TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

BRL-CAD: An Open Source Perspective

MIL-OSS Working Group 2, August 2010

Mr. Sean Morrison
morrison@arl.army.mil
410-278-6678

Approved for public release; distribution unlimited.

Main Topics

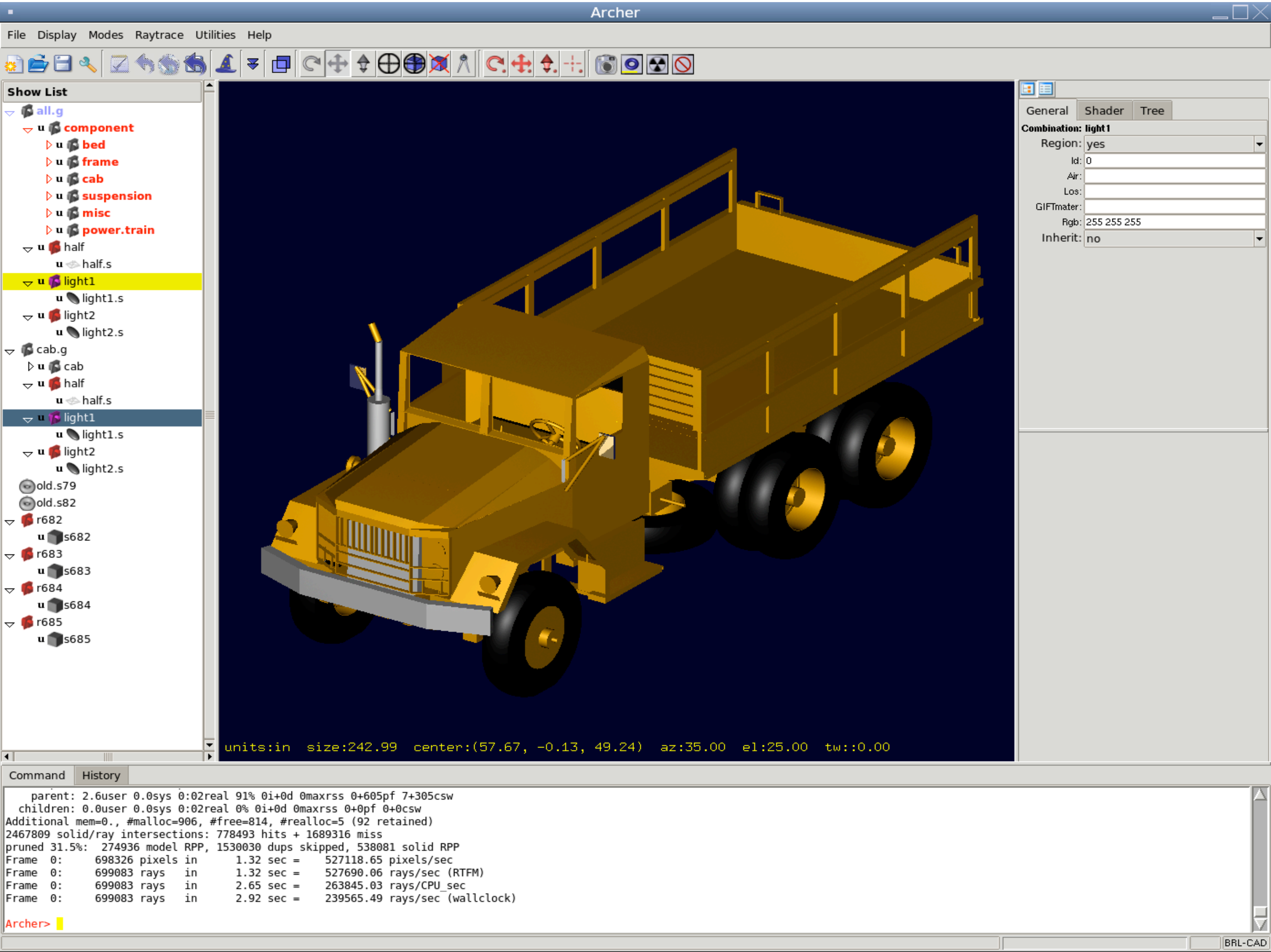
Background on BRL-CAD
What, who, why?

Tools & Techniques
for Geometry Analysis

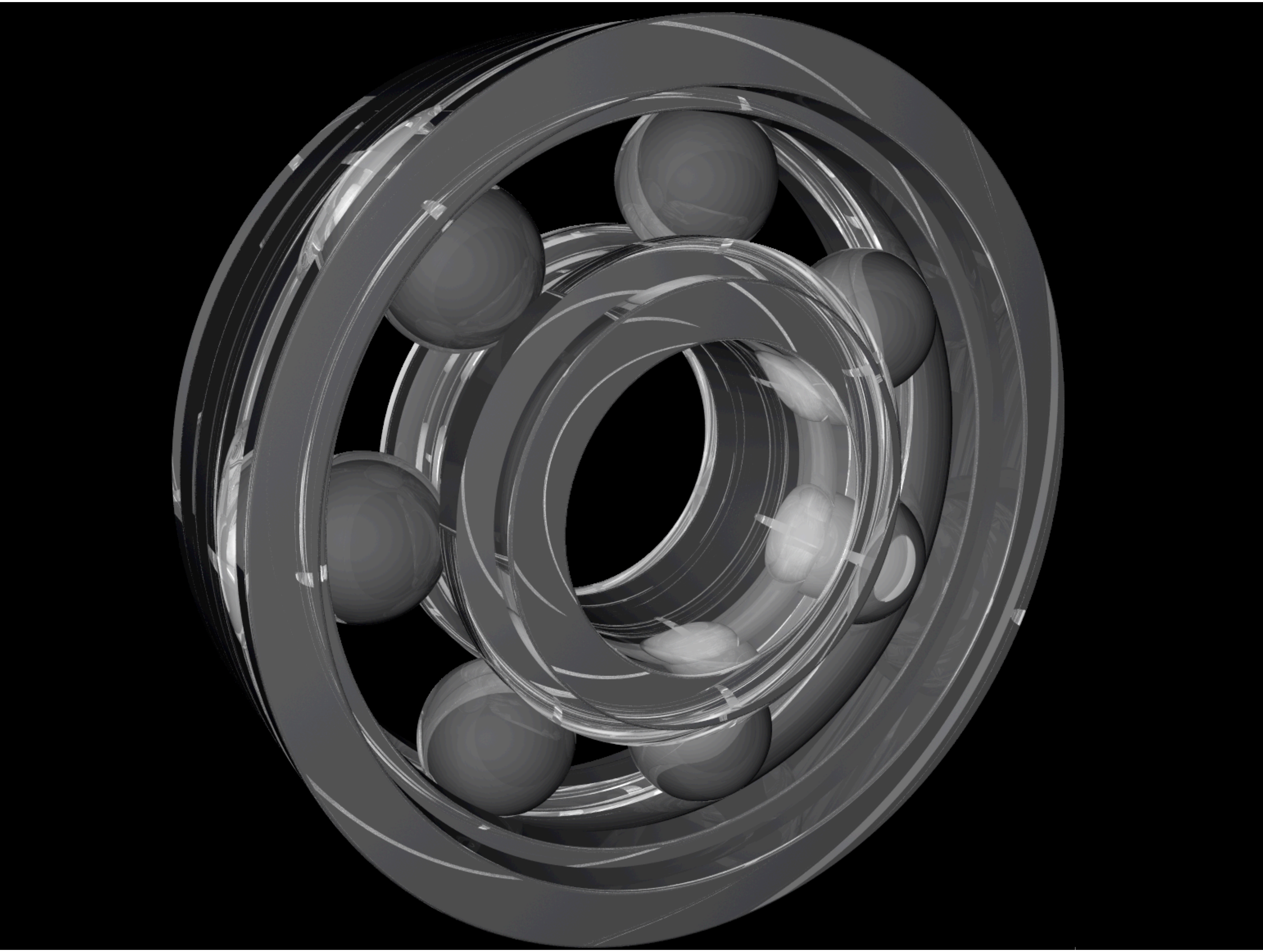
Conversion to Open Source

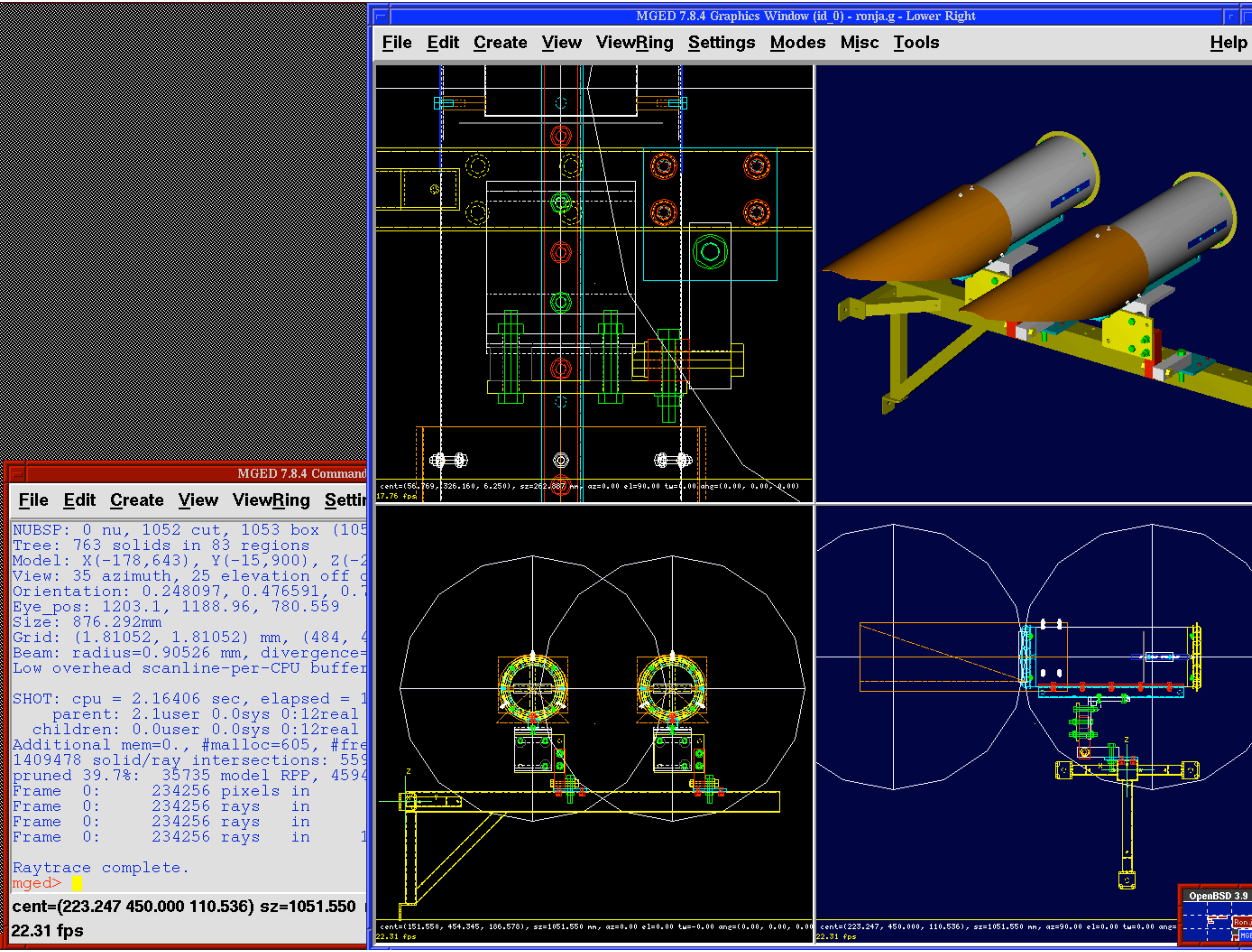
Open Problems & Future Directions

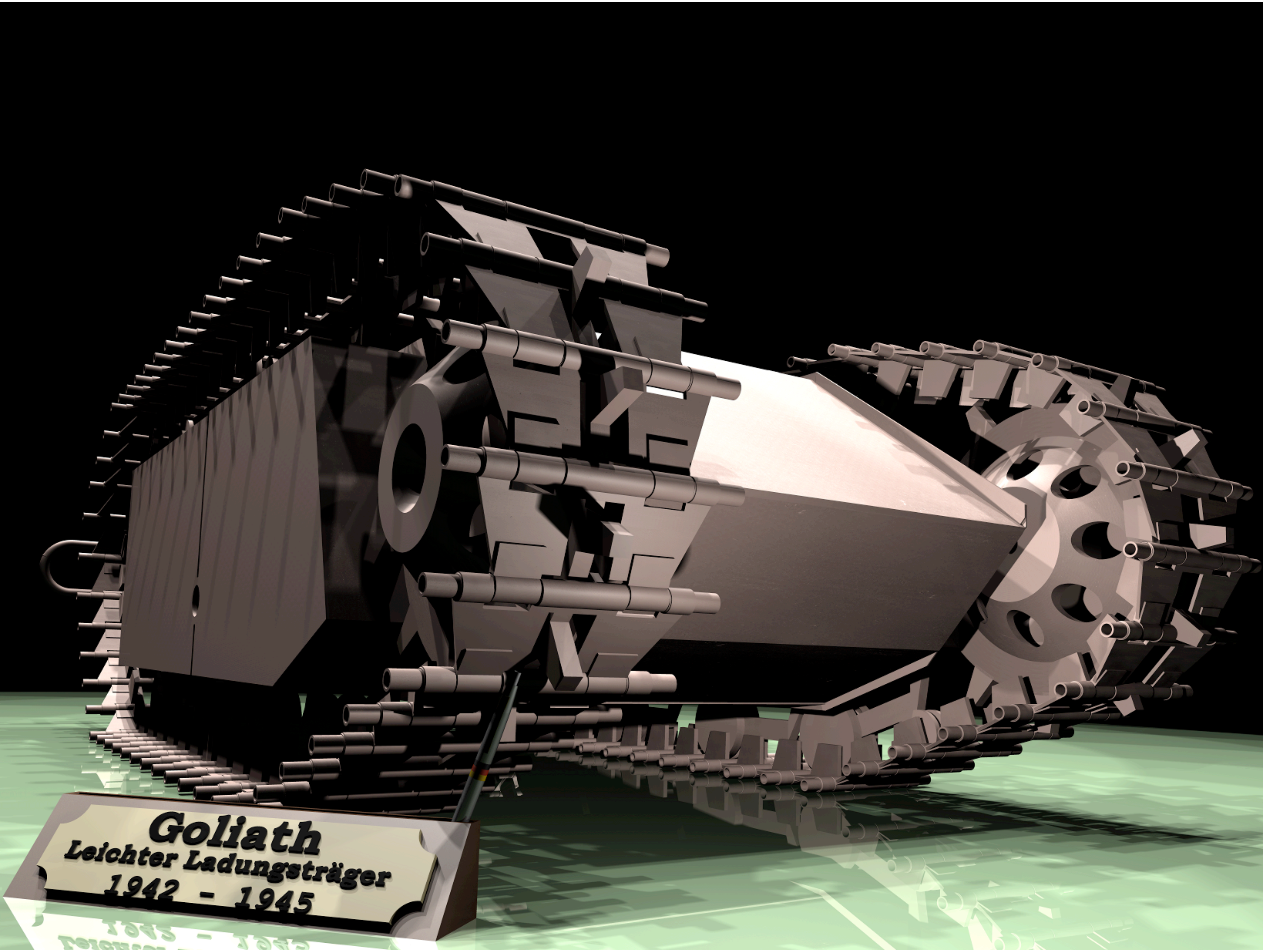
Screenshots!



