

Bemsolver

2.0

Generated by Doxygen 1.7.3

Wed Feb 9 2011 15:37:33

Contents

1	Class Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	calcD3triangle Class Reference	7
4.1.1	Constructor & Destructor Documentation	9
4.1.1.1	calcD3triangle	9
4.1.1.2	~calcD3triangle	9
4.1.2	Member Function Documentation	9
4.1.2.1	dG3Ddx	9
4.1.2.2	dG3Ddy	9
4.1.2.3	dG3Ddz	9
4.1.2.4	G3D	9
4.1.2.5	G3Danalytic	9
4.1.2.6	G3DdnAnalytic	9
4.1.2.7	gauleg	9
4.1.2.8	gauslegsimplextriangle	9
4.1.2.9	GetIntL1	9
4.1.2.10	GetIntL1divR3	9
4.1.2.11	GetIntL1OutOfPlane	9
4.1.2.12	triangleint	9
4.1.2.13	triangleint_pot_xyz	9
4.1.3	Member Data Documentation	9
4.1.3.1	EPS	9
4.1.3.2	eta	9
4.1.3.3	n	9
4.1.3.4	numquad	9
4.1.3.5	w	9
4.1.3.6	xi	9
4.2	D3box Class Reference	10
4.2.1	Constructor & Destructor Documentation	11
4.2.1.1	D3box	11
4.2.1.2	~D3box	12

4.2.2	Member Data Documentation	12
4.2.2.1	xl	12
4.2.2.2	yl	12
4.2.2.3	zl	12
4.3	D3cylinder Class Reference	12
4.3.1	Constructor & Destructor Documentation	13
4.3.1.1	D3cylinder	13
4.3.1.2	~D3cylinder	14
4.3.2	Member Data Documentation	14
4.3.2.1	l	14
4.3.2.2	r	14
4.3.2.3	sectors	14
4.4	D3disk Class Reference	14
4.4.1	Constructor & Destructor Documentation	15
4.4.1.1	D3disk	15
4.4.1.2	~D3disk	16
4.4.2	Member Data Documentation	16
4.4.2.1	r	16
4.4.2.2	sectors	16
4.5	D3electrode Class Reference	16
4.5.1	Constructor & Destructor Documentation	18
4.5.1.1	D3electrode	18
4.5.1.2	~D3electrode	18
4.5.2	Member Function Documentation	18
4.5.2.1	GetCardinalNumber	18
4.5.2.2	GetSymmetryWith	18
4.5.2.3	GetTotalrot	18
4.5.2.4	GetVoltage	18
4.5.2.5	insert	18
4.5.2.6	SetCardinalNumber	18
4.5.2.7	SetSymmetry	18
4.5.2.8	SetSymmetryWith	19
4.5.2.9	SetVoltage	20
4.5.3	Member Data Documentation	20
4.5.3.1	cardinalnum	20
4.5.3.2	symel	20
4.5.3.3	totalrot	20
4.5.3.4	voltage	20
4.6	D3element Class Reference	20
4.6.1	Constructor & Destructor Documentation	24
4.6.1.1	D3element	24
4.6.1.2	~D3element	24
4.6.2	Member Function Documentation	24
4.6.2.1	Add	24
4.6.2.2	correctNorm	24
4.6.2.3	createNewSubelements	24
4.6.2.4	deleteSubelements	25
4.6.2.5	GetAmountOfSubelements	25
4.6.2.6	GetArea	25
4.6.2.7	GetCenter	25

4.6.2.8	GetDoubleLayerPotentialAt	25
4.6.2.9	GetInversenorm	25
4.6.2.10	GetListOfBaseElements	25
4.6.2.11	getName	25
4.6.2.12	GetPotentialAndFieldAt	25
4.6.2.13	GetPotentialAt	25
4.6.2.14	GetRectangle	25
4.6.2.15	GetReferencePoint	25
4.6.2.16	GetSelfDoubleLayerPotential	26
4.6.2.17	GetSelfPotential	26
4.6.2.18	GetTriangle	26
4.6.2.19	init	26
4.6.2.20	InsertTGeoVolume	26
4.6.2.21	IntersectWithRay	26
4.6.2.22	PrintAmountOfSubelements	26
4.6.2.23	refine	26
4.6.2.24	refine	26
4.6.2.25	rotate	27
4.6.2.26	rotate2	27
4.6.2.27	rotateinv	27
4.6.2.28	setName	27
4.6.2.29	SetNormTowards	27
4.6.2.30	shift	27
4.6.3	Member Data Documentation	28
4.6.3.1	Color	28
4.6.3.2	epsilon	28
4.6.3.3	h	28
4.6.3.4	inversenorm	28
4.6.3.5	isBaseElement	28
4.6.3.6	name	28
4.6.3.7	parent	28
4.6.3.8	phi	28
4.6.3.9	psi	28
4.6.3.10	refineable	28
4.6.3.11	subelement	28
4.6.3.12	theta	28
4.6.3.13	x	28
4.6.3.14	y	28
4.6.3.15	z	28
4.7	D3icosahedron Class Reference	28
4.7.1	Constructor & Destructor Documentation	30
4.7.1.1	D3icosahedron	30
4.7.1.2	~D3icosahedron	30
4.7.2	Member Function Documentation	30
4.7.2.1	triangle	30
4.7.3	Member Data Documentation	30
4.7.3.1	r	30
4.7.3.2	refinement	30
4.7.3.3	V	30
4.8	D3ImportedElectrodes Class Reference	30

4.8.1	Constructor & Destructor Documentation	31
4.8.1.1	D3ImportedElectrodes	31
4.8.1.2	~D3ImportedElectrodes	32
4.8.2	Member Function Documentation	32
4.8.2.1	add3DFace	32
4.8.2.2	addArc	32
4.8.2.3	addBlockRecord	32
4.8.2.4	addCircle	32
4.8.2.5	addLayer	32
4.8.2.6	addLine	32
4.8.2.7	addPoint	32
4.8.2.8	addPolyline	32
4.8.2.9	addVertex	32
4.8.2.10	endSequence	32
4.8.2.11	FindElectrode	32
4.8.2.12	Import	32
4.8.2.13	ListElectrodes	33
4.8.2.14	printAttributes	33
4.8.3	Member Data Documentation	33
4.8.3.1	electrode	33
4.8.3.2	ignore3DFace	33
4.8.3.3	ignore_Model_Space_Handle	33
4.8.3.4	ignorePolyline	33
4.8.3.5	importingPolyline	33
4.8.3.6	Model_Space_Handle	33
4.8.3.7	vertexcnt	33
4.8.3.8	x	33
4.8.3.9	y	33
4.8.3.10	z	33
4.9	D3rectangle Class Reference	33
4.9.1	Constructor & Destructor Documentation	36
4.9.1.1	D3rectangle	36
4.9.1.2	D3rectangle	36
4.9.1.3	D3rectangle	37
4.9.1.4	~D3rectangle	37
4.9.2	Member Function Documentation	37
4.9.2.1	createNewSubelements	37
4.9.2.2	GetArea	37
4.9.2.3	GetCenter	37
4.9.2.4	GetDoubleLayerPotentialAt	37
4.9.2.5	GetPotentialAndFieldAt	38
4.9.2.6	GetPotentialAt	38
4.9.2.7	GetRectangle	38
4.9.2.8	GetReferencePoint	38
4.9.2.9	GetSelfDoubleLayerPotential	38
4.9.2.10	GetSelfPotential	38
4.9.2.11	InsertTGeoVolume	38
4.9.2.12	IntersectWithRay	38
4.9.2.13	rotate	38
4.9.2.14	SetNormTowards	39

4.9.2.15	shift	39
4.9.3	Member Data Documentation	40
4.9.3.1	x1	40
4.9.3.2	x2	40
4.9.3.3	x3	40
4.9.3.4	x4	40
4.9.3.5	xa	40
4.9.3.6	xb	40
4.9.3.7	xc	40
4.9.3.8	y1	40
4.9.3.9	y2	40
4.9.3.10	y3	40
4.9.3.11	y4	40
4.9.3.12	ya	40
4.9.3.13	yb	40
4.9.3.14	yc	40
4.9.3.15	z1	40
4.9.3.16	z2	40
4.9.3.17	z3	40
4.9.3.18	z4	40
4.10	D3slowworld Class Reference	40
4.10.1	Constructor & Destructor Documentation	42
4.10.1.1	D3slowworld	42
4.10.1.2	~D3slowworld	42
4.10.2	Member Function Documentation	42
4.10.2.1	calc	42
4.10.2.2	draw	42
4.10.2.3	GetListOfBaseElements	42
4.10.2.4	insert	42
4.10.2.5	solve	42
4.10.3	Member Data Documentation	42
4.10.3.1	amountOfElectrodes	42
4.10.3.2	dfdnAll	42
4.10.3.3	el	42
4.10.3.4	electrodeIndexLimit	42
4.10.3.5	f	42
4.10.3.6	n	42
4.11	D3sortedelement Class Reference	43
4.11.1	Constructor & Destructor Documentation	44
4.11.1.1	D3sortedelement	44
4.11.1.2	~D3sortedelement	44
4.11.2	Friends And Related Function Documentation	44
4.11.2.1	SmallerThan	44
4.11.3	Member Data Documentation	44
4.11.3.1	axis	44
4.11.3.2	done	44
4.11.3.3	electrodeptr	44
4.11.3.4	index	44
4.11.3.5	radius	44
4.12	D3sorter Class Reference	44

