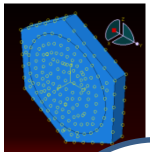
University of Nottingham
UK | CHINA | MALAYSIA

TexGen Workshop

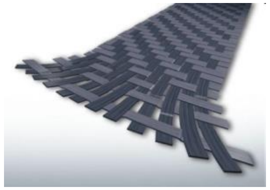
Louise Brown

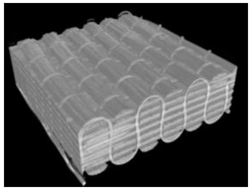
TexGen Geometric Textile Modelling Software



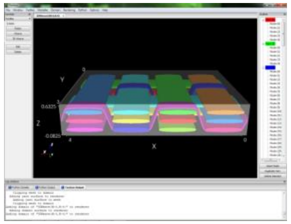
Fibre/Micro-Scale

Micro-scale FEA simulations or analytical methods determine yarn properties

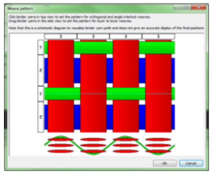




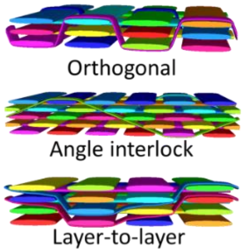
Unit Cell/Meso-Scale

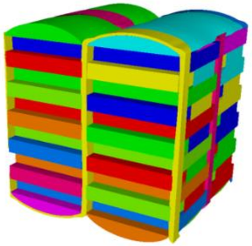


Generate textile geometry using TexGen GUI or script

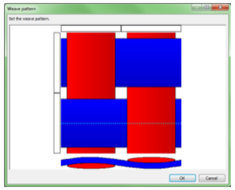


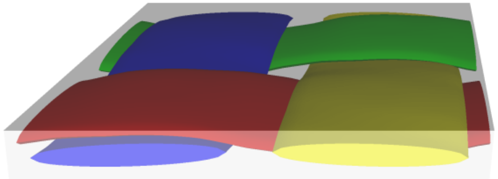
3D wizard generates idealised 3D textiles



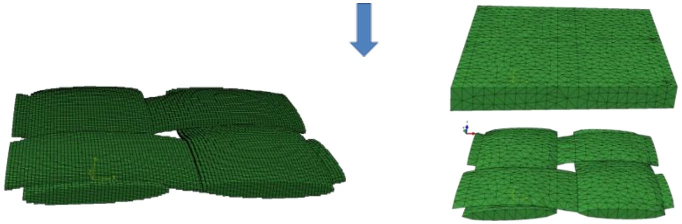


Refinement of orthogonal weave to simulate compaction





Automatically generate 2D and 2D sheared textiles



Generate mesh and input files for FEA or CFD to predict material properties

Composite material properties extracted from meso-scale predictions are used to model structural components