Archer
File Display Modes Raytrace Utilities Help
Show List
all.g
u component
u bed
u frame
u cab
u suspension
u misc
u power.train
u half
u half.s
u light1
u light1.s
u light2
u light2.s
cab.g
u cab
u half
u half.s
u light1
u light1.s
u light2
u light2.s
old.s79
old.s82
r682
u s682
r683
u s683
r684
u s684
r685
u s685
units:in size:242.99 center:(57.67, -0.13, 49.24) az:35.00 el:25.00 tw::0.00
General Shader Tree
Combination: light1
Region: yes
Id: 0
Air:
Los:
GIFTmater:
Rgb: 255 255 255
Inherit: no
Command History
parent: 2.6user 0.0sys 0:02real 91% 0i+0d 0maxrss 0+605pf 7+305csw
children: 0.0user 0.0sys 0:02real 0% 0i+0d 0maxrss 0+0pf 0+0csw
Additional mem=0., #malloc=906, #free=814, #realloc=5 (92 retained)
2467809 solid/ray intersections: 778493 hits + 1689316 miss
pruned 31.5%: 274936 model RPP, 1530030 dups skipped, 538081 solid RPP
Frame 0: 698326 pixels in 1.32 sec = 527118.65 pixels/sec
Frame 0: 699083 rays in 1.32 sec = 527690.06 rays/sec (RTFM)
Frame 0: 699083 rays in 2.65 sec = 263845.03 rays/CPU_sec
Frame 0: 699083 rays in 2.92 sec = 239565.49 rays/sec (wallclock)
Archer>
BRL-CAD




MGED 7.8.4 Graphics Window (id_0) - ronja.g - Lower Right
File Edit Create View ViewRing Settings Modes Misc Tools Help
MGED 7.8.4 Command
File Edit Create View ViewRing Settin
NUBSP: 0 nu, 1052 cut, 1053 box (105
Tree: 763 solids in 83 regions
Model: X(-178,643), Y(-15,900), Z(-2
View: 35 azimuth, 25 elevation off c
Orientation: 0.248097, 0.476591, 0.7
Eye_pos: 1203.1, 1188.96, 780.559
Size: 876.292mm
Grid: (1.81052, 1.81052) mm, (484, 4
Beam: radius=0.90526 mm, divergence=
Low overhead scanline-per-CPU buffer
SHOT: cpu = 2.16406 sec, elapsed = 1
parent: 2.1user 0.0sys 0:12real
children: 0.0user 0.0sys 0:12real
Additional mem=0., #malloc=605, #fre
1409478 solid/ray intersections: 559
pruned 39.7%: 35735 model RPP, 4594
Frame 0: 234256 pixels in
Frame 0: 234256 rays in
Frame 0: 234256 rays in
Frame 0: 234256 rays in 1
Raytrace complete.
mged>
cent=(223.247 450.000 110.536) sz=1051.550
22.31 fps
17.76 fps
22.31 fps
22.31 fps
OpenBSD 3.9


Goliath
Leichter Ladungsträger
1942 - 1945

Main Topics

Background on BRL-CAD
What, who, why?

Tools & Techniques
for Geometry Analysis

Conversion to Open Source

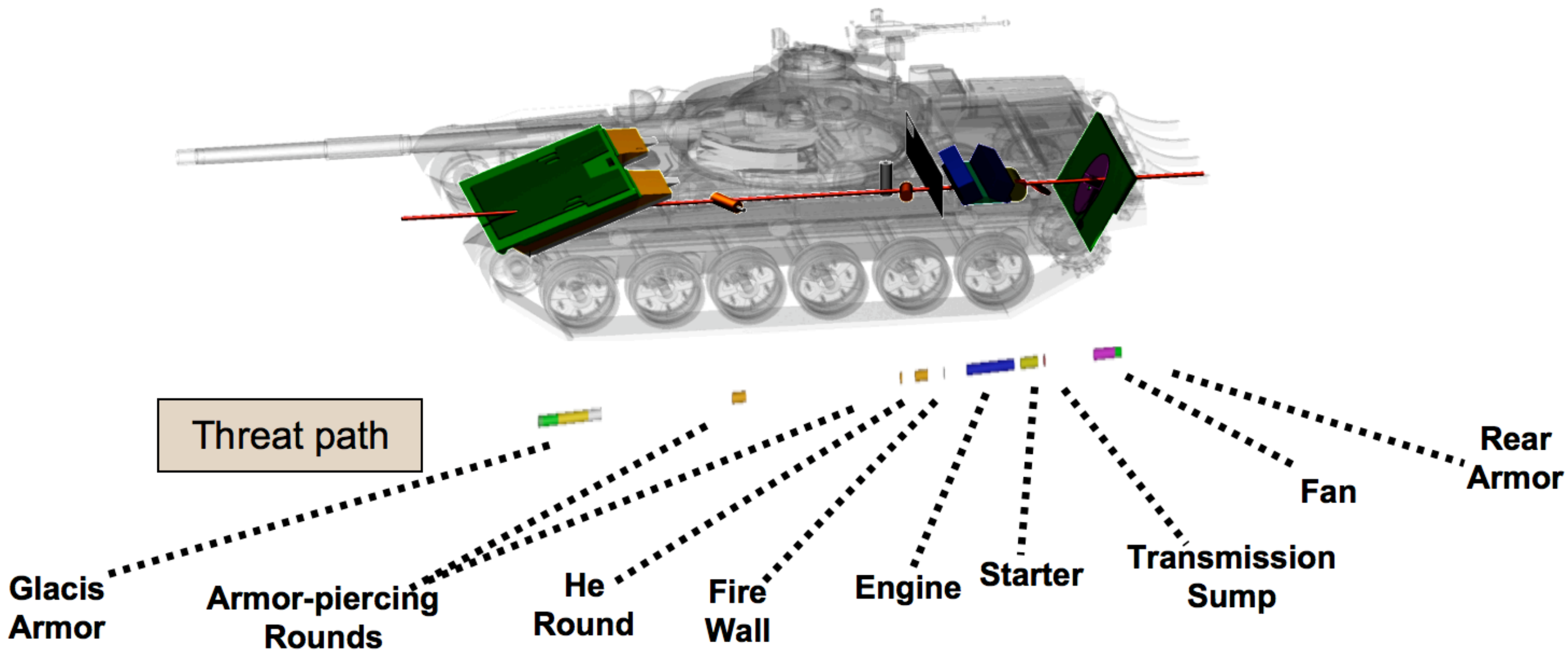
Open Problems & Future Directions

Background

- BRL-CAD* is a powerful open source solid modeling system that includes interactive geometry editing, ray tracing for rendering & geometric analyses, a robust geometric representation, data processing tools, and **more than 25 years** of development history.

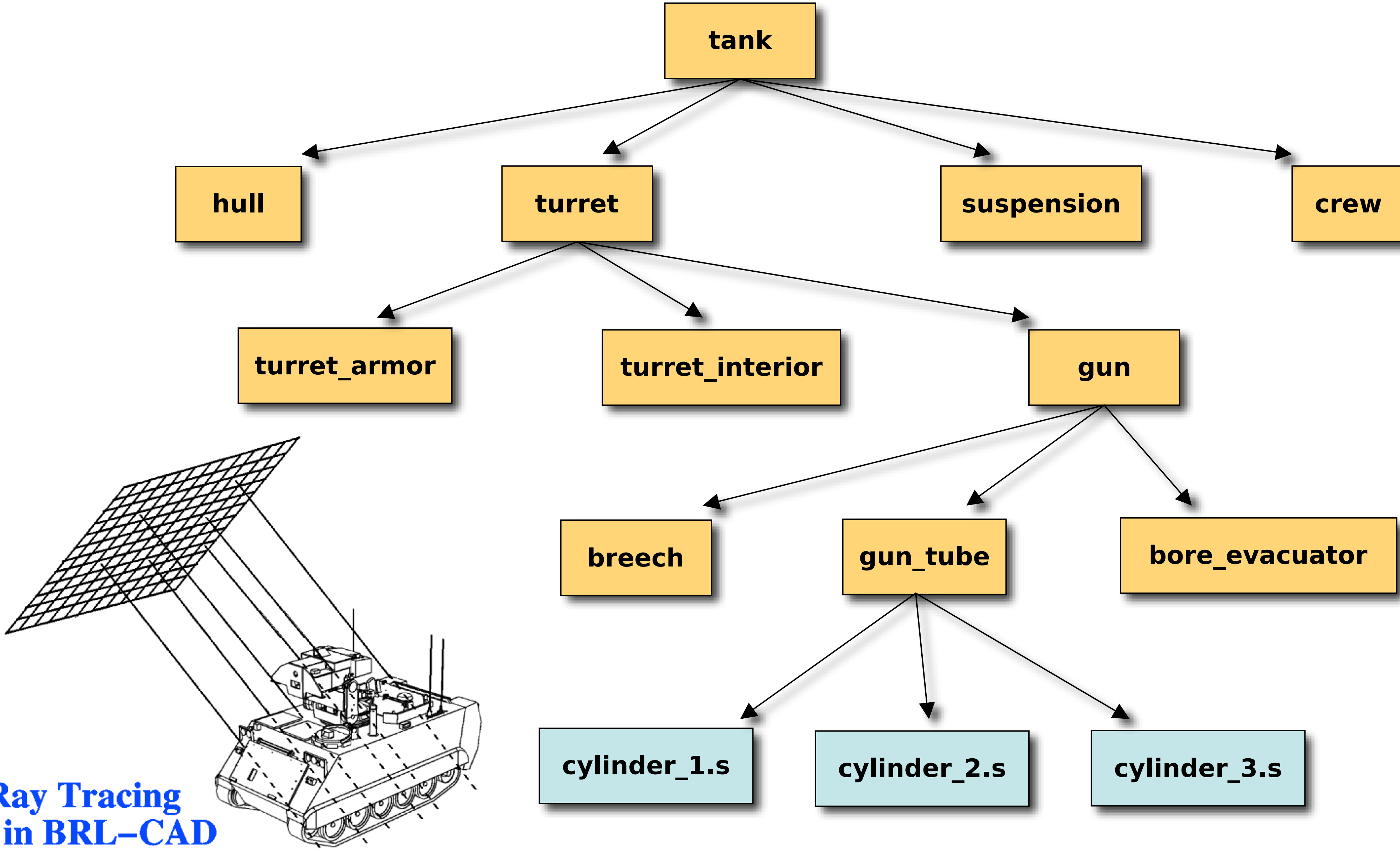
- Development of BRL-CAD started in 1979 by departed wizard **Mike Muuss**, author of many contributions to computing including the infamous “ping” network tool.

- BRL-CAD was designed to provide tools, techniques, and methodology for representing, visualizing, and analyzing geometry. Particular emphasis is placed on validity, performance, and robustness for vulnerability and lethality analyses.



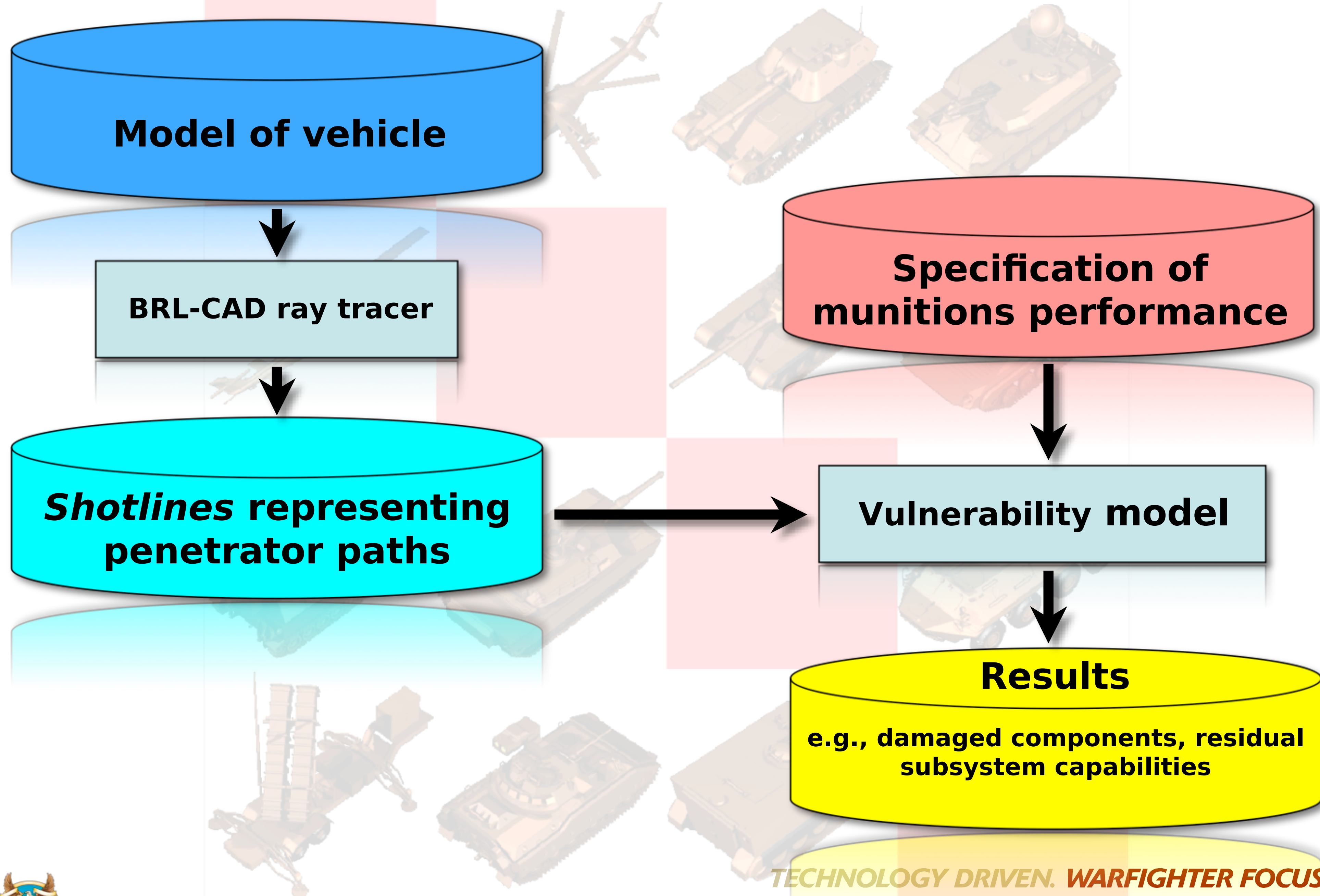
* BRL-CAD is correctly pronounced as “be are el cad”

Structured Geometry



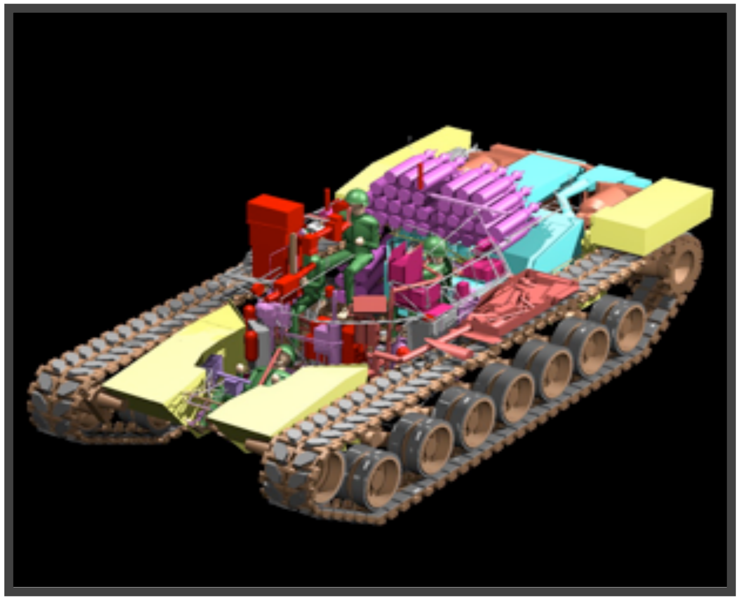
Ray Tracing
in BRL–CAD

Vulnerability / Lethality Analysis



Why Develop BRL-CAD?

