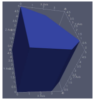
- `win1.offsetRefWin(2)`
- `win1.stretchRefWin(tm.position(2.5,0,0))`
- `win1.stretchRefWinAtPoint`($\vec{p}$,$\vec{d}$) : stretch a point $\vec{p}$ along direction $\vec{d}$
- `win1.limitRefWinSpan`($\vec{c}_m$,$\vec{c}_p$)



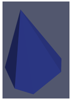
(g) shape of "win1"



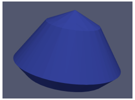
(h) sweep along a path



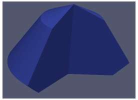
(i) make prism



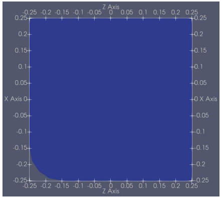
(j) make pyramid



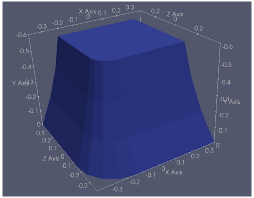
(k) revolve 360°



(l) revolve 230°



(m) Rounded etch mask



(n) Custom etch shape

Transforming 2D RefWin to 3D –

- `win.sweep2DRefWin` ($[\vec{c}_1,\vec{c}_2,\vec{c}_3]$) : sweep along a piecewise linear path defined by $[\vec{c}_1,\vec{c}_2,\vec{c}_3]$.
- `win1.makePrism2DRefWin` ($\vec{d}$) : sweep along $\vec{d}$
- `win1.makePyramid2DRefWin` ($\vec{c}$) : make pyramid with "win1" as base and $\vec{c}$ as apex point.
- `win1.revolve2DRefWin` ($\vec{c}$,$\vec{d}$,360.)

Custom etch profiles can be obtained from the mask RefWins as follows.

- `win1.sweep2DMaskDepthAngle([d1,a1,d2,a2,.])` : etch profile as depth, side-wall angle list.
- `win1.sweep2DMaskPositionList` ($[\vec{c}_1, \vec{c}_2, \vec{c}_3, \vec{c}_4]$) : etch profile as points.

Note: The 2D mask RefWin must be in XZ plane, and the sweep direction is always along y-axis.

Table of Contents

Analytical doping profiles

Masked dopant implantation + diffusion creates a Gaussian distribution and lateral straggle of *error function*.

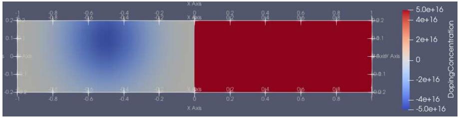
In `Gaussian` profile, doping at each vertex $C_i(\vec{r})$ is given by,

$$C_i(\vec{r}) = C \cdot \operatorname{erf}\left(-(\frac{d_{\|}}{\sigma})^2\right) \cdot \exp\left(-\frac{1}{2}(\frac{d_{\perp} - \mu}{\sigma})^2\right)$$

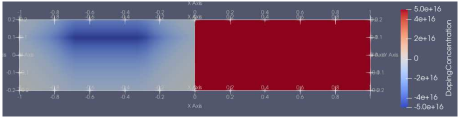
- $d_{\perp}$: perpendicular distance to the `BaseWin`.
- $d_{\|}$: lateral distance from the edges/end-points of the `BaseWin`.
- μ : Implantation depth from Si surface set by `Depth`.
- σ : Diffusion length set by `DecayLength`.
- C : scaling factor set by `Concentration`.

Alternately, `Linear`, `Exponential`, or `Constant` profiles can be set. Following syntax is used for creating a `DopingDef`.