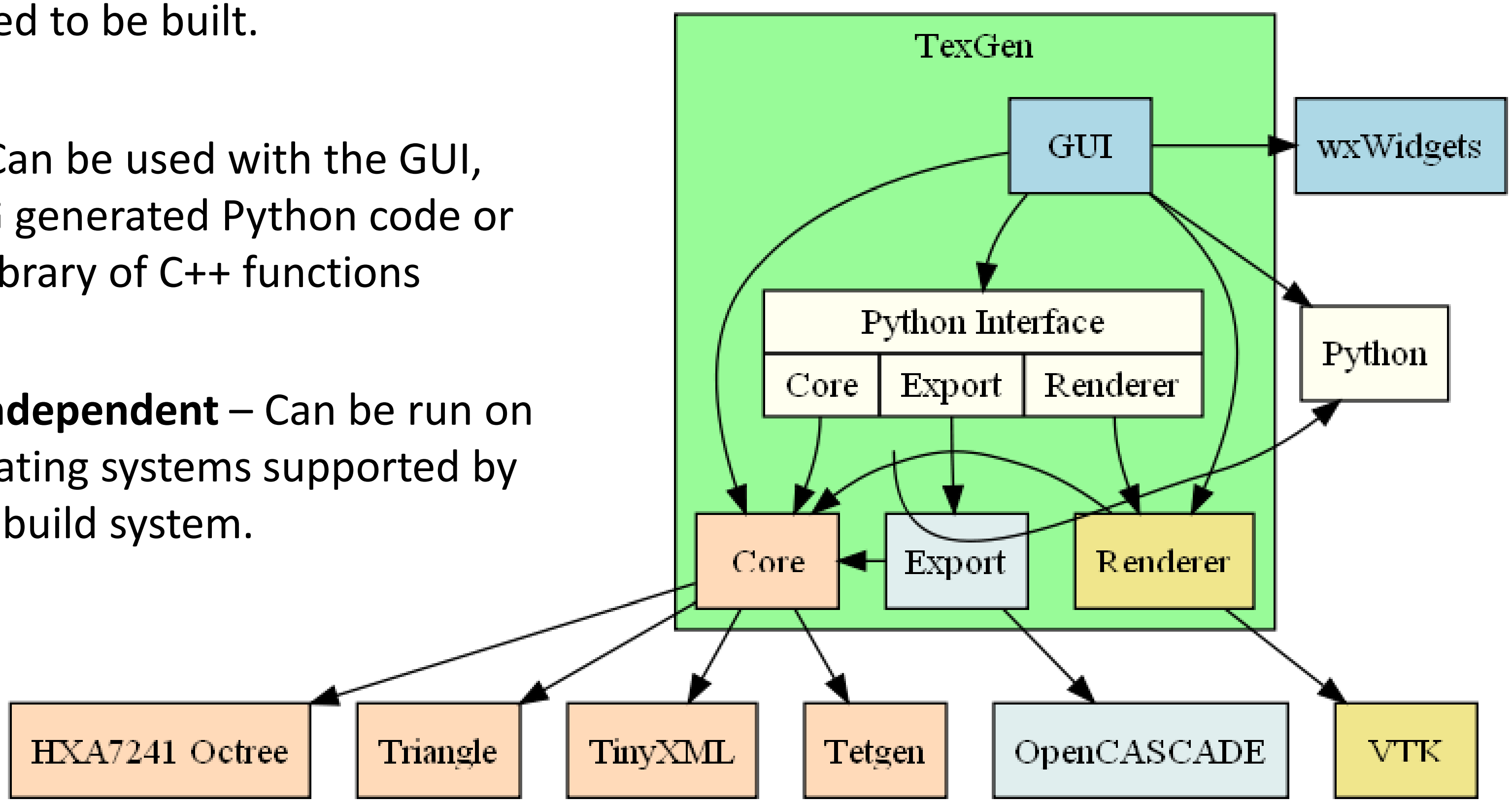
Component/Macro-Scale

Implementation

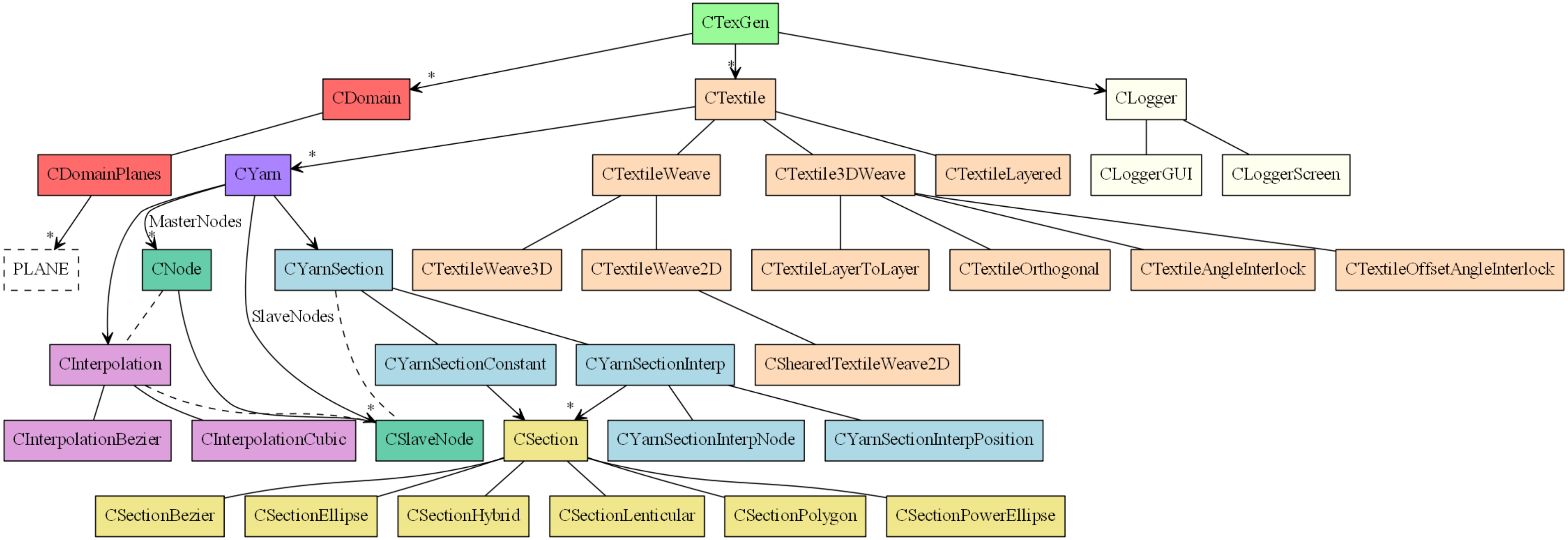
Modular -Core functionality is in the core module, graphics are in a renderer module; if not using visualisation, the renderer doesn't need to be built.

Flexible – Can be used with the GUI, using SWIG generated Python code or used as a library of C++ functions

Platform independent – Can be run on most operating systems supported by the Cmake build system.



