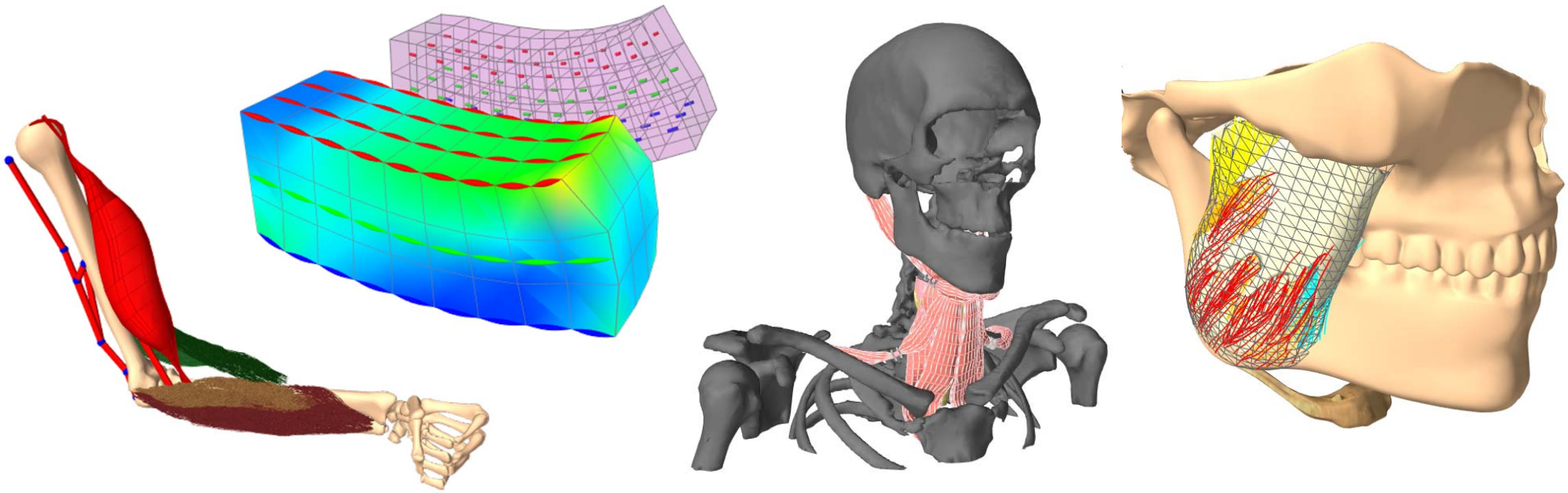
ArtiSynth: finite-element models



Overview

PMHA 2014

FEM Basics
- FEM sub-components
- Creating simple models
- Loading models from files
- Constitutive materials

Integrating FEMs into larger models
- Boundary conditions
- Attachments
- Collisions

FEM Muscles
- Fibre-based muscles
- Muscle materials

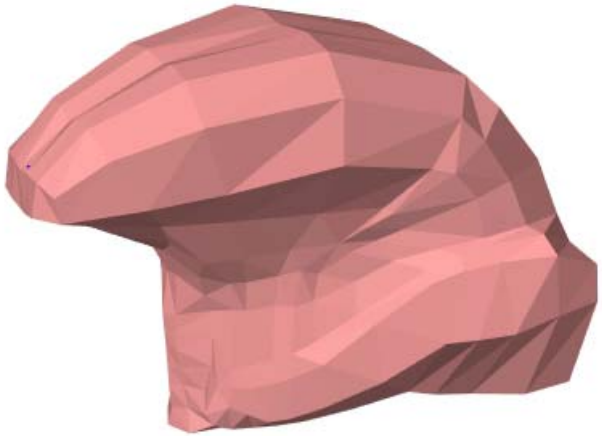
Visualizations
- Rendering properties
- Showing stress/strain

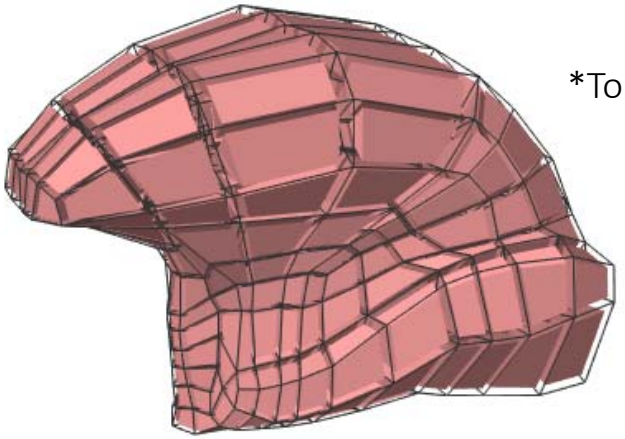
FEM Overview

PMHA 2014

Finite element method: numerical technique to solve system of partial differential equations over some domain.

