Leveraging web technologies for rapid visual debugging

JOWOG 34 – Applied Computer Science

February 3, 2015

Weiss, Kenneth

Lawrence Livermore National Laboratory

LLNL-PRES-666762

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Debugging mesh problems

State of the art

Need to locate the problem

1. Logging
   - Sprinkle `printf` statements throughout code
   - Compile, run, inspect, repeat, …, [clean up]
2. VisIt/Paraview
   - Load entire mesh / set of meshes
   - Filter, focus, render, …
   - Tool might not support your new feature

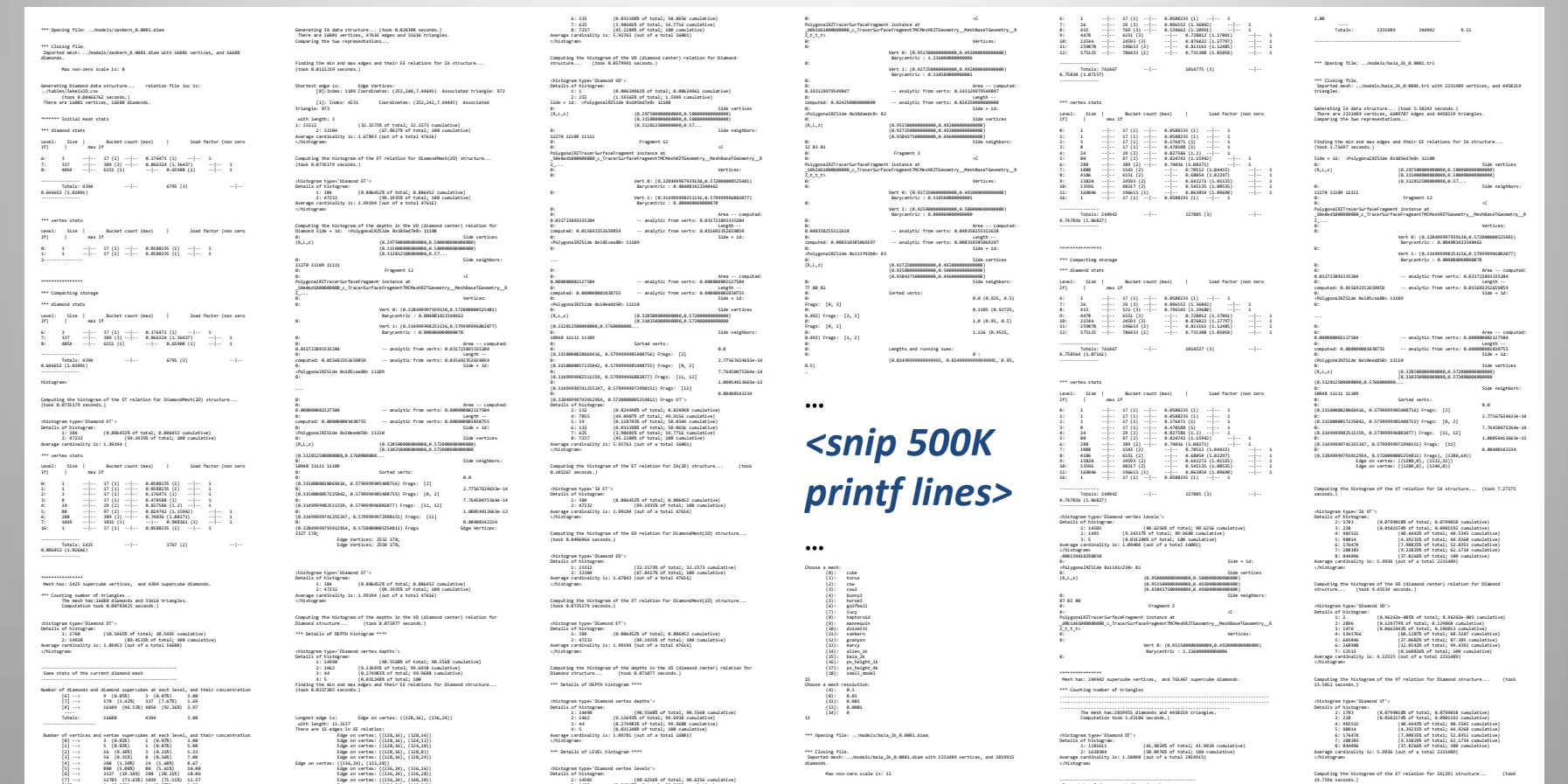
Difficult and time consuming for users and developers

Add lots of `printf` statements…

```
...
0:              Side + id:  <PolygonalRZSide 0x105ed7e0> 11108
0:              Side vertices (R,L,z)    (0.297500000000000,0.580000000000000)    (0.315000000000000,0.580000000000000)    (0.312812500000000,0.57...
0:              Side neighbors:  11270 11109 11111
0:        Fragment 12
0:              <C PolygonalRZTracerSurfaceFragment instance at _50e0ed1000000000_c_TracerSurfaceFragmentTMCMeshRZTGeometry__MeshBaseTGeometry__RZ_...
0:              Vertices:
0:                      Vert 0: (0.328499997939130,0.572000000525481)     Barycentric : 0.804081422340462
0:                      Vert 1: (0.314999998251136,0.579999996802077)     Barycentric : 0.000000000000078
0:
0:              Area -- computed: 0.031723893335204       -- analytic from verts: 0.031723893335204
0:              Length -- computed: 0.015692352659859     -- analytic from verts: 0.015692352659859
0:              Side + id:  <PolygonalRZSide 0x105cea80> 11109
0:

...

0:
0:              Area -- computed: 0.000000002127504       -- analytic from verts: 0.000000002127504
0:              Length -- computed: 0.000000001030755     -- analytic from verts: 0.000000001030755
0:              Side + id:  <PolygonalRZSide 0x10eedd50> 11110
0:              Side vertices (R,L,z)    (0.328500000000000,0.572000000000000)    (0.310250000000000,0.572000000000000 (0.312812500000000,0.5760000000...
0:              Side neighbors:  10948 11111 11109
0:      Sorted verts:
0:              0.0 (0.3150000028069416, 0.5799999985408756) Frags:  [2]
0:              2.77567624633e-14 (0.3150000057225042, 0.5799999985408755) Frags:  [0, 2]
0:              7.76450475364e-14 (0.3149999982511359, 0.579999996802077) Frags:  [11, 12]
0:              1.00054413663e-13 (0.31499999741255247, 0.5799999972990151) Frags:  [11]
0:              0.80408142234 (0.32849999793912954, 0.5720000005254811) Frags:  [12]
0:              0.80408142234 (0.3284999979391296, 0.5720000005254811) Frags:  [13]
...
```