TexGen Class Heirarchy



Generating a Textile

1 - Create Textile

2 - Add yarns

3 - Specify domain

4 - Output data

Steps combined and performed automatically in 2D and 3D wizards

Create yarn path
Assign sections
Select interpolation
Assign repeats
Assign fibre properties

Each step individually using either GUI, Python script or C++ API functions

Create Textile

Each textile is created in a Ctextile object

GUI: Select *Textiles -> Create Empty* (Step 1)

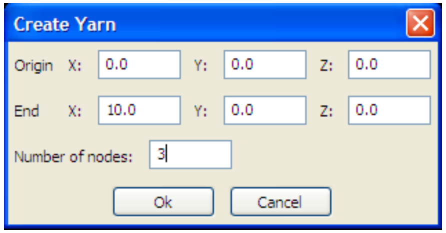
Python:

```
Textile = CTextile()
```

Create Yarns

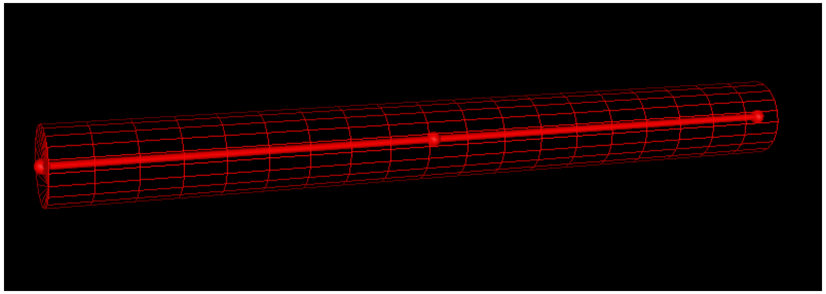
Yarns are denotes by a set of Master Nodes

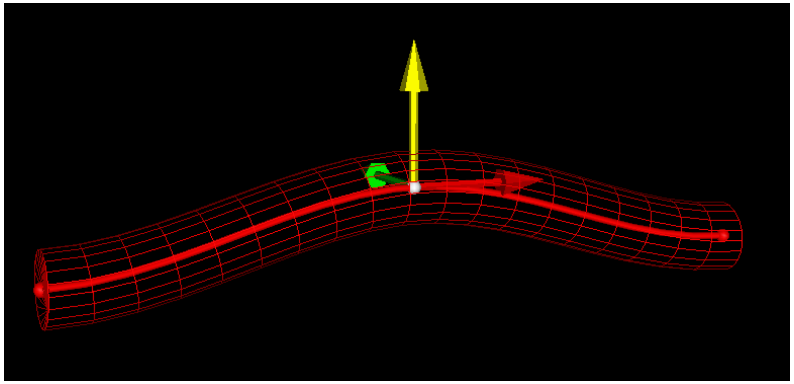
GUI: Select *Modeller->Create Yarn* (Step 2)



Python:

```
Yarn = CYarn()
Yarn.AddNode(CNode(XYZ(0,0,0)))
Yarn.AddNode(CNode(XYZ(5,0,1)))
Yarn.AddNode(CNode(XYZ(10,0,0)))
```

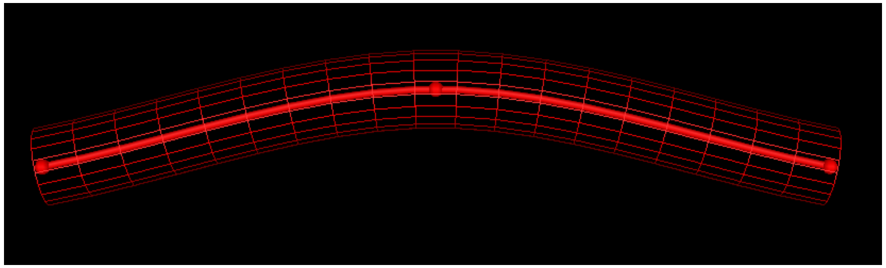




Interpolation Between Nodes

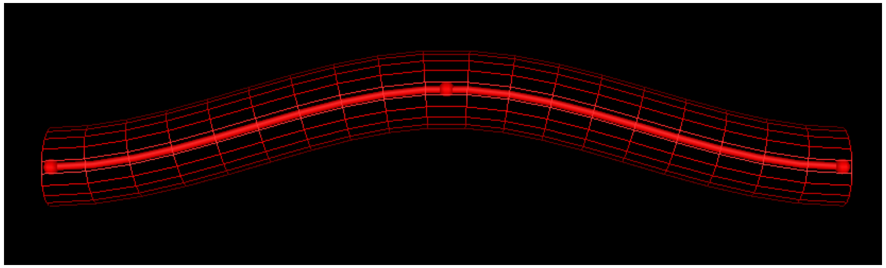
A path is generated between the master nodes by an interpolation function

GUI: Select *Modeller -> Interpolation*



- Bezier spline
- Natural cubic spline
- Linear spline

- Periodic – select to maintain continuity across yarn repeats



Python:

```
Yarn.AssignInterpolation(CInterpolationCubic())
```

Defaults to periodic, send False as parameter to CInterpolationCubic() for non-periodic interpolation