4.12.1	Member Function Documentation	46
4.12.1.1	exchange	46
4.12.1.2	quicksort	46
4.12.1.3	sort	46
4.12.2	Member Data Documentation	46
4.12.2.1	a	46
4.12.2.2	eps	46
4.12.2.3	n	46
4.13	D3sphere Class Reference	46
4.13.1	Constructor & Destructor Documentation	47
4.13.1.1	D3sphere	47
4.13.1.2	~D3sphere	48
4.13.2	Member Data Documentation	48
4.13.2.1	r	48
4.13.2.2	refinement	48
4.14	D3thicktriangle Class Reference	48
4.14.1	Constructor & Destructor Documentation	50
4.14.1.1	D3thicktriangle	50
4.14.1.2	~D3thicktriangle	50
4.14.2	Member Data Documentation	50
4.14.2.1	htriangle	50
4.14.2.2	xa	50
4.14.2.3	xb	50
4.14.2.4	ya	50
4.14.2.5	yb	50
4.15	D3triangle Class Reference	51
4.15.1	Constructor & Destructor Documentation	53
4.15.1.1	D3triangle	53
4.15.1.2	D3triangle	53
4.15.1.3	~D3triangle	54
4.15.2	Member Function Documentation	54
4.15.2.1	createNewSubelements	54
4.15.2.2	GetArea	54
4.15.2.3	GetCenter	54
4.15.2.4	GetDoubleLayerPotentialAt	54
4.15.2.5	GetPotentialAndFieldAt	54
4.15.2.6	GetPotentialAt	54
4.15.2.7	GetReferencePoint	54
4.15.2.8	GetSelfDoubleLayerPotential	54
4.15.2.9	GetSelfPotential	54
4.15.2.10	GetTriangle	55
4.15.2.11	InsertTGeoVolume	55
4.15.2.12	IntersectWithRay	55
4.15.2.13	IntersectWithRay	55
4.15.2.14	Newtriangles	55
4.15.2.15	rotate	55
4.15.2.16	SetNormTowards	55
4.15.2.17	shift	55
4.15.2.18	Stripeit	56
4.15.3	Member Data Documentation	56

4.15.3.1	stripeit	56
4.15.3.2	x1	56
4.15.3.3	x2	56
4.15.3.4	x3	56
4.15.3.5	xa	56
4.15.3.6	xb	56
4.15.3.7	y1	56
4.15.3.8	y2	56
4.15.3.9	y3	56
4.15.3.10	ya	56
4.15.3.11	yb	56
4.15.3.12	z1	56
4.15.3.13	z2	56
4.15.3.14	z3	56
4.16	D3tube Class Reference	56
4.16.1	Constructor & Destructor Documentation	58
4.16.1.1	D3tube	58
4.16.1.2	~D3tube	58
4.16.2	Member Data Documentation	58
4.16.2.1	l	58
4.16.2.2	r	58
4.16.2.3	r2	58
4.16.2.4	sectors	58
4.17	D3world Class Reference	59
4.17.1	Constructor & Destructor Documentation	63
4.17.1.1	D3world	63
4.17.1.2	D3world	63
4.17.1.3	~D3world	63
4.17.2	Member Function Documentation	63
4.17.2.1	AssignColors	63
4.17.2.2	calc	63
4.17.2.3	calc	63
4.17.2.4	calc	63
4.17.2.5	calc	64
4.17.2.6	calc	64
4.17.2.7	calc2	64
4.17.2.8	calc_slow	64
4.17.2.9	calc_slow	64
4.17.2.10	calc_slow	64
4.17.2.11	calc_slow2	64
4.17.2.12	cut	65
4.17.2.13	Dcentroid	65
4.17.2.14	draw	65
4.17.2.15	exch	65
4.17.2.16	exportGeometry	65
4.17.2.17	GetListOfBaseElements	65
4.17.2.18	insert	65
4.17.2.19	IsEqualSurfaceElement	65
4.17.2.20	load	65
4.17.2.21	loadcalc	65

4.17.2.22	propagateForwardEuler	65
4.17.2.23	propagateForwardVerlet	65
4.17.2.24	propagateForwardVerletRotSymX	66
4.17.2.25	propagateForwardVerletRotSymY	66
4.17.2.26	propagateForwardVerletRotSymZ	66
4.17.2.27	RefreshChecksum	66
4.17.2.28	RotMirrorSurfaceElement	66
4.17.2.29	save	66
4.17.2.30	savecalc	66
4.17.2.31	SetScalePostCalc	66
4.17.2.32	solve	66
4.17.2.33	SymmetrizeCharges	66
4.17.2.34	update_adler32	66
4.17.2.35	update_adler32double	66
4.17.3	Member Data Documentation	66
4.17.3.1	amountOfElectrodes	66
4.17.3.2	ave_diri	66
4.17.3.3	ave_neum	66
4.17.3.4	cachefilename	66
4.17.3.5	checksum	66
4.17.3.6	checksumcalc	66
4.17.3.7	cnt_diri	66
4.17.3.8	cnt_neum	66
4.17.3.9	currentel	66
4.17.3.10	dbydbpoten	66
4.17.3.11	dbydnpotenAll	66
4.17.3.12	dfdAll	66
4.17.3.13	docache	66
4.17.3.14	dtype	66
4.17.3.15	el	66
4.17.3.16	electrodeIndexLimit	66
4.17.3.17	electrodes	66
4.17.3.18	error	66
4.17.3.19	f	66
4.17.3.20	feldxcache	66
4.17.3.21	feldycache	66
4.17.3.22	feldzcache	66
4.17.3.23	fljob	66
4.17.3.24	i	66
4.17.3.25	j	66
4.17.3.26	job	66
4.17.3.27	lhsindex	66
4.17.3.28	lhstype	66
4.17.3.29	lhsvect	66
4.17.3.30	max_diri	66
4.17.3.31	max_neum	66
4.17.3.32	maxit	66
4.17.3.33	n	66
4.17.3.34	nlhs	66
4.17.3.35	nrhs	66

4.17.3.36	numit	66
4.17.3.37	numLev	66
4.17.3.38	numMom	66
4.17.3.39	nx	66
4.17.3.40	ny	66
4.17.3.41	nz	66
4.17.3.42	potcache	66
4.17.3.43	poten	66
4.17.3.44	rangeerror	66
4.17.3.45	refreshchecksum	66
4.17.3.46	rhsindex	66
4.17.3.47	rhstype	66
4.17.3.48	rhsvect	66
4.17.3.49	segmentation	66
4.17.3.50	shape	66
4.17.3.51	shapechar	66
4.17.3.52	size	66
4.17.3.53	spaceunit	66
4.17.3.54	tol	66
4.17.3.55	totalrot	66
4.17.3.56	type	66
4.17.3.57	x	66
4.17.3.58	xcoll	67
4.17.3.59	xmax	67
4.17.3.60	xmin	67
4.17.3.61	xnrm	67
4.17.3.62	xscale	67
4.17.3.63	ymax	67
4.17.3.64	ymin	67
4.17.3.65	yscale	67
4.17.3.66	zmax	67
4.17.3.67	zmin	67
4.17.3.68	zscale	67
4.18	xv Struct Reference	67
4.18.1	Member Data Documentation	68
4.18.1.1	vx	68
4.18.1.2	vy	68
4.18.1.3	vz	68
4.18.1.4	x	68
4.18.1.5	y	68
4.18.1.6	z	68
5	File Documentation	69
5.1	bem.h File Reference	69
5.1.1	Define Documentation	72
5.1.1.1	BEMREVISION	72
5.1.1.2	eee	72
5.1.1.3	mmm	72
5.1.1.4	size_	72
5.1.1.5	sqr	72