- BRL-CAD is custom-tailored for engineering analysis work providing robust accurate geometric representation and high-performance geometry evaluation.
  - There is no conversion necessary which is crucial for ensuring robust, consistent, and correct analytic results. *Data conversions introduce errors and complicate validation & verification.*
- Solid modeling and ray tracing are used to represent material interactions and determine paths of propagation. Analyses require shooting millions of rays through highly detailed target descriptions (with millimeter accuracy).
  - This is a **niche** requirement not strongly supported by other CAD systems. BRL-CAD's solid ray tracing is designed to significantly *outperform* commercial ray tracers.
- Extensive model repositories exist at ARL and AFRL of foreign and U.S. assets.
  - Hundreds of *highly* detailed target descriptions represent a major investment in BRL-CAD spanning more than two decades.



- BRL-CAD provides scalability, portability, robustness, and verifiable accuracy that can perpetually and independently be adapted.
  - This allows the U.S. Government to not favor any CAD vendor, avoids expensive licensing, and protects ARL from corporate restructuring.
- **BRL-CAD is highly tuned to DoD needs, more than any other CAD system.**

Main Topics

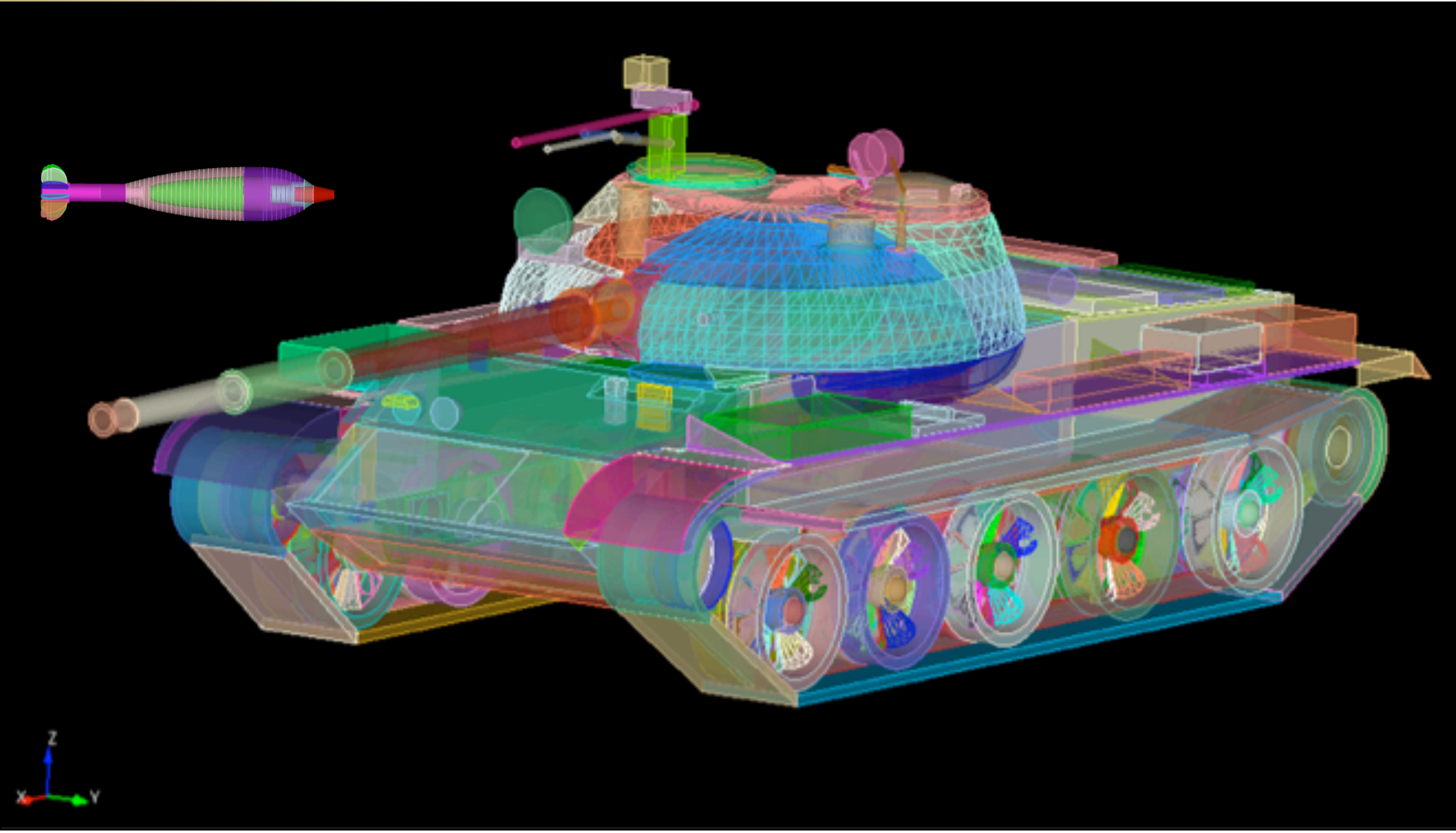
Background on BRL-CAD
What, who, why?

Tools & Techniques
for Geometry Analysis

Conversion to Open Source

Open Problems & Future Directions

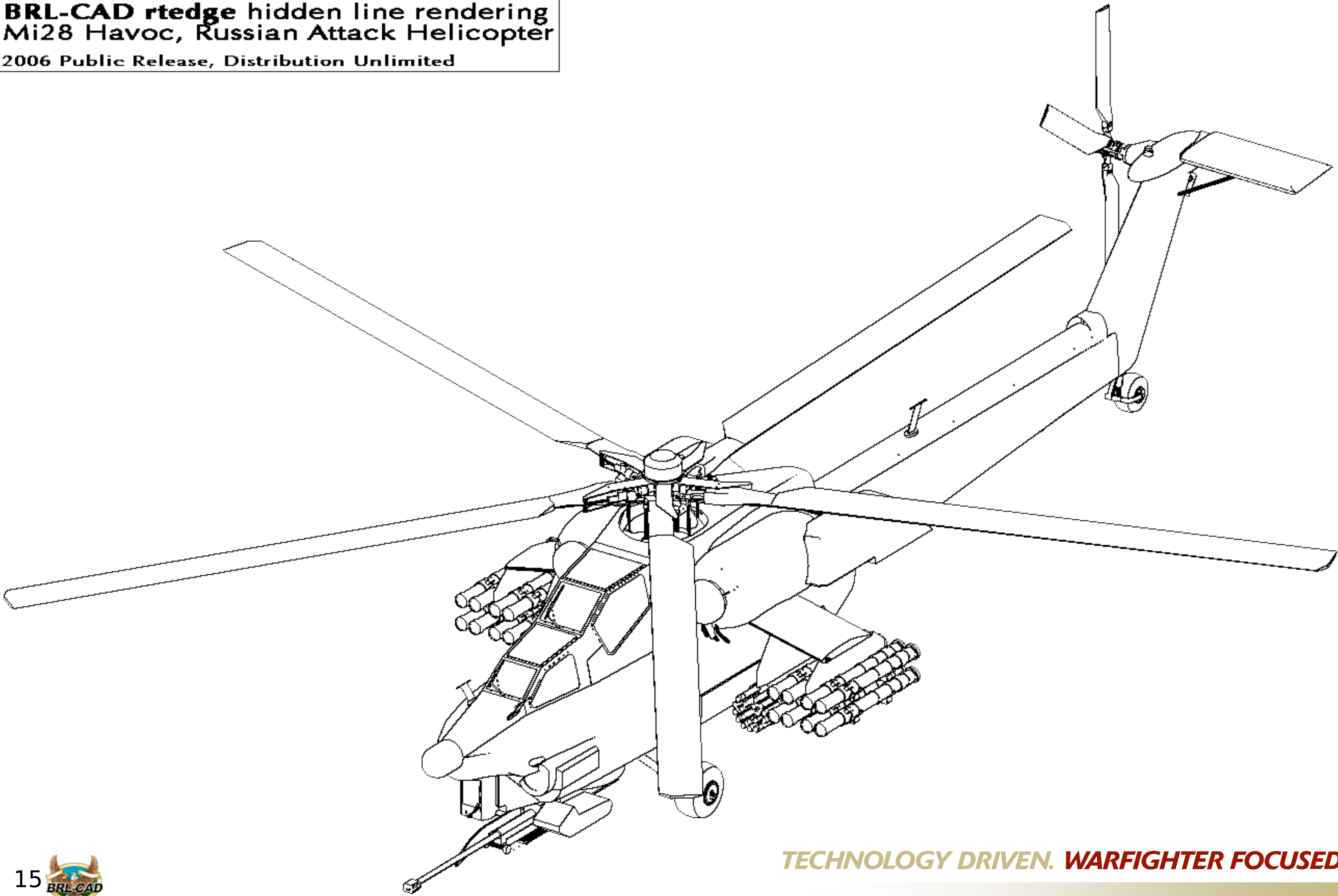
Finite Element Analysis of Target Descriptions



Converted from a BRL-CAD CSG representation to an explicit finite element mesh

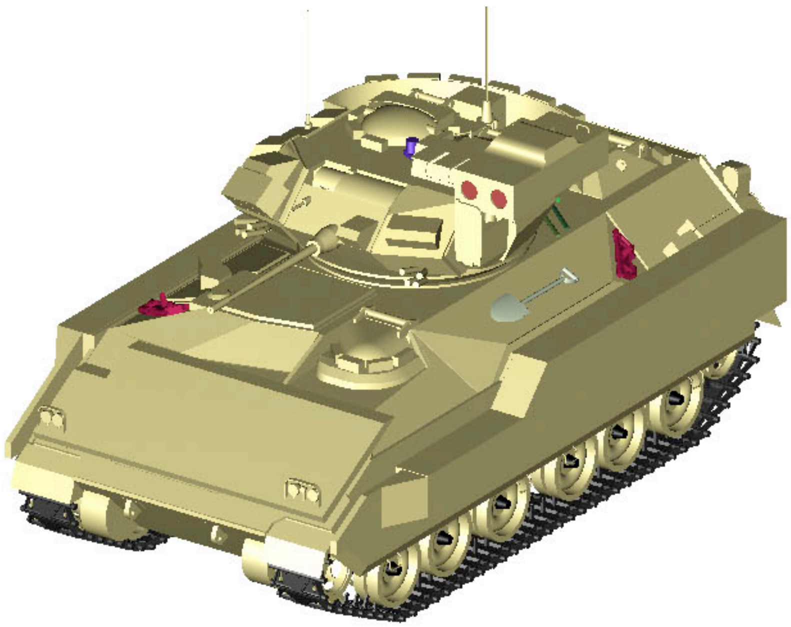
Hidden Line Diagrams

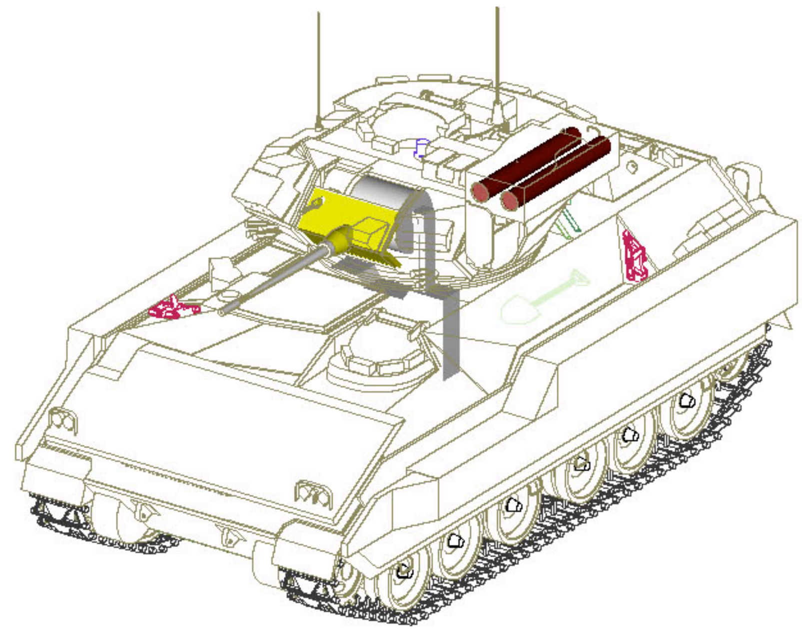
BRL-CAD rtedge hidden line rendering
Mi28 Havoc, Russian Attack Helicopter
2006 Public Release, Distribution Unlimited