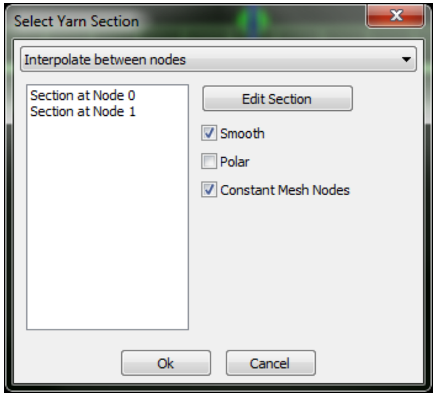
Yarn Cross-Sections - Interpolation

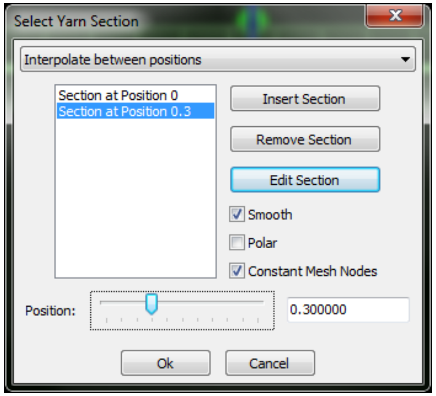
Yarn cross-sections are specified as 2D sections perpendicular to the yarn tangent

By default the cross-section is constant along the length of the yarn or an interpolation method can be chosen

GUI: Select *Modeller -> Assign Section*

- Select interpolation
  - Constant
  - Interpolate between nodes
  - Interpolate between positions



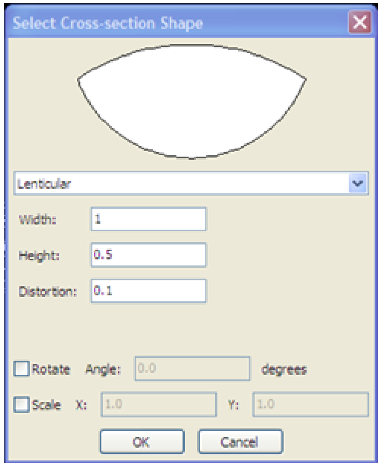


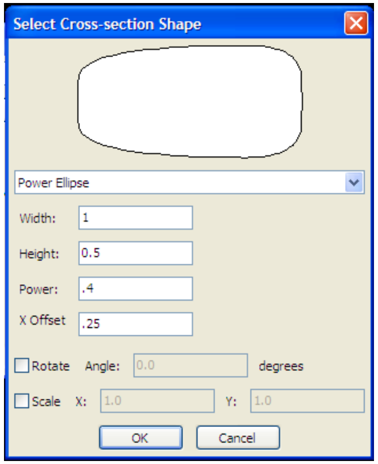
Yarn Cross-Sections – Section Specification

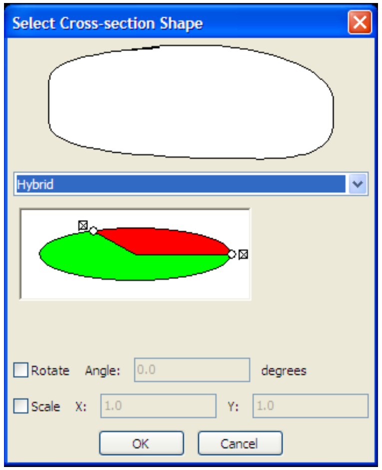
Cross-sections are specified at the locations given by the section interpolation

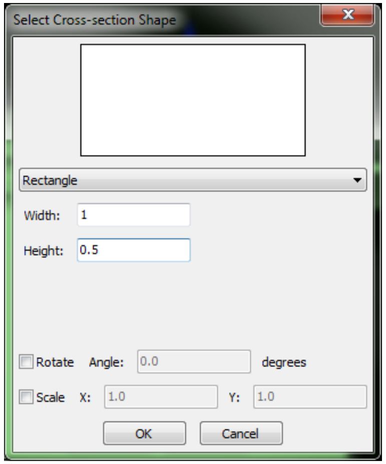
Available cross-sections:

- Ellipse
- Lenticular
- Power ellipse
- Hybrid
- Rectangle
    - Use rather than power ellipse with power = 0 to generate uniform section meshes
- Polygon
    - Only by scripting









