New CAD features, Gmsh 3.0 and API

C. Geuzaine

University of Liège, Belgium

I ADDED ALL OF THE PRODUCT FEATURES THAT EACH OF YOU DEMANDED.
PRODUCT
NOW OUR PRODUCT IS A WORTHLESS HODGEPODGE OF COMPLEXITY.
I APPRECIATE YOUR INPUT. I COULDN'T HAVE FAILED WITHOUT YOU.
TEAM-WORK!
Dilbert.com DilbertCartoonist@gmail.com
2-25-13 ©2013 Scott Adams, Inc./Dist. by Universal Uclick

First, a little bit of history

- Gmsh was started in 1996, as a toy project by JF and myself
- First public release in 1998
- Open Source (GNU GPL) in 2003
- Gmsh 2.0 in 2006
- IJNME paper and switch to CMake in 2009
- Python wrappers in 2011
- Curvilinear meshing: 2012–
- Quad-Hex meshing: 2012–
- Homology: 2013–
- Parallel meshing: 2015–
- New CAD interface and switch to Git/Gitlab: 2017

First, a little bit of history

- Today, Gmsh represents about 400k lines of C++ code
  - still same 2 core developers; about 100 with >= 1 commit
  - about 1,000 people on mailing lists
  - about 10,000 downloads per month (75% Windows)
  - about 400 citations per year – the Gmsh paper is cited about 2,200 times
  - Gmsh probably one of the most popular open source finite element mesh generator in the world?

First, a little bit of history

First workshop held in Braives, September 15-16, 2011

https://gitlab.onelab.info/gmsh/gmsh/wikis/FirstGmshWorkshop

First, a little bit of history

Second workshop held in Moressée, May 23-24, 2013

https://gitlab.onelab.info/gmsh/gmsh/wikis/SecondGmshWorkshop

First, a little bit of history

New CAD features

- The whole internal CAD interface has been rewriten over the last 2 months
  - `GEO_Internals` interfaces all the legacy features of the built-in Gmsh CAD engine
  - `OCC_Internals` interfaces top-level OpenCASCADE features, including boolean operations
- All CAD operations are handled internally in these classes; the only function that interacts with GModel is `synchronize(GModel *)`
- The .geo file parser has been rewritten to only use the `GEO_Internal/OCC_Internal` API
- This API does not expose any structures or classes: only int, double, string, pair and vector

New CAD features

```
class GEO_Internals{
 public:
  // add shapes
  bool addVertex(int num, double x, double y, double z, double lc);
  bool addLine(int num, int startTag, int endTag);
  bool addCircleArc(int num, int startTag, int centerTag, int EndTag,
                    double nx=0., double ny=0., double nz=0.);
  bool addBSpline(int num, const std::vector<int> &vertexTags);
  bool addLineLoop(int num, const std::vector<int> &edgeTags);
  bool addPlaneSurface(int num, const std::vector<int> &wireTags);
  ...
  // extrude and revolve
  bool extrude(const std::vector<std::pair<int, int> > &inDimTags,
               double dx, double dy, double dz,
               std::vector<std::pair<int, int> > &outDimTags);
  ...
  // apply transformations
  bool translate(const std::vector<std::pair<int, int> > &dimTags,
                 double dx, double dy, double dz);
  ...
  // remove
  bool remove(const std::vector<std::pair<int, int> > &dimTags, bool recursive=false);
  ...
  // synchronize internal CAD data with the given GModel
  void synchronize(GModel *model);
  ...
}
```

New CAD features

```
class OCC_Internals{
 public:
  // add shapes
  bool addVertex(int tag, double x, double y, double z, double meshSize=MAX_LC);
  bool addLine(int tag, int startTag, int endTag);
  ...
  bool addSphere(int tag, double xc, double yc, double zc, double radius,
                 double angle1, double angle2, double angle3);
  bool addThruSections(int tag, const std::vector<int> &wireTags,
                       std::vector<std::pair<int, int> > &outDimTags,
                       bool makeSolid, bool makeRuled);
  ...
  // extrude and revolve
  bool extrude(const std::vector<std::pair<int, int> > &inDimTags,
               double dx, double dy, double dz,
               std::vector<std::pair<int, int> > &outDimTags);
  ...
  // apply boolean operator
  bool applyBooleanOperator(int tag, BooleanOperator op,
                            const std::vector<std::pair<int, int> > &objectDimTags,
                            const std::vector<std::pair<int, int> > &toolDimTags,
                            std::vector<std::pair<int, int> > &outDimTags,
                            bool removeObject, bool removeTool);
  ...
  // synchronize internal CAD data with the given GModel
  void synchronize(GModel *model);
}
```