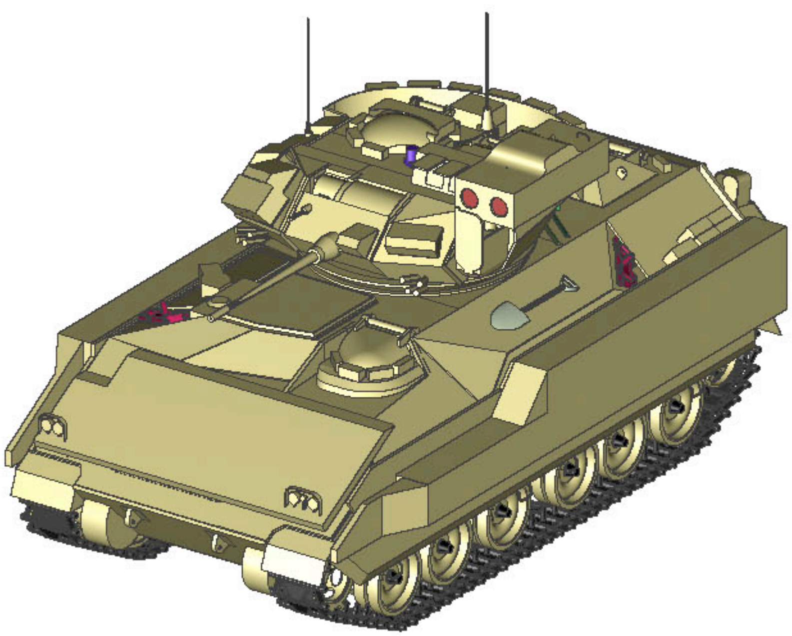


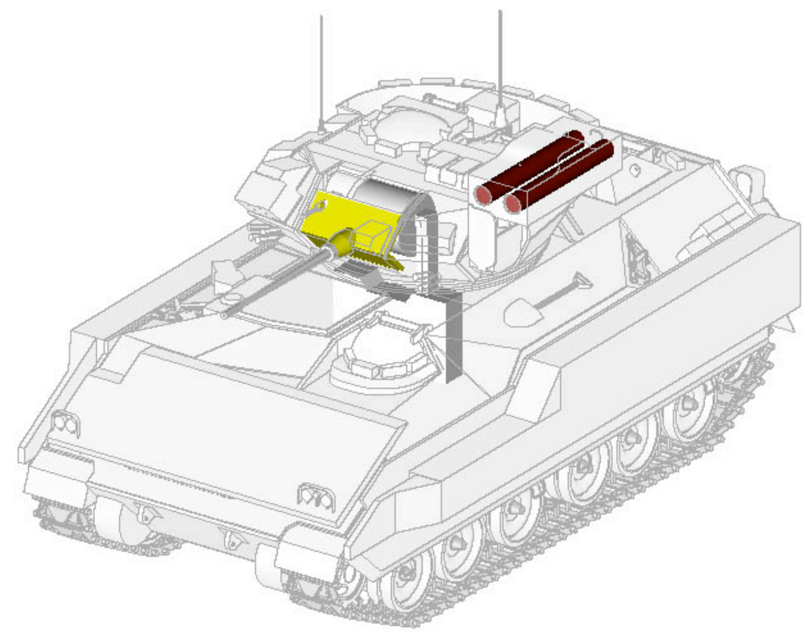
TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

Hybrid Visualizations









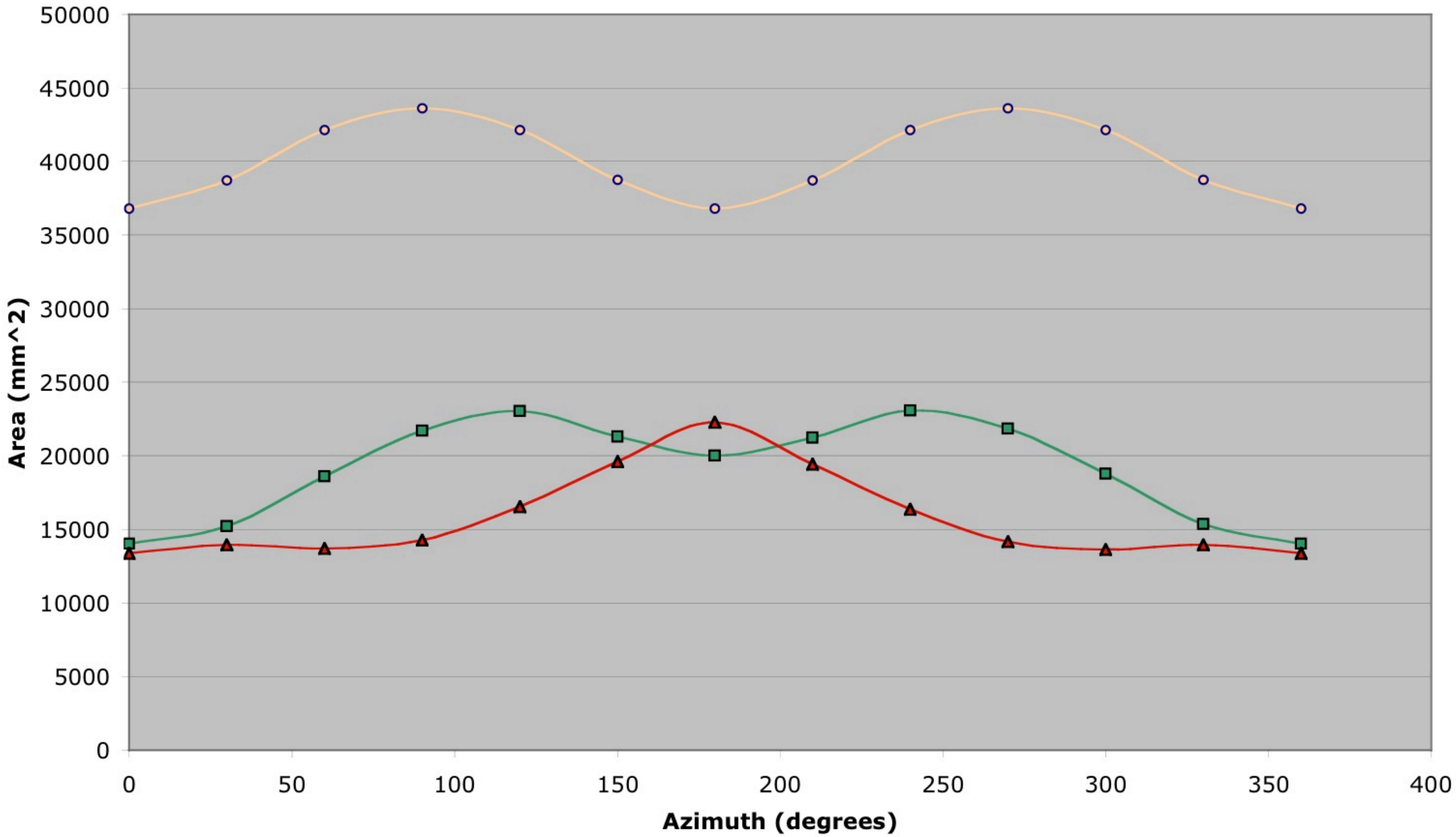
Full Light Simulation

Geometric Analysis

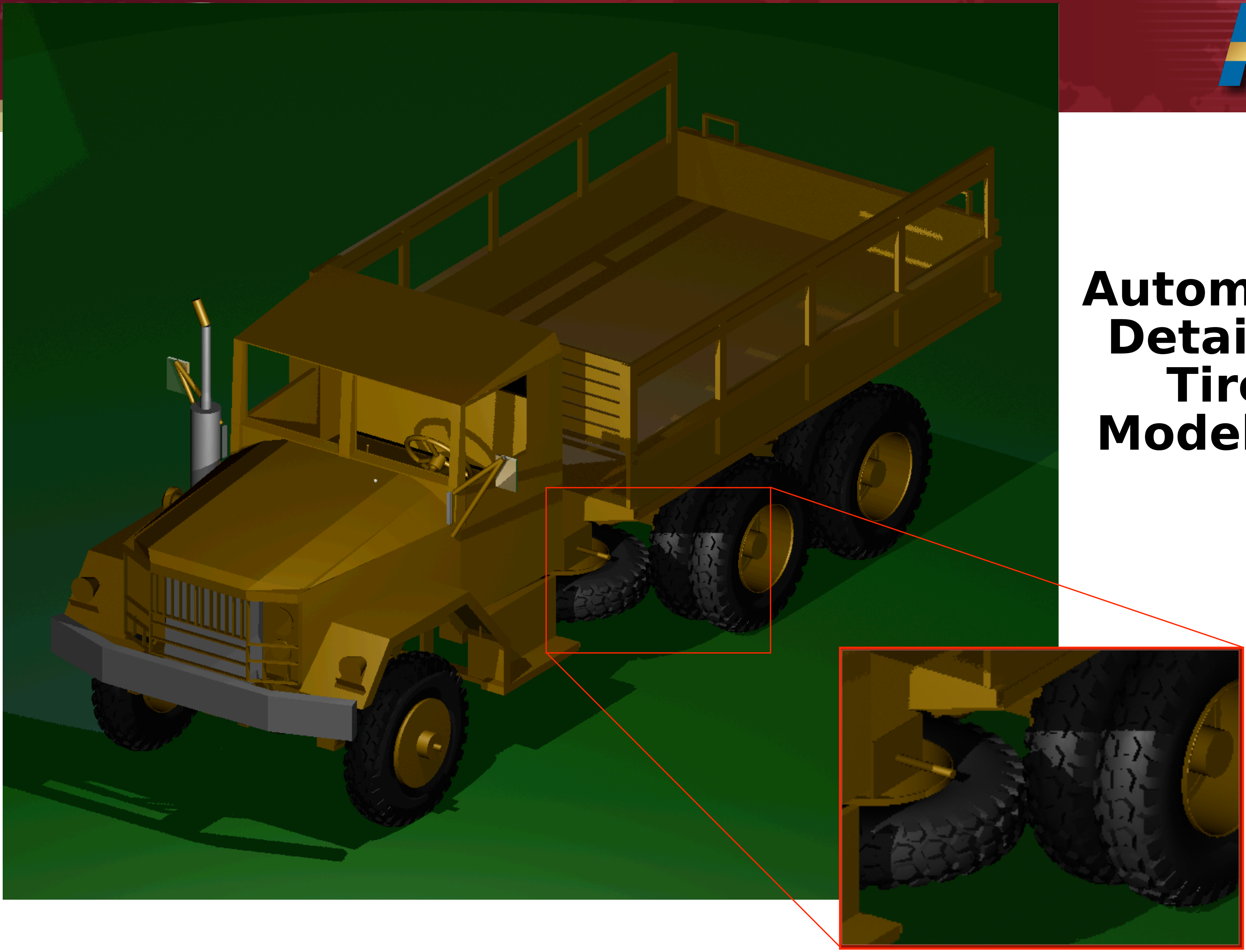
Presented Area, Volume, Centroid, Weight/Mass, Moments of Inertia, Line-of-sight Equivalence, Shape Factors, … (and much more)

Comparing Area Exposure

Helmet presented area coverage comparisons



TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.



Automatic Detailed Tire Modeling

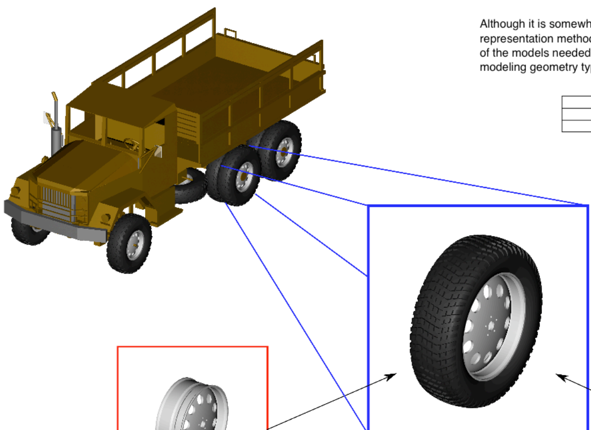
Automated Procedural Modeling of Tires Using Constructive Solid Geometry

Introduction

Computer Aided Design Systems traditionally employ any of three geometric representation techniques when describing geometry - Constructive Solid Geometry (CSG) employing implicit primitives and boolean operators, Non-Uniform Rational B-Splines (NURBS), and Polygonal Meshes. Each technique has advantages and disadvantages - CSG models tend to be very efficient from the standpoint of disk storage, but models are limited to shapes that can be expressed using available primitives. NURBS models are very flexible in terms of shapes they can represent but visualizing them directly is challenging. Polygonal Meshes are relatively simple to manipulate and view but can be very expensive from the standpoint of storage - it may require thousands of vertices and edges to describe the geometry represented in a single CSG primitive.

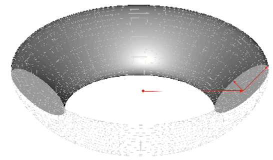
Examples of procedurally generated tires - from left to right, dimensions are 505/35R24, 315/30R30 and 325/60R20

Vehicle tires represent a challenge for CSG representation - the shape of a tire does not map in a straightforward way to most CSG primitives. The solution is an automated procedure that implements a generalized CSG representation of a tire and uses standard dimensional information used to describe tires to deduce all necessary specific geometric parameters needed to produce that specific tire geometry. This allows for a wide variety of tires to be modeled using the same core geometric design, producing very detailed and compact representations of a wide variety of tires with very little modeler effort.



The Elliptical Torus

A CSG tire representation involves cylinders, truncated general cones, ellipsoids, extruded sketches (if tread is modeled) and most importantly the Elliptical Torus. Because most modern tires do not have circular cross sections, and self-intersecting tori are not allowed as valid geometry in some CAD systems, the only available primitive that can approximate the necessary surface curvatures is the Elliptical Torus. The Elliptical Torus also has the constraint of not self intersecting, but its ability to represent shallow curves using large semimajor/seminor axis ratios means this constraint is seldom a problem when representing a reasonable tire shape.



Elliptical Torus structure - vectors represent control parameters of the primitive in BRL-CAD

Size Advantages of CSG

Constructive Solid Geometry Storage Requirements

eto params	eto # floats	cyl params	cyl # floats	trc params	trc # floats	ellipsoid params	ellipsoid # floats	sketch (large) params	sketch (large) # floats	sketch (large) #params / sketch	sketch (small) params	sketch (small) # floats	sketch (small) #params / sketch	extrude params	extrude # floats	total
V	3	V	3	V	3	V	3	V	3	1	V	3	1	V	3	
N	3	H	3	H	3	a	3	u_vec	3	1	u_vec	3	1	H	3	
C	3	r	1	r1	1	b	3	v_vec	3	1	v_vec	3	1	u_vec	3	
r	1			r2	1	c	3	verticies	2	8	verticies	2	4	v_vec	3	
rd	1							segments	2	8	segments	2	4			
Total	11	Total	7	Total	8	Total	12	Total		19	Total		11	Total	12	
# etos	11	# cyls	27	#trcs	8	#ells	2	# sketches		2	# sketches		7	# extrudes	27	84
Total	121	Total	189	Total	64	Total	24	Total		38	Total		77	Total	324	837

Geometric Primitives	Total:		837
Combinations (est. 1 float) x	x Qty	42	42
Matricies (16 floats)	x Qty	60	960
Total Storage Needed:			1839

Alternative Geometry Storage Requirements

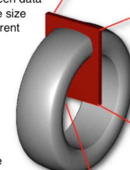
	#Faces	Storage/Face	Total
NURBS	2562	32	81984
Polymesh	47946	9	431514

Although it is somewhat difficult to do a direct size comparison between data representation methods, these numbers represent an estimate of the size of the models needed to represent an example tire with tread in different modeling geometry types.