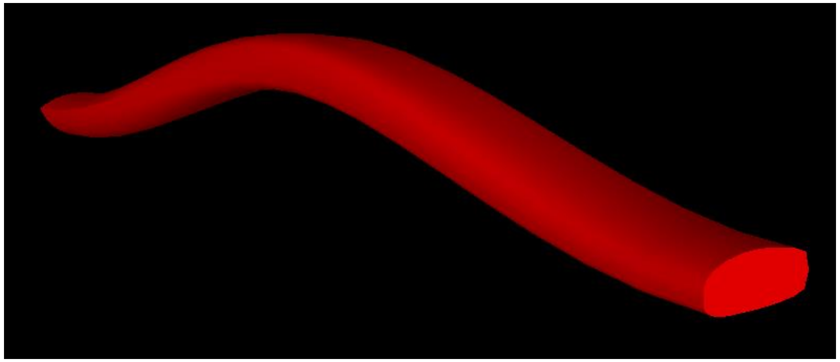
Yarn Cross-Sections – Python Implementation

Python:

```
YarnSections = CYarnSectionInterpNode()
YarnSections.AddSection( CSectionLenticular(1.0,0.5,0.1) )
YarnSections.AddSection( CSectionPowerEllipse(1.0,0.5,0.4,0.25) )

# Hybrid Section
Top = CSectionEllipse( 1.0, 0.4 )
Bottom = CSectionPowerEllipse( 1.0, 0.4, 0.4, 0.25 )
YarnSections.AddSection( CSectionHybrid( Top, Bottom ) )

Yarn.AssignSection( YarnSections )
```

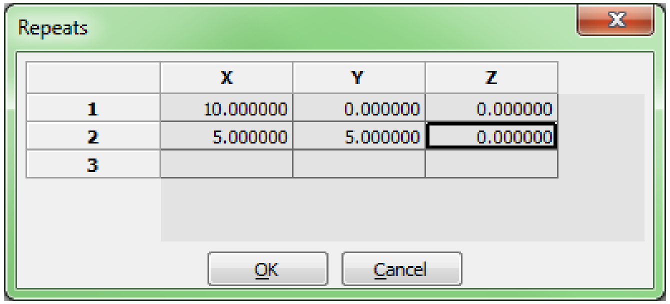


Yarn Repeats

Yarn repeats allow a given yarn section to be repeated as specified by a set of vectors (in theory, allowing an infinite textile)

GUI: Select *Modeller -> Assign Repeats*

- Specify a set of repeat vectors



Python:

```
Yarn.AddRepeat(XYZ(10,0,0))
Yarn.AddRepeat(XYZ(5,5,0))

Textile.AddYarn(Yarn)
```

Domain

The domain restricts the model to a specific region

- Specified by a set of convex planes
- Typically, but not always, the unit cell

GUI:

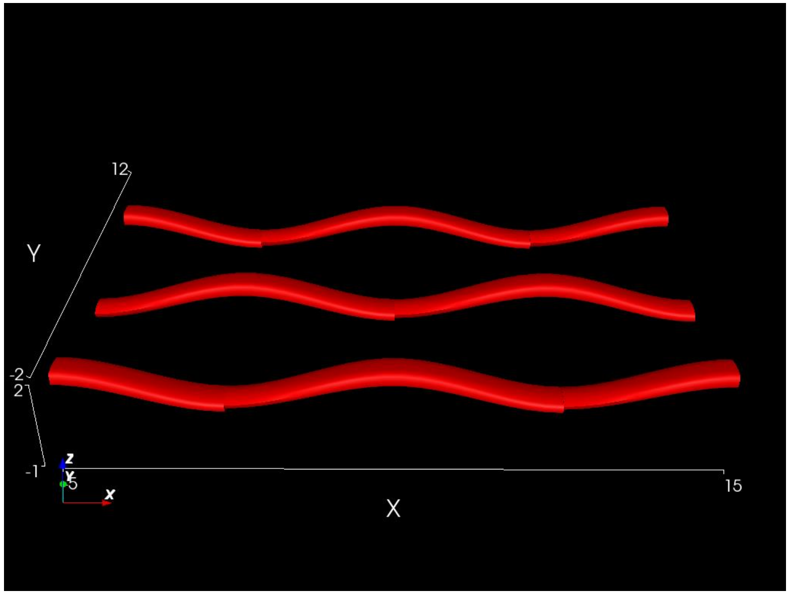
Select *Domain -> Create Box*

- Input minimum and maximum x,y,z values for bounding box

Or

Select *Domain -> Create Planes*

- Input required number of planes specified by the unit normal to the plane and its distance from the origin