5.1.2	Typedef Documentation	72
5.1.2.1	PD3element	72
5.1.2.2	PD3elementList	72
5.1.2.3	trianglefn	72
5.1.3	Function Documentation	72
5.1.3.1	q	72
5.1.3.2	testfn	72
5.1.4	Variable Documentation	72
5.1.4.1	calc	72
5.1.4.2	pi	72
5.2	linearPaulTrap.cxx File Reference	72
5.2.1	Define Documentation	74
5.2.1.1	ELCNT	74
5.2.2	Function Documentation	74
5.2.2.1	main	74
5.2.3	Variable Documentation	75
5.2.3.1	eldca	75
5.2.3.2	eldcb	75
5.2.3.3	elrfa	75
5.2.3.4	elrfb	75
5.2.3.5	wr	75
5.3	linearPaulTrap_edit.cxx File Reference	75
5.3.1	Define Documentation	76
5.3.1.1	PI	76
5.3.2	Function Documentation	76
5.3.2.1	calcit	76
5.3.2.2	ExportAxialPot	77
5.3.2.3	free	77
5.3.2.4	help	77
5.3.2.5	PlotPot	77
5.3.2.6	PlotPseudo	78
5.3.2.7	setTrappingVoltage	78
5.3.2.8	setZero	79
5.3.2.9	Trajectory	79
5.4	ringTrap.cxx File Reference	79
5.4.1	Define Documentation	81
5.4.1.1	ELCNT	81
5.4.2	Function Documentation	81
5.4.2.1	main	81
5.4.3	Variable Documentation	82
5.4.3.1	eldc1	82
5.4.3.2	eldc2	82
5.4.3.3	eldrf	82
5.4.3.4	wr	82
5.5	ringTrap_edit.cxx File Reference	82
5.5.1	Define Documentation	83
5.5.1.1	PI	83
5.5.2	Function Documentation	83
5.5.2.1	calcit	83
5.5.2.2	ExportAxialPot	83

5.5.2.3	free	84
5.5.2.4	help	84
5.5.2.5	PlotPot	84
5.5.2.6	setTrappingVoltage	84
5.5.2.7	setZero	85
5.5.2.8	Trajectory	85

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

calcD3triangle	7
D3element	20
D3box	10
D3cylinder	12
D3disk	14
D3electrode	16
D3icosahedron	28
D3rectangle	33
D3slowworld	40
D3sphere	46
D3thicktriangle	48
D3triangle	51
D3tube	56
D3world	59
D3ImportedElectrodes	30
D3sortedelement	43
D3sorter	44
xv	67

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

calcD3triangle	7
D3box	10
D3cylinder	12
D3disk	14
D3electrode	16
D3element	20
D3icosahedron	28
D3ImportedElectrodes	30
D3rectangle	33
D3slowworld	40
D3sortedelement	43
D3sorter	44
D3sphere	46
D3thicktriangle	48
D3triangle	51
D3tube	56
D3world	59
xv	67

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

bem.h	69
linearPaulTrap.cxx	72
linearPaulTrap_edit.cxx	75
ringTrap.cxx	79
ringTrap_edit.cxx	82

Chapter 4

Class Documentation

4.1 calcD3triangle Class Reference

```
#include <bem.h>
```

Public Member Functions

- **calcD3triangle** ()
- **~calcD3triangle** ()
- double **GetIntL1** (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3)
- double **GetIntL1OutOfPlane** (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, double rp)
- double **GetIntL1divR3** (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, double rp)
- double **triangleint** (**trianglefn** *fn, double x0, double y0, double z0, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3)
- double **G3Danalytic** (double xp, double yp, double zp, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, bool inversenorm)
- double **G3DdnAnalytic** (double xp, double yp, double zp, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, bool inversenorm)
- void **triangleint_pot_xyz** (double x0, double y0, double z0, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, bool inversenorm, double &pot, double &ex, double &ey, double &ez)

Static Public Member Functions

- static double **G3D** (double x0, double y0, double z0, double x1, double y1, double z1)

- static double **dG3Ddx** (double x0, double y0, double z0, double x1, double y1, double z1)
- static double **dG3Ddy** (double x0, double y0, double z0, double x1, double y1, double z1)
- static double **dG3Ddz** (double x0, double y0, double z0, double x1, double y1, double z1)

Protected Member Functions

- void **gauleg** (double x1, double x2, double x[], double w[], int n)
- void **gauslegsimplextriangle** (double xi[], double eta[], double c[], int n)

Protected Attributes

- double **EPS**
- double * **xi**
- double * **eta**
- double * **w**
- int **numquad**
- int **n**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 `calcD3triangle::calcD3triangle ()`

4.1.1.2 `calcD3triangle::~~calcD3triangle ()`

4.1.2 Member Function Documentation

4.1.2.1 `static double calcD3triangle::dG3Ddx (double x0, double y0, double z0, double x1, double y1, double z1) [static]`

4.1.2.2 `static double calcD3triangle::dG3Ddy (double x0, double y0, double z0, double x1, double y1, double z1) [static]`

4.1.2.3 `static double calcD3triangle::dG3Ddz (double x0, double y0, double z0, double x1, double y1, double z1) [static]`

4.1.2.4 `static double calcD3triangle::G3D (double x0, double y0, double z0, double x1, double y1, double z1) [static]`

4.1.2.5 `double calcD3triangle::G3Danalytic (double xp, double yp, double zp, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, bool inversenorm)`

4.1.2.6 `double calcD3triangle::G3DdnAnalytic (double xp, double yp, double zp, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, bool inversenorm)`

4.1.2.7 `void calcD3triangle::gauleg (double x1, double x2, double x[], double w[], int n) [protected]`

4.1.2.8 `void calcD3triangle::gauslegsimplertriangle (double xi[], double eta[], double c[], int n) [protected]`

4.1.2.9 `double calcD3triangle::GetIntL1 (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3)`

4.1.2.10 `double calcD3triangle::GetIntL1divR3 (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, double rp)`

4.1.2.11 `double calcD3triangle::GetIntL1OutOfPlane (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, double rp)`

4.1.2.12 `double calcD3triangle::triangleint (trianglefn * fn, double x0, double y0, double z0, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3)`

4.1.2.13 `void calcD3triangle::triangleint_pot_xyz (double x0, double y0, double z0, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, bool inversenorm, double & pot, double & ex, double & ey, double & ez)`

Generated on Wed Feb 9 2011 15:37:33 for Bemsolver by Doxygen

4.1.3 Member Data Documentation

4.1.3.1 `double calcD3triangle::EPS [protected]`

4.1.3.2 `double* calcD3triangle::eta [protected]`

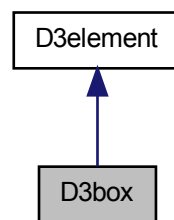
4.1.3.3 `int calcD3triangle::n [protected]`

- **bem.h**

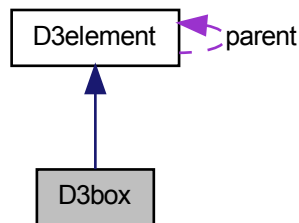
4.2 D3box Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3box:



Collaboration diagram for D3box:



Public Member Functions

- **D3box** (double x1_, double y1_, double z1_, double x, double y, double z, double phi_, double theta_, double psi_, bool **inversenorm**=false, **PD3element parent**=NULL)

Creates a 3-dimensional symmetric box.

- `~D3box ()`

Protected Attributes

- double `xl`
- double `yl`
- double `zl`

4.2.1 Constructor & Destructor Documentation

4.2.1.1 `D3box::D3box (double xl_, double yl_, double zl_, double x, double y, double z, double phi_, double theta_, double psi_, bool inversenorm = false, PD3element parent = NULL) [inline]`

Creates a 3-dimensional symmetric box.

Initial orientation of the Box and coordinate system:

x-axis: towards the viewer

y-axis: upwards

z-axis: to the right

x,y,z: Coordinates of the boxes origin

`xl_`: defines length of box in x-direction

`yl_`: defines length of box in y-direction

`zl_`: defines length of box in z-direction

negative lengths are also possible

`x + xl_`, `y + yl_`, `z + zl_` define the edges of the box

The angles `phi_`, `theta_` and `psi_` can be declared in degrees and represent Euler-angles.

The Center (x,y,z) specifies the center of the Euler-rotations

The Euler-rotations occur in the following order:

`phi_`:.....Rotation around z-axis.....x -> x'.....y -> y'.....z -> z

`theta_`:.....Rotation around the rotated x-axis (x').....x' -> x''.....y' -> y''.....z -> z'

`psi_`:.....Rotation around z'-axis:.....x' -> x''.....y'' -> y'''.....z' -> z''

4.2.1.2 `D3box::~D3box()` [inline]

4.2.2 Member Data Documentation

4.2.2.1 `double D3box::xl` [protected]

4.2.2.2 `double D3box::yl` [protected]

4.2.2.3 `double D3box::zl` [protected]

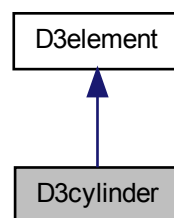
The documentation for this class was generated from the following file:

- `bem.h`

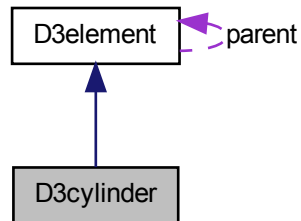
4.3 D3cylinder Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3cylinder:



Collaboration diagram for D3cylinder:



Public Member Functions

- **D3cylinder** (double *r_*, double *l_*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, int *sectors_*=12, bool *subdivide*=true, bool *inversenorm*=false, **PD3element** *parent*=NULL)
*Creates a cylinder with radius *r_* and length *l_*.*
- **~D3cylinder** ()

Protected Attributes

- double **r**
- double **l**
- double **sectors**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 **D3cylinder::D3cylinder** (double *r_*, double *l_*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, int *sectors_* = 12, bool *subdivide* = true, bool *inversenorm* = false, **PD3element** *parent* = NULL) [inline]

Creates a cylinder with radius *r_* and length *l_*.

