



Mini-curso: introdução ao GMSH

Gustavo Peixoto de Oliveira, Dr.

gustavo.oliveira@uerj.br

gustavopeixotodeoliveira.com

Programa de Pós-Graduação em Eng. Mecânica - PPG-EM

Universidade do Estado do Rio de Janeiro - UERJ



Outline

- Considerações Iniciais
- Visão Geral
- Instalação e Execução
- Ferramentas Gerais
- Módulo Geometry
- Módulo Mesh
- Miscelânea
- Créditos
- Considerações Finais

Considerações Iniciais

Esta apresentação:

- destina-se a **usuários iniciantes**, com pouca ou nenhuma experiência com o Gmsh
- **não discute todas** as capacidades do software, nem é exaustivo
- segue a documentação do software, mas de modo **simplificado**

Visão Geral

- Descrição dos desenvolvedores: “Gmsh é um gerador de malha 3D para elementos finitos com um motor CAD e pós-processador integrados.”
- Entrada de dados:
 - built-in scripting language (loops, conditionals, arrays, external calls)
 - GUI

Modelo “pinched drop” @gustavopeixoto

script

```
/* Defaults */

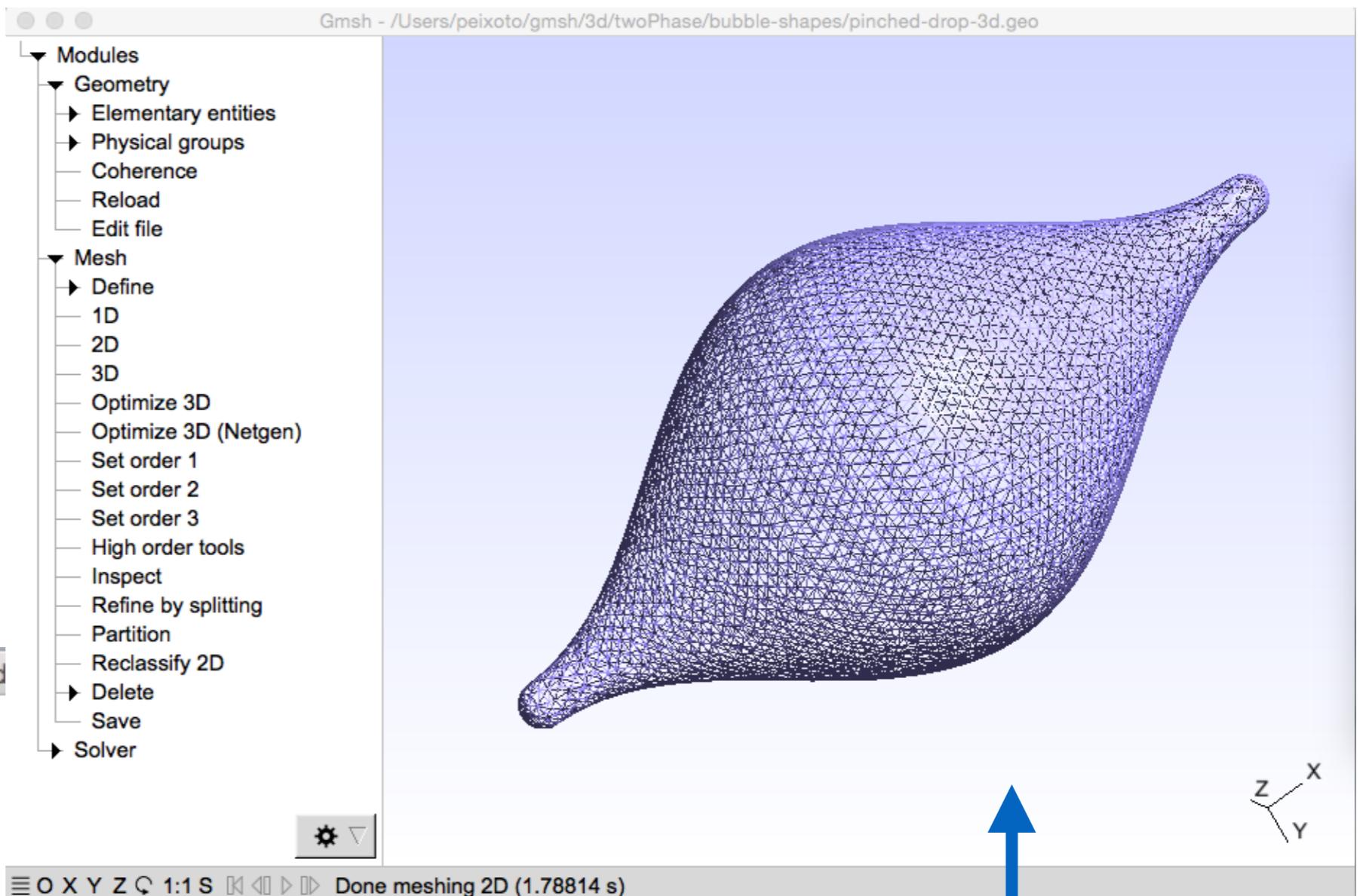
Geometry.Surfaces = 1; // 
Geometry.Points = 0;
Geometry.Lines = 0;
Mesh.SurfaceNumbers = 0;
Mesh.Normals = 20;

/* Mesh parameters */

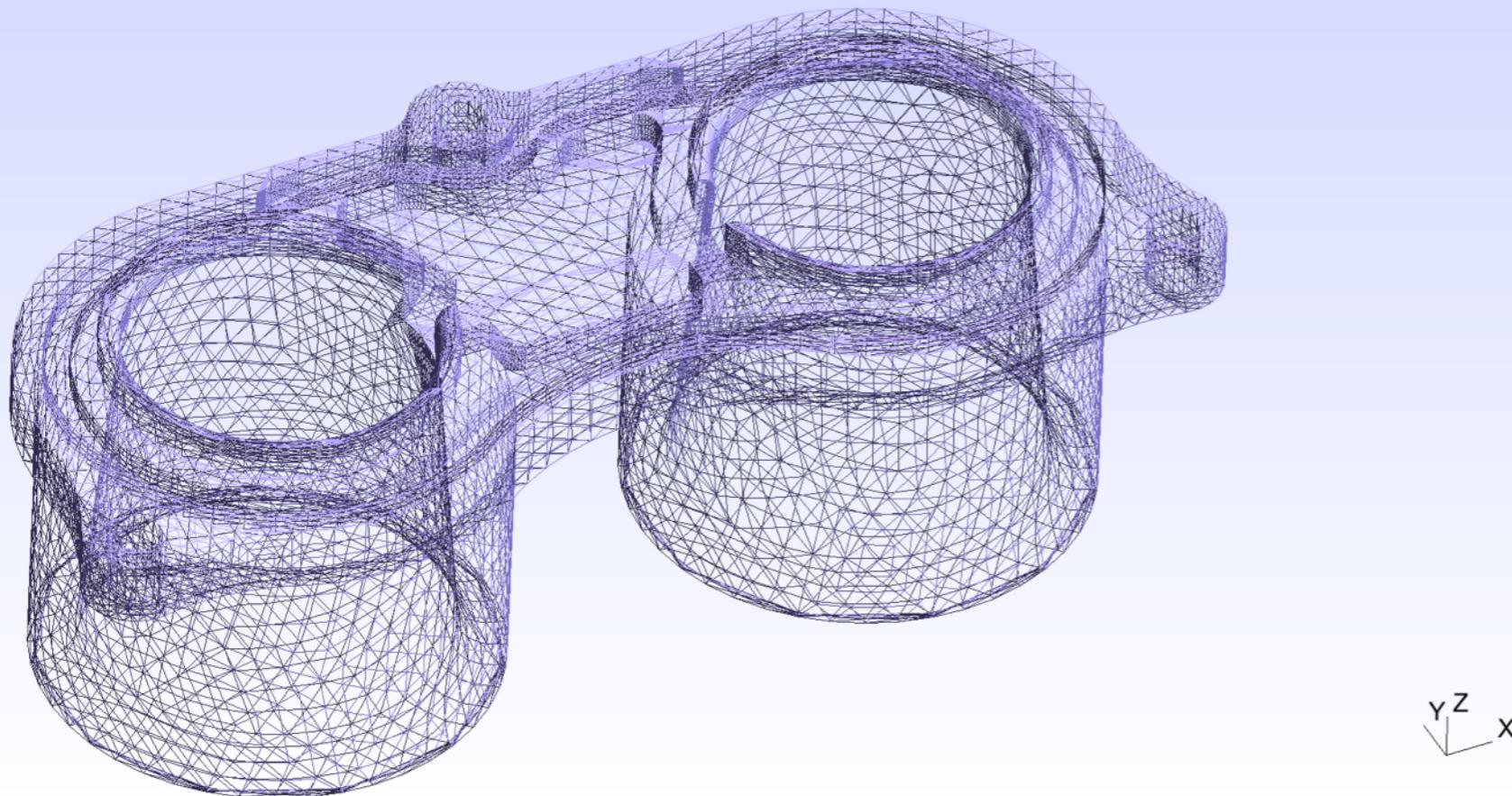
x0 = 0.0;
y0 = 0.0;
z0 = 0.0;
c1 = 0.08; // refinement body
c2 = 0.005; // refinement pinch

/* alpha-lambda: depend on jet density, surface tension and R0 */

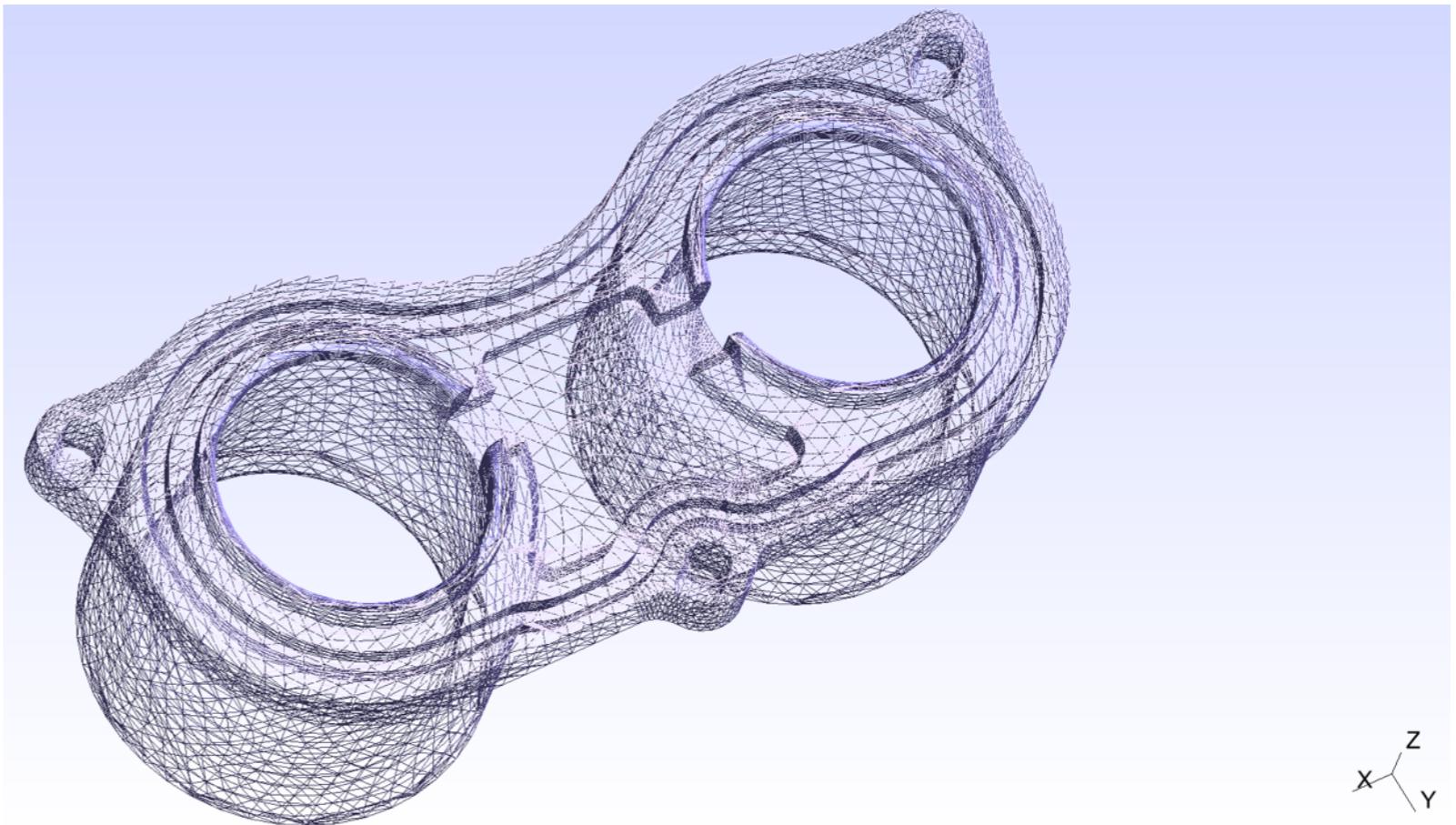
R0 = 0.5; // jet initial radius
DeltaU = 0.001; // velocity disturbance amplitude
alpha = 0.0270; // max growth rate (Middleton)
lambda = 4.44; // wavelength of most unstable mode
amp = DeltaU*Pi/(alpha*lambda); // amplitude
t0 = 30; // breakup time
r1 = 0.25*R0; // pinch radius
L = lambda; // periodic length (drop)
```