Comparison of Storage Requirements

CSG	NURBS	Polymesh
1839	81984	431514

These numbers support the general rule of thumb - one order of magnitude increase for NURBS surfaces, two for Polymeshes. It should be noted that the Polymesh estimates here are based on a rather course tessellation of the tire model and more accurate meshing would tend to rather steeply increase the size of the model.

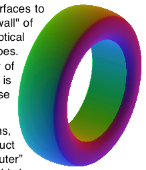


Tire CSG Boolean Tree

This illustrates the structure of the geometric tree used to form a tire using CSG boolean operations. The "U", "-", and "+" notations indicate a union, subtraction, and intersection operation respectively. The red boxes indicate where the model describes a single homogenous region of material in real space. The blue box indicates an assembly of regions.

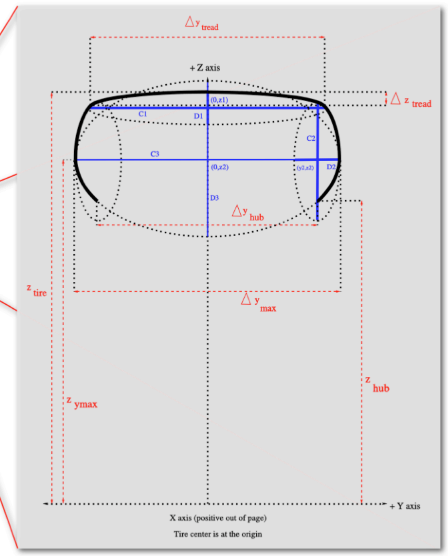
Defining the Tire Shape

Although the Elliptical Torus is a critical part of defining the tire shape, by itself it is not sufficient. Most modern tires have a relatively "flat" surface area that is intended to contact the road surface. This "flat" area then curves into the "walls" of the tire, which in turn terminate at the point where they connect with the wheel rim. In order to define these two areas, three Elliptical Tori must be precisely aligned and defined in such a fashion that they are "smooth". In this case, "smooth" is defined as no surface discontinuities at the intersection curves.



Constraining the surfaces and the first derivatives of the surfaces to be equal at the point where the "tread" surface meets the "wall" of the tire is sufficient to result in a unique solution to the elliptical equations governing that aspect of the elliptical tori's shapes. Although the joining is not "perfect" - a surface normal view of the tire shape clearly reveals the transition point - it is sufficient to ensure a solid tire with a surface profile very close to the expected shape for a common vehicle tire.

Once the surface shape has been defined in general terms, several specific instances of it are typically used to construct the final wheel surface. The first tire surface defines the "outer" surface of the tire. In the case of a "slick" tire without tread this is



actually the outer tire surface and its maximum z height is equal to the maximum radius of the tire. In the case of a tire with tread this surface has its maximum z height equal to the radius of the tire minus the tread depth. Then a second surface is used to "cut out" the volume defined by the first surface and form the basic tire shape. If tread is included a similar process of top and bottom surfaces is used to define the volume that will contain tread, and the tread pattern (based on extrusions of a sketch repeated around the circumference of the tire) is subtracted from the tread volume. This results in the final tread shape, which is combined with the non- tread volume already defined to form the complete wheel solid. The possible shapes supported by this set of primitives are primarily limited by the constraint on the elliptical torus not to self intersect - very extreme configurations (an extremely tall, narrow tire on a small wheel) will not be expressible. This is seldom a problem in practice. Other potential problems arise when pushing the extremes of thickness, tread depth, etc. but those too seldom arise in real-world situations.

Solving for Continuous Alignment

In order to align the individual elliptical tori to form a smooth surface, it is necessary to solve systems of equations describing the conditions the elliptical tori must satisfy. There are four elliptical tori forming a single tire surface. To form systems of linear equations, the first step is to recognize that the radial symmetry of the tire means we can reduce the problem to two dimensional alignment of ellipses. This means the parameters for the elliptical tori can be found from the General Conic Equation (GCE) coefficients:

$$E(y,z) = Ay^2 + Byz + Cz^2 + Dy + Ez + F = 0$$

The first ellipse to solve for is the ellipse forming the "top" of the tire. The constant F is divided out:

$$E_{\text{top}}(y,z) = A_{\text{top}}\,y^2 + B_{\text{top}}\,yz + C_{\text{top}}\,z^2 + D_{\text{top}}\,y + E_{\text{top}}\,z + 1 = 0$$

Solving for the coefficients means there are five unknowns, requiring five equations for the linear system. Examining the available known intersection points and symmetries immediately supplies three points: the maximum tire height at y=0 and the two intersection points with the upper side ellipses. Two more are needed - in order to define them the length of the minor axis of the ellipse is set to a fraction of the distance between the maximum tire radius and the hub radius. Based on experiment, the following value for d1 seems to be practical:

$$d_1 = 0.8(z_{\text{tire}} - z_{\text{hub}})$$

This provides another symmetry axis for two more points, and completes the system of equations:

$$z_r = z_{\text{tire}},\ \ \Delta y_t = \Delta y_{\text{tread}},\ \ \Delta z_t = \Delta z_{\text{tread}},\ \ z_o = d_1 - \Delta z_t$$

$$\begin{pmatrix} \left(\frac{\Delta y_t}{2}\right)^2 & \left(\frac{\Delta y_t}{2}\right)(z_r - \Delta z_t) & (z_r - \Delta z_t)^2 & \frac{\Delta y_t}{2} & (z_r - \Delta z_t) \\ \left(\frac{\Delta y_t}{2}\right)^2 & \left(\frac{\Delta y_t}{2}\right)(z_r - z_o) & (z_r - z_o)^2 & \frac{\Delta y_t}{2} & (z_r - z_o) \\ \left(\frac{-\Delta y_t}{2}\right)^2 & \left(\frac{-\Delta y_t}{2}\right)(z_r - \Delta z_t) & (z_r - \Delta z_t)^2 & \frac{-\Delta y_t}{2} & (z_r - \Delta z_t) \\ 0 & 0 & z_r^2 & 0 & z_r \\ 0 & 0 & (z_r - d_1)^2 & 0 & z_r - d_1 \end{pmatrix} \begin{pmatrix} A_{\text{top}} \\ B_{\text{top}} \\ C_{\text{top}} \\ D_{\text{top}} \\ E_{\text{top}} \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

Once this system is solved, the resulting equation can be solved for z in terms of y:

$$E_{\text{top}}(y,z) \Rightarrow f_{\text{top}}(y) = F(A,B,C,D,E,y), \frac{\partial f_{\text{top}}(y)}{\partial y} = F'(A,B,C,D,E,y)$$

For the solution of the upper side ellipses, the constraints also involve the partial derivative of the GCE with respect to y:

$$\frac{\partial}{\partial y}(E(y,z)) = A\frac{\partial}{\partial y}y^2 + B\frac{\partial}{\partial y}yz + C\frac{\partial}{\partial y}z^2 + D\frac{\partial}{\partial y}y + E\frac{\partial}{\partial y}z + \frac{\partial}{\partial y}F = 0$$

$$\frac{\partial}{\partial y}(E(y,z)) = 2Ay + B\left(z + y\frac{\partial z}{\partial y}\right) + C\left(2z + \frac{\partial z}{\partial y}\right) + D + E\frac{\partial z}{\partial y}z = 0$$

The immediately apparent intersection points for the ellipse to match are the side maximum and the intersection with the top ellipse but those only supply two of the necessary five equations. For the "smoothness" criteria to be satisfied on the tire surface, the partial derivative of the side ellipse at the intersection point is forced to be equal to the partial derivative of the top ellipse at the same point:

$$\frac{\partial f_{\text{side}}}{\partial y}\Big|_{\frac{\Delta y_t}{2}} = \frac{\partial f_{\text{top}}}{\partial y}\Big|_{\frac{\Delta y_t}{2}}$$

This allows the equation for to work in the solution of the linear system, and supplies three of the five equations. The final two result from the constraint that needs to be the maximum point of the ellipse in the positive y direction. This provides an ellipse axis over which the intersection point can be mirrored, providing a fourth equation. The fifth constraint comes from constraining the partial derivative at the mirrored point to satisfy the symmetry of the ellipse. These constraints define the second system of equations:

$$z_{s_2} = (z_{\text{tire}} - \Delta z_{\text{tread}}) - 2(z_{\text{tire}} - \Delta z_{\text{tread}} - z_{\text{ymax}}) = 2z_{\text{ymax}} - z_{\text{tire}} + \Delta z_{\text{tread}}$$

$$z_r = z_{\text{tire}} \quad \Delta y_t = \Delta y_{\text{tread}} \quad \Delta z_t = \Delta z_{\text{tread}} \quad \Delta y_m = \Delta y_{\text{max}} \quad z_m = z_{\text{ymax}}$$

$$\begin{pmatrix} \left(\frac{\Delta y_m}{2}\right)^2 & \left(\frac{\Delta y_m}{2}\right)z_m & z_m^2 & \frac{\Delta y_m}{2} & z_m \\ \left(\frac{\Delta y_t}{2}\right)^2 & \left(\frac{\Delta y_t}{2}\right)(z_r - \Delta z_t) & (z_r - \Delta z_t)^2 & \left(\frac{\Delta y_t}{2}\right) & (z_r - \Delta z_t) \\ \left(\frac{\Delta y_t}{2}\right)^2 & \left(\frac{\Delta y_t}{2}\right)z_{s_2} & z_{s_2}^2 & \left(\frac{\Delta y_t}{2}\right) & z_{s_2} \\ \Delta y_t & (z_r - \Delta z_t) + \left(\frac{\Delta y_t}{2}\right)\frac{\partial f}{\partial y}\Big|_{\frac{\Delta y_t}{2}} & 2(z_r - \Delta z_t) + \frac{\partial f}{\partial y}\Big|_{\frac{\Delta y_t}{2}} & 1 & \frac{\partial f}{\partial y}\Big|_{\frac{\Delta y_t}{2}} \\ \Delta y_t & z_{s_2} + \left(\frac{\Delta y_t}{2}\right) * -\frac{\partial f}{\partial y}\Big|_{\frac{\Delta y_t}{2}} & 2z_{s_2} - \frac{\partial f}{\partial y}\Big|_{\frac{\Delta y_t}{2}} & 1 & -\frac{\partial f}{\partial y}\Big|_{\frac{\Delta y_t}{2}} \end{pmatrix} \begin{pmatrix} A_{\text{side}} \\ B_{\text{side}} \\ C_{\text{side}} \\ D_{\text{side}} \\ E_{\text{side}} \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

The third ellipse is the mirror of the second ellipse over the z axis. The final ellipse is defined using points derived from the maximum width dimensions and hub dimensions and solved in the same manner as the first ellipse. The General Conic Equation coefficients are then used to solve for the ellipse input parameters needed by the CAD system primitives.

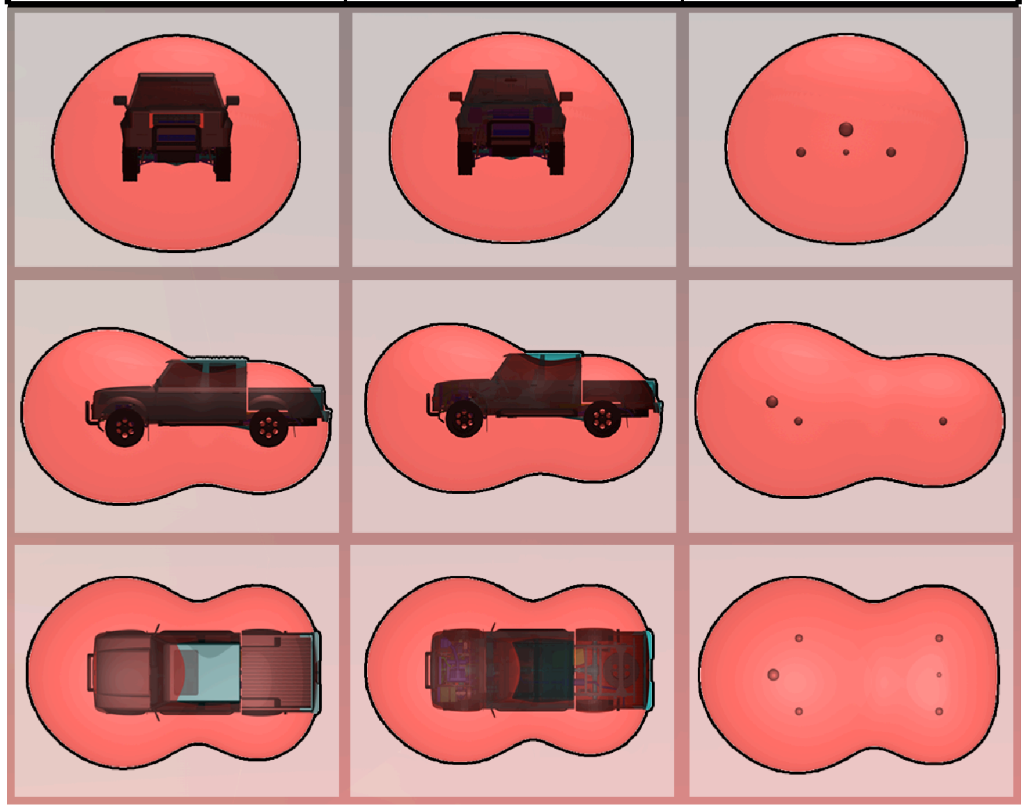
Clifford Yapp
U.S. Army Research Laboratory
Aberdeen Proving Ground, MD
clifford.yapp@arl.army.mil
http://brlcad.org

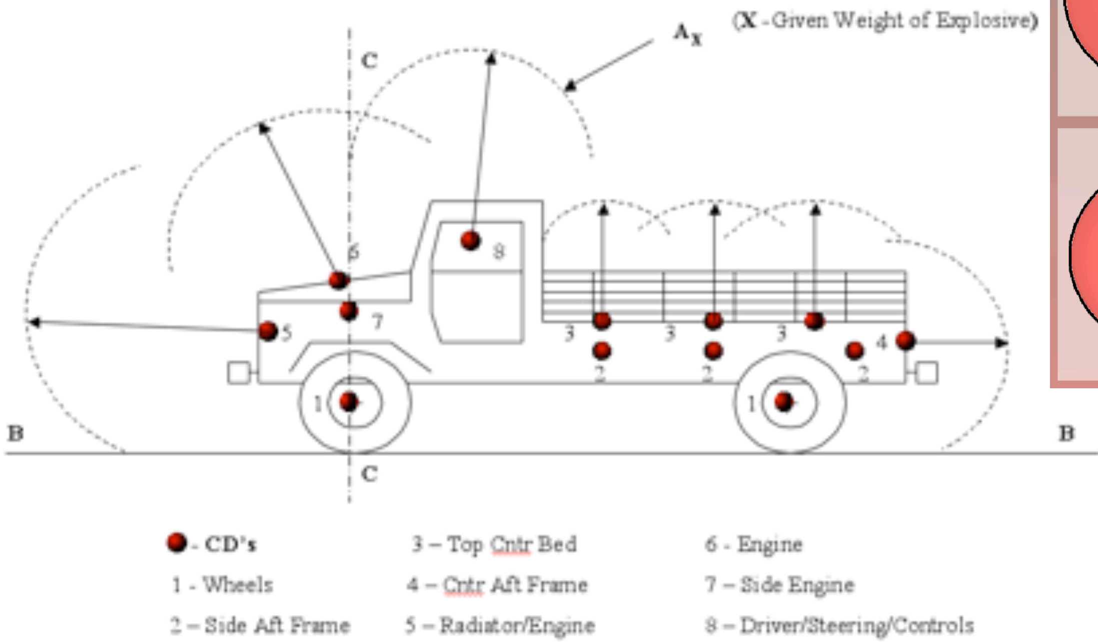
Metaballs for Blast Volumes

- The engineer traditionally:
  - *Specifies a center of damage and lethal miss distance*
  - *Interpolates a 2D curve by hand*
  - *Extrapolates the 2D curves back into a 3D surface*