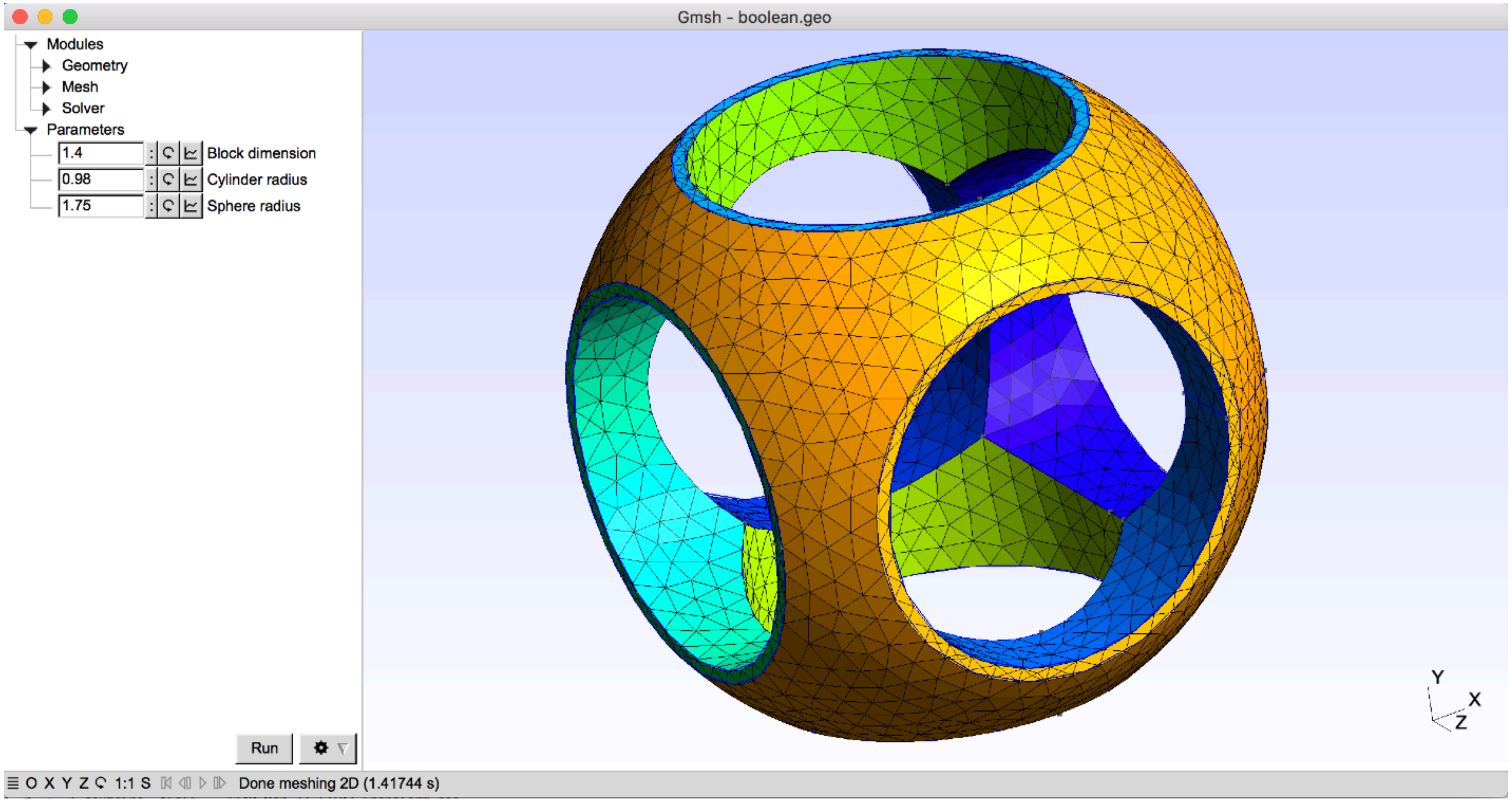
New CAD features

- Some meshing constraints are stored in `{GEO,OCC}_Internals` (they must can be copied around), but most have been moved directly to `GModel`
- `GEO_Internals` is always allocated; `OCC_Internals` is only created on-demand (even if Gmsh is compiled with OCC support)
- In .geo files
  - Switch between internal CAD modelers: `SetFactory("name")`
    - Currently: `"Built-in"` or `"OpenCASCADE"`
  - Force a synchronization: `SyncModel`
  - By default: `"Built-in"`
  - 99.99% backward compatible with existing .geo files