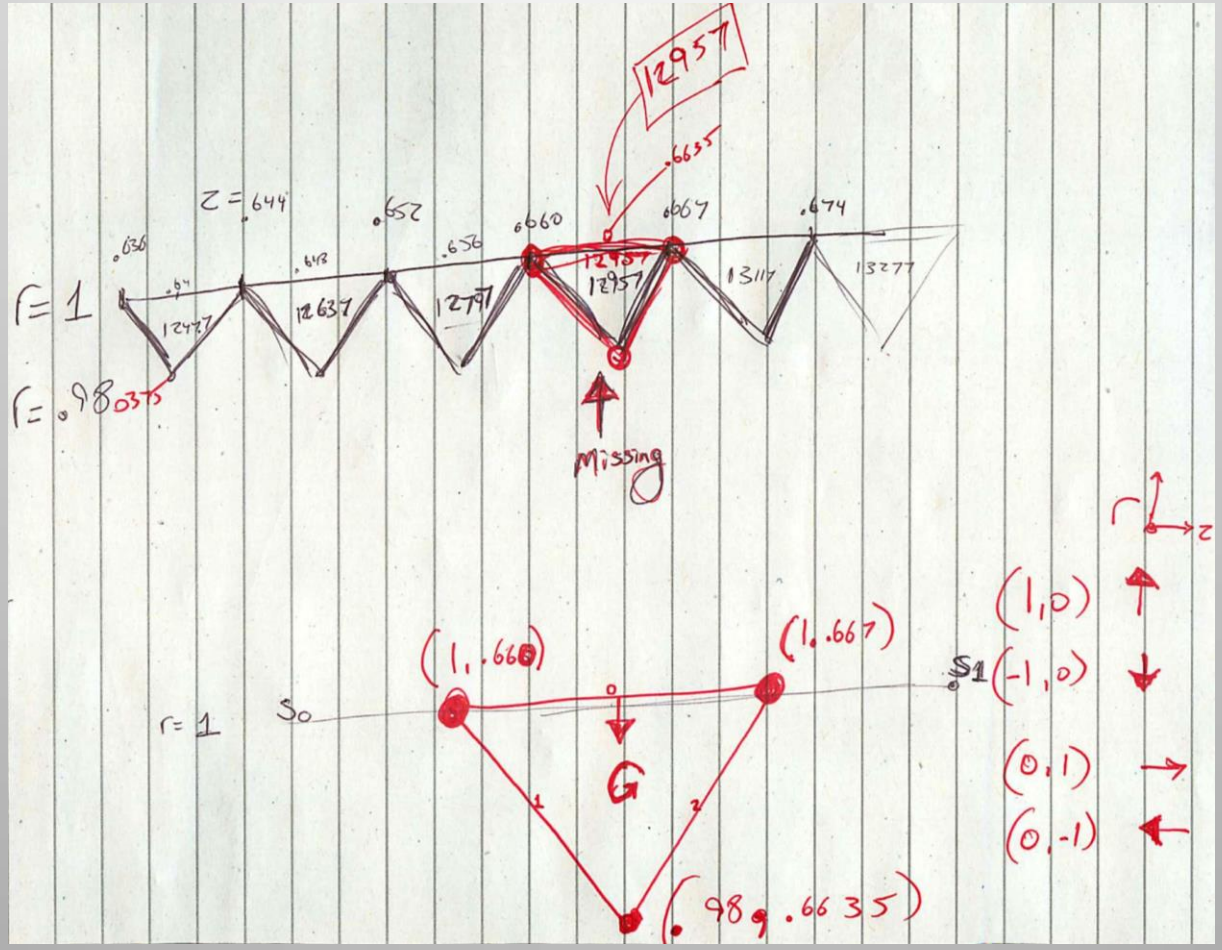
..search for connectivity information

```
...
0:              Side + id:  <PolygonalRZSide 0x105ed7e0> 11108
0:              Side vertices (R,L,z)    (0.297500000000000,0.580000000000000)    (0.315000000000000,0.580000000000000)    (0.312812500000000,0.57...
0:              Side neighbors:  11270 11109 11111
0:        Fragment 12
0:              <C PolygonalRZTracerSurfaceFragment instance at _50e0ed1000000000_c_TracerSurfaceFragmentTMCMeshRZTGeometry__MeshBaseTGeometry__RZ_...
0:              Vertices:
0:                      Vert 0: (0.328499997939130,0.572000000525481)     Barycentric : 0.804081422340462
0:                      Vert 1: (0.314999998251136,0.579999996802077)     Barycentric : 0.000000000000078
0:
0:              Area -- computed: 0.031723893335204       -- analytic from verts: 0.031723893335204
0:              Length -- computed: 0.015692352659859     -- analytic from verts: 0.015692352659859
0:              Side + id:  <PolygonalRZSide 0x105cea80> 11109
0:

...

0:
0:              Area -- computed: 0.000000002127504       -- analytic from verts: 0.000000002127504
0:              Length -- computed: 0.000000001030755     -- analytic from verts: 0.000000001030755
0:              Side + id:  <PolygonalRZSide 0x10eedd50> 11110
0:              Side vertices (R,L,z)    (0.328500000000000,0.572000000000000)    (0.310250000000000,0.572000000000000 (0.312812500000000,0.5760000000...
0:              Side neighbors:  10948 11111 11109
0:      Sorted verts:
0:              0.0 (0.3150000028069416, 0.5799999985408756) Frags:  [2]
0:              2.77567624633e-14 (0.3150000057225042, 0.5799999985408755) Frags:  [0, 2]
0:              7.76450475364e-14 (0.3149999982511359, 0.579999996802077) Frags:  [11, 12]
0:              1.00054413663e-13 (0.31499999741255247, 0.5799999972990151) Frags:  [11]
0:              0.80408142234 (0.32849999793912954, 0.5720000005254811) Frags:  [12]
0:              0.80408142234 (0.3284999979391296, 0.5720000005254811) Frags:  [13]
...
```