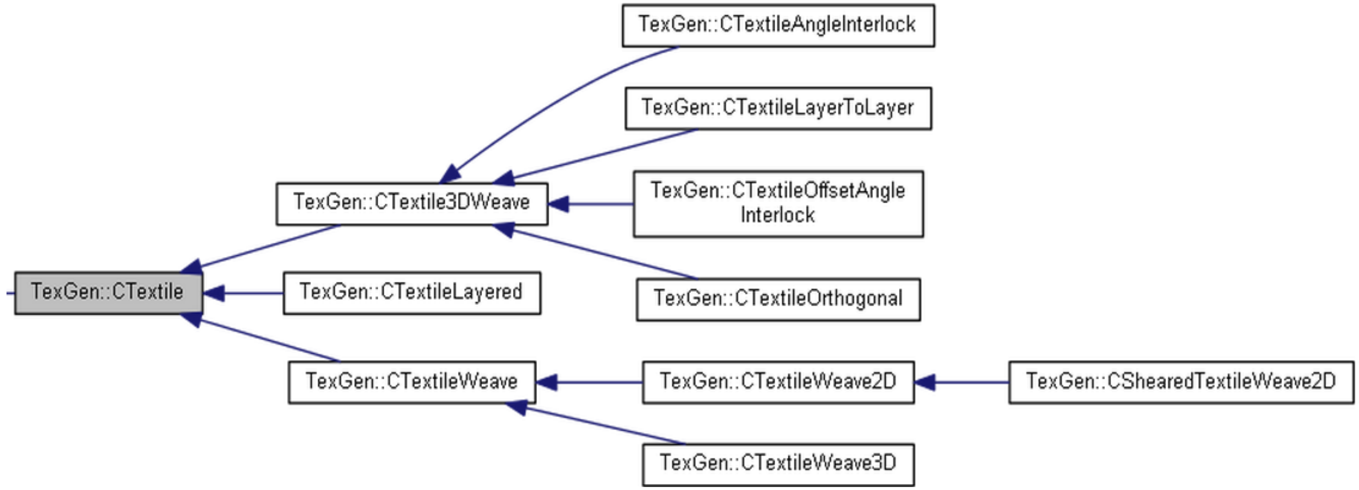
Python:

```
Textile.AssignDomain(CDomainPlanes(XYZ(-5,-2,-1),XYZ(15,12,2)))

AddTextile("Workshop", Textile)
```

Automatically Generated Textiles

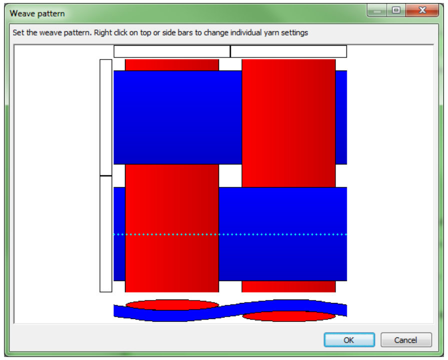
Predefined weave patterns are generated using classes which use Ctextile as a base class. They are used to input weave pattern information which then automatically generate the yarns.

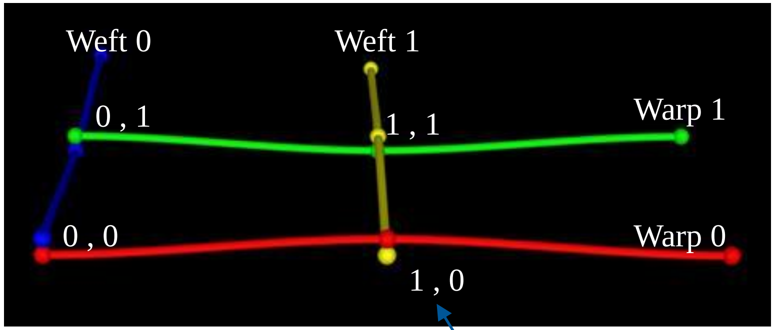


CTextileWeave2D

The 2D wizard in the TexGen GUI creates weaves using the CTextileWeave2D class

Whether the warp and weft are up or down is stored for each x,y position





GUI: Set using Weave Pattern dialog

Python:

```
weave = CTextileWeave2D( numWefts,…
numWarps,spacing, thickness )
weave.SwapPosition(0, 0)
weave.SwapPosition(1, 1)
```

x, y position

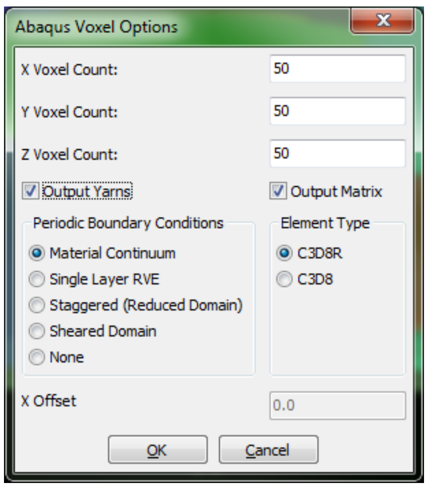
Values stored: 0, 1

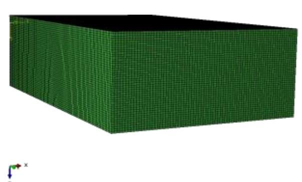
(Weft down, warp up)