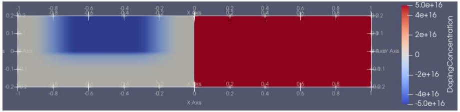
```
DopingDef*Pdoping: {
RefWin = "RefPDoping";
Type =  "Gaussian";
Concentration = 5E16; // 1/cm3
Depth = -0.1;
DecayLength = 0.2;
BaseWin = "PBaseLine";
Dopant = "Boron";
}
```



(o) Gaussian



(p) Exponential



(q) Linear

Figure: Doping profiles obtained by (a) Gaussian, (b) Exponential, and (c) linear distributions.

User-defined analytic doping profiles

Dopant distribution of any user-defined analytic expression is initialized in the `DopingDef` as follows.

```
DopingDef*Pdoping: {
RefWin = "RefPDoping";
Type =  "Analytic";
Concentration = 5E16;
AnalyticExpr = "exp(-(X+0.2)*(X+0.2)/0.02)";
Dopant = "Boron";
}
```

- `AnalyticExpr` : text containing user-defined analytic expression.
- `X`, `Y`, `Z` variables : x-, y-, and z- coordinates of each vertex.
- `Concentration` : scaling factor which internally multiplies the analytic expression.

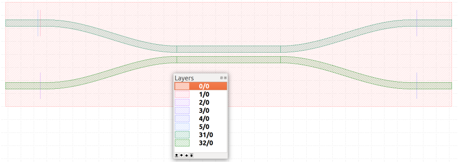
(a) User-defined expression

Figure: Doping profiles obtained by a user-defined expression.

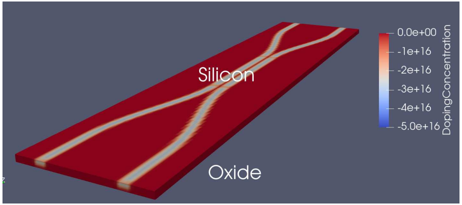
Table of Contents

```
GDSLayerDef*C1: {
GDSFileName = "coupler.gds";
Layer*L1: {Cell="test", Layer=0, Y=0, dY=0.2};
Layer*L2: {Cell="test", Layer=31, Y=0.2, dY=0.0};
Layer*L3: {Cell="test", Layer=32, Y=0.2, dY=0.0};
Layer*L4: {Cell="test", Layer=0, Y=0.2, dY=0.2};
}

DopingDef*Pdoping: {
RefWin = "RefAll";
Type =  "Gaussian";
Concentration = 5E16;
Depth = 0.1;
DecayLength = 0.1;
BaseWinGDS = ["C1_L2", "C1_L3"];
Dopant = "Boron";
}
```

- `GDSLayerDef`: Specifies settings to create polygons from GDS mask `Layers`.
- `Layer`: Each layer creates a `RefWin`.
- `dY`: Thickness of the layer. If 0, it creates a 2D `RefWin`, else, a 3D `RefWin`.
- `RefWinGDS` in `Region`: Creates region from a 3D RefWin.
- `BaseWinGDS` in `Doping`: Sets which 2D RefWins are used as masks.



(a) GDS file of Silicon wave-guide



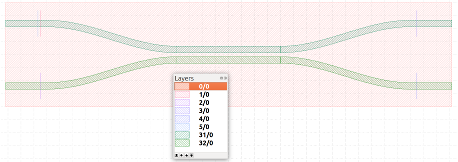
(b) Doping profile from GDS file

Figure: Doping profile in (b) using GDS mask in (a).

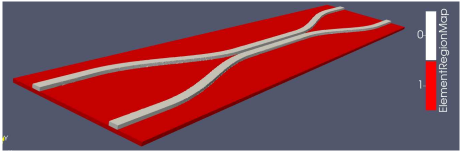
Import GDS layout – region definition (in 3D device only)

```
GDSLayerDef*C1: {
GDSFileName = "coupler.gds";
Layer*L1: {Cell="test", Layer=0, Y=0, dY=0.2};
Layer*L2: {Cell="test", Layer=31, Y=0.2, dY=0.0};
Layer*L3: {Cell="test", Layer=32, Y=0.2, dY=0.0};
Layer*L4: {Cell="test", Layer=0, Y=0.2, dY=0.2};
}

Region*RegSi: {
RefWinGDS = ["C1_L2", "C1_L3"];
Material = "Silicon";
}

Region*RegOxTop: {
RefWinGDS = ["C1_L1"];
Material = "Oxide";
}
```