The bottom of the cylinder is centered at (*x*,*y*,*z*).

The angles *phi_*, *theta_* and *psi_* can be declared in degrees and represent Euler-angles.

The Center (*x*,*y*,*z*) specifies the center of the Euler-rotations

The Euler-rotations occur in the following order:

phi_:.....Rotation around z-axis.....x -> x'.....y -> y'.....z -> z

theta_:.....Rotation around the rotated x-axis (x'):.....x'-> x'y' -> y''z -> z'

psi_:.....Rotation around z'-axis:.....x'-> x''y''-> y'''z'-> z'

For $\phi_ = \theta_ = \psi_ = 0$, the bottom of the cylinder lies in the x,y-plane and the z-extension correspond to the length $l_$.

The parameter `sectors_` provides an option for refinement. It is set to 12 by initialization but can also be set manually. A value of `sectors_ = 8` creates a cylinder with 8 edges on the bottom for example. `sectors_` is not limited to powers of two or even numbers!! 5 or 11 works as well.

4.3.1.2 `D3cylinder::~~D3cylinder()` [inline]

4.3.2 Member Data Documentation

4.3.2.1 `double D3cylinder::l` [protected]

4.3.2.2 `double D3cylinder::r` [protected]

4.3.2.3 `double D3cylinder::sectors` [protected]

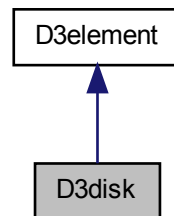
The documentation for this class was generated from the following file:

- `bem.h`

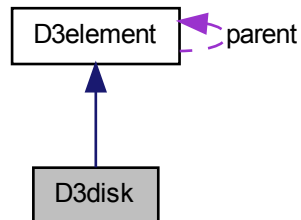
4.4 D3disk Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3disk:



Collaboration diagram for D3disk:



Public Member Functions

- **D3disk** (double *r_*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, int *sectors_*=32, bool *subdivide*=true, bool *inversenorm*=false, **PD3element** *parent*=NULL)
Creates a disk with radius r_.
- **~D3disk** ()

Protected Attributes

- double **r**
- double **sectors**

4.4.1 Constructor & Destructor Documentation

4.4.1.1 **D3disk::D3disk** (double *r_*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, int *sectors_* = 32, bool *subdivide* = true, bool *inversenorm* = false, **PD3element** *parent* = NULL) [inline]

Creates a disk with radius *r_*.

The disk is centered at (*x*,*y*,*z*).

The angles *phi_*, *theta_* and *psi_* can be declared in degrees and represent Euler-angles.

The Center (*x*,*y*,*z*) specifies the center of the Euler-rotations

The Euler-rotations occur in the following order:

phi_:.....Rotation around z-axis.....*x* -> *x'*.....*y* -> *y'*.....*z* -> *z*

theta_:.....Rotation around the rotated x-axis (*x'*):.....*x'* -> *x''*.....*y'* -> *y''*.....*z* -> *z'*

psi_:.....Rotation around z'-axis:.....x'->x''.....y''->y'''.....z'->z'

For $\phi_ = \theta_ = \psi_ = 0$, the disk lies in the x,y-plane and there is no extension in z-direction

The parameter `sectors_` provides an option for refinement. It is set to 32 by initialization but can also be set manually.

A value of `sectors_ = 8` creates a disk with 8 edges. `sectors_` is not limited to powers of two or even numbers!! 5 or 11 for example work as well.

4.4.1.2 `D3disk::~D3disk()` [inline]

4.4.2 Member Data Documentation

4.4.2.1 `double D3disk::r` [protected]

4.4.2.2 `double D3disk::sectors` [protected]

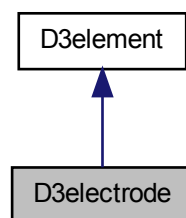
The documentation for this class was generated from the following file:

- `bem.h`

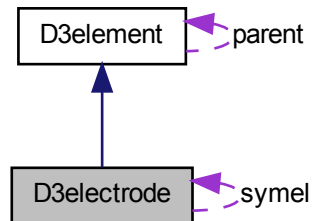
4.5 D3electrode Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3electrode:



Collaboration diagram for D3electrode:



Public Member Functions

- **D3electrode ()**
Constructor for an electrode.
- **~D3electrode ()**
- void **insert (PD3element el)**
Allocates an D3element (p. 20) to an electrode, multiple elements can be allocated to one electrode.
- void **SetVoltage** (double voltage_)
Sets the voltage for the electrode. Unit: Volt.
- double **GetVoltage ()**
Gives the voltage of the electrode. Unit: Volt.
- void **SetSymmetryWith** (bool xsym, bool ysym, bool zsym, bool diagsym, **D3electrode** *symel2, int rotsym=0, int **totalrot**=0)
Eliminates small numerical deviations from perfect symmetry.
- void **SetSymmetry** (bool xsym, bool ysym, bool zsym, bool diagsym, int rotsym=0, int **totalrot2**=0)
Eliminates small numerical deviations from perfect symmetry.
- **D3electrode * GetSymmetryWith** (int num)
- int **GetTotalrot ()**
- void **SetCardinalNumber** (int num)
- int **GetCardinalNumber ()**

Protected Attributes

- int **cardinalnum**
- double **voltage**
- **D3electrode** ** **symel**
- int **totalrot**

4.5.1 Constructor & Destructor Documentation

4.5.1.1 D3electrode::D3electrode ()

Constructor for an electrode.

The father class is **D3element** (p. 20), which means an electrode consists of at least one 3-dimensional element and additional attributes such as voltage etc.

4.5.1.2 D3electrode::~~D3electrode ()

4.5.2 Member Function Documentation

4.5.2.1 int D3electrode::GetCardinalNumber ()

4.5.2.2 D3electrode* D3electrode::GetSymmetryWith (int *num*)

4.5.2.3 int D3electrode::GetTotalrot ()

4.5.2.4 double D3electrode::GetVoltage ()

Gives the voltage of the electrode. Unit: Volt.

4.5.2.5 void D3electrode::insert (PD3element *el*)

Allocates an **D3element** (p. 20) to an electrode, multiple elements can be allocated to one electrode.

Parameters

<i>el</i>	D3element (p. 20) that is being allocated to the electrode
-----------	---

4.5.2.6 void D3electrode::SetCardinalNumber (int *num*)

4.5.2.7 void D3electrode::SetSymmetry (bool *xsym*, bool *ysym*, bool *zsym*, bool *diagmirror*, int *rotsym* = 0, int *totalrot2* = 0)

Eliminates small numerical deviations from perfect symmetry.

If you need an electrode that is symmetric to a certain plane or direction, you have to

place it properly at first.

The method will now only correct small numerical deviations from perfect symmetry! It cannot provide large-scale translations or rotations!

`xsym=true` means symmetric with respect to `x=0` plane

`ysym=true` means symmetric with respect to `y=0` plane

`zsym=true` means symmetric with respect to `z=0` plane

`diagmirror=true` means symmetric with respect of exchange of `x` and `y` (if `z` is set as axis in `SymmetrizeCharges()`)

`diagmirror=true` means symmetric with respect of exchange of `y` and `z` (if `x` is set as axis in `SymmetrizeCharges()`)

`diagmirror=true` means symmetric with respect of exchange of `x` and `z` (if `y` is set as axis in `SymmetrizeCharges()`)

`rotsym` means that symmetric with respect to `rotsym/totalsym*360` degree rotation (set in a loop if you have rotational symmetry)

Parameters

<code>xsym</code>	true, if symmetric with respect to <code>x=0</code> plane
<code>ysym</code>	true, if symmetric with respect to <code>y=0</code> plane
<code>zsym</code>	true, if symmetric with respect to <code>z=0</code> plane
<code>diagmirror</code>	true, if see above
<code>rotsym</code>	?? int ??
<code>totalrot2</code>	??

4.5.2.8 void D3electrode::SetSymmetryWith (bool *xsym*, bool *ysym*, bool *zsym*, bool *diagmirror*, D3electrode * *syme12*, int *rotsym* = 0, int *totalrot* = 0)

Eliminates small numerical deviations from perfect symmetry.

If you need an electrode that is symmetric to a certain plane or direction, you have to place it properly at first.

The method will now only correct small numerical deviations from perfect symmetry! It cannot provide large-scale translations or rotations!