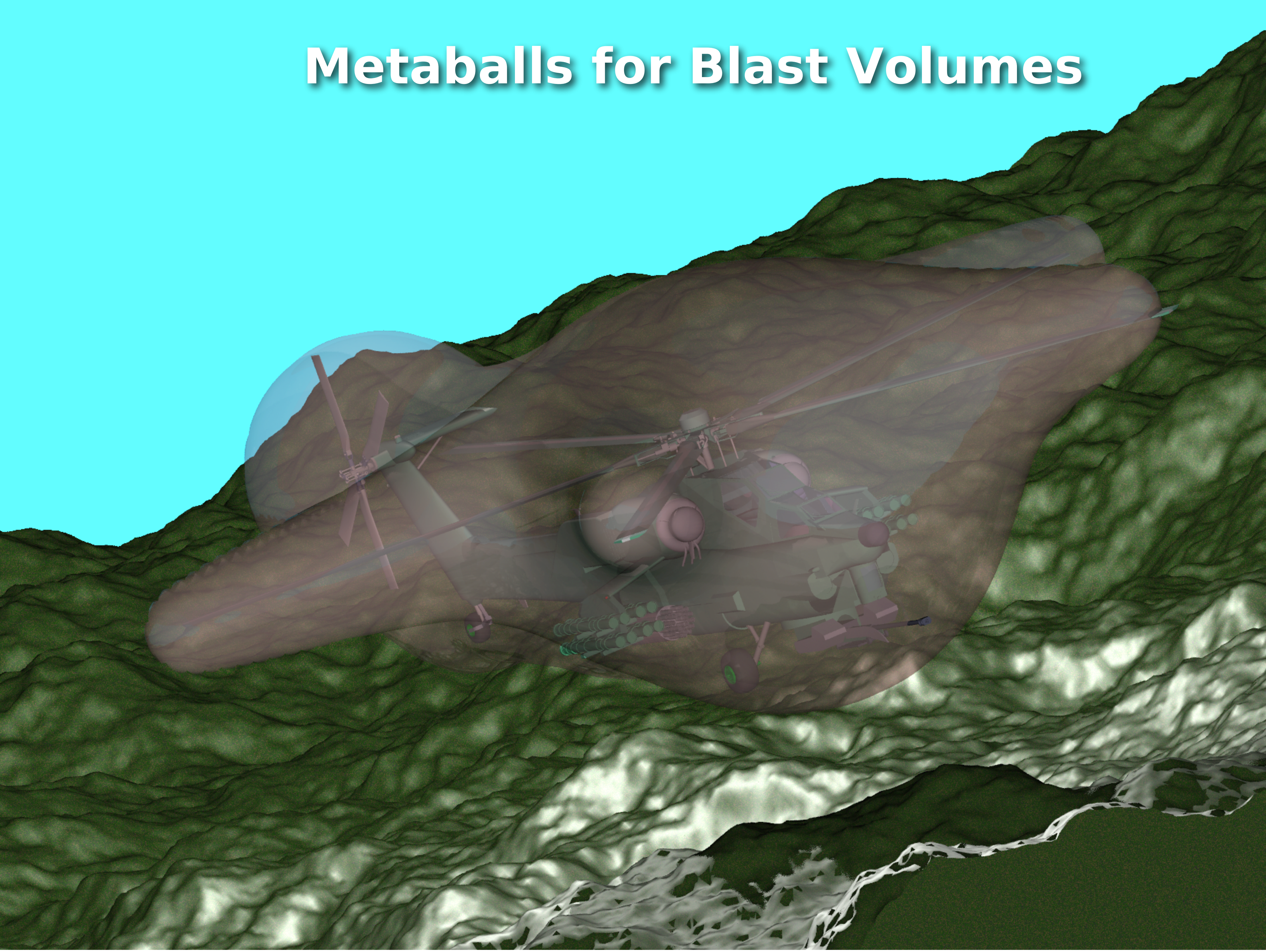
- **With metaballs, the 3D surface is automatically generated based on centers and lethal miss distances**

Truck with Blast Envelope	Glass Truck	Centers of Damage



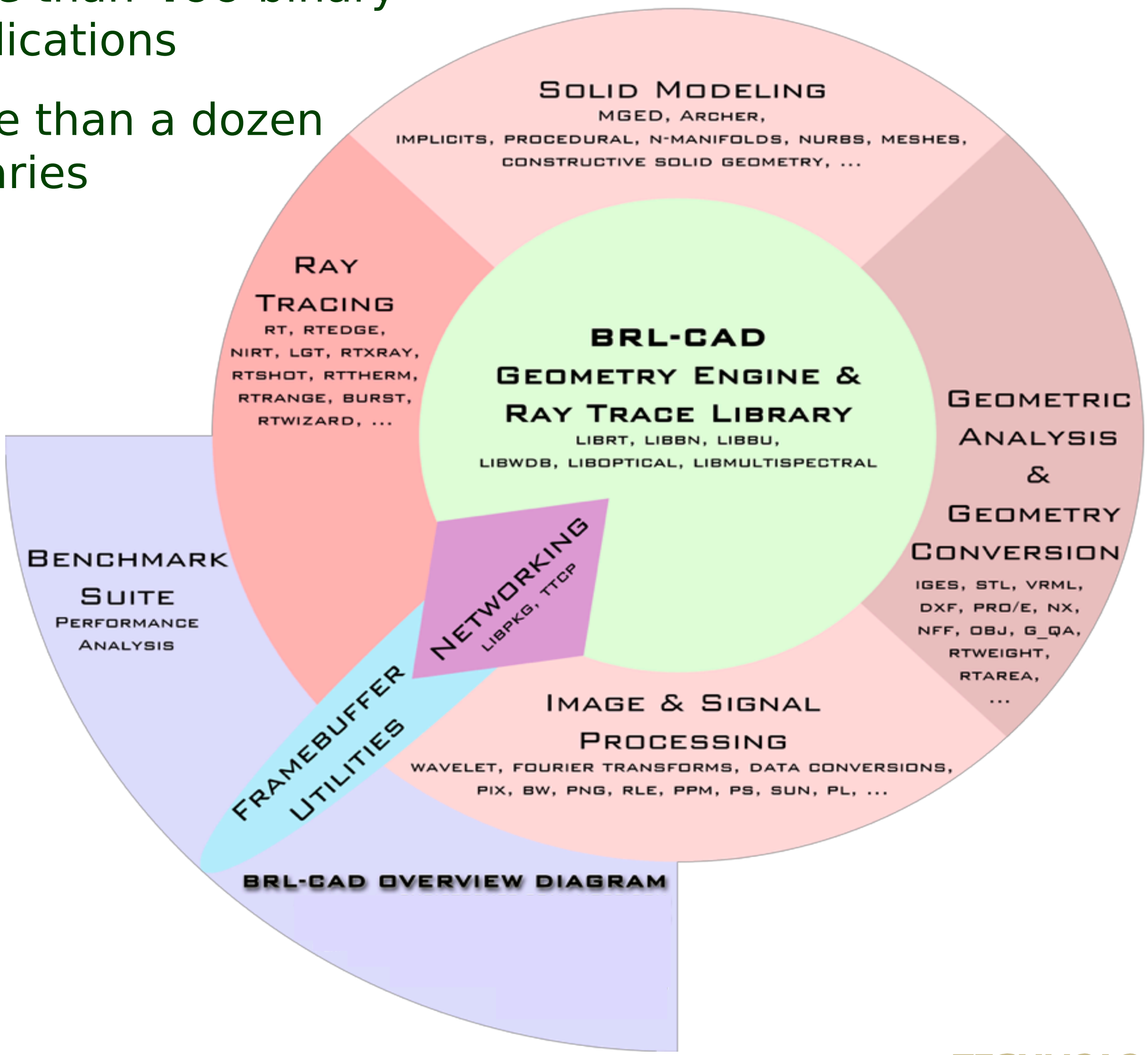


Metaballs for Blast Volumes



BRL-CAD in a Nutshell

- More than *one million* lines of code
- More than **400** binary applications
- More than a dozen libraries



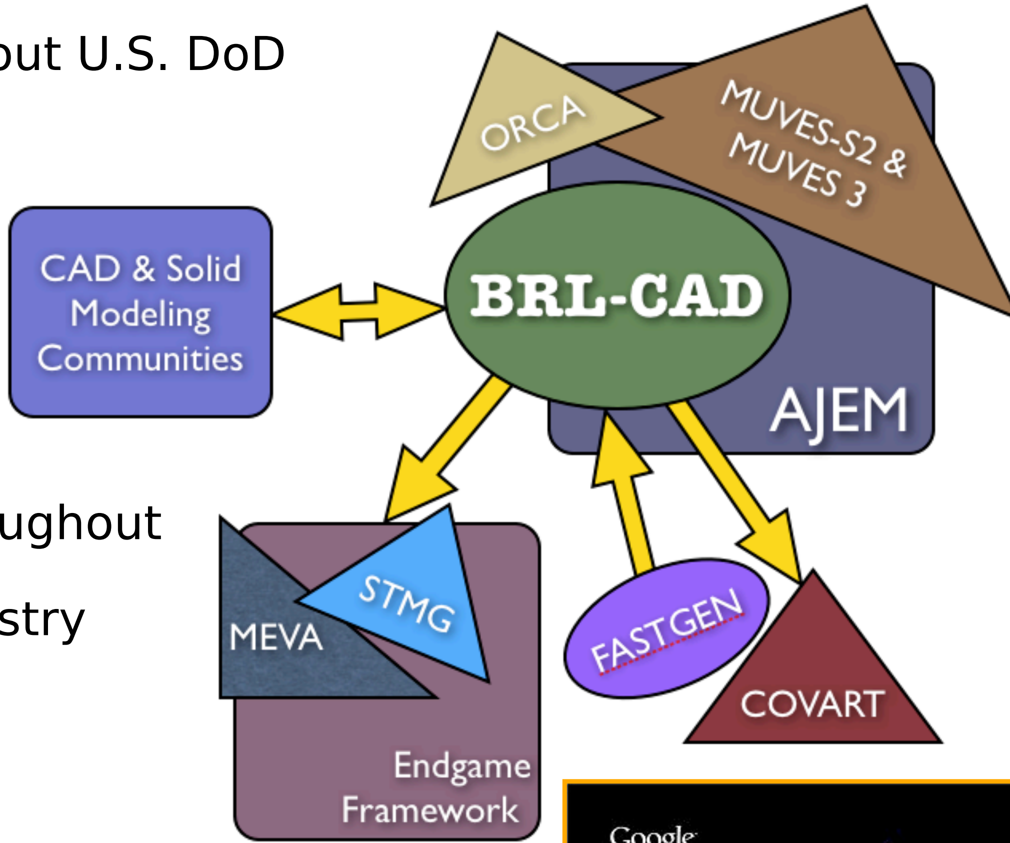
- Extensively cross-platform:
  - Windows,
  - Mac,
  - Linux,
  - UNIX,
  - … *from desktops to supercomputers*
- Became Open Source software in 2004
  - *Open code,*
  - *Open access,*
  - *Open standards*
  - *… It's free!*
- BRL-CAD became the first Open Source solid modeling system in production use under OSI*-approved license terms

* Open Source Initiative http://opensource.org

BRL-CAD Community

- SLAD V/L analysts and target describers
- MUVES and AJEM users throughout U.S. DoD
  - *Army, Air Force, Navy*
- International collaborations: Senior National Representative V/L Assessment Methodology (VLAM) working group
  - *United Kingdom, Germany, Netherlands, …*
- Extensive international ties throughout the Open Source community, academia, and commercial industry
  - *University of Utah*
  - *University of North Carolina at Chapel Hill*
  - *Johns Hopkins University*
  - *Texas A&M University*

  … and many others …
- Google Summer of Code
  - *Exclusive Open Source opportunity*



Main Topics

Background on BRL-CAD
What, who, why?

Tools & Techniques
for Geometry Analysis

Conversion to Open Source

Open Problems & Future Directions

Towards Open Source

… the long, manpower-intensive way …

- Lots of playing “Devil’s Advocate”
- Obtain common understanding
- Continually dismay fears
- Quantify benefits
- Obtain buy-in
- Keep going

The journey to make BRL-CAD Open Source began in **1998**.*

*Fortunately *much* has changed since then.

- Garner support within development team

They think it’s impossible, but okay.
They were *almost* right.

- Obtain initial support from management

They didn’t entirely understand, but didn’t see cause to stop the effort either..

Lots of questions.

RDECOM

ARL

(This is much better now thanks to persistent efforts from folks like Risacher and Wheeler, but still an open issue.)

- Become an open source legal expert

Many, *many* questions and **FUD** to dispel. Required constant shepherding, even amongst the dev team, to keep support going and fears at bay. Nobody knowledgeable to help navigate the issues.

Had to have answers ready for all of the worst possible "***what if***" scenarios.. Many licenses to read in detail.

This was the most meticulously tedious step.

BRL-CAD

TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

- Talk to Tech Transfer

  They really didn't understand, but were willing to listen. Their focus was the worst case possible. They wanted to see evidence of *others* releasing as Open Source.

  We seemed to be the first within the Army … (maybe first within DoD)
  So we were stuck.

- Talk to Security / Public Affairs

Fortunately, BRL-CAD was designed from the beginning to be devoid of sensitive data. Without analysis codes or military data embedded, there were no objections.

In fact, this would mean substantially less overhead for them! (No more licenses to review and file.)

- Talk to Information Assurance

They didn't really understand nor have authority to halt the effort, but were consulted for blessing regardless.

They had the usual questions and fears that we had heard and answered many times over by this point. They didn't have objections.

- Talk to Legal

It took **months** to get a sit down.
Persistence!

Finally, we got to meet with patent counsel and former NSA Secure Linux legal guru, Steve Bloor. (woot!)

He understood everything.

- Consider the legalities of licensing

Bloor made our options clear. Public domain was downright trivial but had major risks. NOSA-style was also easy but had many limitations (less than ideal).