GUI

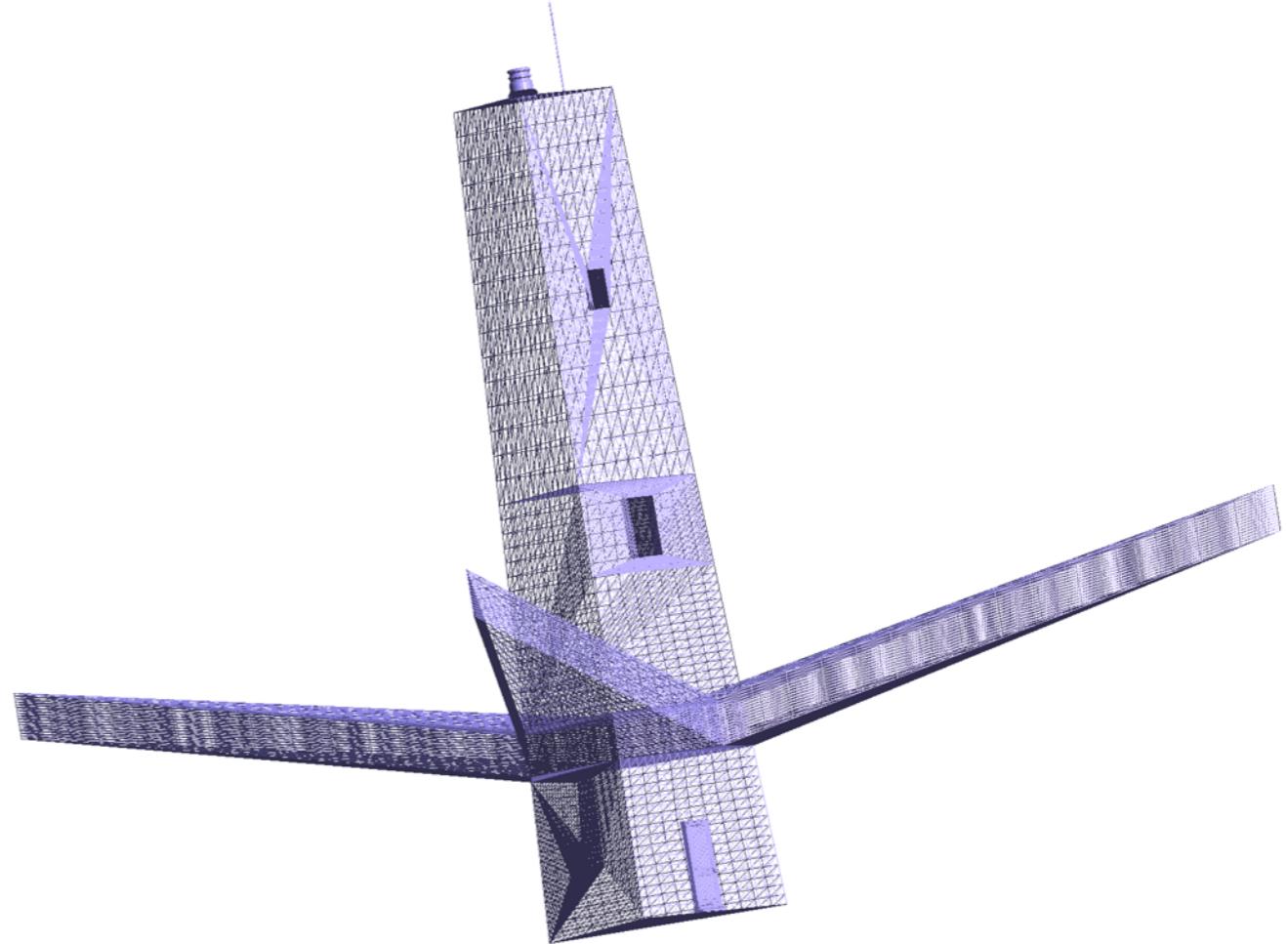
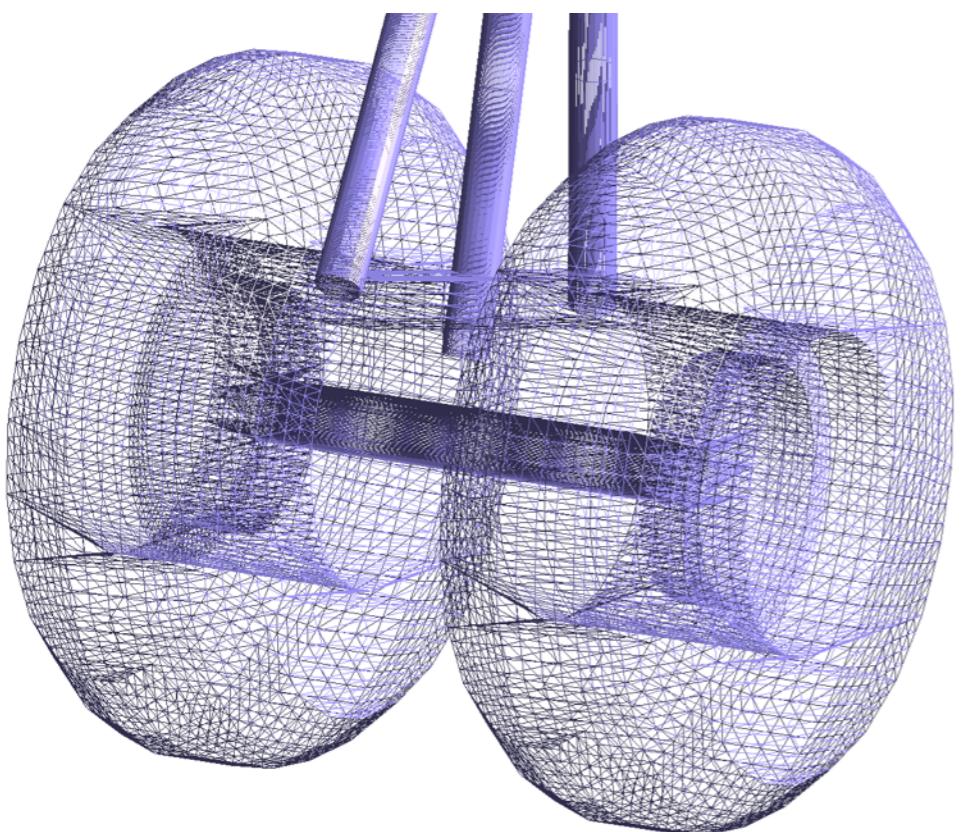


Modelo
"throttle body
adapter-
Honda CBR600"
@sudoshidesign



Modelo
"Farol de Cabo Branco -
João Pessoa, PB, Brasil"

@Sketchup Warehouse 3D



Modelo
"Boeing 717-1990"
@Sketchup Warehouse 3D



Módulos (paradigma de Modelagem Computacional)



Geometry

definição
de
entidades
geométricas:

- pontos
- segmentos
- círculos
- elipses
- splines, etc.

Mesh

geração
de
malha de
EF:

- triângulos
- tetraedros
- prismas
- hexaedros
- pirâmides

Solver

interface
para
solvers
externos

- "getDP"
(padrão)

Post-Processing

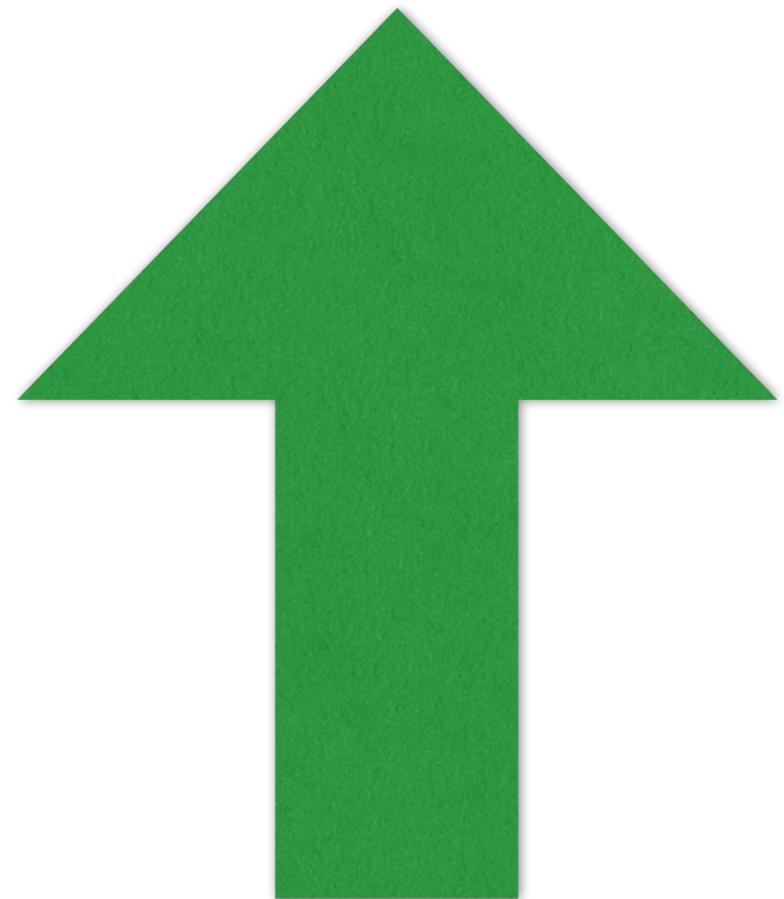
manipulação
e
visualização
de dados

- escalares
- vetores
- tensores
- animações, etc

cobertos neste tutorial

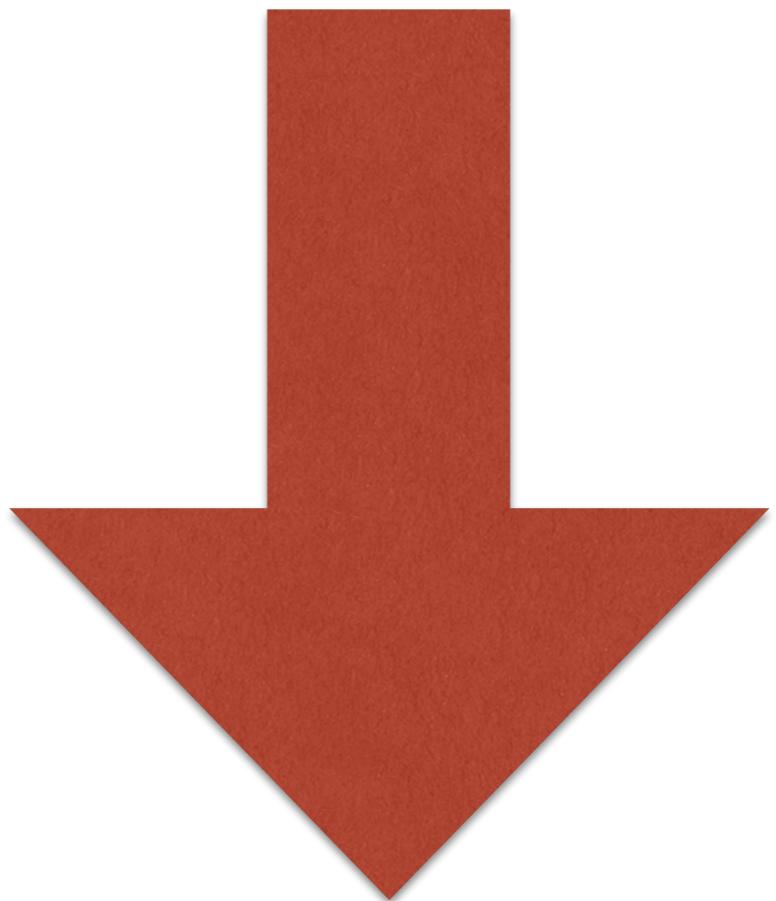
Qualidades do Gmsh

- Geometrias simples
- UDFs (user-defined functions)
- Elementos em {1,2,3}D para modelos CAD
- Mecanismos para refinamento adaptativo
- Extrusão de geometrias e malhas
- Exportação de dados para diversos formatos
- Execução em linha de comando
- Customização
- Portável (Windows, Mac OS and UNIX)
- Free!