Additionally to the methode SetSymmetry , this method also provides to set the current electrode in perfect symmetry with the one specified by *syme12

`xsym=true` means symmetric with respect to `x=0` plane

`ysym=true` means symmetric with respect to `y=0` plane

`zsym=true` means symmetric with respect to `z=0` plane

`diagmirror=true` means symmetric with respect of exchange of `x` and `y` (if `z` is set as axis in `SymmetrizeCharges()`)

`diagmirror=true` means symmetric with respect of exchange of `y` and `z` (if `x` is set as axis in `SymmetrizeCharges()`)

diagmirror=true means symmetric with respect of exchange of x and z (if y is set as axis in SymmetrizeCharges())

symel2 is the other electrode unequal to *this that the symmetry refers to

rotsym means that symmetric with respect to rotsym/totalsym*360 degree rotation (set in a loop if you have rotational symmetry)

Parameters

<i>xsym</i>	true, if symmetric with respect to x=0 plane
<i>ysym</i>	true, if symmetric with respect to y=0 plane
<i>zsym</i>	true, if symmetric with respect to z=0 plane
<i>diagmirror</i>	true, if see above
<i>symel2</i>	other electrode unequal to *this that the symmetry refers to
<i>rotsym</i>	?? int ??
<i>totalrot</i>	??

4.5.2.9 void D3electrode::SetVoltage (double *voltage_*)

Sets the voltage for the electrode. Unit: Volt.

Parameters

<i>voltage_</i>	voltage for the electrode, unit: Volt.
-----------------	--

4.5.3 Member Data Documentation

4.5.3.1 int D3electrode::cardinalnum [protected]

4.5.3.2 D3electrode** D3electrode::symel [protected]

4.5.3.3 int D3electrode::totalrot [protected]

4.5.3.4 double D3electrode::voltage [protected]

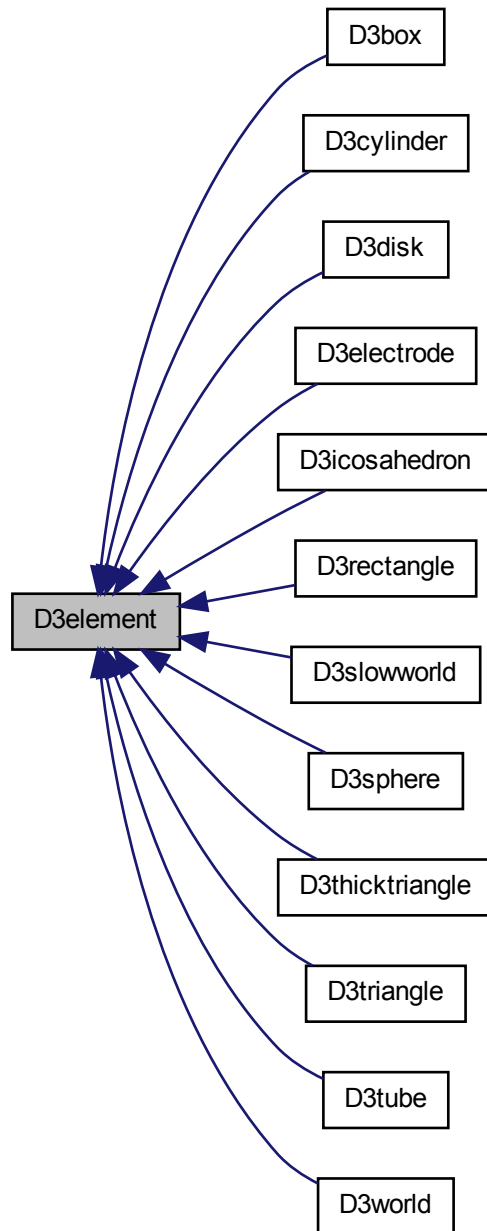
The documentation for this class was generated from the following file:

- **bem.h**

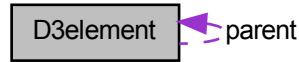
4.6 D3element Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3element:



Collaboration diagram for D3element:



Public Member Functions

- virtual void **correctNorm** (double x0, double y0, double z0)
Uses ray-shooting to determine the norm and should only be used for externally imported geometrical figures.
- virtual bool **IntersectWithRay** (double x0, double y0, double z0, double xdir, double ydir, double zdir, double &mul)
- virtual void **InsertTGeoVolume** (TGeoVolume *top, TGeoMedium *mat, TGeoMedium *matVak, TGeoManager *geom)
- virtual double **GetSelfPotential** ()
- virtual double **GetPotentialAt** (double x, double y, double z)
- virtual double **GetSelfDoubleLayerPotential** ()
- virtual double **GetDoubleLayerPotentialAt** (double x, double y, double z)
- virtual void **GetPotentialAndFieldAt** (double x, double y, double z, double &pot, double &ex, double &ey, double &ez)
- virtual double **GetArea** ()
- virtual void **GetCenter** (double &x, double &y, double &z)
- **D3element** (double x_, double y_, double z_, double phi_, double theta_, double psi_, bool inversenorm_=false, bool refineable_=true, **D3element** *parent_=NULL, double epsilon=0, string name="")
- ~**D3element** ()
- virtual void **GetReferencePoint** (double &x, double &y, double &z)
- virtual void **init** (bool ignorefirst=false)
- virtual void **refine** (double length)
Refines a geometrical figure.
- virtual void **refine** (double length, int num)
- int **GetAmountOfSubelements** ()
- int **PrintAmountOfSubelements** ()
- void **GetListOfBaseElements** (**PD3element** *el, int &cnt)
- virtual void **rotate** (double phi_, double theta_, double psi_)
- virtual void **shift** (double xs, double ys, double zs)
- virtual void **rotate2** (double phi_, double theta_, double psi_, double x, double y, double z, double &x2, double &y2, double &z2)

- virtual void **rotateinv** (double phi_, double theta_, double psi_, double **x**, double **y**, double **z**, double &x2, double &y2, double &z2)
- void **Add** (**PD3element** el)
- virtual void **GetTriangle** (double *A, double *B, double *C, double *COL)
- virtual void **GetRectangle** (double *A, double *B, double *C, double *D, double *COL)
- virtual void **SetNormTowards** (double **x**, double **y**, double **z**, bool towards)
- virtual void **createNewSubelements** (double length)
- virtual string **getName** ()
- virtual void **setName** (string name2)
- virtual bool **GetInversenorm** ()

Public Attributes

- **D3element** * **parent**
- int **Color**
- string **name**
- list< **D3element** * > **subelement**

Protected Member Functions

- virtual void **deleteSubelements** ()

Protected Attributes

- double **epsilon**
- TGeoHMatrix **h**
- double **x**
- double **y**
- double **z**
- double **phi**
- double **theta**
- double **psi**
- bool **isBaseElement**
- bool **refineable**
- bool **inversenorm**

4.6.1 Constructor & Destructor Documentation

4.6.1.1 `D3element::D3element (double x_, double y_, double z_, double phi_, double theta_, double psi_, bool inversenorm_ = false, bool refineable_ = true, D3element * parent_ = NULL, double epsilon = 0, string name = " ")`

4.6.1.2 `D3element::~~D3element ()`

4.6.2 Member Function Documentation

4.6.2.1 `void D3element::Add (PD3element el)`

4.6.2.2 `virtual void D3element::correctNorm (double x0, double y0, double z0)`
[virtual]

Uses ray-shooting to determine the norm and should only be used for externally imported geometrical figures.

From Point (*x0*, *y0*, *z0*) Rays are being shot towards the object in order to determine in which direction the normal vector of each surface is pointing. Generally, all normal vectors can point either to the inside of the object or to the outside of the object. We should make sure that all normal vectors are pointing in the same direction!

A blue color on each outside-surface indicates a correct norm, which means that all normal Vectors are pointing in the same direction. The ions that fly by the electrodes see only the outside-surfaces of the electrodes. If an outside-surface has an incorrect norm, the ion will not be influenced correctly by the surface. **This means that all outside-surfaces should be blue!**

A red color on at least one outside-surface indicates that at least one normal vector is pointing in a different direction than the others. In this case we need to modify the parameters *x0/y0/z0*.

Important for the "shooting point" (*x0,y0,z0*): It must not be in the same plane as one of the surfaces!! Furthermore it must not be inside a geometrical figure!! Otherwise the Ray-Shooting method will not work properly.

Parameters

<i>x0</i>	x-coordinate of the shooting-point
<i>y0</i>	y-coordinate of the shooting-point
<i>z0</i>	z-coordinate of the shooting-point

4.6.2.3 `virtual void D3element::createNewSubelements (double length)` [virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 37).

4.6.2.4 virtual void D3element::deleteSubelements () [protected, virtual]

4.6.2.5 int D3element::GetAmountOfSubelements ()

4.6.2.6 virtual double D3element::GetArea () [virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 37).

4.6.2.7 virtual void D3element::GetCenter (double & x, double & y, double & z)
[virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 37).