Approach: divide the domain into building blocks (elements), solve smaller problems



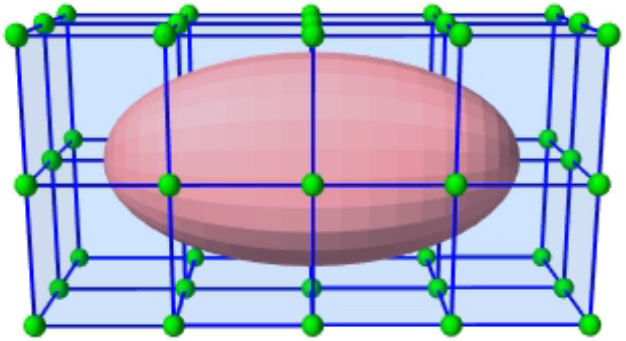


*Tongue by TIMC

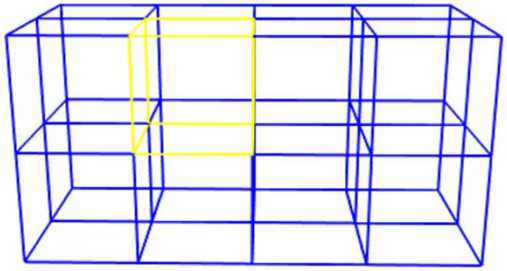
Differential equations:

- governing equations of continuum mechanics: conservation of mass, momentum and energy
- a constitutive law that describes the underlying material

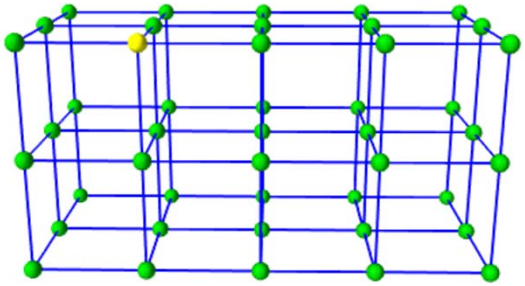
FEM Overview



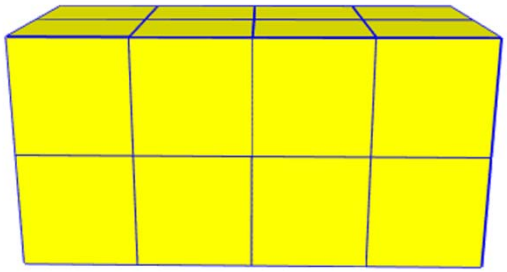
Elements (building blocks)

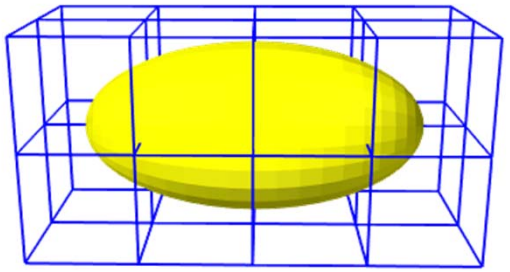


Nodes (dynamic points)



Geometry (meshes)





FEM Creation

PMHA 2014

Basic Template:

```
// Create and add main MechModel
MechModel mech = new MechModel("mech");
addModel(mech);

// Create FEM
FemModel3d fem = new FemModel3d("FEM Example");

// FEM setup
       ⋮

// Add FEM to model
mech.addModel(fem);
```

FEM Setup:

- Build nodes/elements
- Density*
- Material properties* (constitutive law)
- Boundary conditions
- Render properties

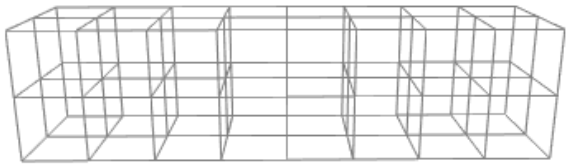
** depends on spatial scale*

FEM Creation: FemFactory

PMHA 2014

FemFactory has many functions for generating simple models

- FemFactory.createGrid(…)
- FemFactory.createEllipsoid(…)
- FemFactory.createTorus(…)
- FemFactory.createTube(…)
- FemFactory.createWedge(…)



```
// Create FEM
FemModel3d beam = new FemModel3d("beam");

// FEM Setup
double[] size = {1.0, 0.25, 0.25};  // widths
int[] res = {8, 2, 2};              // resolution

FemFactory.createGrid(beam, FemElementType.Hex,
   size[0], size[1], size[2],
   res[0], res[1], res[2]);

// Set density, properties...
          ⋮

// Add FEM to model
mech.addModel(beam);
```