Everything else required full rights. Copyright was even better.

- Make it happen, acquire rights

The U.S. Government has no default copyright. At least, not nationally… BUT, copyright can be established and assigned.

How? Contract modification, *substantial* derivative work, copyright claim on derivative, assignment.

(There's even standard clauses!)

- Get the okay from Management

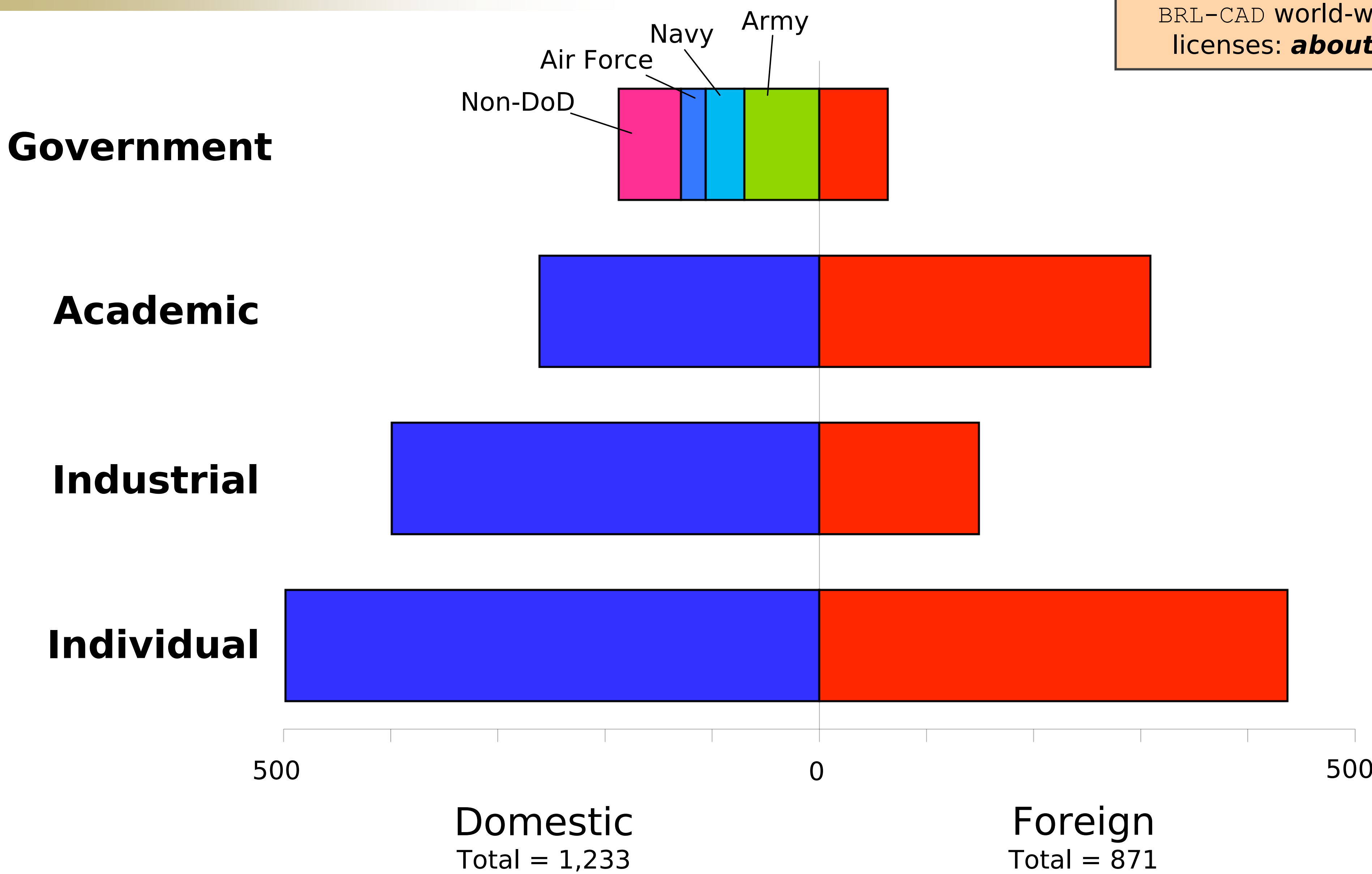
Once copyright was acquired, it finally became a "simple" matter of release approval from management.

We showed a substantial quantifiable process savings in addition to numerous "hypothetical" potential benefits. We'd done our homework. The code was ready.

After more than five years of persistent effort, we got the okay to release as Open Source in **December 2004.**

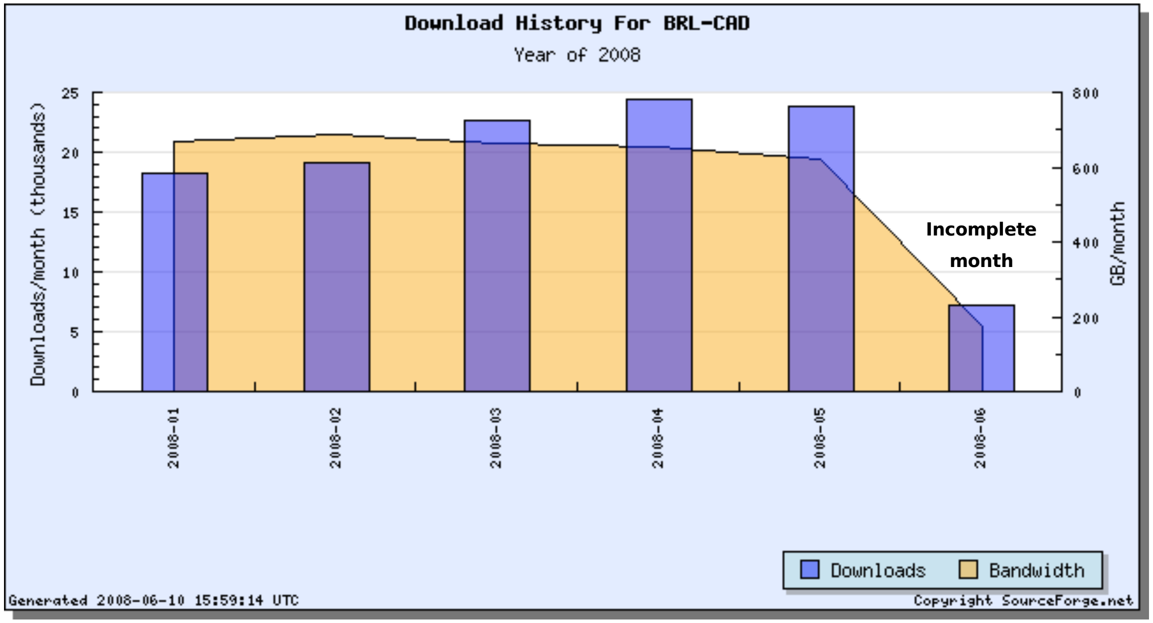
Before Open Source

BRL-CAD world-wide *site* licenses: ***about 2000***



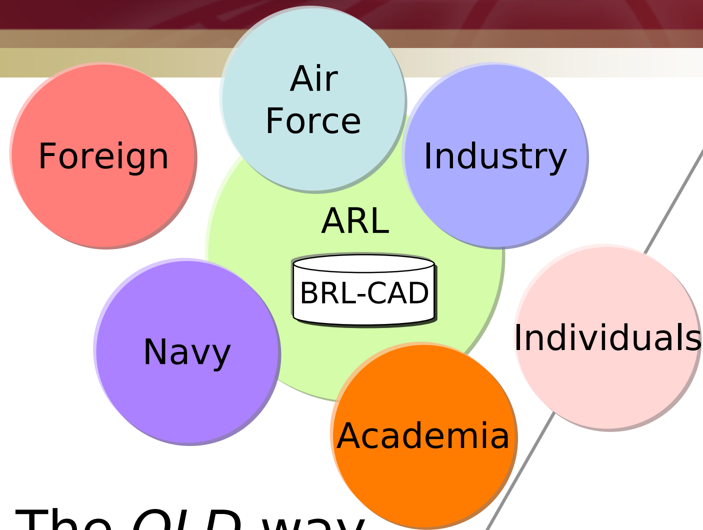
After Open Source

BRL-CAD downloads in first <u>month</u>: ***more than 2000***

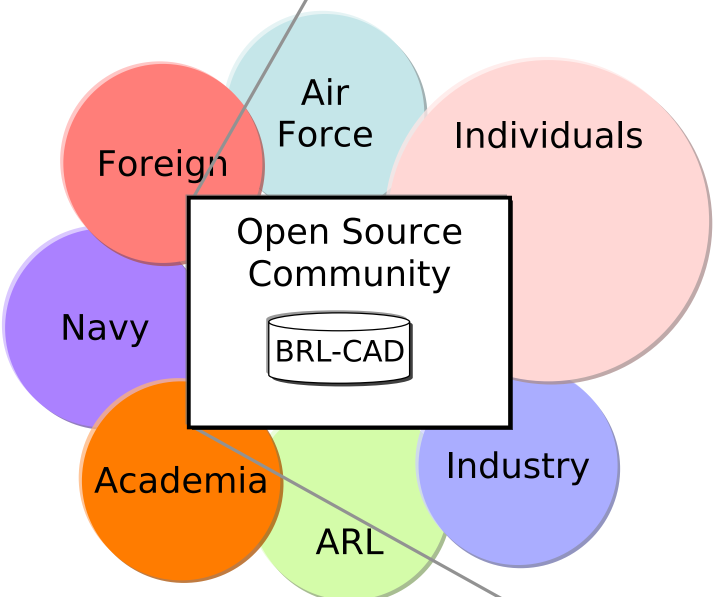


"the world's oldest source code repository"
– August 2007,
Robin Luckey, Ohloh Inc.

- More than **200,000** downloads and **2,000,000** website hits per year (as of 2010)
- Activity (both interest and development) is ***increasing*** year over year
- Presently receiving about **three-to-five staff-years** of contributed effort from the Open Source community including source code enhancements, bug fixes, documentation, website development, and more…
- Received roughly an additional staff-year of effort in 2008 and 2009 by being accepted into the **Google Summer of Code**

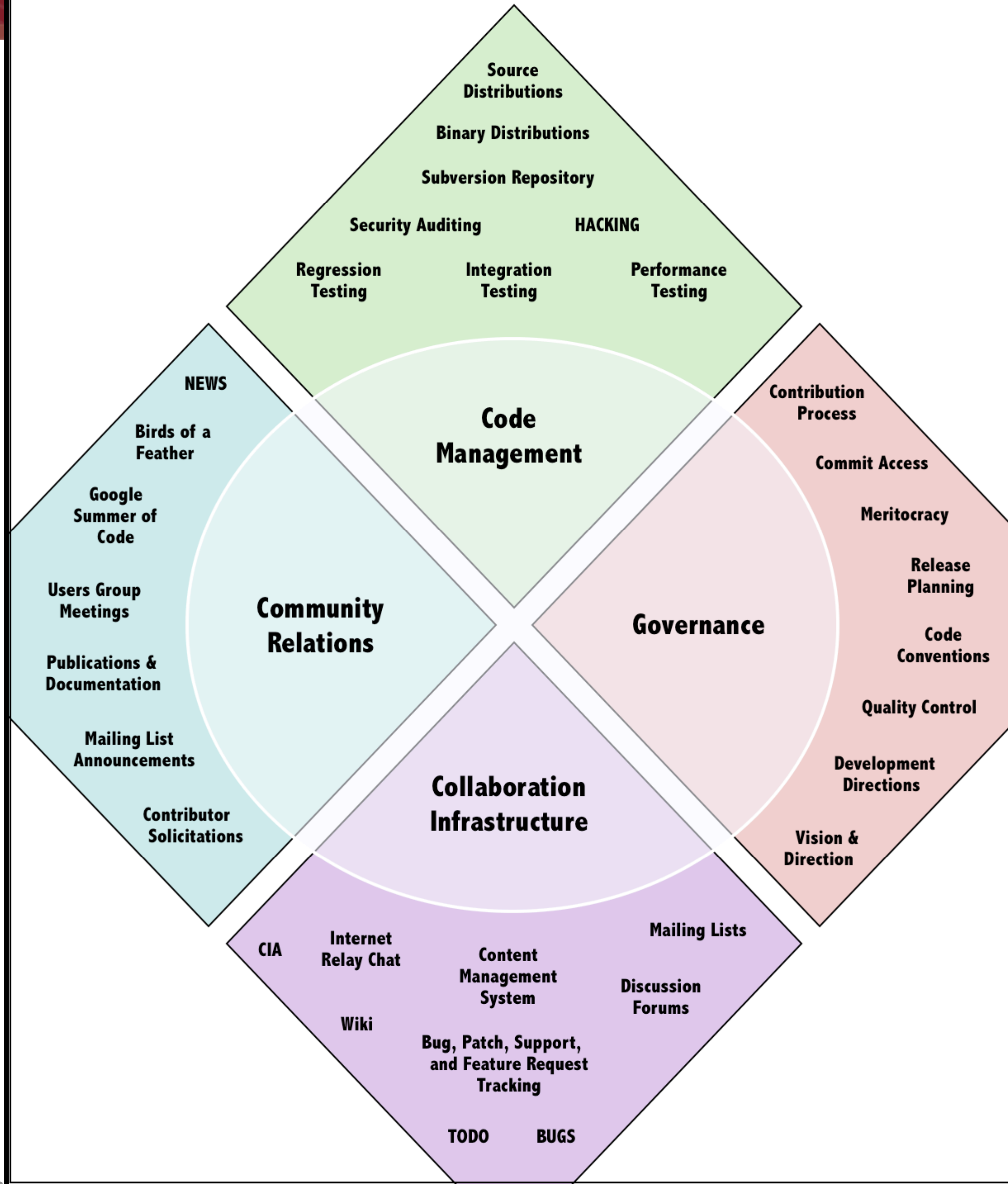


The *OLD* way



The *NEW* way

BRL-CAD Open Source Project Organization



Main Topics

Background on BRL-CAD
What, who, why?