Fraquezas do Gmsh

- Motor CAD interno limitado para geometrias complexas
- Malha geradas são apenas conformes
- Interface limitada
- Scripting limitado (escopos simples; sem variáveis locais)
- Inexistência de uma função global de "desfazer" ("undo") - correção via texto



Pode ser usado de 3 maneiras

1. Como programa gráfico com uma GUI padrão

Ver [screencasts](#)

2. Como um programa baseado em scripting

3. Como uma biblioteca

Ver [API](#)

3º nível: API; desenvolvimento

2º nível: script; sintaxe; comandos

1º nível: usuário padrão; GUI

Formatos de arquivo: open/save as...

CAD/Geometry

.geo .opt .brep .step



Mesh
.msh .inp .celum .diff
.unv .ir3 .med .mesh
.mail .bdf .p3d .stl
.vtk .ply2 .su2

 ABAQUS

 ParaView

Várias opções:
free or paid

Pós-Processamento

.pos .rmed .txt

Imagen

.eps .jpg .tex .pdf
.ps .ppm .svg .yuv
.png .pgf .mpg



Instalação

e

Execução

- Onde obter? [Gmsh webpage](#)
- GUI ou linha de comando?
 - **GUI**: instalação direta por executável;
 - **Linha de comando**: compilação/construção do código-fonte (CMake+PREFIX)

Abrindo em modo interativo



via interface
gráfica

```
tenderspeech-mac-mini:MacOS peixoto$ pwd
/Applications/Gmsh.app/Contents/MacOS
tenderspeech-mac-mini:MacOS peixoto$ ls
gmsh
tenderspeech-mac-mini:MacOS peixoto$ ./gmsh &
[3] 7647
tenderspeech-mac-mini:MacOS peixoto$ |
```

A screenshot of a terminal window titled "bash". The window shows a command-line session. The user is in their home directory ("~/MacOS") and lists the contents ("ls"). They then run the "gmsh" command with the "&" operator to run it in the background. The process ID "[3] 7647" is shown. The terminal prompt "tenderspeech-mac-mini:MacOS peixoto\$ |" is at the bottom.

via terminal (diretório-base)

Abrindo em modo não-interativo (ex.:em máquina remota)

```
gcpoliveira@ORION:~$ gmsh -help
Gmsh, a 3D mesh generator with pre- and post-processing facilities
Copyright (C) 1997-2011 Christophe Geuzaine and Jean-Francois Remacle
Usage: gmsh [options] [files]
Geometry options:
  -0                         Output unrolled geometry, then exit
  -tol float                  Set geometrical tolerance
  -match                      Match geometries and meshes
```

A screenshot of a terminal window titled "bash". The user runs the "gmsh -help" command. The output provides information about Gmsh, its copyright, usage, and various command-line options. The "Geometry options" section is expanded, showing three options: "-0", "-tol float", and "-match". The descriptions for these options are: "-0" outputs unrolled geometry and exits, "-tol float" sets geometrical tolerance, and "-match" matches geometries and meshes.

Comandos úteis (linha de comando)

- Abrir arquivo:
gmsh <filename> // .geo, em geral
- Malhar "n"D:
gmsh -n <filename> // n=1,2,3; retorna um .msh
- Malhar e renomear saída:
gmsh -n <filename> -o <filename_out> // retorna um .msh
- Malhar restringindo nível mínimo(máximo) de refinamento
gmsh -n -clmin(clmax) 0.01(0.8) <filename> // compr. caract. mín.(máx)=0.01(0.8)
- Ajuda:
gmsh -help

Ferramentas Gerais

- Comentários (scripting): estilo C/C++

```
/* block comment */  
// line comment
```

- Tipos de dados (não há tipo “int”)

real
string

Expressões úteis (floating point)

Expressão	Descrição
<pre>a = 1; b = 2.0; c = "my_string";</pre>	float float string
<pre>x~{1} = 0.1; (x_1 = 0.1;)</pre>	"index-notation"
<pre>ls = {1,2.0,3,4.0};</pre>	list
<pre>lsz = #ls1[];</pre>	list size
<pre>ls[i] i.e., ls[0] = 1; ls[1] = 2.0;</pre>	i-th component of list $0 \leq i < lsz$

Expressão	Descrição
<pre>StrCmp(e,f)</pre>	string comparison returns integer $\{>,=,<\}0$ if $e\{>,=,<\}f$
<pre>Exists(a)</pre>	is variable a defined?
<pre>FileExists("/foo.bar")</pre>	is this file defined?
<pre>StrCat(stra,strb) StrCat("/home","dir")</pre>	string concatenation
<pre>GetValue(str_call,val) GetValue("Give me a number", 43.21)</pre>	pop-up window for user's input data; if nothing, sets a standard value

Ver *expressions.geo*

Expressões úteis (char)

Expressão	Descrição
<code>tdy = Today;</code>	current date/time
<code>fn = StrPrefix("/zoo/ foo.bar"); i.e., "/zoo/foo"</code>	gets file without extension
<code>ffn = StrRelative("/zoo/ foo.bar"); i.e., "foo.bar"</code>	gets file name
<code>sstr = Str("alfa","beta", "gamma","delta");</code>	adds newline after each string
<code>sctr = StrChoice(1>2, "beta","gamma"); i.e., returns 'gamma'</code>	evaluates expression, then chooses

Expressão	Descrição
<code>sprf = Sprintf("epsilon"); i.e., prints 'epsilon'</code>	C-style Sprintf
<code>sprf2 = Sprintf("value%g, %g",a,b);</code>	ditto
<code>genv = GetEnv("HOME");</code>	get specified environment variable
<code>get = GetString("Tell me", "this is it");</code>	pop-up window for user's input data; if nothing, sets a standard value
<code>strpl = StrReplace("find_us","us ","yourself");</code>	replaces substring inside a string

Operadores

unary-left	-	minus
	!	logical not
unary-right	++	post-incrementation
	--	post-decrementation
	^	exponentiation
	*	multiplication
	/	division
	%	modulo
	+	addition
	-	subtraction
binary	==	equality
	!=	inequality
	>	greater than
	≥	greater or equality
	<	less
	≤	less or equality
	&&	logical "and"
		logical "or"
ternary-left	?	
ternary-right	:	decision; used as ?:

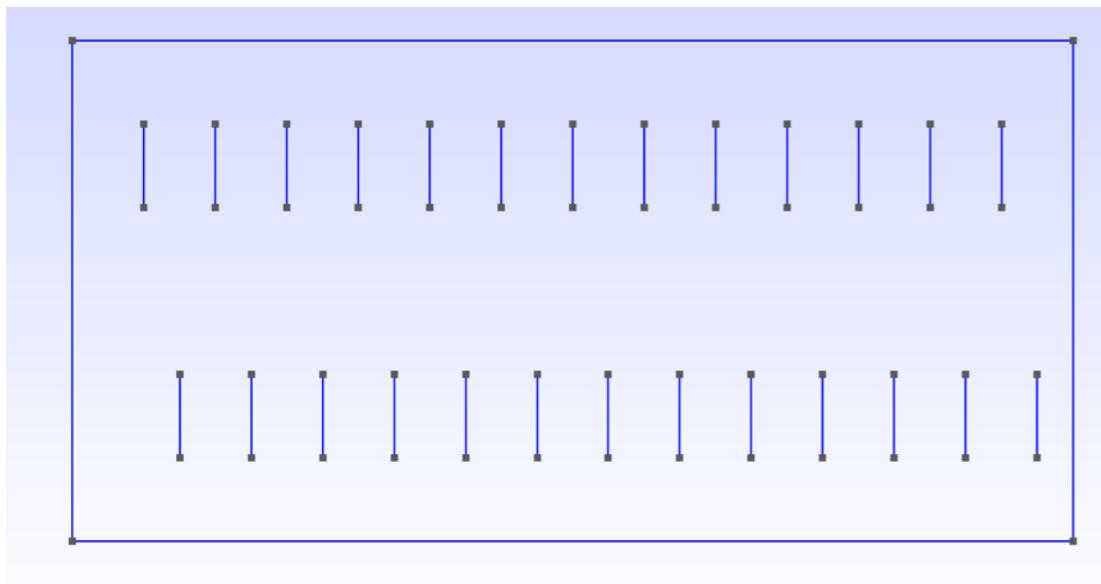
Funções predefinidas

Acos(a)	-1 <= a <= 1
Asin(a)	returns - pi/2 <= a <= pi/2
Atan(a)	returns - pi/2 <= a <= pi/2
Atan2(a)	- pi <= a <= pi
Ceil(a)	rounds to nearest int
Cos(a)	cosine
Cosh(a)	hyperb. cosine
Exp(a)	exponential
Fabs(a)	absolute value
Fmod(a)	modulus
Hypot(u,v)	Sqrt(u*u + v*v)
Log(a)	a > 0
Log10(a)	a > 0
Modulo(a)	ditto Fmod(a)
Rand(a)	random 0 < R < a
Round(a)	rounds
Sqrt(a)	square root, a > 0
Sin(a)	sin
Sinh(a)	hyperb. sin
Tan(a)	tan
Tanh(a)	hyperb. tan

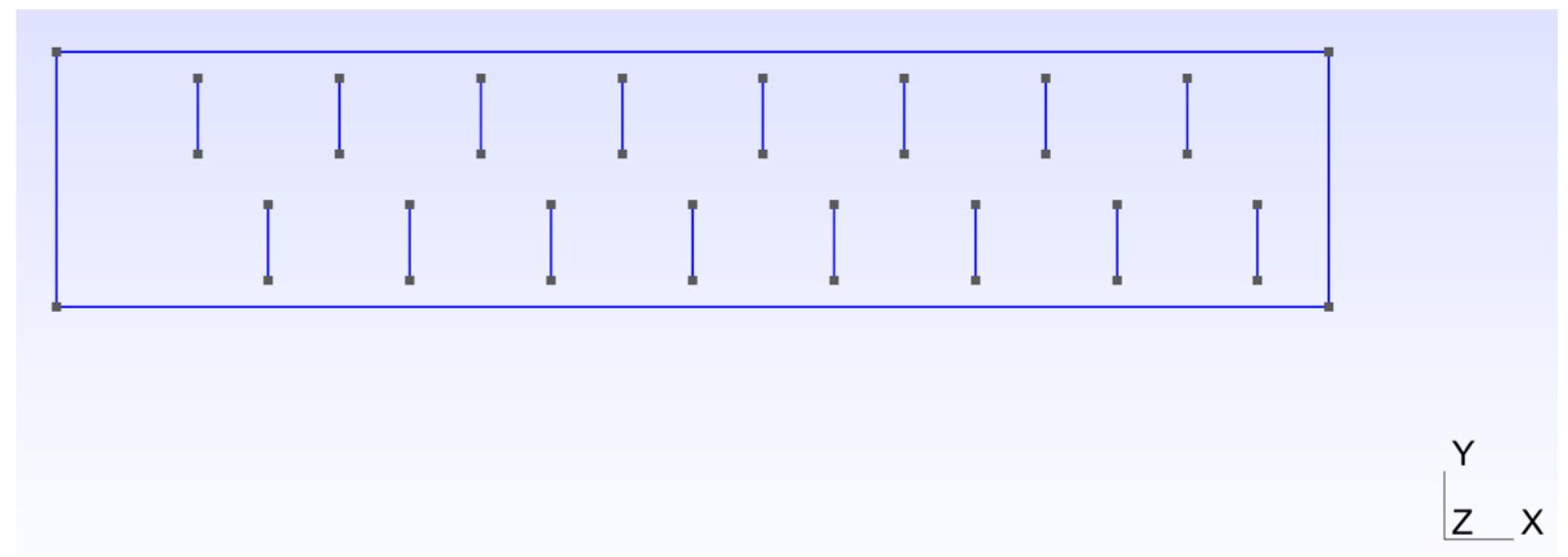
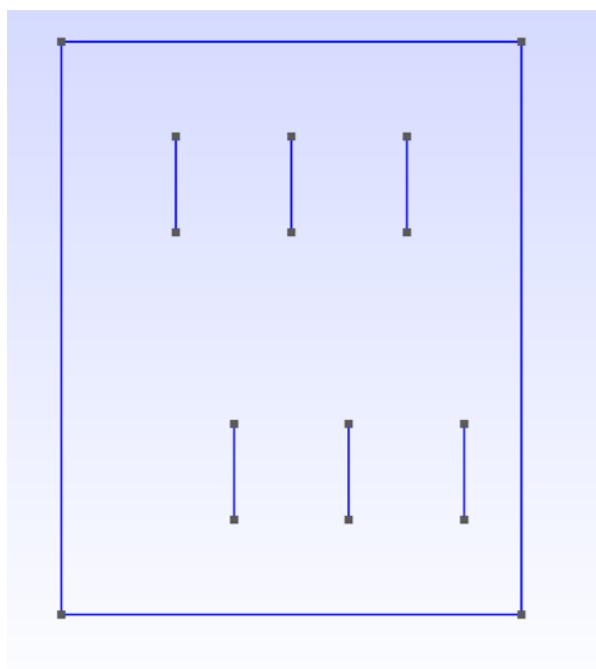
User-Defined Functions