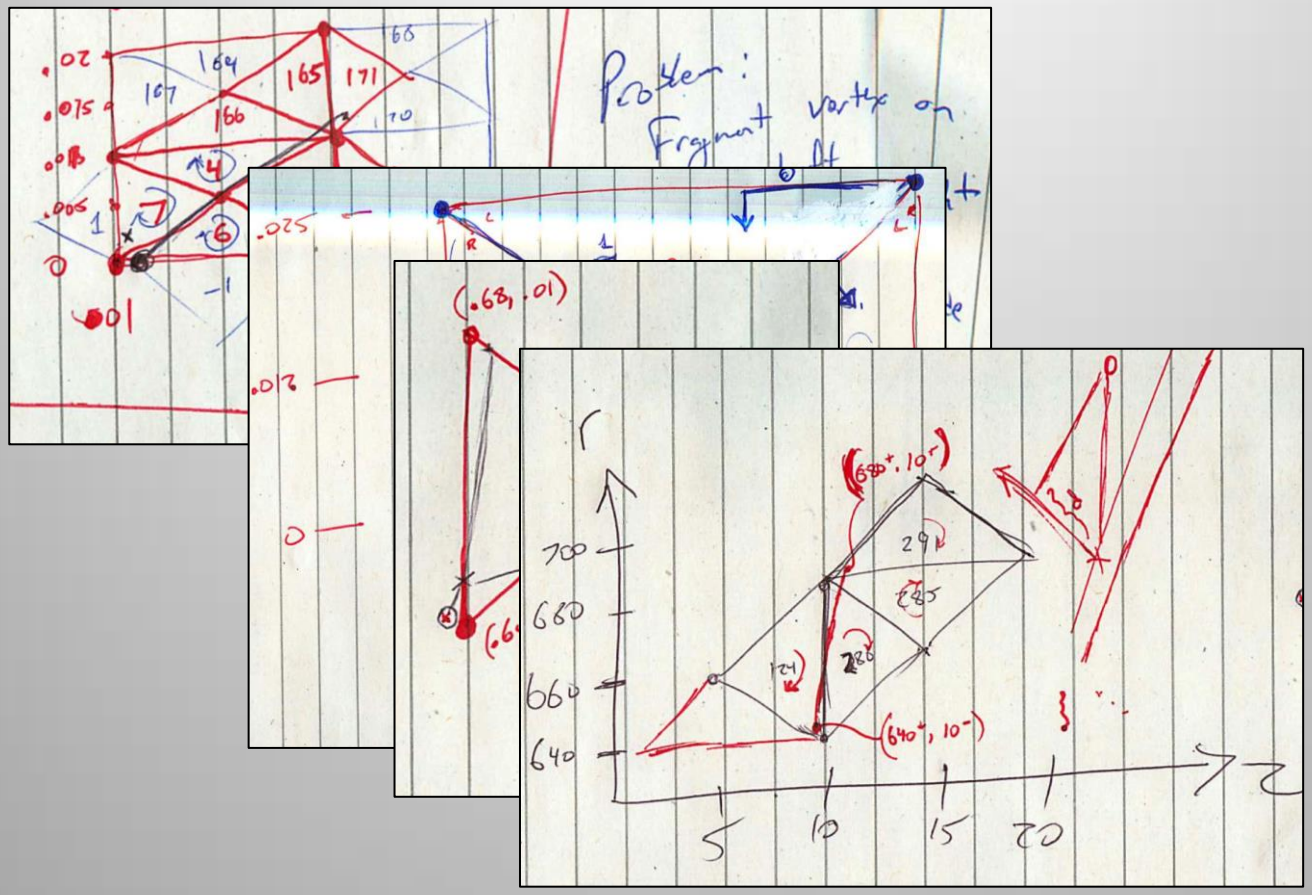


<snip 500K
printf lines>

Reconstruct local mesh (by hand)



Mesh changed … start again



Lightweight visual debugging tool

- Output small subset of local mesh from simulation
  - with desired geometric and topological features
  - simple text based representation of local mesh
- Leverage existing web technologies to visualize and interact with this information
  - reconstruct local geometry and connectivity
  - ‘free’ GUI
    - [practically] no new tool to maintain
    - firefox available on all platforms

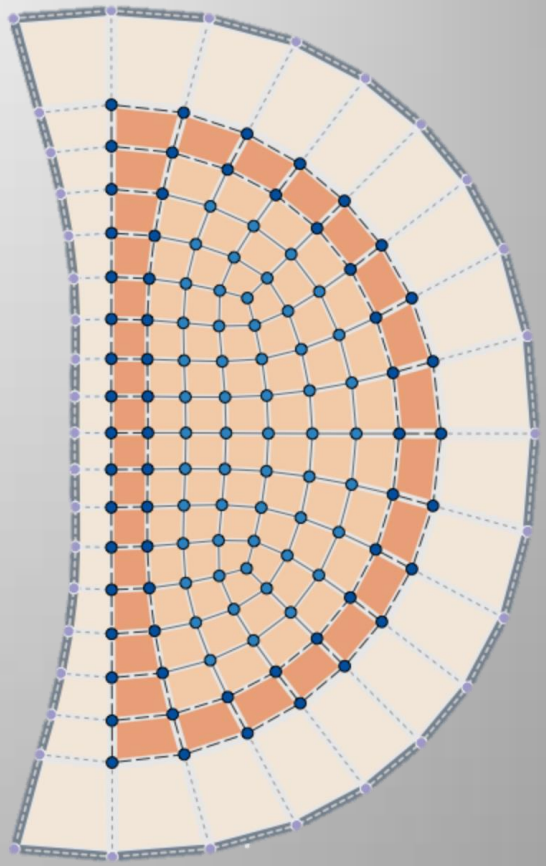
Text output mesh (JSON format)


```
{
        "zones": [
                {"idx":0, "position":{"x":0.027,"y":0.205}, "classification":["owned","send"], "domainID":0,
                 "connectivity":{"n":[3,24,53,12],"e":[0,1,2,3],"f":[0,1,2,3],"s":[0,1,2,3],"c":[0,1,2,3],"z_n":[1,4,5,32,36,96,97,104]}},
                {"idx":1, "position":{"x":0.083,"y":0.201}, "classification":["owned"], "domainID":0,
                 "connectivity":{"n":[12,53,54,13],"e":[2,4,5,6],"f":[4,5,6,7],"s":[4,5,6,7],"c":[4,5,6,7],"z_n":[0,2,4,5,6,32,36,40]}},
                {"idx":2, "position":{"x":0.137,"y":0.194}, "classification":["owned"], "domainID":0,
                 "connectivity":{"n":[13,54,55,14],"e":[5,7,8,9],"f":[8,9,10,11],"s":[8,9,10,11],"c":[8,9,10,11],"z_n":[1,3,5,6,7,36,40,44]} },
        ],
        "nodes": [
                {"idx":0, "position":{"x":0.5,"y":0.0}, "classification":["owned","send"], "domainID":0,
                 "connectivity":{"e":[199,200,201,243],"f":[365,504,507],"z":[91,92,112,126],"s":[365,366,504,507],"c":[366,369,451,504],"n_n":[41,52,131,146]} },
                {"idx":1, "position":{"x":0.0004,"y":-0.499}, "classification":["owned","send"], "domainID":0,
                 "connectivity":{"e":[112,115,240,267],"f":[192,500,501],"z":[48,111,124,125],"s":[192,195,501],"c":[192,446,498,501],"n_n":[15,44,145]} },
                {"idx":2, "position":{"x":-0.0005,"y":0.499}, "classification":["owned","send"], "domainID":0,
                 "connectivity":{"e":[86,87,231,251],"f":[468,469],"z":[35,107,116,117],"s":[142,143,469],"c":[143,429,466,469],"n_n":[136,137]} },
        ],
        "edges": [
                {"idx":0, "position":{"x":-0.00059,"y":0.2061}, "classification":["owned","send"], "domainID":0,
                 "connectivity":{"n":[3,24],"f":[0,386],"z":[0,96],"s":[0,386],"c":[0,1,386,387]} },
                {"idx":1, "position":{"x":0.0266,"y":0.1734},   "classification":["owned","send"], "domainID":0,
                 "connectivity":{"n":[24,53],"f":[1,19],"z":[0,4],"s":[1,19],"c":[1,2,19,16]} },
                {"idx":2, "position":{"x":0.05545,"y":0.2040},  "classification":["owned","send"], "domainID":0,
                 "connectivity":{"n":[12,53],"f":[2,4],"z":[0,1],"s":[2,4],"c":[2,3,4,5]} },
        ],

        "views": [
                {"name":"ownedView", "xMin":-0.000856344631497, "yMin":-0.499999773836, "xMax":0.5, "yMax":0.499999651374 },
                {"name":"domainView", "xMin":-0.149575632365, "yMin":-0.643248713534, "xMax":0.643076796639, "yMax":0.643232259745},
                {"name":"problemView", "xMin":-1.0, "yMin":-1.0, "xMax":1.0, "yMax":1.0}
        ]
}
```