Tools & Techniques
for Geometry Analysis

Conversion to Open Source

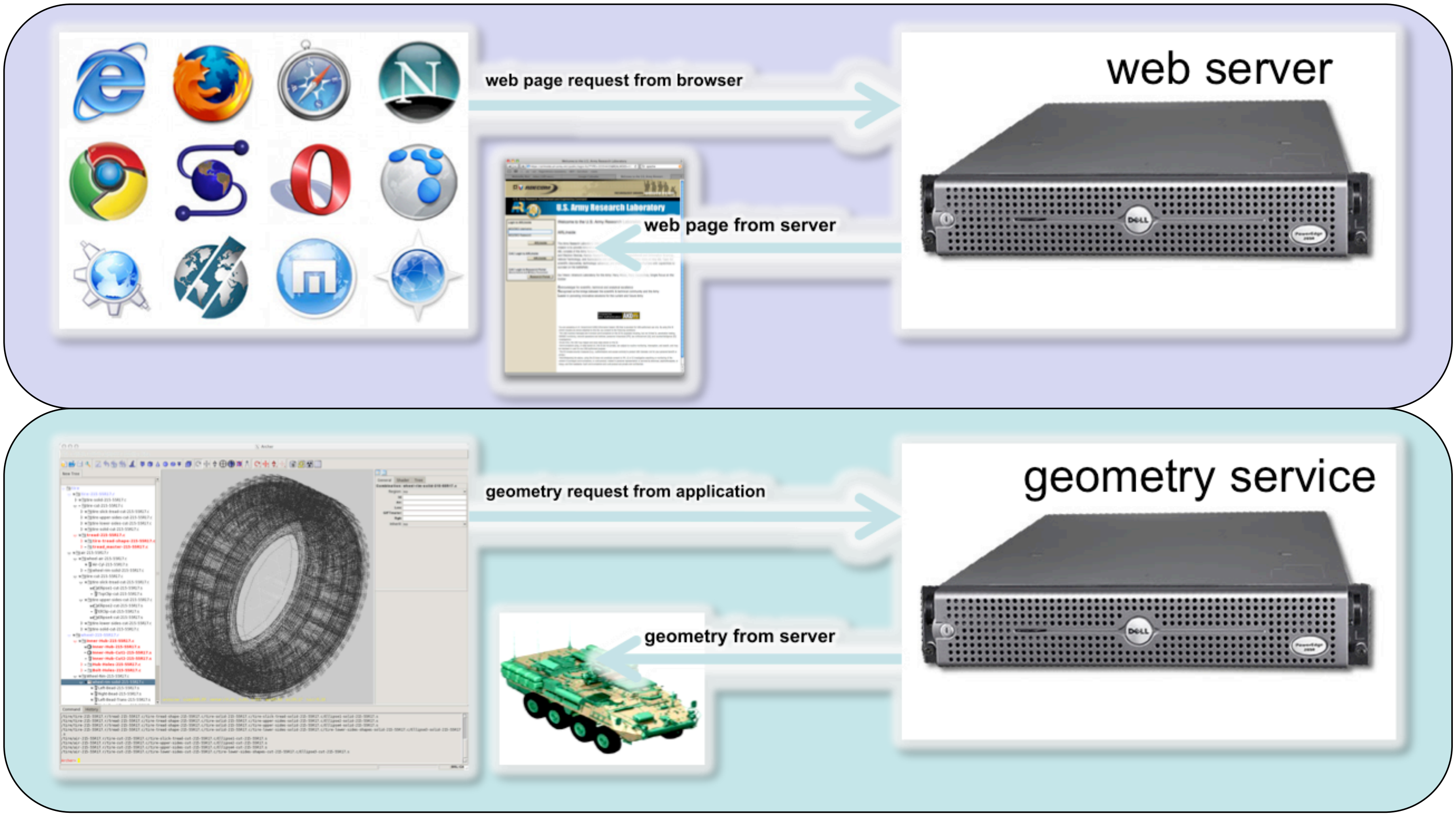
Open Problems & Future Directions

Open Problems

- NIH: "Not Invented Here"
- Perceptions of government code
- Open communication with the community
- Avoiding the waterfalls with agility
- Collaboration
- Meritocracy
- Diversity
- "**GLOSS**"

Blatant Solicitation
(er, Potential for Collaboration)

The *BRL-CAD Geometry Service* is an interface for accessing revision-controlled geometry from a data repository. Sitting on top of an embeddable *geometry engine*, the service provides and stores geometry data similar to how a web server delivers web pages.



Thank you!
Questions? Comments?

Christopher Sean Morrison
morrison@arl.army.mil
410-278-6678

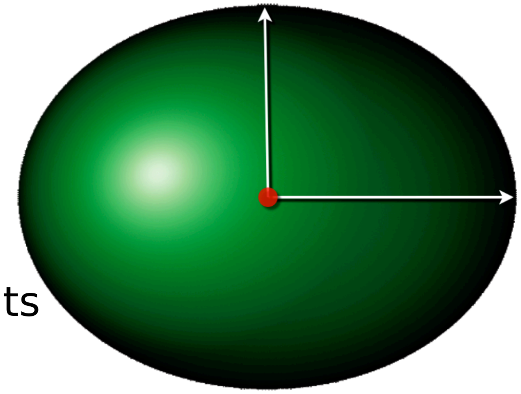
Several of the images contained within this presentation were created with the support and efforts of many individuals. The following deserve special recognition and thanks:

Mike Muuss
Lee Butler
Erik Greenwald
Cliff Yapp
Ron Bowers
Mike Gillich
Justin Shumaker
Stephen Kennedy
Karel Kulhavy
Edwin Davisson

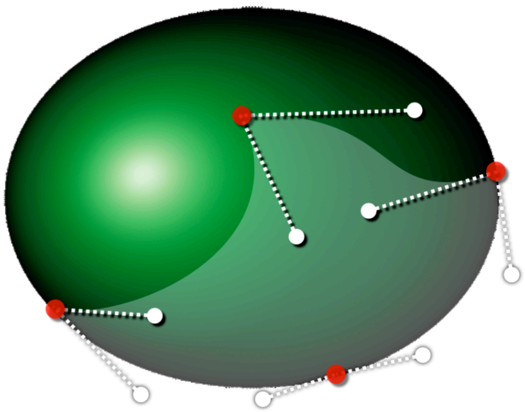
Additional Information about BRL-CAD

Geometric Representations

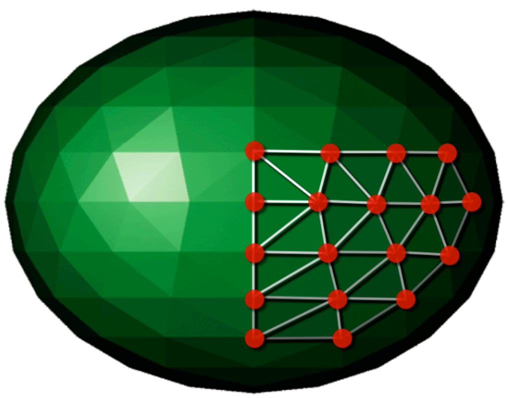
CSG implicits



BREP NURBS



BREP facets

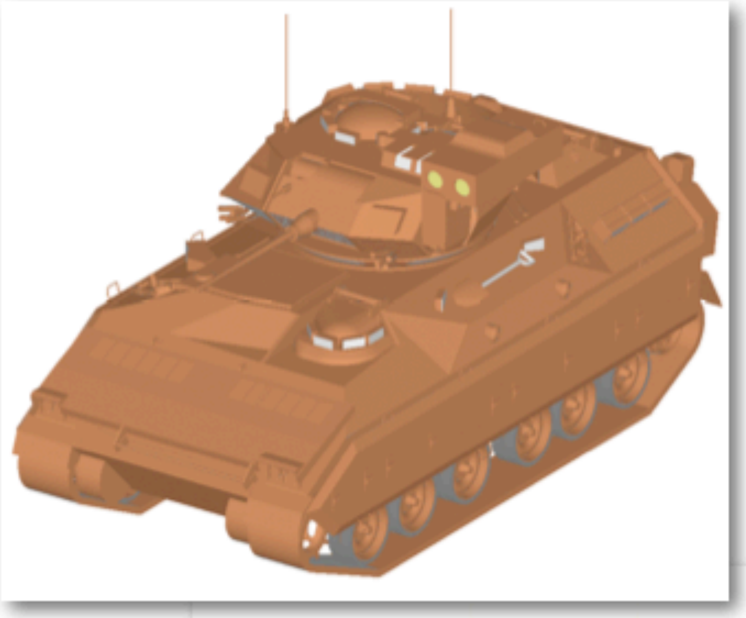


Geometric Definition	Applications of Use
4 values • Radius • Position	• Implicit primitives with constructive solid geometry (CSG) provide a representation format that is very compact and numerically robust (no cracks) — Solidity constraint is guaranteed making it well-suited for solid modeling and engineering analysis purposes
200 values • Surface • Patches • Knot • Values • Weights	• Spline surface boundary representations are prevalent in commercial CAD systems for their modeling flexibility — More recently they are also the subject of real-time ray tracing computer graphics research
1000 values or more (configurable) • Individual • Polygons • Vertices • Normal values	• Polygonal boundary models are commonly used by display systems (e.g., OpenGL and DirectX) for interactive rendering and real-time visualization — Many advancements have been made over the years on high-performance ray tracing of triangle models

Representation Data Implications

Exponential Geometric Growth ↓

CSG implicts	2 MB
BREP NURBS	20 MB
BREP facets	200 MB



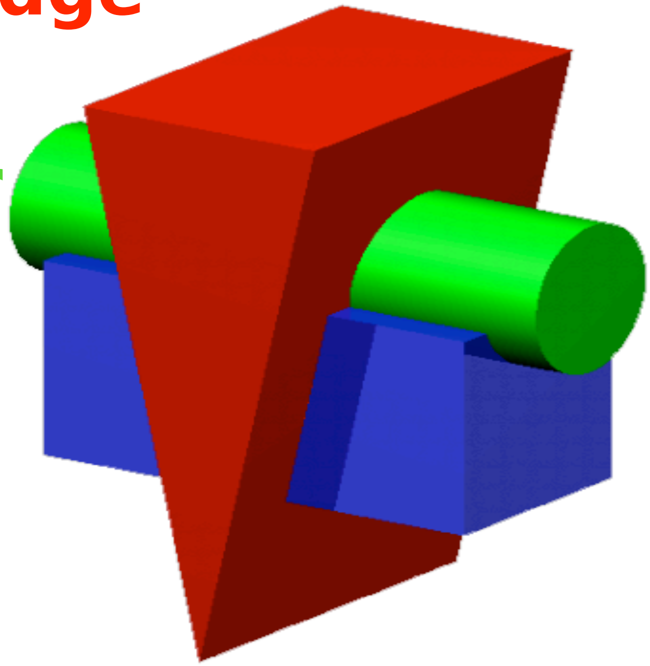
TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

Constructive Solid Geometry

wedge

cylinder

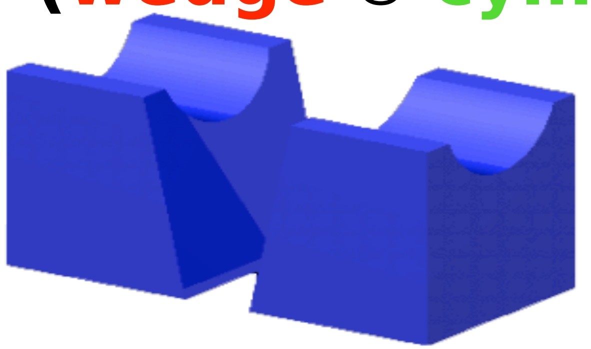
block



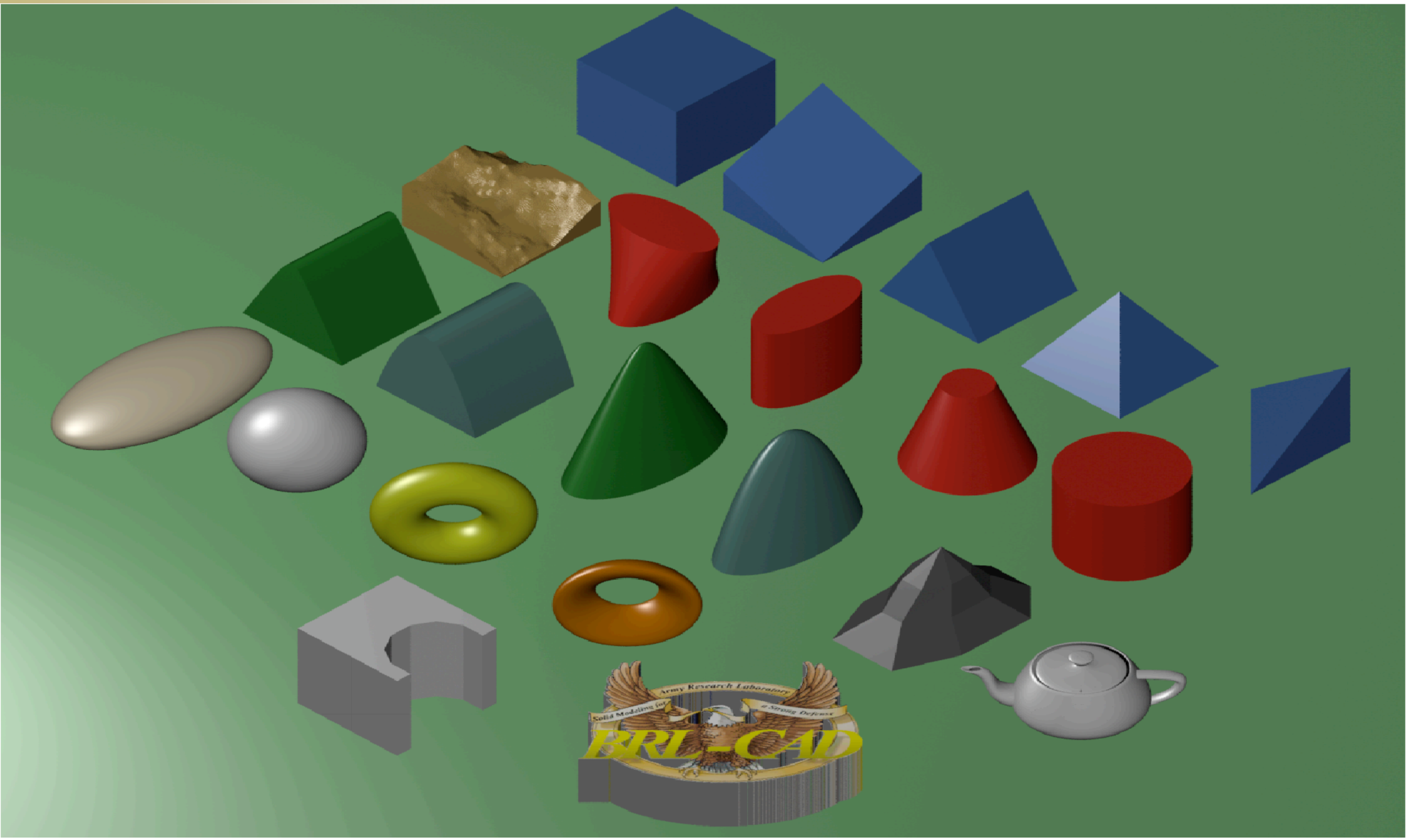
(wedge ∩ block) – cylinder

wedge – block – cylinder

block – (wedge ∪ cylinder)



Some of BRL-CAD's Primitives



Who is this guy?

Morrison is the open source project lead and principal software architect for BRL-CAD. His primary areas of expertise are in computer graphics, open source project management, solid modeling, system administration, information security, human-computer interface and interaction design, software testing, and computational geometry.

He is involved numerous open source projects including BRL-CAD, BZFlag, FTGL, and with ties throughout the open source community.

'brlcad' on Freenode IRC