- Funções criadas pelo usuário
- Ideais para automatização de rotinas
- Combinacão de recursos
- Customização geométrica

ex: Modelo geométrico customizável (comprimento, espaçamento, largura) p/ aletas intercaladas



Ver udfbuildstripes.geo



Loops I - ()

For (expr:expr) // iterates each 1

...

EndFor

For (expr:st:expr) // iterates each
increment st (pos. or neg.)

...

EndFor

Loops II - {}

For string In {expr:expr} // iterates each 1

...

EndFor

For string In {expr:st:expr} // iterates
each increment st (pos. or neg.)

...

EndFor

Condicionais

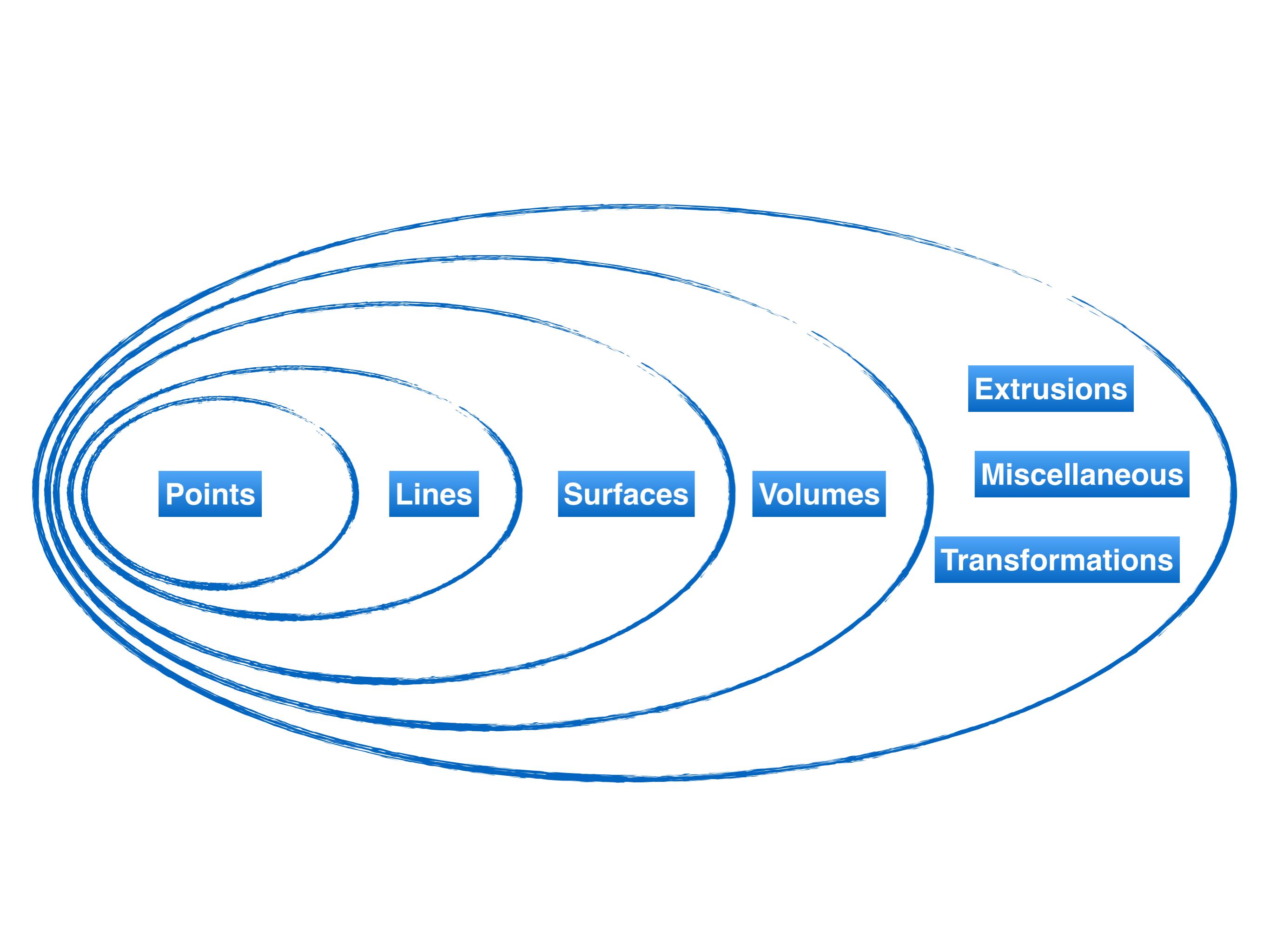
```
If (expr) // if true  
{scope}  
EndIf  
  
// there's no else or elseif conditions
```

Alguns comandos gerais

Pi	3.1415926535897932
Cpu	current CPU time (in seconds)
Memory	current memory usage (in Mb)
TotalMemory	the total memory available (in Mb)
newp	next available point number
newl	ditto line number
news	ditto surface number
newv	ditto volume number
newll	line loop number
newsI	surface loop number
newreg	region number

string = {};	empty list
Abort;	aborts the current script
Exit	exits Gmsh.
Printf(expr)	print in the window - C-like
Delete <entity>	delete specified entity
RefineMesh	refine mesh by splitting elements
Include <char>	include a file; if not full path, append to current dir

Módulo Geometry



Points

Lines

Surfaces

Volumes

Extrusions

Miscellaneous

Transformations

Entidades elementares

Um único número de identificação (sempre positivo) para cada objeto:

- Point
- Line
- Surface
- Volume

Entidades podem ser transformadas com certos comandos, tais como:

- Translate
- Rotate
- Scale
- Symmetry
- Delete

Grupos de entidades geométricas elementares podem ser definidos como “grupos físicos”

- Não podem ser modificadas por comandos geométricos
- Podem modificar orientações
- Facilitam o malhamento no sentido de serem entidades únicas
- Também recebem um único número positivo identificador

Regra sintática geral para a definição de entidades geométricas

- “Se uma expressão **define** uma nova entidade, ela vem entre parênteses. Se uma expressão **refere-se** à uma nova entidade, ela vem entre chaves.”

Pontos geométricos

define

```
Point(expr) = {x,y,z,c1};
```

// cria um ponto com coordenadas x,y,z e
tamanho característico c1

// expr é o identificador do ponto; o
primeiro ponto do modelo seria

```
Point(1) = {x,y,z,c1};
```

// c1 relaciona-se ao comprimento
característico local desejado para o
elemento de malha

Pontos físicos

```
Physical Point(expr) = {id};
```

// define o ponto id como físico

// expr é o identificador do ponto físico, podendo ser um número ou uma string. Podemos definir

```
Physical Point(10) = {1};
```

// ou

```
Physical Point("PhysicalPoints") = {1};
```

// para que o ponto 1 torne-se físico.



referencia

// Se “PhysicalPoints” definisse um grupo de vários pontos, são formas usuais:

Physical Point(“PhysicalPoints”) = {1,2,32,16,1232}; // aleatoriamente

Physical Point(“PhysicalPoints”) = {1:10}; // sequencialmente

Physical Point(“PhysicalPoints”) = {points[]}; // uma lista

Physical Point(“PhysicalPoints”) = {ps[], points[], 1:10, 1232}; // vários

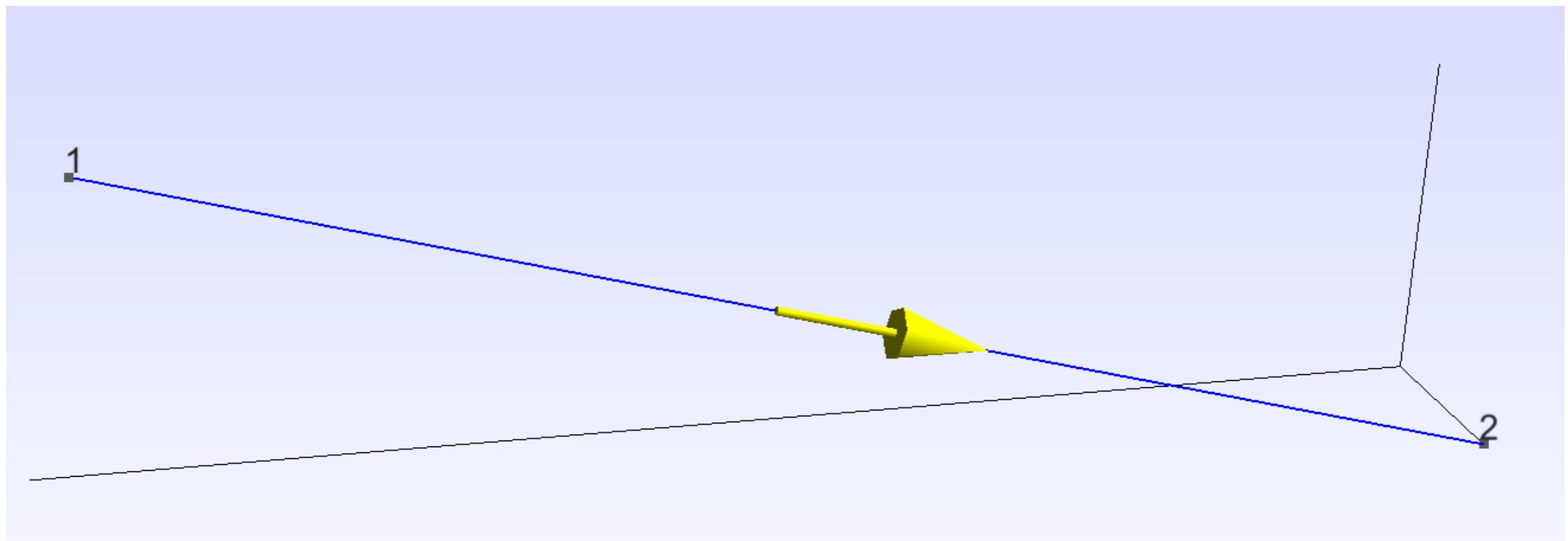
Linhas geométricas

define

```
Line(expr) = {p1,p2};
```

// cria um segmento de reta orientado de p1 a p2

// expr é o identificador da “reta”