FEM Creation: FEM files

PMHA 2014

Supported Formats:

- ANSYS
- TetGen
- ABAQUS (limited)
- VTK ASCII (limited)

```
AnsysReader.read(fem, ...);
TetGenReader.read(fem, ...);
AbaqusReader.read(fem, ...);
VtkAsciiFemReader.read(fem, ...);
```

File IO may throw IOException

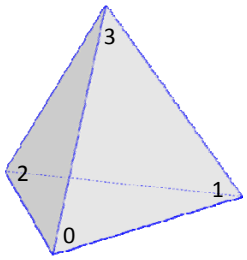
```
// Create FEM
FemModel3d tongue = new FemModel3d("tongue");

// Read FEM from file
try {
   // Get files relative to THIS class
   String nodeFileName = ArtisynthPath.getSrcRelativePath(this,
                              "data/tongue.node");
   String elemFileName = ArtisynthPath.getSrcRelativePath(this,
                              "data/tongue.elem");

   AnsysReader.read(tongue, nodeFileName, elemFileName);

} catch (IOException ioe) {
   // Wrap error, fail to create model
   throw new RuntimeException("Failed to read model", ioe);
}

// Add to model
mech.addModel(tongue);
```

FEM Creation: Manual

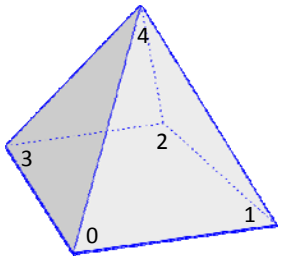
PMHA 2014

Manual creation:

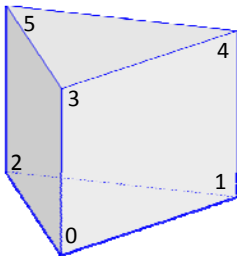
- Add nodes: `FemModel3d.addNode(...)`
- Add elements: `FemModel3d.addElement(...)`



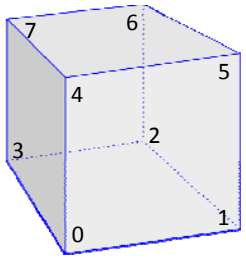
TetElement



PyramidElement



WedgeElement



HexElement

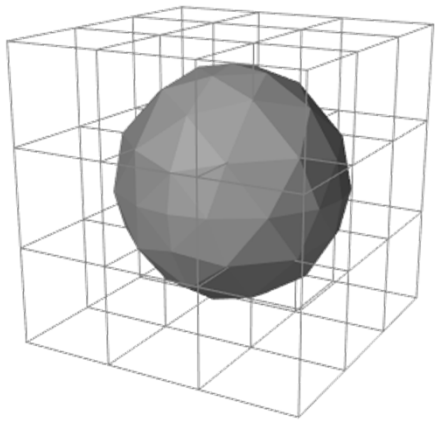
```
Point3d[] pnts = ...
int [][] elems = ... // { {0, 1, 2, 3}, {2, 3, 4, 5} ...

// Add nodes from a set of points
for (Point3d pnt : pnts) {
   fem.addNode(new FemNode3d(pnt));
}

// Add elements from a 2D array of node indices
for (int i=0; i<elems.length; i++) {

   // Collect nodes based on indices
   FemNode3d[] nodes = new FemNode3d[elems[i].length];
   for (int j=0; j<elems[i].length; j++) {
      nodes[j] = fem.getNode( elems[i][j] );
   }

   // Create element from supplied nodes
   FemElement3d elem = FemElement3d.createElement(nodes);

   // add element to model
   fem.addElement(elem);
}
```

FEM Creation: Add Geometry

PMHA 2014

Additional geometry can be added to the FEM

- Align geometry inside FEM
- `FemModel3d.addMesh(MeshBase)`



`MeshBase: PolygonalMesh, PolylineMesh, PointMesh`

```
 // Create FEM block
FemModel3d block = new FemModel3d("block");
FemFactory.createHexGrid(block, 1.0, 1.0, 1.0, 3, 3, 3);

// Set properties
block.setDensity(40);
block.setMaterial(new LinearMaterial(10000, 0.33));

// Create sphere geometry
double r = 0.4;
int refinement = 2;      // level of refinement
PolygonalMesh sphere = MeshFactory.createOctahedralSphere(
                           r, refinement);

// Add sphere geometry to FEM
FemMesh embeddedSphere = block.addMesh(sphere);
embeddedSphere.setName("sphere");

// Add FEM to model
mech.addModel(block);
```