- `GDSLayerDef`: Specifies settings to create polygons from GDS mask `Layers`.
- `Layer`: Each layer creates a `RefWin`.
- `dY`: Thickness of the layer. If 0, it creates a 2D `RefWin`, else, a 3D `RefWin`.
- `RefWinGDS` in `Region`: Creates region from a 3D RefWin.
- `BaseWinGDS` in `Doping`: Sets which 2D RefWins are used as masks.



(a) GDS file of Silicon wave-guide

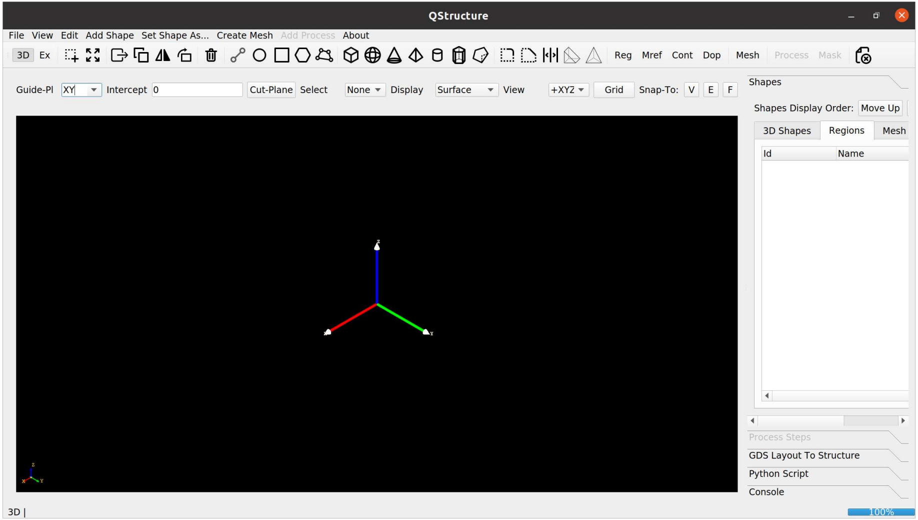


(b) Silicon wave-guide structure

Figure: Silicon wave-guide structure in (b) using GDS mask in (a).

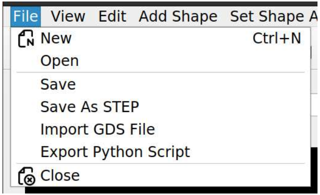
Table of Contents

GUI - Main window



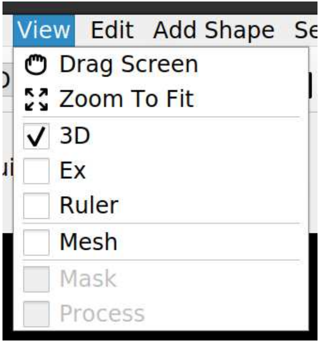
(a) Main window

GUI - File Menu



(b) File menu

- `New` Open new `circuitdraw` window.
- `Open File` Selected '*.qstr' file opened in a new window.
- `Save (As)` Current (New) '*.qstr' file is stored.
- `Save As STEP` Save current structure as a STEP file.
- `Import GDS File` Import a layout file in GDS format.
- `Export Python Script` Export python script which can generate the structure.

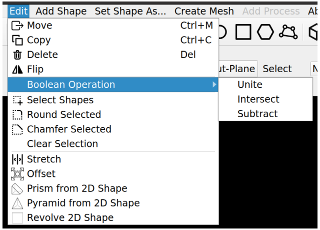


(c) File menu

To change screen viewpoint -

- `Middle Mouse Button` : Drag screen.
- `Right Mouse Button` : Rotate camera view-point.