4.6.2.8 virtual double D3element::GetDoubleLayerPotentialAt (double x, double y, double z)
[virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 37).

4.6.2.9 virtual bool D3element::GetInversenorm () [virtual]

4.6.2.10 void D3element::GetListOfBaseElements (PD3element * el, int & cnt)

4.6.2.11 virtual string D3element::getName () [virtual]

4.6.2.12 virtual void D3element::GetPotentialAndFieldAt (double x, double y, double z, double
& pot, double & ex, double & ey, double & ez) [virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 38).

4.6.2.13 virtual double D3element::GetPotentialAt (double x, double y, double z)
[virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 38).

4.6.2.14 virtual void D3element::GetRectangle (double * A, double * B, double * C, double *
D, double * COL) [virtual]

Reimplemented in **D3rectangle** (p. 38).

4.6.2.15 virtual void D3element::GetReferencePoint (double & x, double & y, double & z)
[virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 38).

4.6.2.16 virtual double D3element::GetSelfDoubleLayerPotential () [virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 38).

4.6.2.17 virtual double D3element::GetSelfPotential () [virtual]

Reimplemented in **D3triangle** (p. 54), and **D3rectangle** (p. 38).

4.6.2.18 virtual void D3element::GetTriangle (double * *A*, double * *B*, double * *C*, double * *COL*) [virtual]

Reimplemented in **D3triangle** (p. 55).

4.6.2.19 virtual void D3element::init (bool *ignorefirst* = false) [virtual]

4.6.2.20 virtual void D3element::InsertTGeoVolume (TGeoVolume * *top*, TGeoMedium * *mat*, TGeoMedium * *matVak*, TGeoManager * *geom*) [virtual]

Reimplemented in **D3triangle** (p. 55), and **D3rectangle** (p. 38).

4.6.2.21 virtual bool D3element::IntersectWithRay (double *x0*, double *y0*, double *z0*, double *xdir*, double *ydir*, double *zdir*, double & *mul*) [virtual]

Reimplemented in **D3triangle** (p. 55), and **D3rectangle** (p. 38).

4.6.2.22 int D3element::PrintAmountOfSubelements ()

4.6.2.23 virtual void D3element::refine (double *length*, int *num*) [virtual]

4.6.2.24 virtual void D3element::refine (double *length*) [virtual]

Refines a geometrical figure.

This means that the original geometrical object will be filled with smaller versions of itself. Works good with cubic objects.

Does not work good with other objects like cylinders etc..?? //x

The parameter length determines the size of the smaller objects.

Example: Cube with sides (1x1x1) and parameter length 0.1 => 10 small cubes will be fit along each side, so the cube will then contain 10*10*10=1000 small cubes.

Press ctrl+w to see the lattice model and get an impression of the refinement. Press ctrl+r to see the original model again

Parameters

<i>length</i>	size of the smaller objects that fill up the incident object
---------------	--

4.6.2.25 `virtual void D3element::rotate (double phi_, double theta_, double psi_)`
[virtual]

Reimplemented in **D3triangle** (p. 55), and **D3rectangle** (p. 38).

4.6.2.26 `virtual void D3element::rotate2 (double phi_, double theta_, double psi_, double x,
double y, double z, double & x2, double & y2, double & z2)` [virtual]

4.6.2.27 `virtual void D3element::rotateinv (double phi_, double theta_, double psi_, double x,
double y, double z, double & x2, double & y2, double & z2)` [virtual]

4.6.2.28 `virtual void D3element::setName (string name2)` [virtual]

4.6.2.29 `virtual void D3element::SetNormTowards (double x, double y, double z, bool towards
)` [virtual]

Reimplemented in **D3triangle** (p. 55), and **D3rectangle** (p. 39).

4.6.2.30 `virtual void D3element::shift (double xs, double ys, double zs)` [virtual]

Reimplemented in **D3triangle** (p. 55), and **D3rectangle** (p. 39).

4.6.3 Member Data Documentation

4.6.3.1 `int D3element::Color`

4.6.3.2 `double D3element::epsilon` [protected]

4.6.3.3 `TGeoHMatrix D3element::h` [protected]

4.6.3.4 `bool D3element::inversenorm` [protected]

4.6.3.5 `bool D3element::isBaseElement` [protected]

4.6.3.6 `string D3element::name`

4.6.3.7 `D3element* D3element::parent`

4.6.3.8 `double D3element::phi` [protected]

4.6.3.9 `double D3element::psi` [protected]

4.6.3.10 `bool D3element::refineable` [protected]

4.6.3.11 `list<D3element*> D3element::subelement`

4.6.3.12 `double D3element::theta` [protected]

4.6.3.13 `double D3element::x` [protected]

Reimplemented in `D3world` (p. 66).

4.6.3.14 `double D3element::y` [protected]

4.6.3.15 `double D3element::z` [protected]

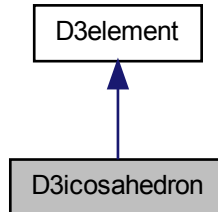
The documentation for this class was generated from the following file:

- `bem.h`

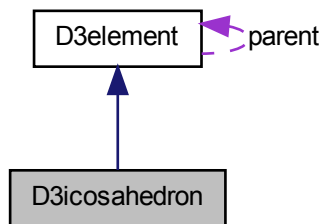
4.7 D3icosahedron Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3icosahedron:



Collaboration diagram for D3icosahedron:



Public Member Functions

- **D3icosahedron** (double edgelength, double **refinement**, double **x**, double **y**, double **z**, double **phi_**, double **theta_**, double **psi_**, bool **inversenorm**=false, **PD3element** **parent**=NULL)
- **~D3icosahedron** ()
- void **triangle** (int a, int b, int c, bool **inversenorm**=false)

Protected Attributes

- double **r**
- double **refinement**
- double(* **V**)[12][3]

4.7.1 Constructor & Destructor Documentation

4.7.1.1 `D3icosahedron::D3icosahedron (double edglength, double refinement, double x, double y, double z, double phi_, double theta_, double psi_, bool inversenorm = false, PD3element parent = NULL)` [inline]

4.7.1.2 `D3icosahedron::~~D3icosahedron ()` [inline]

4.7.2 Member Function Documentation

4.7.2.1 `void D3icosahedron::triangle (int a, int b, int c, bool inversenorm = false)` [inline]

4.7.3 Member Data Documentation

4.7.3.1 `double D3icosahedron::r` [protected]

4.7.3.2 `double D3icosahedron::refinement` [protected]

4.7.3.3 `double(* D3icosahedron::V)[12][3]` [protected]

The documentation for this class was generated from the following file:

- **bem.h**

4.8 D3ImportedElectrodes Class Reference

```
#include <bem.h>
```

Public Member Functions

- **D3ImportedElectrodes ()**
Can be used to import an Electrode from an external .dxf file, that was created by AutoCAD.
- *** ~D3ImportedElectrodes ()**
- **bool Import** (const char *file, bool ignore3DFace_=false, bool ignorePolyline_=false)
Loads the .dxf file into memory.
- virtual void **addLayer** (const DL_LayerData &data)
- virtual void **addPoint** (const DL_PointData &data)
- virtual void **addLine** (const DL_LineData &data)
- virtual void **addArc** (const DL_ArcData &data)
- virtual void **addCircle** (const DL_CircleData &data)
- virtual void **addPolyline** (const DL_PolylineData &data)

- virtual void **addVertex** (const DL_VertexData &data)
- virtual void **add3DFace** (const DL_3DFaceData &data)
- virtual void **addBlockRecord** (const DL_BlockRecordData &data)
- void **endSequence** ()
- void **ListElectrodes** ()

Iterates over all electrodes and displays their names in a command line output.

- **D3electrode & FindElectrode** (const char *name)
- void **printAttributes** ()

Public Attributes

- map< string, **D3electrode** > **electrode**

Protected Attributes

- int **Model_Space_Handle**
- bool **ignore_Model_Space_Handle**
- bool **ignore3DFace**
- bool **ignorePolyline**
- bool **importingPolyline**
- int **vertexcnt**
- double **x** [4]
- double **y** [4]
- double **z** [4]

4.8.1 Constructor & Destructor Documentation

4.8.1.1 D3ImportedElectrodes::D3ImportedElectrodes ()

Can be used to import an Electrode from an external .dxf file, that was created by AutoCAD.