- Output is arrays of javascript objects (JSON)
- Uses desired set of properties with actual mesh values
- Bonus: can `diff` against when changing mesh features

Visual output

- Transform JSON to 2D vector graphics (SVG) in browser
  - d3 visualization library (javascript)
  - Custom functions (javascript)
  - Mesh elements styled in CSS
- View and interact in browser
  - Pan, zoom, shrink, hover to get details



Transforming the mesh
with the d3 javascript library

```
//// Functions to setup the various element types
function setupZones(){

    // Generate the zones (closed polyline paths)
    d3.select("#zoneGroup").selectAll(".zone")
        .data( visualCMI.model.zones ).enter()
        .append("path")
          .attr("id", function(d,i) { return elementID("z", d.idx, d.globalID); } )
          .attr("class", function(d) { return classificationClasses("zone", d); })
          .attr("d", function(d) {  return svgPathFunction(d, true); } )
          .attr("title", function(d,i) { return generateElementTooltip(d,i,"Zone"); })
          .on("mouseover", function(d,i) { onElementHover(d,i,"Zone"); } )
          .on("mouseout", function(d) { onElementUnhover(d); })
          .on("click", function(d,i) { onElementClick(d,i,"Zone", this); })
        ;

    addElementsToMap("z", visualCMI.model.zones);
}
```

```
/*******************
 *    Zone styles   *
 *******************/
.zone.owned {  fill: rgba(253, 174, 107, 0.5);}
.zone.owned.selected {  fill: rgba(43, 43, 43, 0.5);}
.zone.owned.clicked,.zone.owned:hover {  fill: rgba(250, 255, 233, 0.5);}
.zone.owned.send {  fill: rgba(230, 85, 13, 0.5);}
.zone.owned.send.selected {  fill: rgba(0, 0, 0, 0.5);}
.zone.owned.send.clicked,.zone.owned.send:hover {  fill: rgba(237, 248, 123, 0.5);}
.zone.receive {  fill: rgba(254, 230, 206, 0.5);}
.zone.receive.selected {  fill: rgba(142, 142, 142, 0.5);}
.zone.receive.clicked,.zone.receive:hover {  fill: rgba(255, 255, 255, 0.5);}
.zone.externalSurface {  stroke-width: 20;  stroke: rgba(1, 25, 49, 0.5);}
.zone.externalSurface.problemBoundary {  stroke-width: 30;  stroke: rgba(25, 1, 49, 0.7);}
.zone.externalSurface.selected {  stroke: rgba(0, 0, 0, 0.5);}
.zone.externalSurface.clicked,.zone.externalSurface:hover {  stroke: rgba(46, 4, 174, 0.5);}
```

Transforming the mesh
with the d3 javascript library

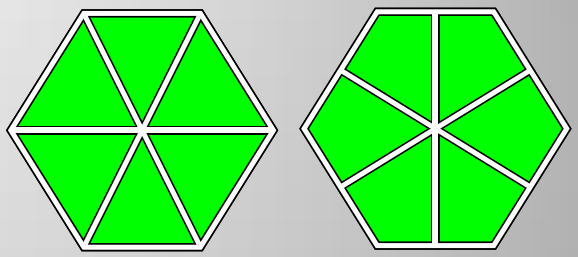
```
//// Functions to setup the various element types
function setupZones(){

    // Generate the zones (closed polyline paths)
    d3.select("#zoneGroup").selectAll(".zone")
        .data( visualCMI.model.zones ).enter()
        .append("path")
          .attr("id", function(d,i) { return elementID("z", d.idx, d.globalID); } )
          .attr("class", function(d) { return classificationClasses("zone", d); })
          .attr("d", function(d) {  return svgPathFunction(d, true); } )
          .attr("title", function(d,i) { return generateElementTooltip(d,i,"Zone"); })
          .on("mouseover", function(d,i) { onElementHover(d,i,"Zone"); } )
          .on("mouseout", function(d) { onElementUnhover(d); })
          .on("click", function(d,i) { onElementClick(d,i,"Zone", this); })
        ;

    addElement
}
```

```
<path id="z_44_140" class="zone owned send" d="M-8.072351821603153,440.71406464995L-
8.155464771921501,494.4…645366499,483.98271494520003L-86.53911350515,432.4634424669Z" title="[44] Zone 44 GID 140"></path>
<path id="z_45_141" class="zone owned send" d="M-100.858008282985,429.45011408399995L-
115.6476012315,480.96…18529997,453.5614589605L-172.10471292745,407.25834488289996Z" title="[45] Zone 45 GID 141"></path>
<path id="z_46_142" class="zone owned send" d="M-184.6009637059,401.90678954515L-
209.81875896374999,448.209….9911303983,409.3827685343L-246.75533165035,369.03170044495Z" title="[46] Zone 46 GID 142"></path>
<path id="z_47_143" class="zone owned send" d="M-257.3708265445,361.84645096779997L-
288.60662529245,402.197…817584995,354.1397176974L-310.39888855034997,318.7408217025Z" title="[47] Zone 47 GID 143"></path>
<path id="z_48_144" class="zone owned selected" d="M-209.50099524670003,204.22566568825L-
235.326857909,230.0039…060762285,187.4286701895L-219.15716420459998,168.9283878283Z" title="[48] Zone 48 GID 144"></path>
<path id="z_49_145" class="zone owned" d="M-221.9256837954,161.42059801135L-
264.70912721365,179.920880…83919556,130.9189640871L-231.10961110735002,118.11849840985Z" title="[49] Zone 49 GID 145"></path>
<path id="z_50_146" class="zone owned" d="M-233.0447354635,109.62706259701L-
283.94351631175,122.427528….8057448186,69.316291406855L-239.5621336413,62.5697702596452" title="[50] Zone 50 GID 146"></path>
```