Constitutive Materials

PMHA 2014

- Defines the stress-strain relationship
- `FemModel3d.setMaterial(FemMaterial)`

`FemMaterial` subclasses:

- `LinearMaterial`
- `StVenantKirchoffMaterial`
- `MooneyRivlinMaterial`
- `CubicHyperelastic`
- `OgdenMaterial`
- `FungMaterial`
- `NeoHookeanMaterial`
- `IncompNeoHookeanMaterial`

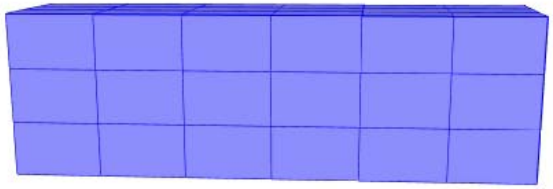
```
// Basic corotated linear material
double youngsModulus = 4000;
double poissonsRatio = 0.49;
boolean corotated = true;     // rotation-invariant

fem.setMaterial(
   new LinearMaterial(
      youngsModulus, poissonsRatio, corotated) );
```

```
// Incompressible Mooney-Rivlin material
double c10 = 1000;
double c01 = 0;
double c11 = 0;
double c20 = 500;
double c02 = 0;
double kappa = 100*c10;

fem.setMaterial(
         new MooneyRivlinMaterial(
            c10, c01, c11, c20, c02, kappa) );
```

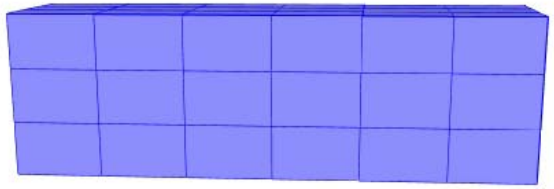
Constitutive Materials

PMHA 2014

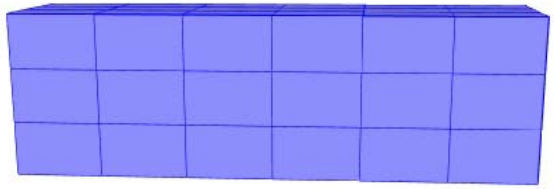
Pure Linear



Corotated Linear



Mooney-Rivlin
(incompressible)



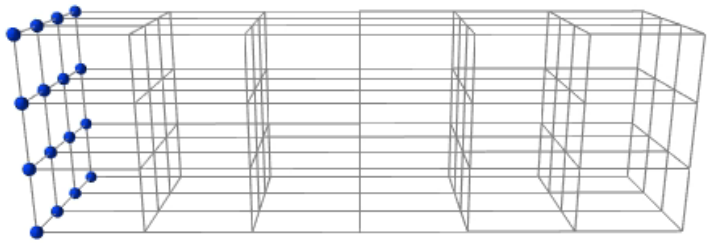
Boundary Conditions

PMHA 2014

- Fixed or parametrically-controlled node positions
  `FemNode3d.setDynamic(false)`

- Parametric control through:
  - `Controller`
  - `InputProbe`

- Can apply forces to nodes
  `FemNode3d.setExternalForce(Vector3d);`



```
// Anchor -x side of beam
double epsilon = 1e-15;   // small offset
for (FemNode3d node : beam.getNodes()) {
   if (node.getPosition().x < -size[0]/2 + epsilon) {
      node.setDynamic(false);
   }
}
```

Attachments

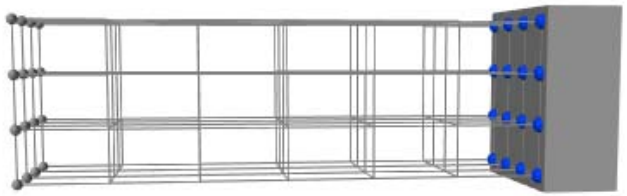
PMHA 2014

`PointParticleAttachment:` Attach one "point" to a "particle" (includes `FemNode3d`)

`PointFrameAttachment:` Attach one "point" to a "frame" (includes `RigidBody`)

`PointFem3dAttachment:` Attach one "point" to some linear combination of FEM nodes

```
// Create a rigid block and move to the side of FEM
RigidBody block = RigidBody.createBox (
   "block", width/2, 1.2*width, 1.2*width, 2*density);
block.setPose (new RigidTransform3d (length/2+width/4, 0, 0));
mech.addRigidBody (block);

// Attach right-side nodes to rigid block
for (FemNode3d node : fem.getNodes()) {
   if (node.getPosition().x > length/2-EPS) {
      mech.addAttachment (
         new PointFrameAttachment (block, node) );
   }
}
```