The lines below show an example for proper use of the method

```
D3ImportedElectrodes (p. 30) *impel=new D3ImportedElectrodes() (p. 31);
```

```
TString importfilename;
```

```
logfile.AbsoluteFileName("./Ansaug_endcap.dxf",importfilename);
```

```
if(!impel->Import(importfilename)) return 0;
```

```
endcapl=& (impel->FindElectrode("endcap"));
```

```
endcapl->correctNorm(0,0,-10);
```

```
endcapl->refine(0.02);
```

```
so genug??
```

4.8.1.2 * D3ImportedElectrodes::~~D3ImportedElectrodes ()

4.8.2 Member Function Documentation

4.8.2.1 virtual void D3ImportedElectrodes::add3DFace (const DL_3DFaceData & *data*)
[virtual]

4.8.2.2 virtual void D3ImportedElectrodes::addArc (const DL_ArcData & *data*) [virtual]

4.8.2.3 virtual void D3ImportedElectrodes::addBlockRecord (const DL_BlockRecordData & *data*) [virtual]

4.8.2.4 virtual void D3ImportedElectrodes::addCircle (const DL_CircleData & *data*)
[virtual]

4.8.2.5 virtual void D3ImportedElectrodes::addLayer (const DL_LayerData & *data*)
[virtual]

4.8.2.6 virtual void D3ImportedElectrodes::addLine (const DL_LineData & *data*)
[virtual]

4.8.2.7 virtual void D3ImportedElectrodes::addPoint (const DL_PointData & *data*)
[virtual]

4.8.2.8 virtual void D3ImportedElectrodes::addPolyline (const DL_PolylineData & *data*)
[virtual]

4.8.2.9 virtual void D3ImportedElectrodes::addVertex (const DL_VertexData & *data*)
[virtual]

4.8.2.10 void D3ImportedElectrodes::endSequence ()

4.8.2.11 D3electrode& D3ImportedElectrodes::FindElectrode (const char * *name*)

4.8.2.12 bool D3ImportedElectrodes::Import (const char * *file*, bool *ignore3DFace_* = false,
bool *ignorePolyline_* = false)

Loads the .dxf file into memory.

Displays the filename given by *file* , in a command line output.

Returns false and gives an error message if the file cannot be opened.

so genug??

Parameters

<i>file</i>	name of the .dxf file
-------------	-----------------------

4.8.2.13 `void D3ImportedElectrodes::ListElectrodes ()`

Iterates over all electrodes and displays their names in a command line output.

4.8.2.14 `void D3ImportedElectrodes::printAttributes ()`

4.8.3 Member Data Documentation

4.8.3.1 `map<string,D3electrode> D3ImportedElectrodes::electrode`

4.8.3.2 `bool D3ImportedElectrodes::ignore3DFace` [protected]

4.8.3.3 `bool D3ImportedElectrodes::ignore_Model_Space_Handle`
[protected]

4.8.3.4 `bool D3ImportedElectrodes::ignorePolyline` [protected]

4.8.3.5 `bool D3ImportedElectrodes::importingPolyline` [protected]

4.8.3.6 `int D3ImportedElectrodes::Model_Space_Handle` [protected]

4.8.3.7 `int D3ImportedElectrodes::vertexcnt` [protected]

4.8.3.8 `double D3ImportedElectrodes::x[4]` [protected]

4.8.3.9 `double D3ImportedElectrodes::y[4]` [protected]

4.8.3.10 `double D3ImportedElectrodes::z[4]` [protected]

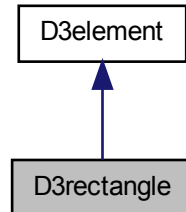
The documentation for this class was generated from the following file:

- `bem.h`

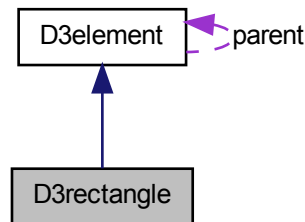
4.9 D3rectangle Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3rectangle:



Collaboration diagram for D3rectangle:



Public Member Functions

- virtual void **InsertTGeoVolume** (TGeoVolume *top, TGeoMedium *matVak, TGeoMedium *mat, TGeoManager *geom)
- virtual bool **IntersectWithRay** (double x0, double y0, double z0, double xdir, double ydir, double zdir, double &mul)
- virtual void **SetNormTowards** (double x, double y, double z, bool towards)
- virtual void **GetReferencePoint** (double &x, double &y, double &z)
- virtual double **GetArea** ()
- virtual void **GetCenter** (double &x, double &y, double &z)
- virtual double **GetSelfPotential** ()
- virtual double **GetPotentialAt** (double x, double y, double z)
- virtual double **GetSelfDoubleLayerPotential** ()

- virtual double **GetDoubleLayerPotentialAt** (double **x**, double **y**, double **z**)
- virtual void **GetPotentialAndFieldAt** (double **x**, double **y**, double **z**, double &pot, double &ex, double &ey, double &ez)
- virtual void **GetRectangle** (double *A, double *B, double *C, double *D, double *COL)
- virtual void **rotate** (double phi_, double theta_, double psi_)
- virtual void **shift** (double xs, double ys, double zs)
- **D3rectangle** (double xa_, double ya_, double xc_, double yc_, double **x**, double **y**, double **z**, double phi_, double theta_, double psi_, bool **inversenorm**=false, **PD3element parent**=NULL, bool subdivideInSquares=false)

Creates a rectangle from 2 2-dimensional vectors, 1 3-dimensional center vector and 3 Euler angles.
- **D3rectangle** (double xa_, double ya_, double xb_, double yb_, double xc_, double yc_, double **x**, double **y**, double **z**, double phi_, double theta_, double psi_, bool **inversenorm**=false, **PD3element parent**=NULL)

Creates a rectangle from 3 2-dimensional vectors, 1 center vector and 3 Euler angles.
- **D3rectangle** (char *str, double x1_, double y1_, double z1_, double x2_, double y2_, double z2_, double x3_, double y3_, double z3_, double x4_, double y4_, double z4_, bool **inversenorm**=false, **PD3element parent**=NULL)

Creates a rectangle from coordinates for each corner.
- ~**D3rectangle** ()
- virtual void **createNewSubelements** (double length)

Public Attributes

- double **x1**
- double **y1**
- double **z1**
- double **x2**
- double **y2**
- double **z2**
- double **x3**
- double **y3**
- double **z3**
- double **x4**
- double **y4**
- double **z4**

Protected Attributes

- double **xa**
- double **ya**
- double **xb**

- double **yb**
- double **xc**
- double **yc**

4.9.1 Constructor & Destructor Documentation

4.9.1.1 D3rectangle::D3rectangle (double *xa_*, double *ya_*, double *xc_*, double *yc_*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, bool *inversenorm* = false, PD3element *parent* = NULL, bool *subdivideInSquares* = false)

Creates a rectangle from 2 2-dimensional vectors, 1 3-dimensional center vector and 3 Euler angles.

The parameters x,y,z determine the 1. corner of the rectangle. The Euler-rotations are also performed with respect to this corner.

(x,y,z) can also be interpreted as a center of a coordinate system.

The 2. corner is determined by (*xa_* + x, *ya_* + y).

The 3. corner is determined by (*xb_* + x, *yb_* + y).

The 4. corner is determined by (*xa_* + *xb_* + x, *ya_* + *yb_* + y). This represents a vector addition of the vectors for the first 2 corners.

The ordering of the corners does not matter!!

The angles *phi_*, *theta_* and *psi_* can be declared in degrees and represent Euler-angles.

The Center (x,y,z) specifies the center of the Euler-rotations

They can be used to generate z-coordinates for the rectangle

The Euler-rotations occur in the following order:

phi_:.....Rotation around z-axis.....x -> x'y -> y'z -> z

theta_:.....Rotation around the rotated x-axis (x'):.....x'-> x'y' -> y''z -> z'

psi_:.....Rotation around z'-axis:.....x'-> x''y'' -> y'''z' -> z''

4.9.1.2 D3rectangle::D3rectangle (double *xa_*, double *ya_*, double *xb_*, double *yb_*, double *xc_*, double *yc_*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, bool *inversenorm* = false, PD3element *parent* = NULL)

Creates a rectangle from 3 2-dimensional vectors, 1 center vector and 3 Euler angles.

The parameters x,y,z determine the 1. corner of the rectangle. The Euler-rotations are also performed with respect to this corner.

(x,y,z) can also be interpreted as a center of a coordinate system.

The 2. corner is determined by (*xa_* + x, *ya_* + y).

The 3. corner is determined by (*xb_* + x, *yb_* + y).

The 4. corner is not being calculated automatically, it is specified via (*xc_*,*yc_*)!

The ordering of the corners does not matter!!

If the 3 corner-vectors would result in a non-convex object only a triangle will be drawn!

The angles phi_, theta_ and psi_ can be declared in degrees and represent Euler-angles.

The Center (x,y,z) specifies the center of the Euler-rotations

They can be used to generate z-coordinates for the rectangle

The Euler-rotations occur in the following order:

phi_:.....Rotation around z-axis:.....x -> x'.....y -> y'.....z -> z
 theta_:.....Rotation around the rotated x-axis (x'):.....x' -> x''.....y' -> y''.....z -> z'
 psi_:.....Rotation around z'-axis:.....x'' -> x'''.....y'' -> y'''.....z' -> z''

4.9.1.3 D3rectangle::D3rectangle (char * str, double x1_, double y1_, double z1_, double x2_, double y2_, double z2_, double x3_, double y3_, double z3_, double x4_, double y4_, double z4_, bool inversenorm = false, PD3element parent = NULL)

Creates a rectangle from coordinates for each corner.