New CAD features

- All existing .geo commands are conserved
- New or modified .geo commands:
  - Shapes (with explicit numbering): `Circle, Ellipse, Wire, Surface, Sphere, Block, Torus, Rectangle, Disk, Cylinder, Cone, Wedge, ThickSolid, ThruSections, Ruled ThruSections`
  - Operations (implicit numbering): `ThruSections, Ruled ThruSections, Fillet, Extrude`
  - Boolean operations (explicit or implicit numbering): `BooleanUnion, BooleanIntersection, BooleanDifference, BooleanFragments`
  - Other: `ShapeFromFile, Recursive Delete, In Surface, In Volume`

New CAD features


Gmsh - boolean.geo

1.4 Block dimension
0.98 Cylinder radius
1.75 Sphere radius

Done meshing 2D (1.41744 s)

http://gitlab.onelab.info/gmsh/gmsh/tree/master/demos/boolean/boolean.geo

New CAD features

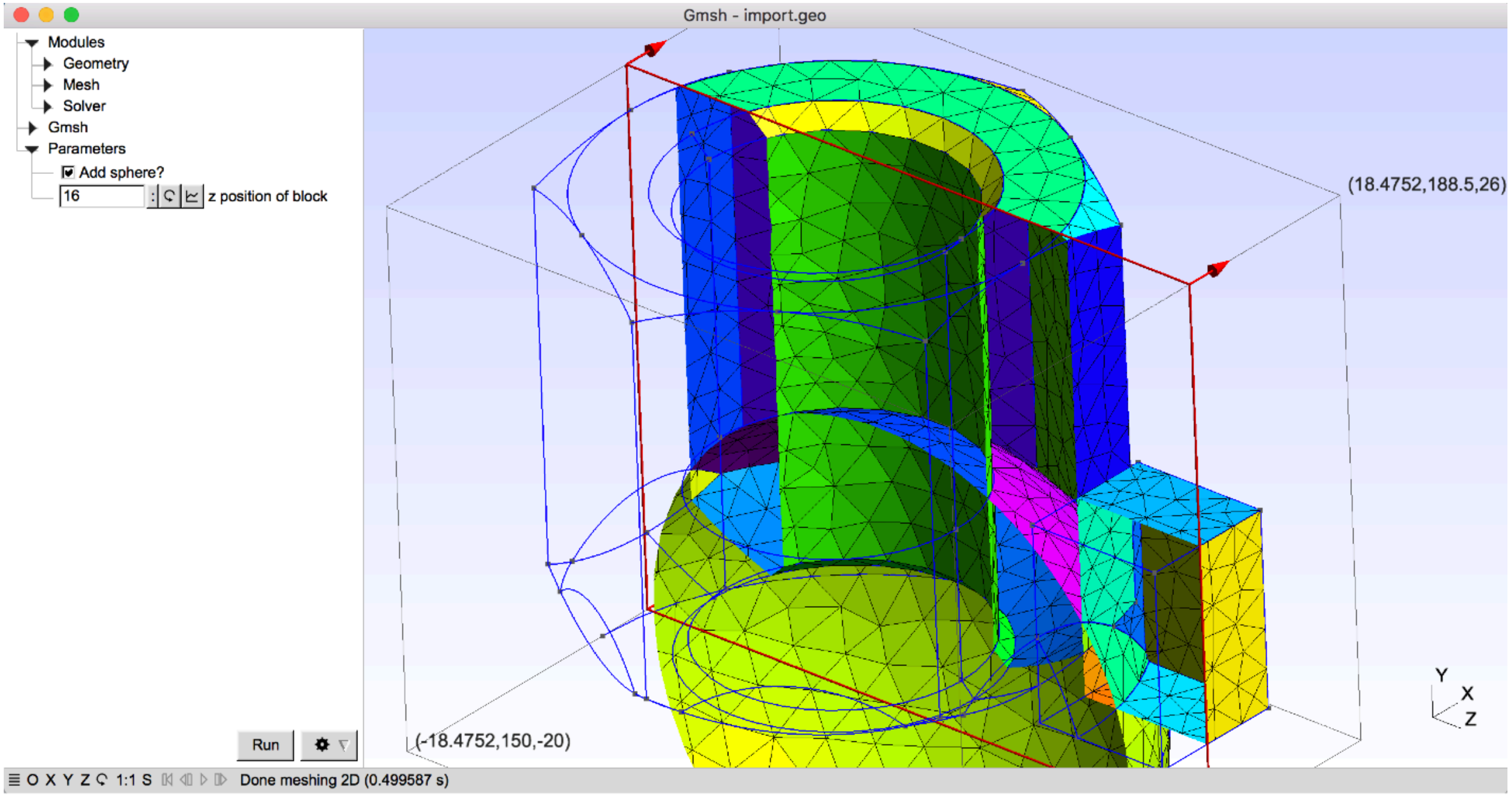
Set OpenCASCADE internal CAD modeler

```
SetFactory("OpenCASCADE");

// from http://en.wikipedia.org/wiki/Constructive_solid_geometry

R = DefineNumber[ 1.4 , Min 0.1, Max 2, Step 0.01, Name "Parameters/Block dimension" ];
Rs = DefineNumber[ R*.7 , Min 0.1, Max 2, Step 0.01, Name "Parameters/Cylinder radius" ];
Rt = DefineNumber[ R*1.25, Min 0.1, Max 2, Step 0.01, Name "Parameters/Sphere radius" ];

Block(1) = {-R,-R,-R, R,R,R};

Sphere(2) = {0,0,0,Rt};

BooleanIntersection(3) = { Volume{1}; Delete; }{ Volume{2}; Delete; };

Cylinder(4) = {-2*R,0,0, 2*R,0,0, Rs};
Cylinder(5) = {0,-2*R,0, 0,2*R,0, Rs};
Cylinder(6) = {0,0,-2*R, 0,0,2*R, Rs};

BooleanUnion(7) = { Volume{4}; Delete; }{ Volume{5,6}; Delete; };
BooleanDifference(8) = { Volume{3}; Delete; }{ Volume{7}; Delete; };
```