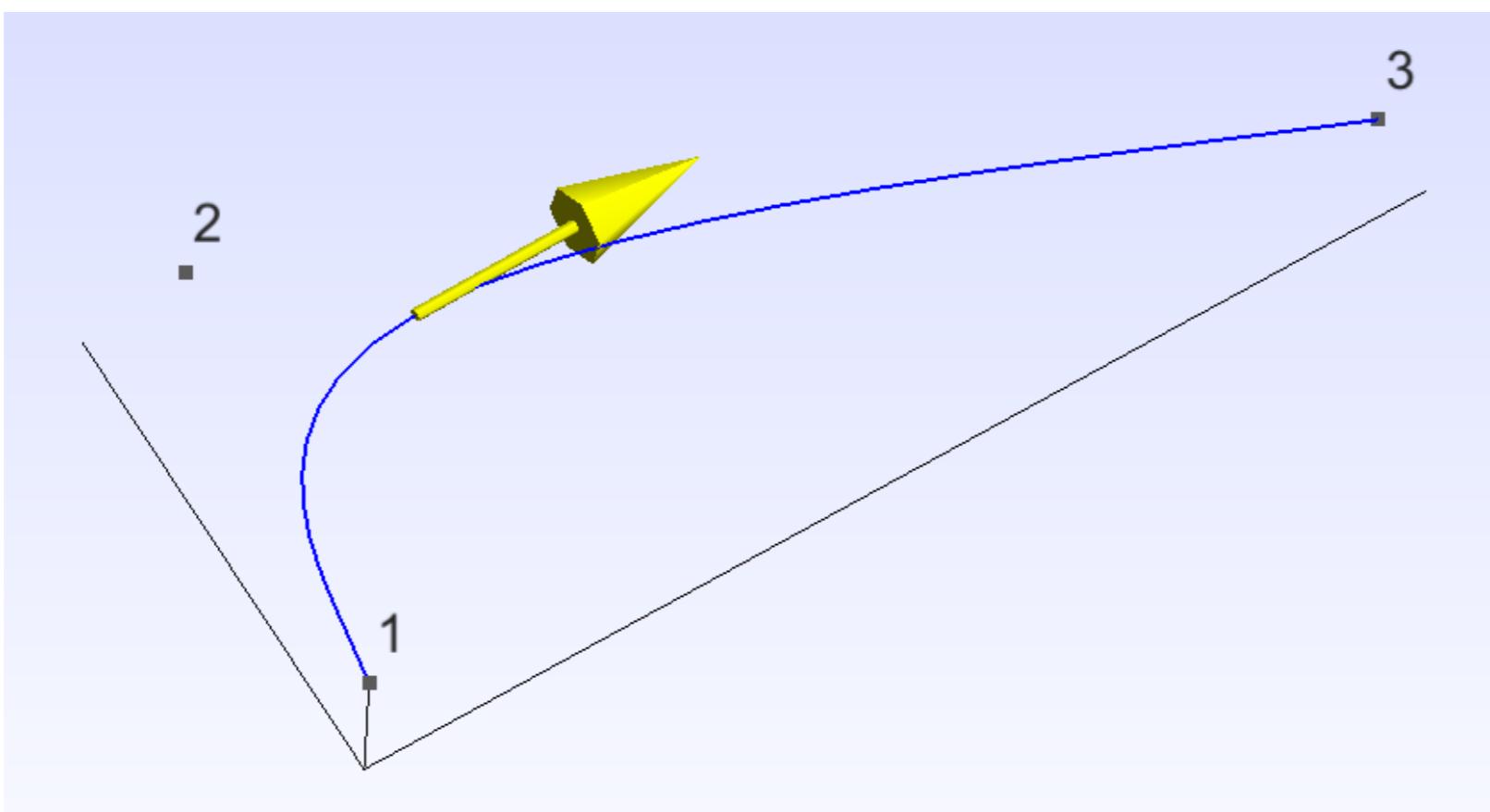
Splines/BSplines

```
Spline(expr) = {p1,p2,p3};
```

```
BSpline(expr) = {p1,p2,p3};
```

```
// cria uma spline/b-spline com pontos de controle  
p1,p2,p3.
```

```
// expr é o identificador da spline/b-spline
```

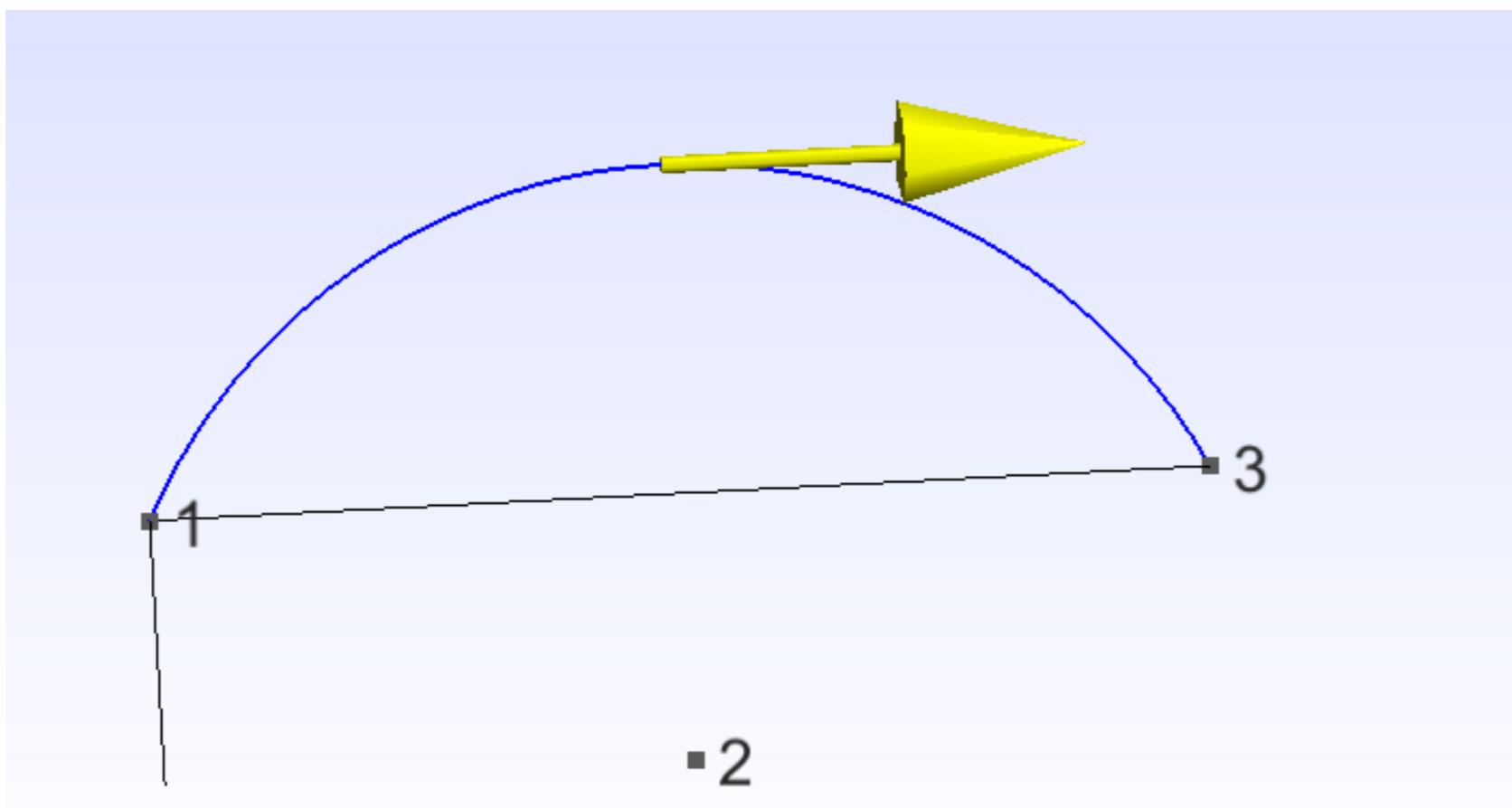


Arcos/Círculos

```
Circle(expr) = {p1,p2,p3};
```

// cria um arco de 180º orientado de p1 a p3 com centro em p2. Um círculo pode ser obtido construindo um arco complementar.

// expr é o identificador do arco

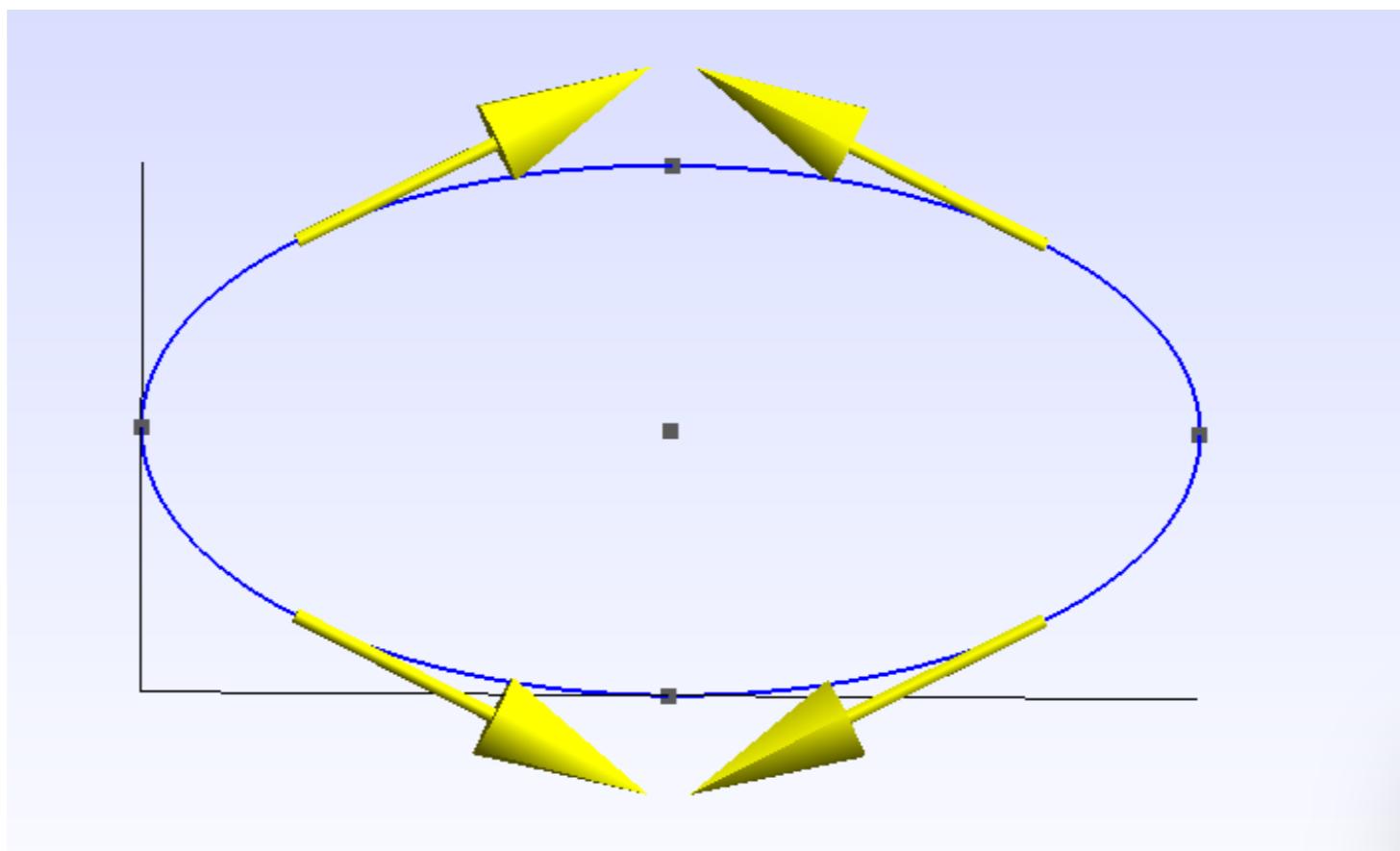


Elipses

```
Ellipse(expr) = {p1,p2,p3,p4};
```

// cria um arco de elipse: p1 é o ponto inicial; p2 o centro; p3 um ponto sobre o eixo maior; p4 um ponto sobre o eixo menor. Para uma elipse completa, ou se criam os arcos complementares, ou se trabalha com transformações. (ex. c/ orientação não preservada)

// expr é o identificador do arco de elipse



Line Loops

```
Line Loop(expr) = {id_lines};
```

// cria um loop orientado passando os identificadores das entidades do tipo Line como parâmetros.

