**Purdue TexGen Workshop 15th November 2019**

* Composites group intro – Slides 2-4
* General introduction
  + Overview slide – Slide 6
  + Webpage - Slide 7
  + Download – Slide 8
  + Code on Github – Slide 9
  + Forum – Slide 10
  + Software overview – Slides 11,12
* Use Cases – S13
  + Permeability and mechanical modelling – Slides 14-27
  + CTE of sheared fabric – Slides 30-43
  + Braiding – Slides 46-49
  + Optimisation – Slides 50-53
  + T-Piece modelling – Slides 54-59
  + Knitted fabrics – 60-64
* Modelling Theory
  + Slides 46-54
* TexGen as a Pre-processor
  + Elastic analysis Slides 56-62
  + Flow simulation Slides 62-70
* Open TexGen
  + Windows – make sure have Controls, Outliner and Log windows open
  + Controls – most commonly used are Textiles and Modeller
  + Outliner – shows yarns and nodes
  + Log window
    - Demonstrate Python console
    - Python output – TexGen is written in C++ but functions are wrapped to give access via Python scripting. GUI calls via Python & output recorded here
      * Executing commands and looking at output can be a good way of seeing how to call Python commands
    - TexGen output – shows messages output by TexGen – useful to see if any error messages
* Most common starting point – 2D Wizard with default settings
* Slides showing CTextileWeave class
  + Classes inherited from CTextile – 72
  + CTextileWeave2D – 73-74
  + Yarn spacing is distance between centrelines of yarns
  + The yarn height will be the fabric thickness / 2
    - Show plain weave without refine
    - Weave pattern dialog
    - Create weave
    - Defaults to elliptical cross-sections
  + Python output shows commands which have been executed
  + Show outliner
  + Render x-ray, nodes, paths
  + Render interference
  + Right button – zoom
  + Centre button – pan
* Show same plain weave with refine
  + Gap size if want space between yarn surfaces
  + Cross-sections have been changed
  + Show changed to interp between nodes
  + Show use of options in modeller window to assign sections etc
* 2D wizard 4x4 textile
  + Show selecting bars – ctrl-click to select multiple
  + Change 2nd & 4th yarns to 0.9 width, 0.08 height
* Exercise to create 2D weave – 75
  + Either select *Textiles->Create Weave...* from the main menu or *Weave* from the *Textiles* tab of the Controls menu.
  + Set the number of warp and weft yarns to 4 (to create a 4x4 satin weave).
  + Use the warp yarn data to set the yarn spacing and yarn width (as these parameters are different for the warp and weft yarns the weft dimensions will be set later using the Pattern Dialog). The *Yarn Spacing* is set to 1.0 based on 10 yarns/cm.
  + The combined heights of the yarns is 0.5 mm so this is selected as the *Fabric Thickness.* The wizard will automatically set both yarn heights to 0.25mm (half the thickness) but, again, these can be adjusted in the Pattern Dialog.
  + Click the *Next* button to move to the *Weave pattern dialog.*
  + *Shift-click* on the bars at the left side of the weave pattern to select all of the warp yarns.
  + *Right-click* on one of the selected side bars to show options to change yarn parameters for these yarns
  + Select *Set yarn height...* and enter the value 0.3
  + Repeat steps 6-8 to select the bars at the top of the weave pattern and set the weft yarn width, height and spacing to 0.7, 0.2 and 0.769 respectively. (The spacing is calculated using 13 yarns/cm)
  + Click on the crossovers on the weave pattern to create the desired weave configuration
  + Select *OK.* The textile can also be loaded from the SatinWeave4x4.tg3 file.
* 3D textiles
  + 3D weave wizard
  + Start with orthogonal
  + 4 wefts, 4 layers
  + Weft spacing needs to be sufficient to accommodate height of binder
  + Explain warp/binder ratio
  + Slide showing CTextileWeave3D - 76
  + All use same base class – can’t create using this in GUI
  + Refine (record) – same as 3DOrthogonal example in scripting guide
    - Weft – yarns = 4, layers = 3, spacing = 2.8, width = 2.58, height = 0.25, power = 0.6
    - Warp – yarns = 6, layers = 2, binder ratio = 1, warp ratio = 2, spacing = 3.8, width = 3.6, height = 0.35, power = 0.6
    - Binder – width = 1.375, height = 0.16, spacing = 1.4, power = 0.8, thickness = 1.5
    - All fibre diameter = 0.007 mm
    - Fibres/yarn – weft=5000, warp=8000, binder=3500
  + Demonstrate layer to layer
    - Increased number of layers governs max number of binder layers
    - Warp layers linked to number of weft layers
* Exercise to create 3D weave - 77
  + Either select *Textiles->Create 3D Weave…* from the main menu or *3D Weave* from the *Textiles* tab of the Controls menu.