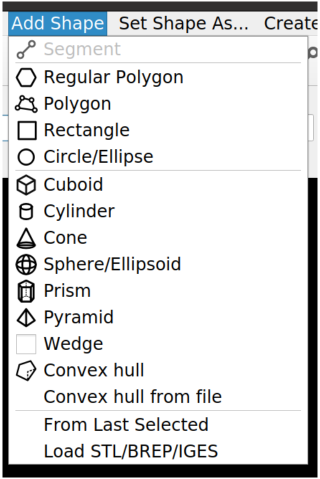
- `Drag screen` : (if middle mouse button is not available) Drag screen.
- `Zoom To Fit` : Click to zoom the structure to fit the entire window.
- `3D` : Set given structure as a 3D structure. It cannot be changed mid-way.
- `Ex` : Activate 'exact coordinates'. When active, any shape drawing action would prompt a dialog box to specify x,y,z coordinates of the point.
- `Ruler` : Activate it. Click start point of the ruler and end point to measure distance.
- `Mesh` : Click to display/hide mesh generated by the mesher.



(d) Edit menu

- `Move`, `Copy`: Activate it. Click on shape to be moved/copied and click where to place it.
- `Delete` : Activate it. Click on the shape to delete.
- `Flip` : Activate it. Click on the shape to be flipped w.r.t. guide-plane at given `intercept`.
- `Select` : Activate it. Click on shapes to be selected.
- `Clear Selection` : Click on it to clear all selected shapes.
- `Boolean operation` : *After selecting the shapes*, click on the operation.
    - `Unite` : Join all selected shapes and store in the first shape.
    - `Intersect` : Find common areas of selected shapes and store in the first shape.
    - `Subtract` : Subtract selected shapes from the first shape.
- `Round Selected`, `Chamfer Selected` : Round/chamfer all corners of selected shapes with the given rounding/chamfer radius (in μm).
- `Stretch` : Activate it. Click on the shape to be stretched at the point where to stretch. Specify stretch distance.
- `Offset` : Activate it. Specify offset distance. Click on the shape to be offset.
- `Prism/Pyramid from 2D Shape` : Activate it. click on the 2D shape and click at the apex point. A pyramid/prism would be created.
- `Revolve 2D shape` : Activate it. Click on the 2D shape. Fix location of axis of revolution.

GUI - Add Shape Menu

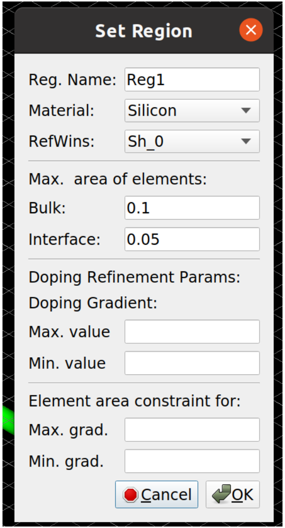


(e) Add Shape menu

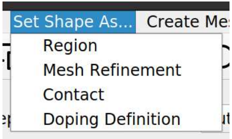
To add a `RefWin` of a given shape, activate its icon by click. Then perform –

- `Segment` : Click on two opposite points of the segment.
- `Regular Polygon`, `Rectangle`, `Ellipse`: Click on two diagonally opposite points of the axis-aligned rectangle to add the respective shape inscribing the rectangle.
- `Polygon` : Click on the consecutive corners of the polygon. Double click on the last corner to add polygon.
- `Cuboid`, `Cylinder`, `Cone`, `Ellipsoid`, `Prism`, `Pyramid`, `Wedge` : Click on two diagonally opposite points of the axis-aligned rectangle to add the *base of* the respective shape. Then click on the third point (not in the same plane) to specify height of the shape.
- `Convex Hull` : Click on the points of the convex hull. Double click on the last point to add convex hull.
- `Convex Hull From File` : Load points of the convex hull from a csv file. It has N x 3 table. Each row has one point and x,y,z co-ordinates are stored in 3 columns.
- `From Last Selected` : Select a shape, an edge, or a face. Then click to convert that item to a `RefWin`.
- `Load STL/BREP/IGES` : Load the files in the specified format to create a new `RefWin`.