// expr é o identificador do loop

// id_lines são os identificadores dos objetos Line

Linhas físicas

```
Physical Line(expr) = {id};
```

// define objeto Line como físico

// expr é o identificador do objeto físico,
// podendo ser um número ou uma string. Podemos
// definir

```
Physical Line(10) = {1};
```

// ou

```
Physical Line("PhysicalLines") = {1};
```

// just as for points

Physical Line("PhysicalLines") = {1,2,32,16,1232}; // aleatoriamente

Physical Line("PhysicalLines") = {1:10}; // sequencialmente

Physical Line("PhysicalLines") = {lines[]}; // uma lista

Physical Line("PhysicalLines") = {ls[], lines[], 1:10, 1232}; // vários

Superfícies geométricas

```
Plane Surface(expr) = {1p};
```

// cria uma superfície plana com base em
um Line Loop fechado ordenado.

// expr é o identificador da superfície;

// 1p é o identificador do loop;

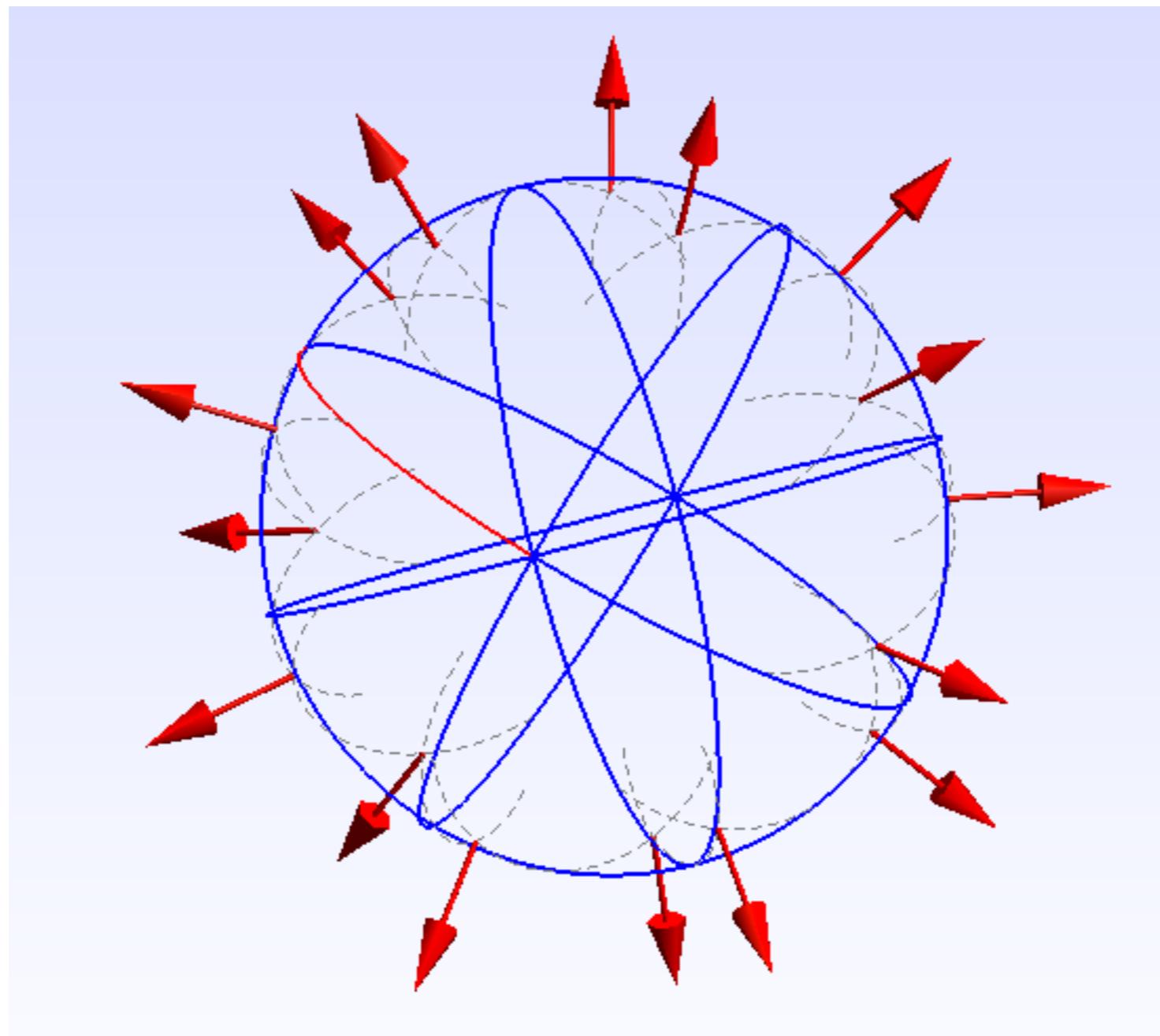
```
Ruled Surface(expr) = {1p};
```

```
// cria uma superfície que pode ser  
interpolada de modo transfinito
```

```
// expr é o identificador da superfície;
```

```
// 1p é o identificador do loop;
```

Ruled Surfaces representadas por tracejados
(vetor normal para fora)



Surface Loops

```
Surface Loop(expr) = {id_surfs};
```

// cria um loop orientado passando os identificadores das entidades do tipo Surface como parâmetros.

// expr é o identificador do loop

// id_surfs são os identificadores Surface

Superfícies físicas

```
Physical Surface(expr) = {id};
```

// define objeto Surface como físico

// expr é o identificador do objeto físico,
podendo ser um número ou uma string. Podemos
definir

```
Physical Surface(10) = {1};
```

// ou

```
Physical Surface("PhysicalSurfaces") = {1};
```

// como para Points and Lines

Physical Surface("PhysicalSurfaces") =
{1,2,32,16,1232}; // aleatoriamente

Physical Surface("PhysicalSurfaces") = {1:10}; //
sequencialmente

Physical Surface("PhysicalSurfaces") = {lines[]}; //
uma lista

Physical Surface("PhysicalSurfaces") = {ls[], lines[],
1:10, 1232}; // vários

Volumes geométricos

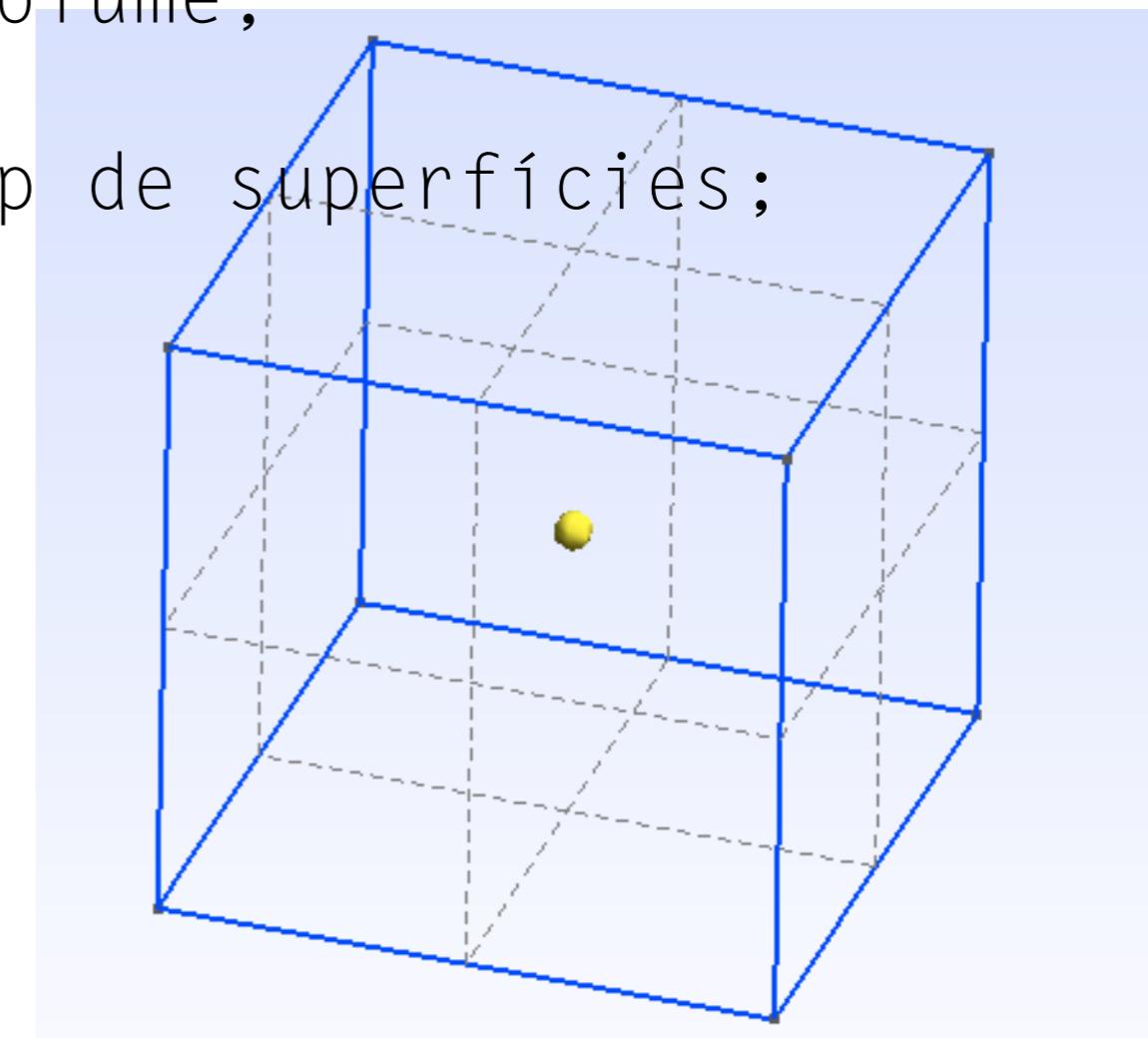
```
Volume(expr) = {1p};
```

// cria um volume com base em um

Surface Loop fechado ordenado.

// expr é o identificador do volume;

// 1p é o identificador do loop de superfícies;



Volumes físicos

Physical Volume(expr) = {id};

// define objeto Volume como físico

// expr é o identificador do objeto físico,
podendo ser um número ou uma string. Podemos
definir

Physical Volume(10) = {1};

// ou

Physical Volume("PhysicalSurfaces") = {1};

Extrusões

extrusão

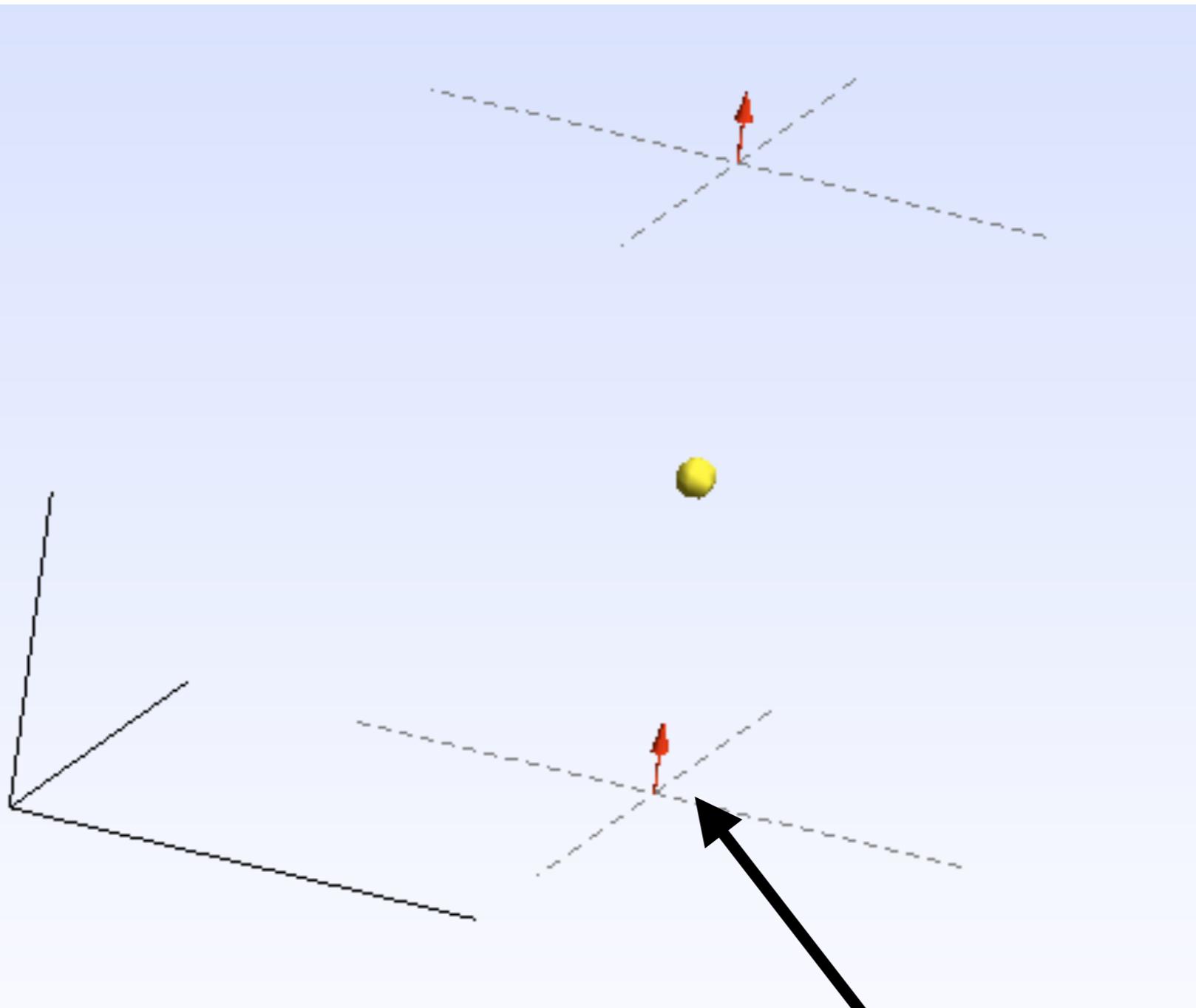
{pontos, linhas, superfícies} → {linhas, superfícies, volumes}

```
Extrude {xt,yt,zt} { Point{id_point}; }
```