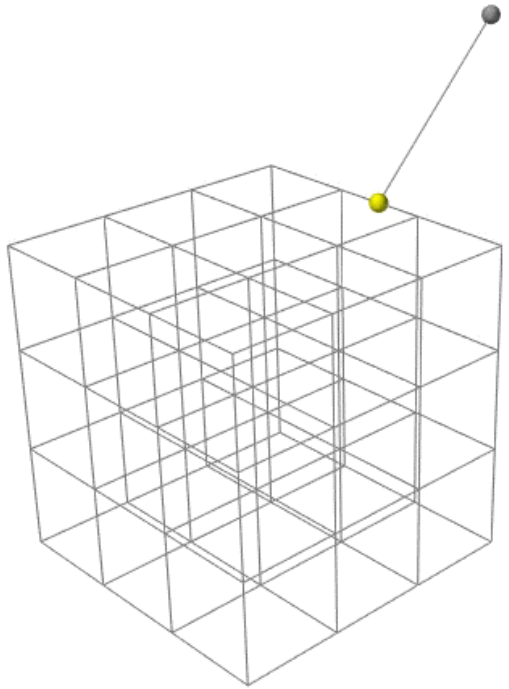
Attachments: Markers

PMHA 2014

Objects can also attach to an arbitrary fixed position inside an FEM using “markers”:

- `FemModel3d.addMarker( Point3d )`

```
 // Create and add FemMarker at (0.5, 0.0, 0.5)
Point3d markerPos = new Point3d(0.5, 0.0, 0.5);
FemMarker marker = block.addMarker(markerPos);

// Add a fixed particle at (1.0, 0.0, 1.0)
Particle particle = new Particle(0, 1.0, 0.0, 1.0);
particle.setDynamic(false);
mech.addParticle(particle);

// Attach a spring between the particle and FemMarker
AxialSpring spring = new AxialSpring("spring");
spring.setMaterial(new LinearAxialMaterial(1000, 0));
spring.setPoints(marker, particle);
mech.addAxialSpring(spring);
```



DEMOS

PMHA 2014

- FemBeam
  - `FemFactory`
  - Fixed boundary condition
- FemBeamWithBlock
  - Attaches to rigid block
- FemBeamWithMuscle
  - `FemMarker`
  - Adds a point-to-point muscle

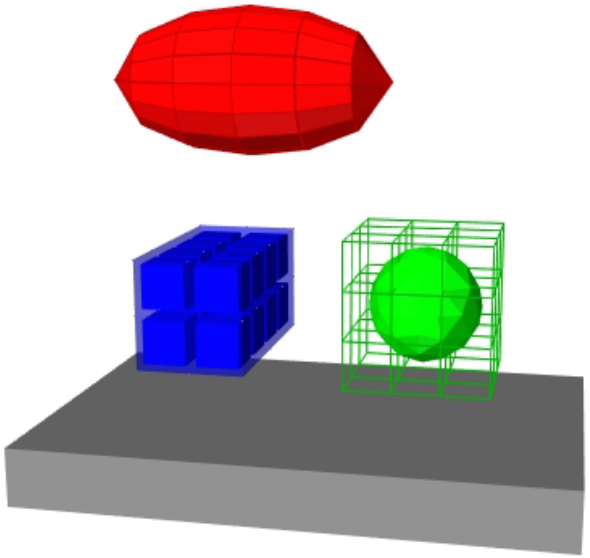
Collisions

PMHA 2014

Collisions can be enabled between any pair of `Collidable` objects

`Collidable`s:

- `FemModel3d`
- `RigidBody`
- `FemMesh` (any mesh in a `FemModel3d`)
- `RigidMesh` (any mesh in a `RigidBody`)



```
// Enable collisions between FEM surface and rigid block
mech.setCollisionBehavior(beam, table, true);
mech.setCollisionBehavior(ellipsoid, table, true);
mech.setCollisionBehavior(ellipsoid, beam, true);

// Enable collisions between embedded mesh and rigid block
FemMesh embeddedMesh = block.getMesh("sphere");
mech.setCollisionBehavior(embeddedMesh, table, true);
mech.setCollisionBehavior(embeddedMesh, ellipsoid, true);
```

Muscles

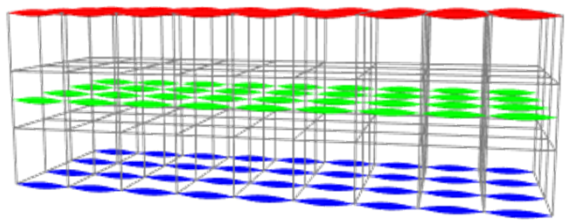
Use `FemMuscleModel` instead of `FemModel3d`