GUI - Set Region



(f) Set Region menu



(g) Set Shape As menu

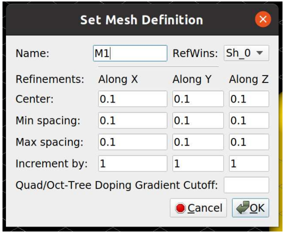
To set a `Region` from one or more `RefWins` (shapes) –

- Edit → Select the desired shapes.
- Click Set Shape As → Region to open the dialog box.

Region dialog box is described below –

- `Reg. Name` : Set name of the region.
- `Material` : Set material from available materials.
- `RefWins` : Names of the selected RefWins are listed here.
- `Max area of elements` : Maximum area of elements in `bulk` and at `interface` for meshing using `Triangle` or `TetGen` mesher.
- `Doping gradient` : `max` and `min` gradient and their corresponding `element area constraints`. Element area is linearly interpolated between max and min gradients (only valid for meshing using `Triangle` or `TetGen` mesher).

GUI - Set Mesh Definition



(h) Set Mesh Definition

To set a `MeshDef` from a `RefWin` –

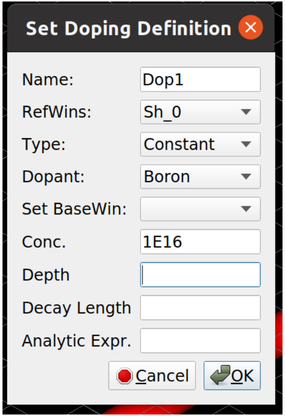
- [Edit] → [Select] the desired shape.
- Click [Set Shape As] → [Mesh Definition] to open the dialog box.

[Mesh Definition] dialog box is described below –

- `MeshDef Name` : Set name of the mesh-definition.
- `RefWins` : Names of the selected RefWins are listed here.
- `Refinements` along X, Y, and Z direction are specified in respective columns.
- `Center` : Center of the refinement point
- `Min spacing` : spacing at the center.
- `Max spacing` : max spacing in the refinement box.
- `Increment by`: As one moves away from `Center`, spacing increases from `min` to `max`. Mutiplying factor is set in `Increment by`.
- `Doping gradient Cutoff` : Quadtree or octtree leaves are split if doping gradient along a side is more than this value.

Note: This mesh definition box is used only in quadtree (2D) or octtree (3D) meshers.

GUI - Set Doping Definition



(i) Set Doping Definition

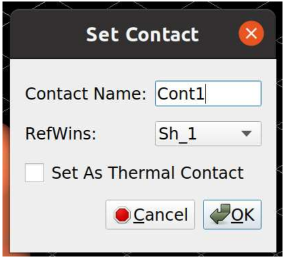
To set a `MeshDef` from a `RefWin` –

- Edit → Select the desired shape.
- Click Set Shape As → Mesh Definition to open the dialog box.

Doping Definition dialog box is described below –

- `DopingDef Name` : Set name of the doping definition.
- `RefWins` : Names of the selected RefWins are listed here.
- `Type` : Set the type of definition - Gaussian, constant, analytic, etc.
- `Dopant` : Specify dopant among available ones.
- `Set BaseWin` : Set basewin for the doping profile
- `Conc` : Set peak concentration (cm^{-3})
- `Depth`and `Decay Length`: For Gaussian and Exponential profiles, set distance of peak from basewin and its decay length.
- `Analytic Expr.` : For analytic profiles, define analytic expression.

GUI - Set Contacts



(j) Set Contact

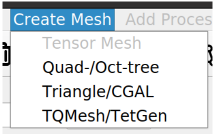
To set a `Contact` from a `RefWin` –

- Edit → Select the desired shape.
- Click Set Shape As → Contact to open the dialog box.