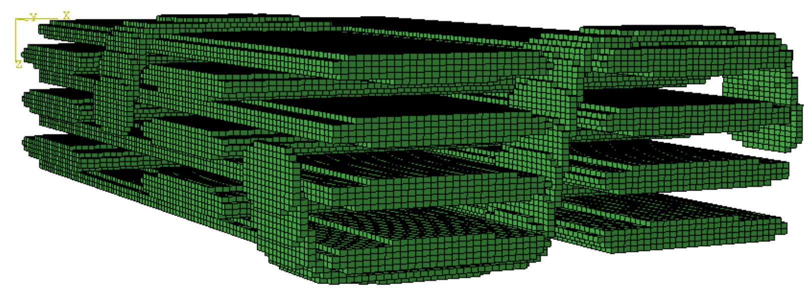
ABAQUS Voxel Export

GUI: Select *File -> Export -> ABAQUS File -> ABAQUS Voxel File*

- Hex elements
- Periodic boundary conditions and steps for extraction of material properties
- http://texgen.sourceforge.net/index.php/Extraction_of_Material_Properties_using_Voxel_Meshing_and_Abaqus





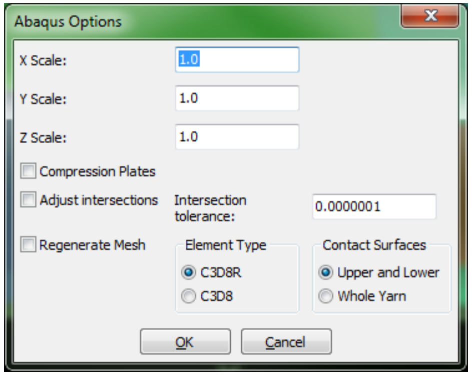


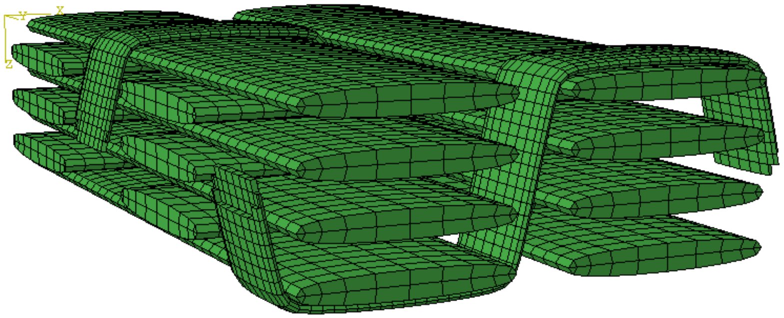
All ABAQUS exports include additional .ori and .eld files containing element orientation, fibre volume fraction and yarn information.

ABAQUS Dry Fibre Export

GUI: Select *File -> Export -> ABAQUS File -> ABAQUS Dry FibreFile*

- Conformal mesh using hex and wedge elements
- Uses weave pattern information to generate contact surfaces
- Correction for small intersections



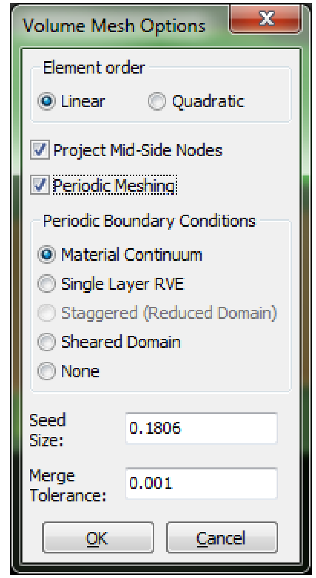


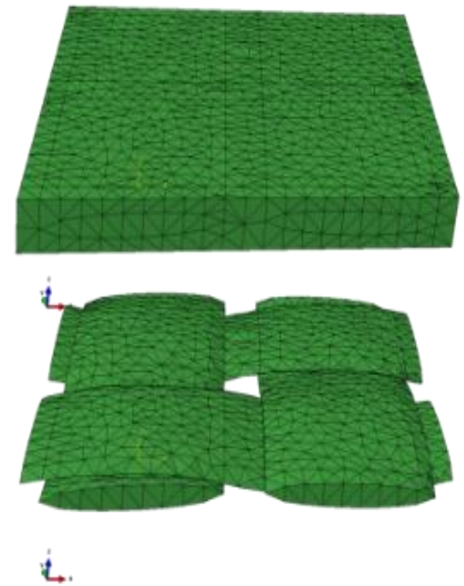
Conformal Mesh

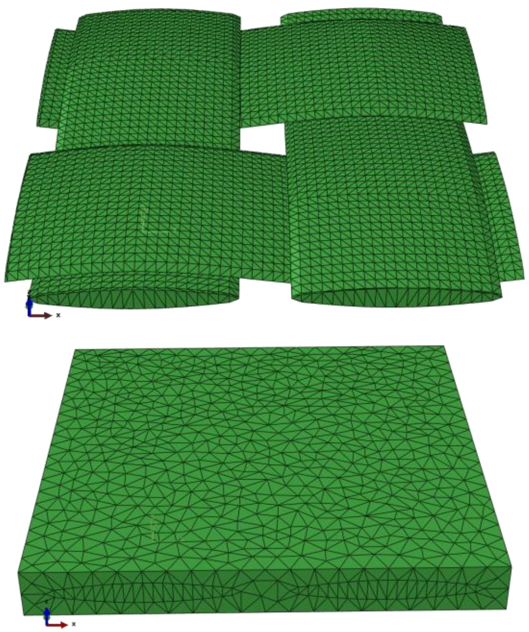
Volume Mesh

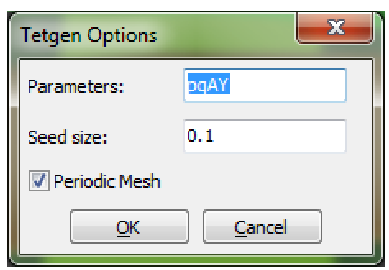
GUI: Select *File -> Export -> Volume Mesh*