Explicit tags

Delete object and tool

New CAD features



Gmsh - import.geo

Done meshing 2D (0.499587 s)

http://gitlab.onelab.info/gmsh/gmsh/tree/master/demos/boolean/import.geo

New CAD features

```
SetFactory("OpenCASCADE");

DefineConstant[
  z = {16, Name "Parameters/z position of block"}
  sph = {0, Choices{0,1}, Name "Parameters/Add sphere?"}
];

a() = ShapeFromFile("component8.step");
b() = 2;
Block(b(0)) = {0,156,z, 10,170,z+10};

If(sph)
  b() += 3;
  Sphere(b(1)) = {0,150,0, 20};
EndIf

r() = BooleanFragments{ Volume{a()}; Delete; }{ Volume{b()}; Delete; };
Save "merged.brep";

Physical Volume("Combined volume", 1) = {r()};
Physical Surface("Combined boundary", 2) = CombinedBoundary{ Volume{r()}; }
```

Import (highest dim) shapes from STEP or BREP

BooleanFragments intersects everything

Implicit tags

New CAD features

- More examples on http://gitlab.onelab.info/gmsh/gmsh/tree/master/demos/boolean
- OpenCASCADE features require OCCT >= 6.9
  - Recommended: OCCT 7.1
  - Multi-threaded 2D mesh generation OK (configure Gmsh with `-DENABLE_OPENMP=1`)
  - Multi-threaded boolean operations (test with `Geometry.OCCParallel=1;`)
  - Fuzzy boolean operations (`Geometry.ToleranceBoolean`)
- Let's use the Git+Debug session this afternoon to get everyone up-to-date

New CAD features

- Other recent features in .geo files:
  - structures (contributed by Patrick); still experimental, but examples avalailable on

    http://gitlab.onelab.info/gmsh/gmsh/tree/master/demos/struct
  - `Unique` operator
  - Ruled Surface -> Surface
- Other recent changes pushed in the master branch
  - Faster 3D tetrahedral mesh optimization enabled by default

Gmsh 3.0

- I would like to release Gmsh 3.0 with these new features right after the workshop
- Intensive debug session this afternoon!
- … and introduction to Git by Jon right after this talk
- I would wait for Gmsh 3.1 (or 4.0) to break things:
  - new reparametrization code from Pierre-Alexandre (cf. his talk tomorrow)… and removal of the old one
  - new initial 3D mesher (collaboration with Hang Si): still some small bugs: please test with `Mesh.Algorithm3D=2;`
  - new 3D refinement code: still some bugs… but much faster: please test with `Mesh.OldRefinement=0;`

API

- I would like to export a C-type API for the new CAD interface, the meshing algorithms (and combine this with HXT) and the post-pro
  - should be purely functional (we should decide on a naming convention)
  - should only use standard types (int, double, vector, pair, map) — no pointers, no classes, no structs
  - should be documented and could be wrapped in *whatever* (Python, Java, Julia, …)
  - dynamic libraries exposing this API should be made available
- The current Python/Java wrappers should be reserved for Gmsh developers, and clearly marked as such; they should not be distributed by default (e.g. in Debian)

Thank you!