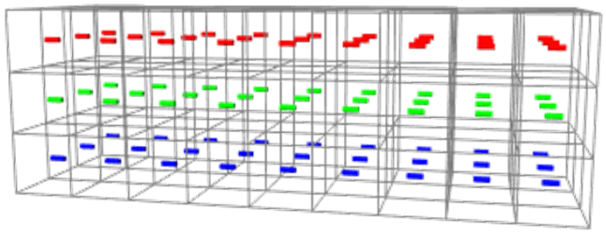
- Adds new properties:
  - Muscle material
  - Muscle bundles & exciters
  - Muscle-specific rendering

```
// Create FEM muscle
FemMuscleModel muscle = new FemMuscleModel("FEM Muscle");

// FEM setup
       ⋮

// Add to model
mech.addModel(muscle);
```

Two "types" of muscles in an FEM:

- Muscle fibres: use point-to-point springs for activation
- Muscle *material*: add an auxiliary material to embed muscle properties



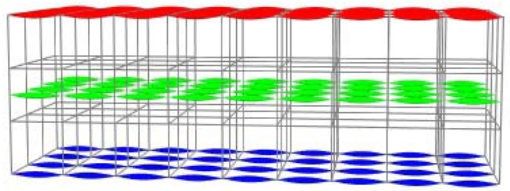


Muscles: Fibres

PMHA 2014

Muscle fibres:

- Point-to-point muscles between pairs of points (nodes or markers)
- Assign an `AxialMuscleMaterial` (same as p2p muscles)
- Enable `MuscleBundle.setFibresActive(true)`



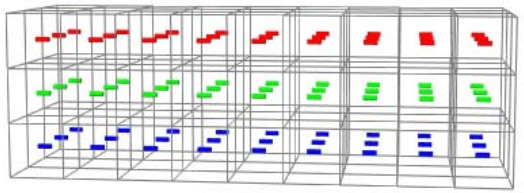
```
// Add muscle bundles
MuscleBundle bundle = new MuscleBundle("fibres");
Point3d[] fibrePoints = ...

// Add fibres
Point pPrev = beam.addMarker(fibrePoints[0]);
for (int i=1; i<=fibrePoints.length; i++) {
   Point pNext = beam.addMarker(fibrePoint[i]);

   double l0 = pNext.distance(pPrev); // rest len
   AxialMuscleMaterial fibreMat =
         new BlemkerAxialMuscle(
               1.4*l0, l0, 3000, 0, 0);

   // add a fibre between pPrev and pNext
   bundle.addFibre(pPrev, pNext, fibreMat);
   pPrev = pNext;
}

// Enable use of fibres (default is disabled)
bundle.setFibresActive(true);
beam.addMuscleBundle(bundle);
```

Muscles: Embedded Material

PMHA 2014

Embedded material bundle:

- Augments constitutive law with muscle props
- Assign a `MuscleMaterial`
  e.g. `BlemkerMuscle` (Blemker, 2005)



```
// Add muscle bundles
MuscleBundle bundle = new MuscleBundle("embedded");

// Embedded material
MuscleMaterial muscleMat = new BlemkerMuscle(
      1.4, 1.0, 3000, 0, 0);
bundle.setMuscleMaterial(muscleMat);

Point3d centroid = new Point3d();
Vector3d dir = Vector3d.X_UNIT;
double eps = 1e-6;

// Add elements if centroid above 0
for (FemElement3d elem : beam.getElements()) {
   elem.computeCentroid(centroid);
   if (centroid.z > eps) {
      bundle.addElement(elem, dir);
   }
}

beam.addMuscleBundle(bundle);
```

DEMOS

PMHA 2014

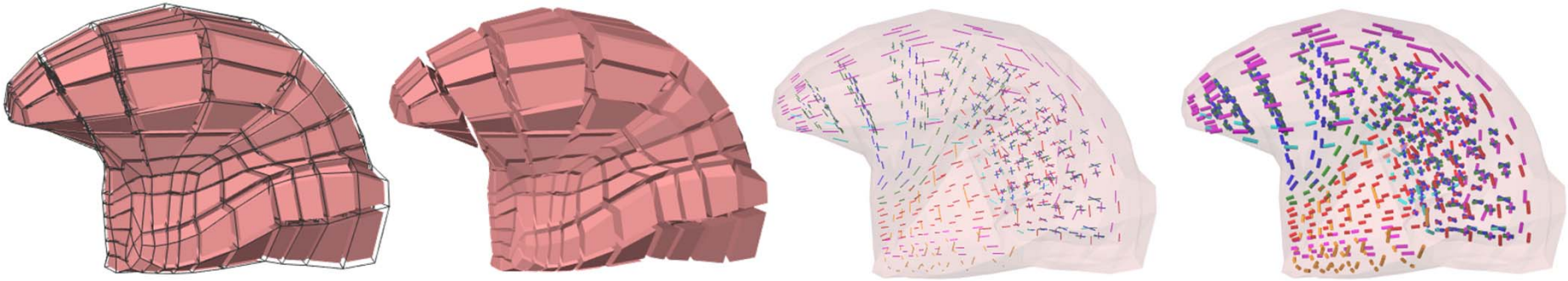
- FemMuscleBeams
  - Fibre muscle
  - Embedded material muscle
- FemMuscleHeart
  - Load FEM from TetGen file
  - Collisions