  + Select the *Orthogonal* weave type and select *Next.*
  + Fill in the weft, warp and binder data.
  + In the binder yarn window select the *Refine* option and set the *Target Thickness* to 1.4.
  + In the three subsequent windows set the yarn properties.
  + In the final weave pattern window click on the points on the binder yarns to create the desired weave configuration
  + Select *OK.* The textile can also be loaded from the OrthogonalThickness\_1.4.tg3 file.
  + Same parameters without refinement: OrthogonalDefaultThickness.tg3
* Save textile
  + Tg3 format = Slide 79
  + File->SaveTexGenFile - save in TexGen/Workshop – satin.tg3
  + Edit file – standard XML file
  + Can edit file
  + Show weave data part of file
  + Save screenshot – Slide 80
* Export options
  + Properties – Slide 81
  + Show yarn properties for textile
  + Select yarns
  + Yarn properties for yarns
    - Need density etc for volume fraction
    - Note don’t necessarily need all of them
  + Show orientations
  + Slides for Export options – 82-86
* Manual creation of yarns
  + Create empty textile
  + Create default yarn
  + Select & duplicate yarn – use button or ctrl-d
  + Drag to y = 5
  + Select yarn 0 from outliner & duplicate
  + Select node 1 of yarn 2 using outliner
  + Type x = 0, y = 10
  + Select yarn 2 using outliner & duplicate
  + Drag yarn 3 to x = 5
  + Select node 1 in each yarn & insert node
  + Select nodes 0 & 2 of yarns 0 & 3 and node 1 of yarns 1 & 2
  + Type z = 2
  + Select all yarns
  + Assign section
  + Change to lenticular, width = 4, height = 2
  + Create box domain max x & y = 10, min z = -1, max z = 3
  + Trim to domain
  + Select all yarns
  + Assign repeats x = 10 & y = 10
  + Save textile
* Edit textile using GUI (will cover editing using Python console later)
  + Show render interference depth – look at node 1
  + Select yarn 1
  + Assign section – change to interpolate between nodes
  + Select yarn 1 – change distortion ( to 0.1 or 0.2?)
  + Refresh view
  + Show interference depth
  + Select hybrid section – divide into 2 horizontally
  + Leave top section
  + Lower section – height = 1.8
  + Refresh view
  + Show interference depth
  + Final model in CorrectInterferenceYarn1.tg3
* Python scripting
  + SingleYarn.py –
    - Open file so attendees can follow
    - Slides 89-95
  + SingleYarnSections.py
    - Adding sections – Slide 96
    - (Use ReplaceSection for interpNode if need to change)
    - Creating sections – Slide 97- 102
    - Move y plane to 6.9 to show hybrid section
  + Polyester.py
    - Adding multiple yarns – Slide 103
    - Loop to set up cross-sections etc – Slide 104
    - Finally set up domain and add to textile (script)
* Exercise to edit polyester.py – Slide 105
* Weave classes in scripts – Slide 106
  + 2dweave.py
* Other textile types
  + TriaxialBraid.py
  + WeftKnit.py
* Python editing of textile
  + Textile = GetTextile(‘name’)
  + yarn = textile.GetYarn(1)
  + yarn.SetResolution(20) (or numSlaveNodes, numSectionPoints)
  + newNode = CNode( XYZ(5, 5, 2.4))
  + yarn.ReplaceNode(1, newNode )
  + Sections = yarn.GetYarnSection()
  + Sections.GetType()
  + InterpSections = CYarnSectionInterpNode()
  + InterpSections.AddSection(CSectionEllipse(2, 0.048))
  + InterpSections.AddSection(CSectionLenticular(2, 0.048))
  + InterpSections.AddSection(CSectionEllipse(2, 0.048))
  + yarn.AssignSection(InterpSections)
  + Refresh view
  + Or could put into a script and run using RunScript
  + Weave classes
  + Use 2dweave.py
  + Textile = GetTextile()
  + Textile.GetRefine(True,True)
    - Fails – need to get inherited class to get at virtual functions
    - Weave = Textile.GetWeave()
    - Weave.RefineTextile(True, True)
    - Refresh view
  + Scripting guide <https://github.com/louisepb/TexGenScriptingGuide>