// faz extrusão de um ponto; xt,yt,zt são as coordenadas
do vetor que determina a direção de translação; id_point
é o identificador do ponto geométrico

```
// ... { Line{id_line}; } ... { Surface{id_surf}}
```

```
// idem para outras entidades
```



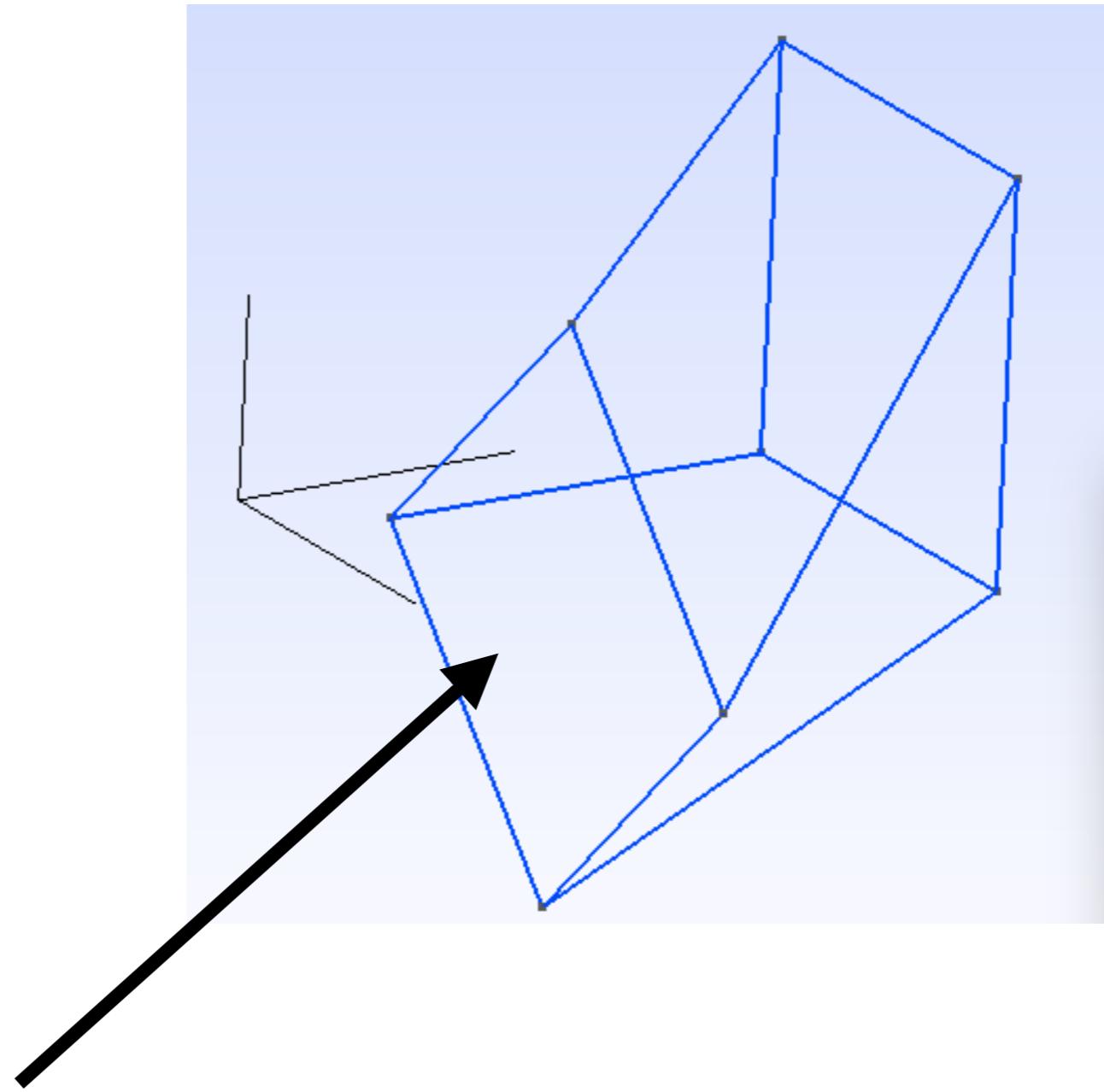
direção do vetor de
translação em relação à superfície
inferior do cubo
mostrado anteriormente

```
Extrude { { xa,ya,za }, { xp,yp,zp}, theta }
{ Point(id_point); }
```

// faz rotação de um ponto; xa,ya,za são as coordenadas
do vetor que determina a direção do eixo de rotação;
xp,yp,zp são coordenadas de um ponto qualquer sobre o
eixo; theta é o ângulo de rotação em radianos; id_point é o
identificador do ponto geométrico

```
// ... { Line{id_line}; } ... { Surface{id_surf}}
```

```
// idem para outras entidades
```



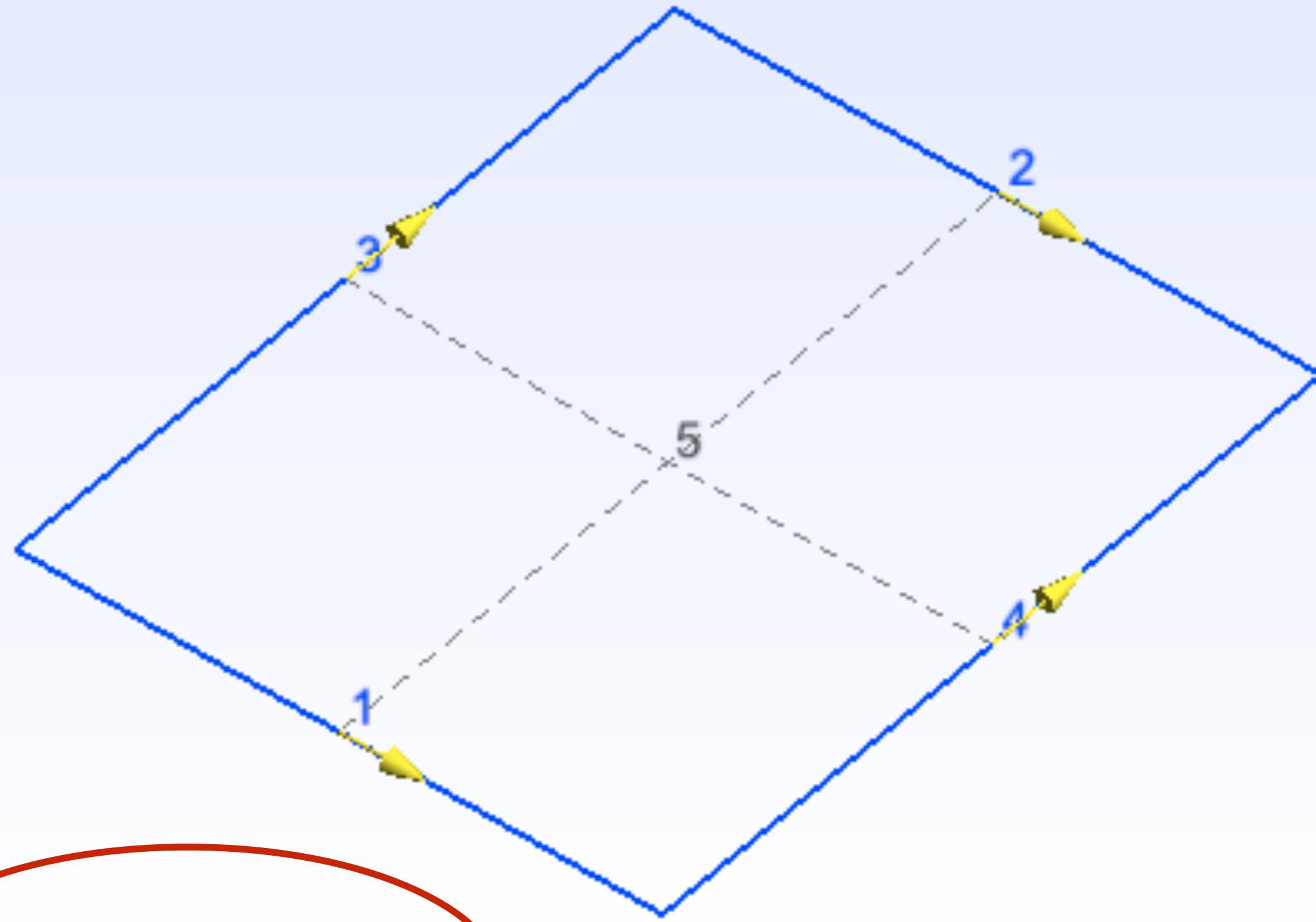
face do cubo mostrado
anteriormente girada de $\pi/4$

```
Extrude { {xt,yt,zt}, { xa,ya,za } ,  
{ xp,yp,zp}, theta } { Point(id_point); }  
  
// faz translação combinada com rotação de  
um ponto;  
  
// ... { Line{id_line}; } ... { Surface{id_surf}  
  
// idem para outras entidades
```

**Nota: observe se a sintaxe do comando está correta:
chaves, pontos-e-virgulas!**

Exemplo

```
Point(1) = {0,0,0};  
Point(2) = {1,0,0};  
Line(1) = {1, 2};  
out[] = Extrude{0,1,0}{ Line{1}; };  
Printf("top line = %g", out[0]);  
Printf("surface = %g", out[1]);  
Printf("side lines = %g and %g", out[2], out[3]);  
  
// por padrão, o Gmsh preserva uma ordem de  
armazenamento nos índices das entidades  
que passam por extrusão. No caso acima,  
out[0], out[1], out[2], out[3] armazena os índices  
respectivos: da linha que sofreu extrusão;  
da superfície gerada, das linhas laterais
```