Visualizations: Rendering

PMHA 2014

- Element Widgets

```
FemModel3d.setElementWidgetSize( size );   // sets the size of elements [0,1]
RenderProps.setLineWidth( fem.getElements(), 0 ); // disable drawing element edges
```



- Muscle fibres

```
FemMuscleModel.setDirectionRenderLen ( size );  // sets the length of fibre lines [0,1]
// Draw fibres as cylinders
RenderProps.setLineStyle ( fem.getMuscleBundles(), LineStyle.CYLINDER );
RenderProps.setLineRadius( fem.getMuscleBundles(), 0.002 );
```

Visualizations: Stress/Strain

Any `FemMesh` (including FEM surface) can show stress/strain

- `FemModel3d.setSurfaceRendering( ... )`
- `FemMesh.setColorRendering( ... )`

Be aware of range / ranging mode

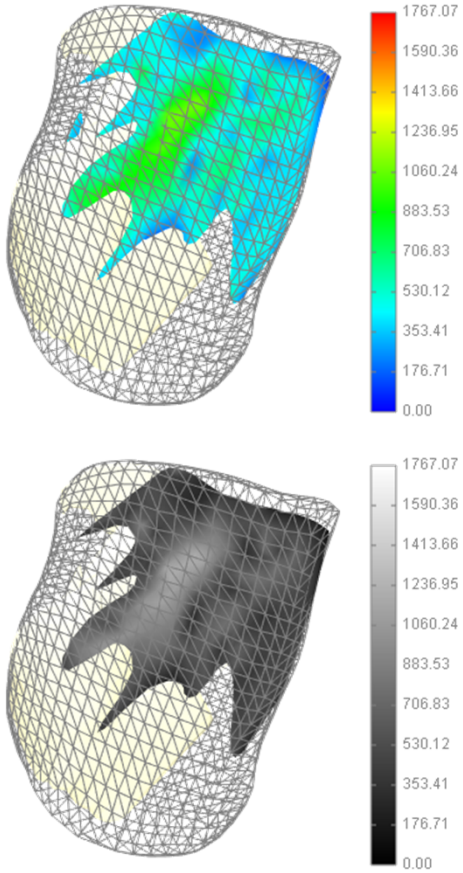
- `FemModel3d.setStressPlotRange( ... )`
- `FemModel3d.setStressPlotRanging( ... )`
  - Fixed or auto-ranging

Colors can be changed using the `ColorMap` property

- `FemModel3d.setColorMap( ColorMap )`
- `FemMesh.setColorMap( ColorMap )`

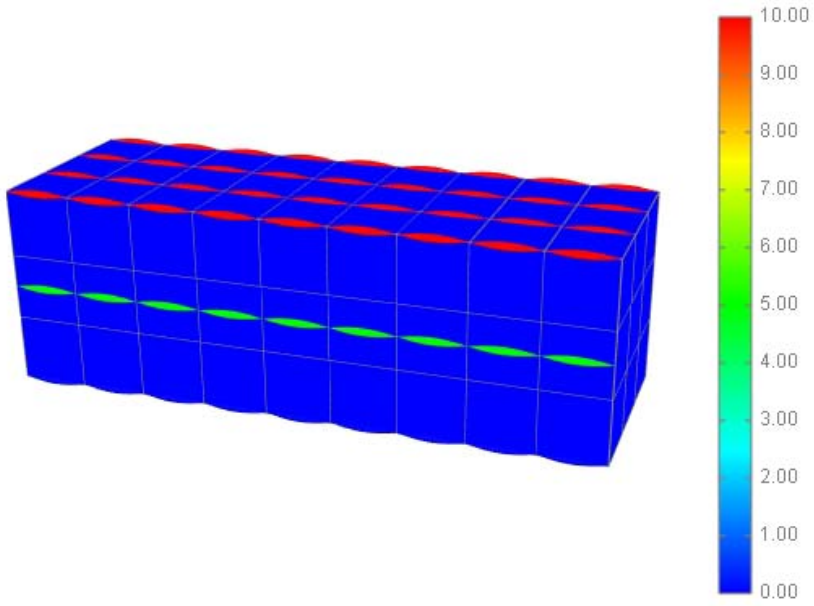
A `ColorBar` can be added to the `RootModel`