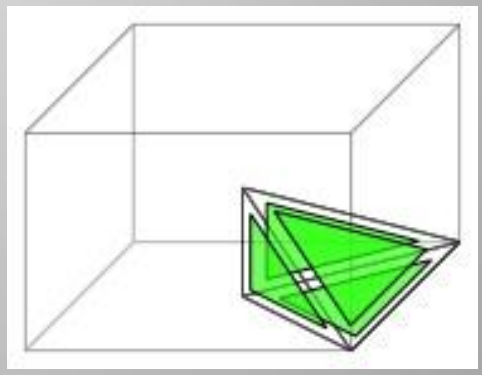
```
/*************
 *    Zone sty
 *************
.zone.owned {
.zone.owned.se
.zone.owned.cl
.zone.owned.se
.zone.owned.se
.zone.owned.se
.zone.receive
.zone.receive.
.zone.receive.clicked,.zone.receive:hover {  fill: rgba(255, 255, 255, 0.5);}
.zone.externalSurface {  stroke-width: 20;  stroke: rgba(1, 25, 49, 0.5);}
.zone.externalSurface.problemBoundary {  stroke-width: 30;  stroke: rgba(25, 1, 49, 0.7);}
.zone.externalSurface.selected {  stroke: rgba(0, 0, 0, 0.5);}
.zone.externalSurface.clicked,.zone.externalSurface:hover {  stroke: rgba(46, 4, 174, 0.5);}
```

Example: Visual CMI

- We have a complicated mesh
  - Unstructured polygonal/polyhedral mesh
  - Subzonal geometry
  - Ghost layers:
    send, receive, external surface
- Difficult to learn
  - Documentation not always consistent w/ implementation
- Visual CMI is a way to interact with our actual mesh
  - …as seen by the code



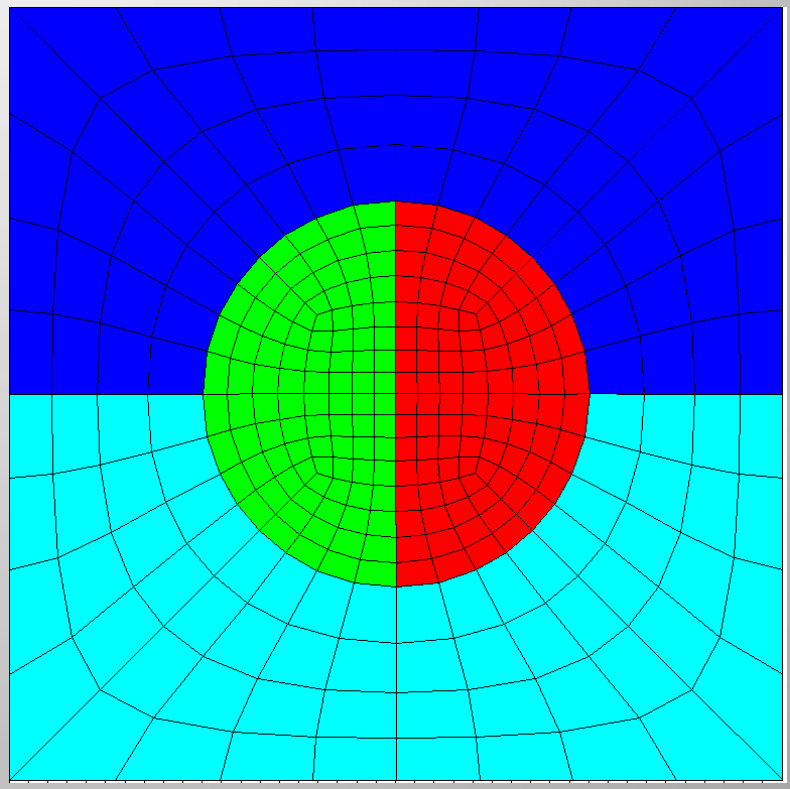
Sides and corners of a hexagonal zone



Facelets of a side of a hexahedral zone

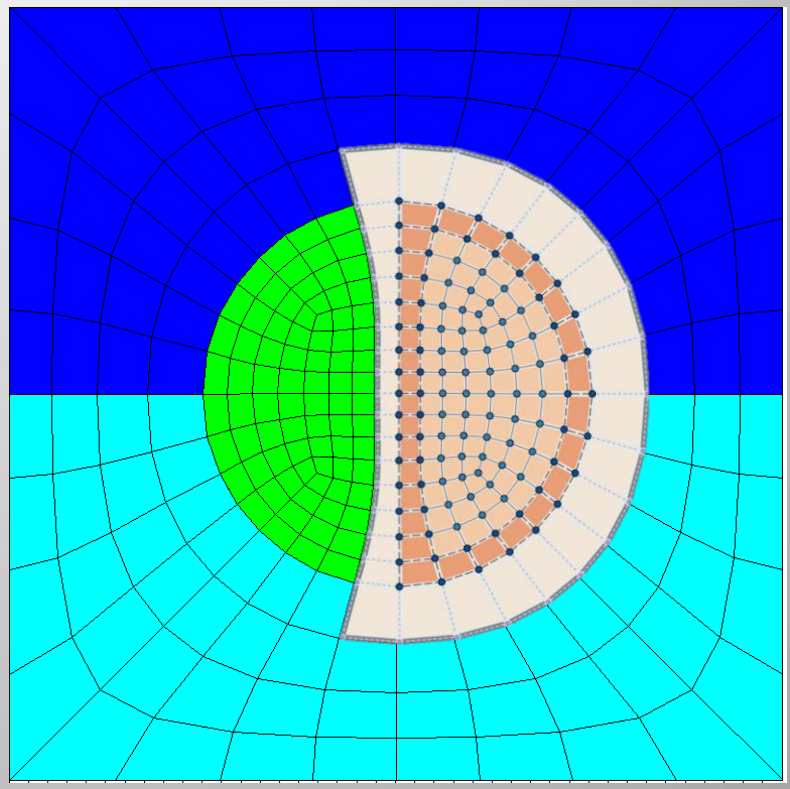
Visual CMI: Demo

- Simple mesh: circle embedded within square
- Four domains
  - Right of circle
  - Left of circle
  - Upper square
  - Lower square