top line = 2
surface = 5
side lines = 4 and -3

Transformações

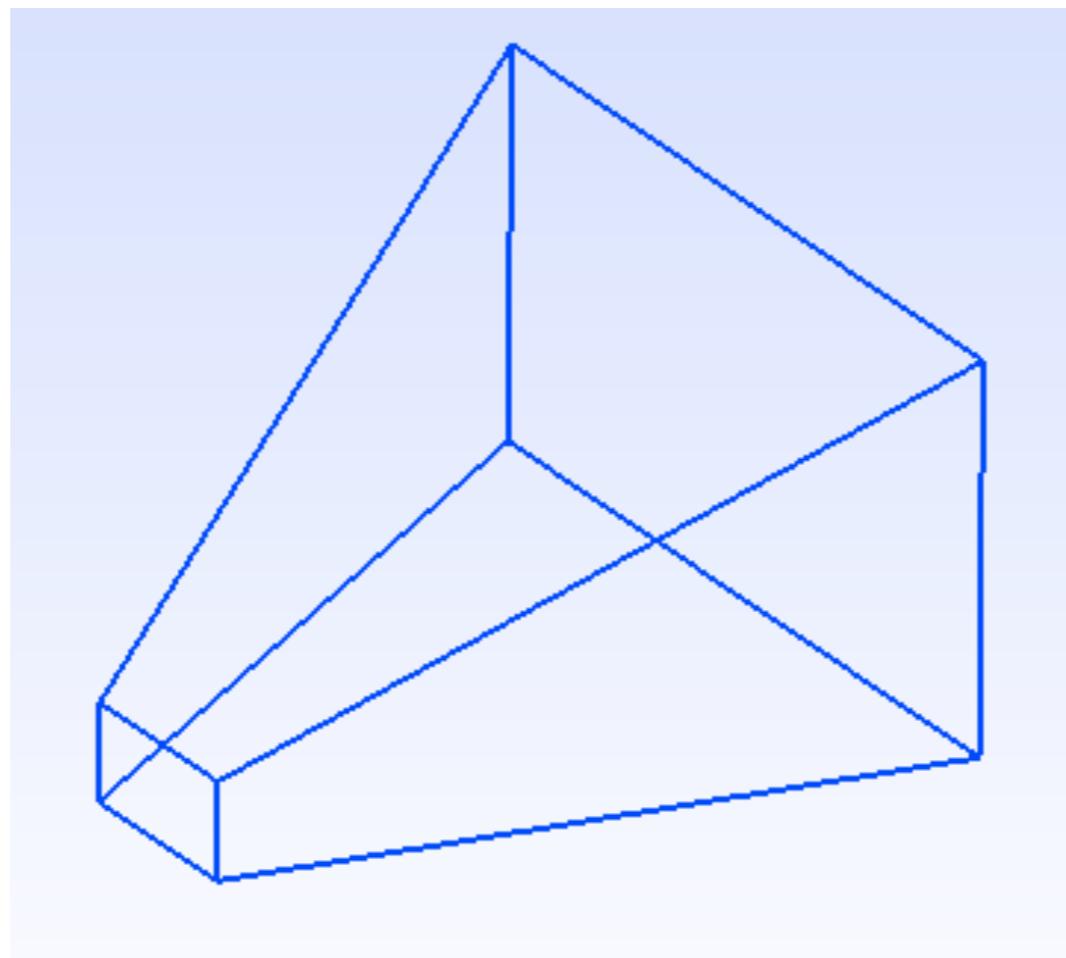
Dilate	escala por um fator
Rotate	gira entidades elementares
Symmetry	transforma simetricamente a um plano
Translate	translada entidades elementares
Boundary*	retorna contorno das entidades elementares
CombinedBoundary*	idem acima, mas de forma combinada
Duplicata**	duplica entidades

*não é, per se, uma "transformação"

**comando útil em loops, translações, etc.

```
Dilate { { xh,yh,zh } , f }
{ Surface{id_surf}; }
```

// dilata uma superfície na direção homotética dada por xh,yh,zh por um fator de f



```
Symmetry { a,b,c,d } { ent(id_ent); }
```

```
// realiza a simetria de entidades  
geométricas 'ent' em relação ao plano  
 $ax + by + cz = d$ 
```

```
Translate { xt,yt,zt } { ent(id_ent); }
```

```
// translada entidades geométricas na  
direção dada pelo vetor cujas  
coordenadas são xt,yt,zt.
```

Outros comandos

Coherence;	remove ent. geométricas duplicadas
Hide	esconde ent. no display da GUI
Show	mostra ent. no display da GUI

Módulo Mesh

Capacidades de malhamento

- Reagrupamento de vários algoritmos para malhamento {1,2,3}D de **elementos finitos conformes**
- Algoritmos 2D(3D) **não-estruturados**: geram triângulos e/ou “quadrângulos”(tetraedros)
- Algoritmos 2D **estruturados** (malhamento transfinito e por extrusão): geram triângulos e/ou “quadrângulos”
- Algoritmos 3D **estruturados**: geram tetraedros, hexaedros, prismas ou pirâmides

Algoritmos disponíveis

- 2D: Delaunay + {MeshAdapt || Delaunay || Frontal}
 - Primeiro passo: malha 2D a partir de 1D
 - Segundo passo: malha final
- 3D: two-step Delaunay || Delaunay + Frontal
 - Primeiro passo: união de volumes
 - Segundo passo: malha final

Ranking dos algoritmos

Método	Robustez	Desempenho	Qualidade do elemento
MeshAdapt	1	3	2
Delaunay	2	1	2
Frontal	3	2	1

Melhores opções on-demand

- Superfícies curvas complexas → MeshAdapt
- Malhas de grandes superfícies → Delaunay*
- Alta qualidade de elementos → Frontal

*o mais robusto e rápido; único que suporta especificação de tamanho de elemento

Entidades físicas x elementares

- Entidades elementares salvam a malha “como é” (todos os índices das entidades)
- Entidades físicas permitem o “assembling” de entidades elementares (tratamento de interseções e controle de orientação), facilitam visualização e manipulação do modelo

Alguns comandos

Field	cria um novo campo de valor
Attractor	calcula distâncias (refinamento adaptativo)
Ball	define uma região esférica
BoundaryLayer	camada-limite (refinamento adaptativo)
Box	define um cuboide
Gradient	calcula o gradiente de um campo (via DF)
Laplacian	calcula Laplaciano de um campo
Threshold	define limiares (refinamento adaptativo)

Comandos para malhas estruturadas

Extrude	faz extrusão de geometria e malha
Layers	cria camadas com tamanhos especificados
Recombine	recombinação de elementos de malha
Transfinite Line	malhamento transfinito 1D
Transfinite Surface	idem em 2D
Transfinite Volume	malhamento transfinito 3D

Outros comandos*

Point Line In Surface	embutir ponto ou linha em superfície
Surface In Volume	embutir superfície em volume
Periodic Line	malhamento de contornos periódico 1D
Periodic Surface	malhamento de contorno periódico 2D
Coherence Mesh	remover vértices de malha duplicados

*ver GUI para opções de malhamento

Miscelânea

**Lista de Tutoriais
por Tópicos
(divisão de conteúdos não é
padrão)**

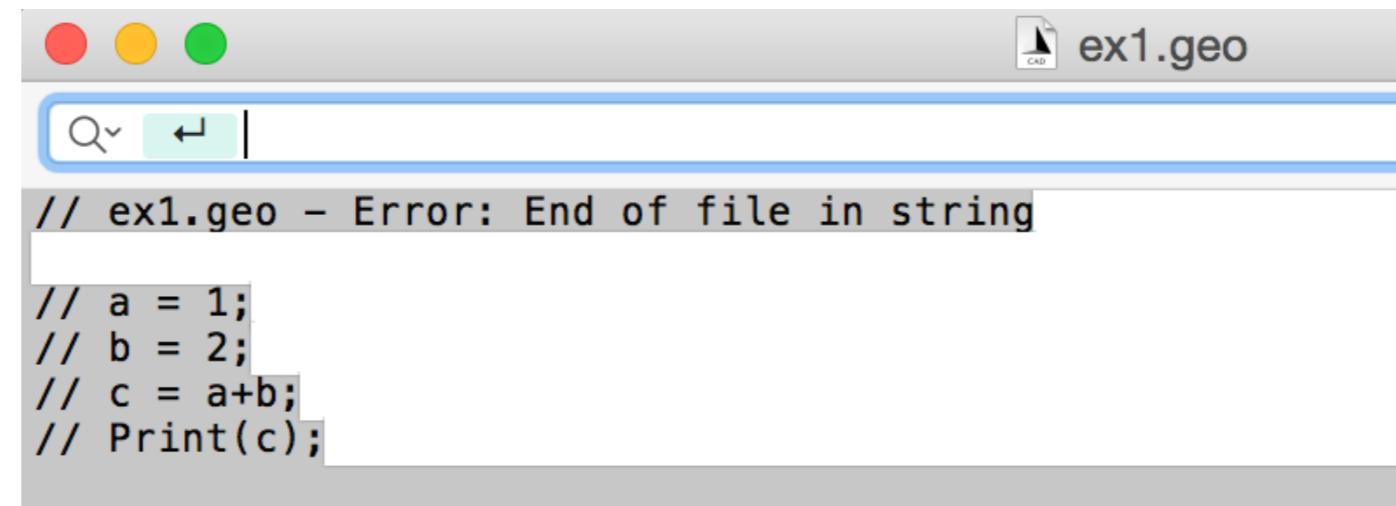
Tutorial**Conceitos Apresentados**

t1.geo	Variáveis, entidades elementares geométricas e físicas (Point, Line, Surface)
t2.geo	Inclusões, transformações geométricas, extrusões
t3.geo	Malhas em extrusão, parâmetros, opções
t4.geo	Funções predefinidas, dominios com furos, strings, malhas coloridas
t5.geo	Comprimentos característicos escalonados, arrays, UDFs, loops
t6.geo	Malhamento transfinito, recombinação, suavização de malha
t7.geo	Malha de background*
t8.geo	Pós-processamento, scripting, animações, opções*
t9.geo	Plugins para pós-processamento (conjuntos de nível, seções, anotações, etc.)
t10.geo	Campos de comprimento característicos gerais (atratores, funções)
t11.geo	Malhas quadrilaterais não-estruturadas
t12.geo	Cross-patch meshing e estruturas compostas
t13.geo	Remalhamento de STL
t14.geo	Homologia/co-homologia computacional
t15.geo	Pontos, linhas e superfícies embutidas

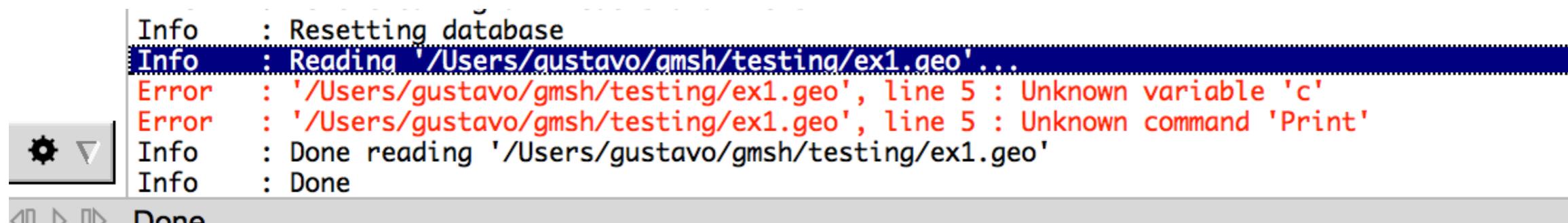
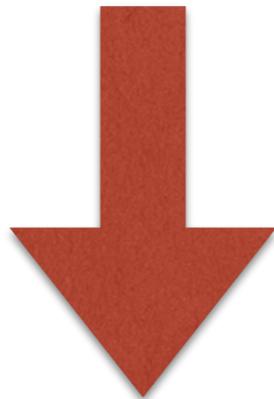
Troubleshooting

Problema: última linha sem quebra
Solução: quebrar na última linha :)

```
// ex1.geo  
// a = 1;  
// b = 2;  
// c = a+b;  
// Print(c);
```



```
// ex1.geo - Error: End of file in string  
// a = 1;  
// b = 2;  
// c = a+b;  
// Print(c);
```



```
Info : Resetting database  
Info : Reading '/Users/gustavo/gmsh/testing/ex1.geo'...  
Error : '/Users/gustavo/gmsh/testing/ex1.geo', line 5 : Unknown variable 'c'  
Error : '/Users/gustavo/gmsh/testing/ex1.geo', line 5 : Unknown command 'Print'  
Info : Done reading '/Users/gustavo/gmsh/testing/ex1.geo'  
Info : Done
```

Onde obter ajuda?

- Help (> Current Options) (interno)
- [Gmsh Wiki](#)
- [Mailing list](#)

Créditos

- Criadores do software:
 - Prof. Christophe Geuzaine, Universidade de Liège
 - Prof. Jean-François Remacle, Universidade Católica de Louvain
- Referência-padrão:
 - C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331, 2009.
 - Versão para este texto: Gmsh 2.9, Abril 2015.
 - Doação ao projeto via PayPal.

Considerações Finais

- Este material estará em contínuo desenvolvimento e aperfeiçoamento;
- Contate o autor e envie seu feedback com sugestões, comentários e/ou correções;

Obrigado!

Contact Card

Gustavo Peixoto de Oliveira, Dr.

gustavo.oliveira@uerj.br

gustavopeixotodeoliveira.com

Programa de Pós-Graduação em Eng. Mecânica - PPG-EM
Universidade do Estado do Rio de Janeiro - UERJ