- Your responsibililty to synchronize colors/labels
- `RootModel.addRenderable( ... )`



Visualizations: Stress/Strain


```
ColorBar colorBar = new ColorBar();
colorBar.populateLabels(0, 10.0, 10);   // initial labels
colorBar.setNumberFormatString("%.2f"); // 2 decimal places

// Set location to RHS (x < 0 from right)
// Non-zero entries override normalized positions/lengths
colorBar.setLocationOverride(-80, 0, 20, 0);
addRenderable(colorBar);

// Initialize stress rendering
beamFibres.setSurfaceRendering(SurfaceRender.Stress);
beamFibres.setStressPlotRange(new DoubleInterval(0, 10));
...

@Override
public void prerender(RenderList list) {
   super.prerender(list);

   // Synchronize colorbar
   FemMesh surface = beam.getMesh("surface");
   colorBar.setColorMap(surface.getColorMap());
   DoubleInterval dval = surface.getStressPlotRange();
   colorBar.updateLabels(
      dval.getLowerBound(), dval.getUpperBound());
}
```

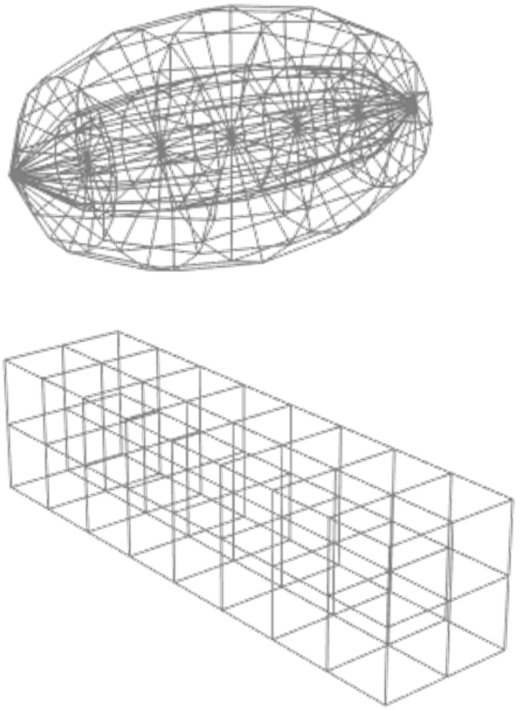


EXTRA SLIDES

FEM Creation: FemFactory

PMHA 2014

```
// Create FEM ellipsoid
FemModel3d ellipsoid = new FemModel3d("ellipsoid");

// FEM Setup
double[] radii = {0.4, 0.2, 0.2}; // radii (z, x, y)
int[] eres = {16, 4, 3};          // resolution (theta, phi, r)

// Create ellipsoid
FemFactory.createEllipsoid(ellipsoid,
   radii[0], radii[1], radii[2],
   eres[0], eres[1], eres[2]);

// Transform: rotate 90 degrees about X-Axis
//            translate up by 0.6
RigidTransform3d trans = new RigidTransform3d();
trans.setRotation(new AxisAngle(1, 0, 0, Math.PI/2));
trans.setTranslation(new Vector3d(0, 0, 0.6));
ellipsoid.transformGeometry(trans);

// Add FEM to model
mech.addModel(ellipsoid);
```



FEM Creation: Surface Geometry

PMHA 2014

Generate FEM given a triangular, closed, manifold polygonal surface:

- Interfaces with TetGen
- `FemFactory.createFromMesh(...)`

Supported polygon files:

- Wavefront (.obj)
- Stereolithography (.stl)
- Stanford polygon (.ply)
- Object file format (.off)

TetGen quality:

- maximum radius-edge ratio
- default: 2.0
- 'good' quality : 1.3

```
// Polygonal surface file
PolygonalMesh surface = new PolygonalMesh();

// Read surface file
try {
   String surfaceFileName = ArtisynthPath.getSrcRelativePath(this,
      "data/tongue.obj");
   WavefrontReader wfr = new WavefrontReader(surfaceFileName);
   wfr.readMesh(surface);

   surface.triangulate(); // triangulate

} catch (IOException ioe) {
   throw new RuntimeException("Failed to load surface", ioe);
}

// Generate FEM
try {
   double quality = 1.2;
   FemFactory.createFromMesh(fem, surface, quality);
} catch (Exception e) {
   throw new RuntimeException("Failed to generate FEM", e);
}
```