Contact dialog box is described below –

- `Contact Name` : Set name of the contact.
- `RefWins` : Names of the selected RefWins are listed here.

GUI - Create Mesh Menu

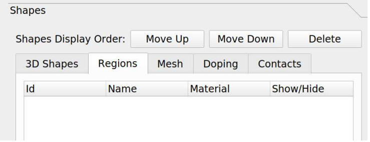


(k) Set Contact

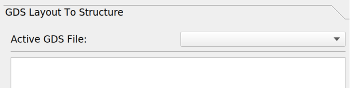
- Once all the regions, mesh definitions, doping definitions, and contacts are set, a mesh can be created.
- Click Create Mesh and then choose the desired mesh engine to perform meshing.
- On succesfull meshing, the Mesh mode is activated and mesh is shown in the drawing board. To toggle to structure view, deactivate Mesh mode.

Following meshing options are available.

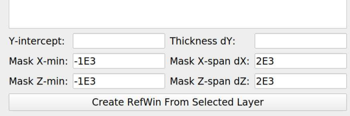
- Tensor mesh : Rectangular (2D) or Cubic (3D) meshing can be performed.
- Quad-/Octtree : FEM mesh is created using quadtree (2D) or octtree (3D) algorithm.
- Triangle/Tetgen : FEM mesh is created using `Triangle` or `Tetgen` package.



(l) Shapes Side-pane



(m) GDS Layout to structure - I



(n) GDS Layout to structure - II

- `Shapes` tab displays all the shapes as well as entities (regions, mesh defs, etc.) which have been added to the structure generator. They are displayed in different tabs in the table.
- To display/hide a shape linked to the given entity, click on the check-box in the last column of the table.
- Order in `Regions` as well as `Contacts` tab is important. If there is an overlap, regions/contacts added later **replace** the region/contact added before.
- Entity order can be changed by clicking on it and click move up/down.
- Delete entity by clicking on it and click Remove.

- Layers defined in imported layout files are listed in this tab. The files can be selected from the drop-down list.
- Check/uncheck the check-box to show/hide the layout of that layer.
- Set the parameters `Y intercept`, `Thickness`, mask `Xmin`, `dX`, `Zmin`, `dZ`.
- Click Create RefWin to add a `RefWin` as per the parameter specifications.

Table of Contents

Conclusions

The structure generator supports creating –

- A 2D/3D device from a rectangle/cuboid, polygon/polyhedron, ellipse (2D), ellipsoid, cylinder, or cone (3D).
- Boolean operations, transformations, and sweeps of 2D/3D devices.
- Doping definitions of analytic profiles such as Gaussian, exponential, linear, or user-defined.
- Mesh refinements globally or in specific regions.
- Python interface for convenient scripting and integration.
- Import *GDS* layout file to create masks for doping, etching, or deposition.
- FEM-mesh, Tensor-mesh, *Triangle* mesh, and, *TetGen* mesh.
- A graphic-user-interface for convenient creation of the structure.

Table of Contents

Easy-to-install licenses

For ordering a license, along with the name and the organization details, please also provide –

- For the node-locked licenses: Ethernet mac address of the client machine on which the software will run.
  OR
- For the server licenses: Ethernet mac address of the server machine at the client organization.

If you purchased one or more node-locked licenses, you will receive the following license file by secured email.

1. NodeLockedLicense_<id>_<Info>.lic, where <id> stands for license id and <Info> stands for customer identification in short.

Copy the license file to `/var/local/oesoft/licenses/` on the machine whose mac-address has been provided and change its access rights to `777`.

If you purchased one or more server licenses, you will receive the following license file by secured email.

1. ServerLicense_<id>_<Info>.lic

Copy the license file to `/usr/share/oesoft/licenses/` on the *server machine* whose mac-address has been provided.

The End

Questions? Comments?