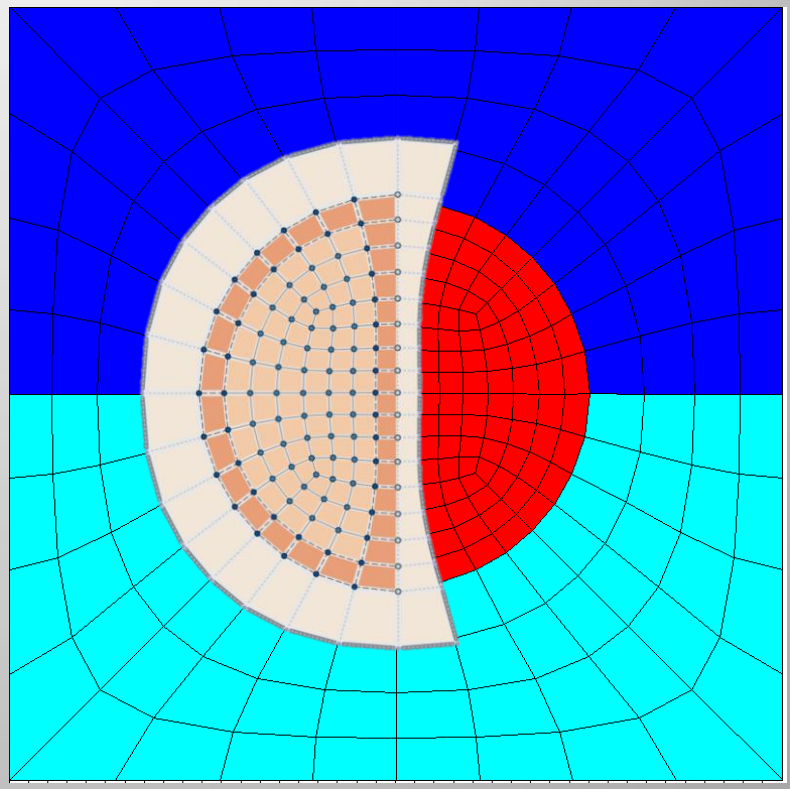
Visual CMI: Demo

- Simple mesh: circle embedded within square
- Four domains
  - Right of circle
  - Left of circle
  - Upper square
  - Lower square



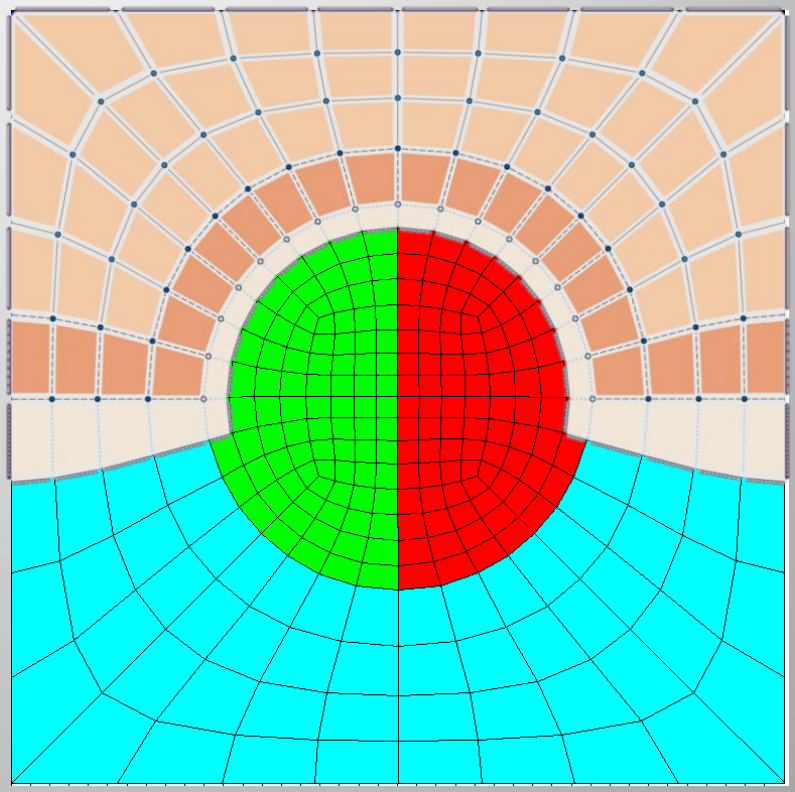
Visual CMI: Demo

- Simple mesh: circle embedded within square
- Four domains
  - Right of circle
  - Left of circle
  - Upper square
  - Lower square



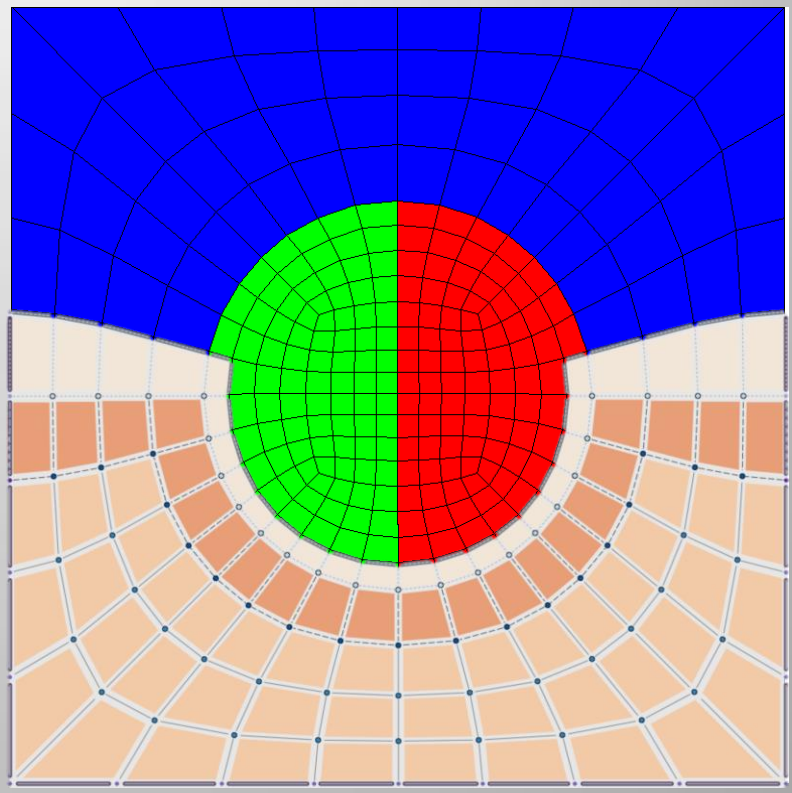
Visual CMI: Demo

- Simple mesh: circle embedded within square
- Four domains
  - Right of circle
  - Left of circle
  - Upper square
  - Lower square