- Tetrahedral elements
- Save as ABAQUS .inp file or .vtu
- Works best for 2D weaves









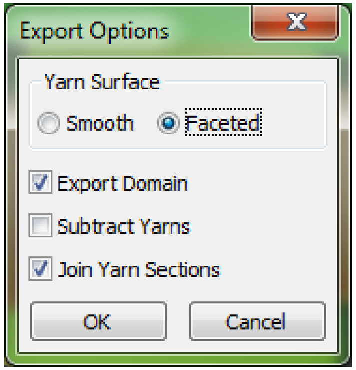
Tetgen Export

GUI: Select *File -> Export -> Tetgen Mesh*

- Tetrahedral elements
- Save as ABAQUS .inp file
- May need to introduce gap between yarns for export to be successful
- Uses Tetgen library: http://wias-berlin.de/software/index.jsp?id=TetGen&lang=1

Geometry Export

The geometry alone can be exported in IGES, STEP or stl format. No orientations, volume fractions or properties are exported.



GUI: Select *File -> Export -> IGES File*
or -> STEP File

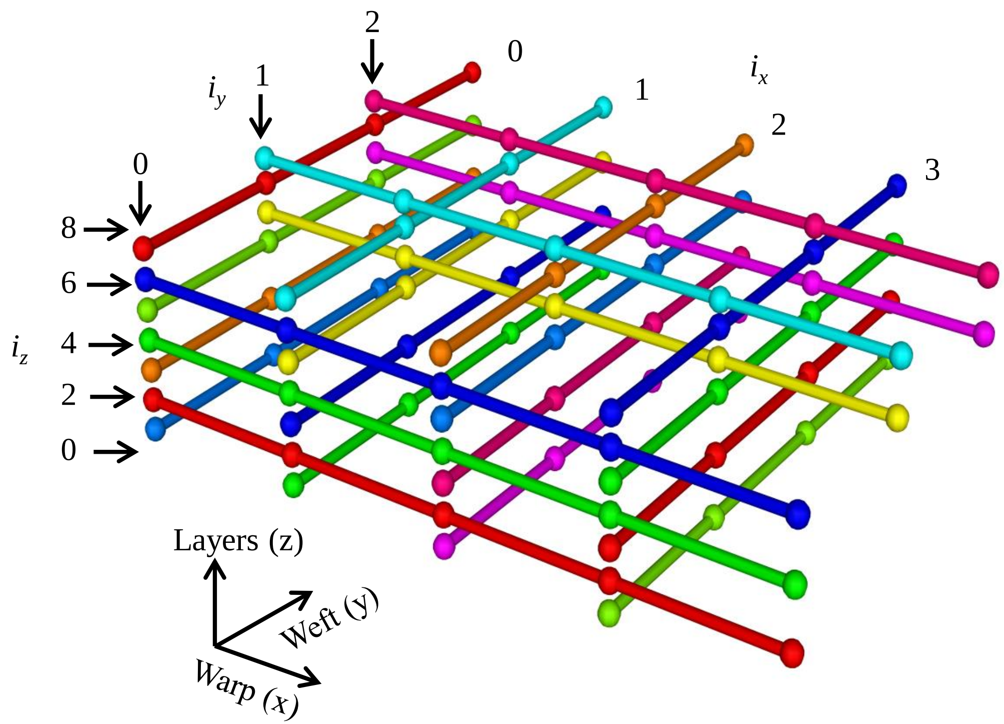
- This option uses the OpenCASCADE library.
- The 'Smooth' option may be unsuccessful for more complex geometries
- 'Join Yarn Sections' will remove joins at repeat boundaries but is <u>much</u> slower

GUI: Select *File -> Export -> Surface Mesh*

- Exports the surface mesh as displayed by *Rendering -> X-Ray*
- Saves in .vtu or .stl format

These all use the CTextile3DWeave base class

- Creates a grid of points at the yarn crossovers, specified in the GUI by the Weave Pattern dialog
- Each point may be warp, weft or no yarn
- The derived classes then automatically generate the yarn paths from this dat
- Textiles using the base class can be created using a Python script



Resources

User Guide:
http://texgen.sourceforge.net/index.php/User_Guide

Scripting Guide:
https://github.com/louisepb/TexGenScriptingGuide

TexGen source code:
https://github.com/louisepb/TexGen

Workshop materials:
https://github.com/louisepb/ICMAC2018-Workshop