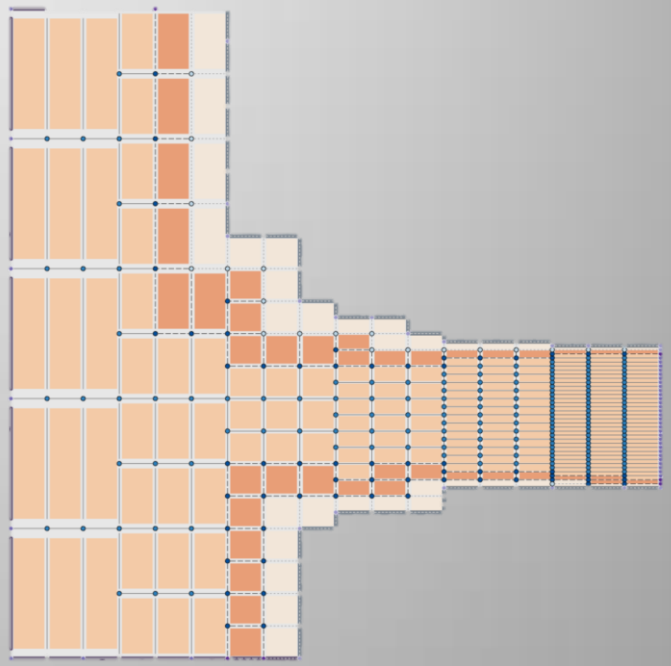
Visual CMI: Demo

- Simple mesh: circle embedded within square
- Four domains
  - Right of circle
  - Left of circle
  - Upper square
  - Lower square



Benefits when modifying mesh API or implementation

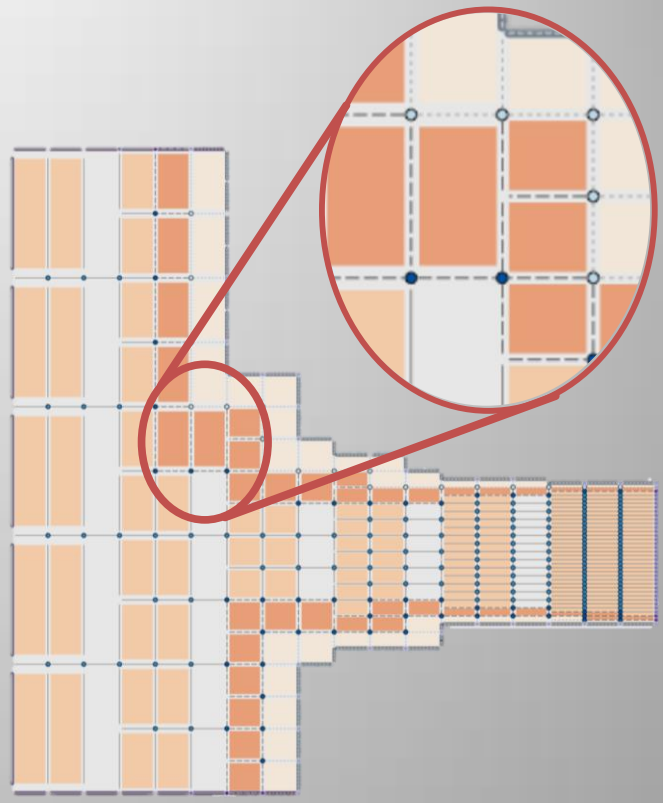
- Add support for dendritic elements to mesh
  - Elements with hanging nodes
    - increase resolution while maintaining good aspect ratios
  - Marked with per element bitflags
- Tool showed problems with ghost elements
  - Logic error in my code!
    - When setting classification tags, I accidentally overwrote all other tags, including dendritic



demo: bug | fixed

Benefits when modifying mesh API or implementation

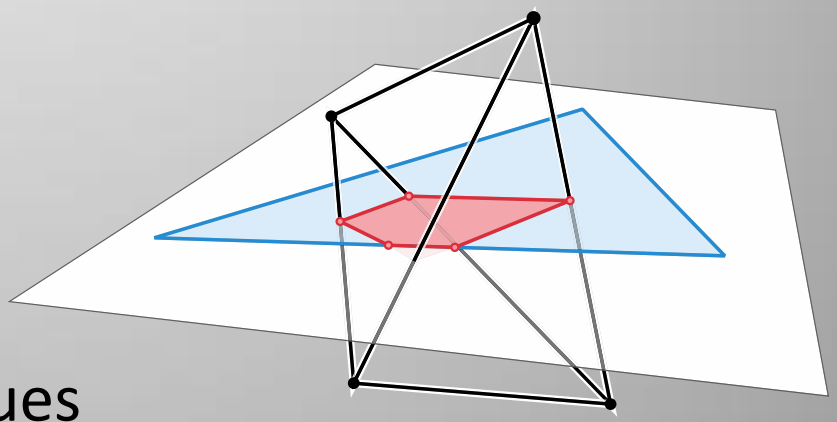
- Add support for dendritic elements to mesh
  - Elements with hanging nodes
    - increase resolution while maintaining good aspect ratios
  - Marked with per element bitflags
- Tool showed problems with ghost elements
  - Logic error in my code!
    - When setting classification tags, I accidentally overwrote all other tags, including dendritic



demo: bug | fixed

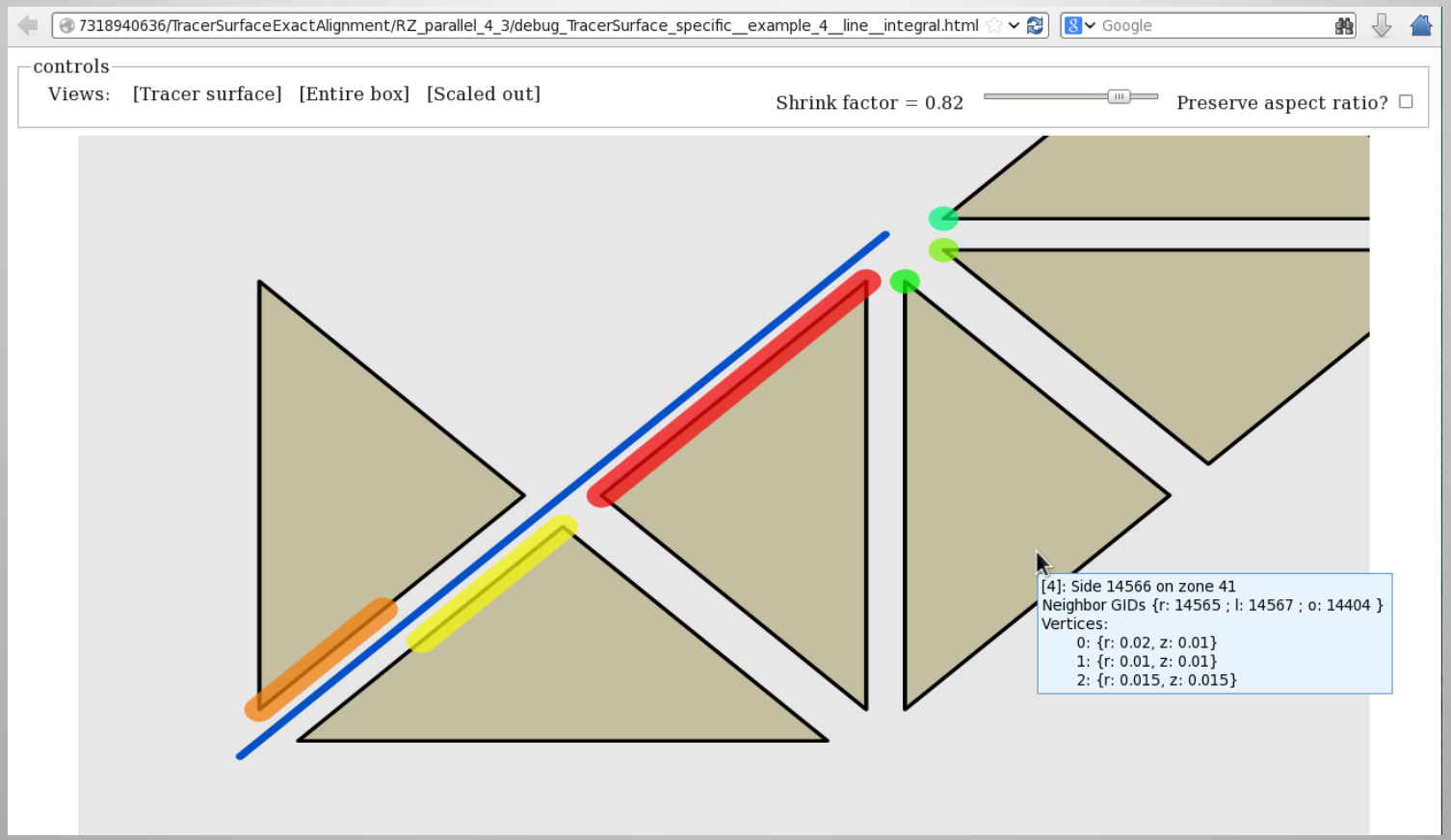
A second example:
Tracer Surfaces

- Project: Improve efficiency of fragmentation algorithm
  - Diagnostic tool measuring flux through dynamic arbitrary surfaces
    - Integrated via decomposition of tracer surface {segments,triangles} into *fragments* intersecting a single mesh element
    - Decomposition based on only local intersections
  - Output mesh contains
    - Set of tracer surface simplices
    - Set of intersected mesh elements
    - Set of tracer surface fragments



- Code had some robustness issues
  - When surface elements almost coincident with mesh edge/face

Tracer Surface visualization demo



Fragmentation demo: problem | fixed

Conclusion

- Lightweight, flexible tool for visual debugging
- KISS
  - Use available tools
  - Don’t duplicate functionality in existing powerful tools
    - Visualization tools: VisIt, paraview, silex
    - Debugging tools: totalview, gdb, …
- Quickly pinpoint local problem in the mesh
  - reconstruct geometry and connectivity
  - along with some field data
- Only for small regions of mesh

Ongoing and future work

- Visualizing tangled zones
  - output problematic zones and all incident neighbors
  - along with field data
  - can be automatically triggered based on local conditions
- Extend tool to handle our regions and field data
- Visualization of polyhedral (3D) meshes using webGL
- Visualize data layout/locality

Thank you.

Questions?

Comments?

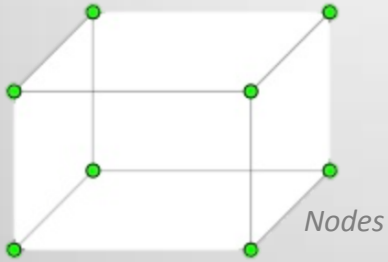
Suggestions?

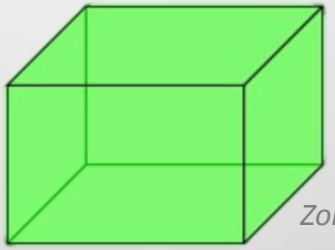
Lawrence Livermore
National Laboratory

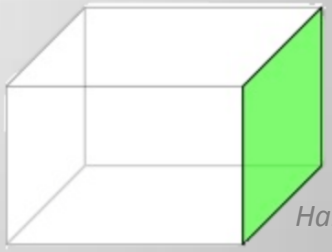
Complicated mesh setup

3D Element Types

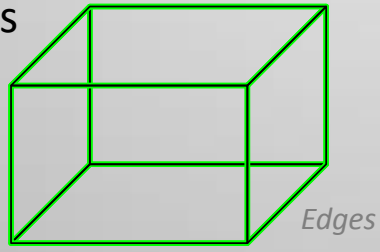
Primary elements

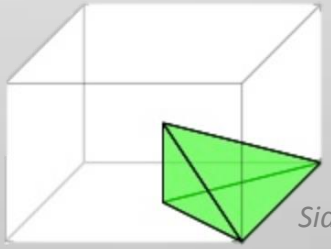


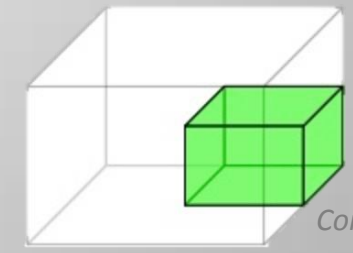




Secondary elements







Tertiary elements