Parameters x1_, y1_, z1_ define the coordinates of the first corner

The remaining paramters define the other corners

wofür der string??!--> nur wegen überladung?

4.9.1.4 D3rectangle::~~D3rectangle () [inline]

4.9.2 Member Function Documentation

4.9.2.1 virtual void D3rectangle::createNewSubelements (double length) [virtual]

Reimplemented from **D3element** (p. 24).

4.9.2.2 virtual double D3rectangle::GetArea () [virtual]

Reimplemented from **D3element** (p. 25).

4.9.2.3 virtual void D3rectangle::GetCenter (double & x, double & y, double & z) [virtual]

Reimplemented from **D3element** (p. 25).

4.9.2.4 virtual double D3rectangle::GetDoubleLayerPotentialAt (double x, double y, double z) [inline, virtual]

Reimplemented from **D3element** (p. 25).

4.9.2.5 `virtual void D3rectangle::GetPotentialAndFieldAt (double x, double y, double z, double & pot, double & ex, double & ey, double & ez) [virtual]`

Reimplemented from **D3element** (p. 25).

4.9.2.6 `virtual double D3rectangle::GetPotentialAt (double x, double y, double z) [inline, virtual]`

Reimplemented from **D3element** (p. 25).

4.9.2.7 `virtual void D3rectangle::GetRectangle (double * A, double * B, double * C, double * D, double * COL) [virtual]`

Reimplemented from **D3element** (p. 25).

4.9.2.8 `virtual void D3rectangle::GetReferencePoint (double & x, double & y, double & z) [virtual]`

Reimplemented from **D3element** (p. 25).

4.9.2.9 `virtual double D3rectangle::GetSelfDoubleLayerPotential () [virtual]`

Reimplemented from **D3element** (p. 26).

4.9.2.10 `virtual double D3rectangle::GetSelfPotential () [virtual]`

Reimplemented from **D3element** (p. 26).

4.9.2.11 `virtual void D3rectangle::InsertTGeoVolume (TGeoVolume * top, TGeoMedium * matVak, TGeoMedium * mat, TGeoManager * geom) [virtual]`

Reimplemented from **D3element** (p. 26).

4.9.2.12 `virtual bool D3rectangle::IntersectWithRay (double x0, double y0, double z0, double xdir, double ydir, double zdir, double & mul) [virtual]`

Reimplemented from **D3element** (p. 26).

4.9.2.13 `virtual void D3rectangle::rotate (double phi_, double theta_, double psi_) [virtual]`

Reimplemented from **D3element** (p. 27).

4.9.2.14 `virtual void D3rectangle::SetNormTowards (double x, double y, double z, bool towards) [virtual]`

Reimplemented from **D3element** (p. 27).

4.9.2.15 `virtual void D3rectangle::shift (double xs, double ys, double zs) [virtual]`

Reimplemented from **D3element** (p. 27).

4.9.3 Member Data Documentation

4.9.3.1 double **D3rectangle::x1**

4.9.3.2 double **D3rectangle::x2**

4.9.3.3 double **D3rectangle::x3**

4.9.3.4 double **D3rectangle::x4**

4.9.3.5 double **D3rectangle::xa** [protected]

4.9.3.6 double **D3rectangle::xb** [protected]

4.9.3.7 double **D3rectangle::xc** [protected]

4.9.3.8 double **D3rectangle::y1**

4.9.3.9 double **D3rectangle::y2**

4.9.3.10 double **D3rectangle::y3**

4.9.3.11 double **D3rectangle::y4**

4.9.3.12 double **D3rectangle::ya** [protected]

4.9.3.13 double **D3rectangle::yb** [protected]

4.9.3.14 double **D3rectangle::yc** [protected]

4.9.3.15 double **D3rectangle::z1**

4.9.3.16 double **D3rectangle::z2**

4.9.3.17 double **D3rectangle::z3**

4.9.3.18 double **D3rectangle::z4**

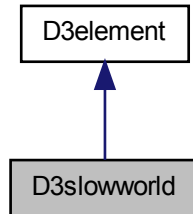
The documentation for this class was generated from the following file:

- **bem.h**

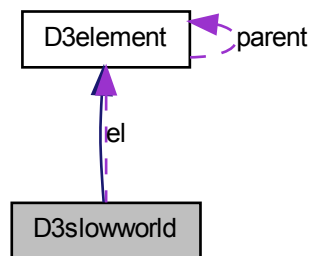
4.10 D3slowworld Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3slowworld:



Collaboration diagram for D3slowworld:



Public Member Functions

- **D3slowworld** ()
- **~D3slowworld** ()
- void **insert** (**D3electrode** *el)
- void **GetListOfBaseElements** (**PD3element** *el, int ***electrodeIndexLimit**, int &cnt)
- void **solve** ()
- double **calc** (double x, double y, double z)
- void **draw** ()

Protected Attributes

- double * **f**
- double * **dfdAll**
- int * **electrodeIndexLimit**
- **PD3element** * **el**
- int **n**
- int **amountOfElectrodes**

4.10.1 Constructor & Destructor Documentation

4.10.1.1 **D3slowworld::D3slowworld** ()

4.10.1.2 **D3slowworld::~~D3slowworld** ()

4.10.2 Member Function Documentation

4.10.2.1 double **D3slowworld::calc** (double *x*, double *y*, double *z*)

4.10.2.2 void **D3slowworld::draw** ()

4.10.2.3 void **D3slowworld::GetListOfBaseElements** (**PD3element** * *el*, int * *electrodeIndexLimit*, int & *cnt*)

4.10.2.4 void **D3slowworld::insert** (**D3electrode** * *el*)

4.10.2.5 void **D3slowworld::solve** ()

4.10.3 Member Data Documentation

4.10.3.1 int **D3slowworld::amountOfElectrodes** [protected]

4.10.3.2 double* **D3slowworld::dfdAll** [protected]

4.10.3.3 **PD3element*** **D3slowworld::el** [protected]

4.10.3.4 int* **D3slowworld::electrodeIndexLimit** [protected]

4.10.3.5 double* **D3slowworld::f** [protected]

4.10.3.6 int **D3slowworld::n** [protected]

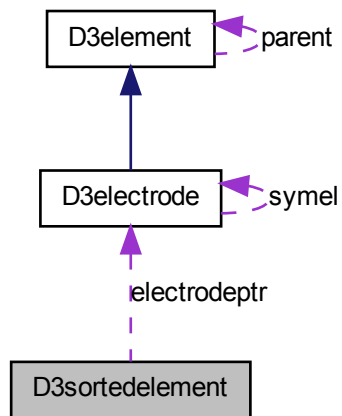
The documentation for this class was generated from the following file:

- **bem.h**

4.11 D3sortedelement Class Reference

```
#include <bem.h>
```

Collaboration diagram for D3sortedelement:



Public Member Functions

- **D3sortedelement** ()
- **~D3sortedelement** ()

Public Attributes

- bool **done**
- double **axis**
- double **radius**
- int **index**
- **D3electrode** * **electrodeptr**

Friends

- bool **SmallerThan** (**D3sortedelement** &a, **D3sortedelement** &b, double eps)

4.11.1 Constructor & Destructor Documentation

4.11.1.1 `D3sortedelement::D3sortedelement ()` [`inline`]

4.11.1.2 `D3sortedelement::~~D3sortedelement ()` [`inline`]

4.11.2 Friends And Related Function Documentation

4.11.2.1 `bool SmallerThan (D3sortedelement & a, D3sortedelement & b, double eps)`
[`friend`]

4.11.3 Member Data Documentation

4.11.3.1 `double D3sortedelement::axis`

4.11.3.2 `bool D3sortedelement::done`

4.11.3.3 `D3electrode* D3sortedelement::electrodeptr`

4.11.3.4 `int D3sortedelement::index`

4.11.3.5 `double D3sortedelement::radius`

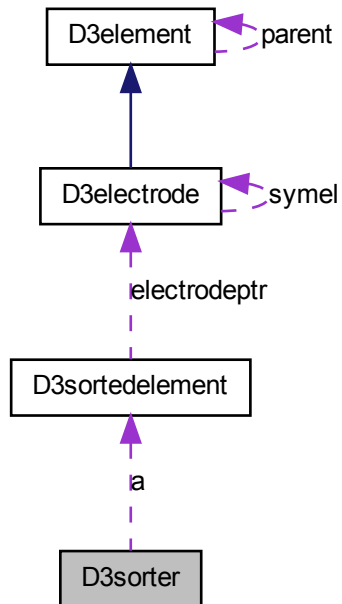
The documentation for this class was generated from the following file:

- `bem.h`

4.12 D3sorter Class Reference

```
#include <bem.h>
```

Collaboration diagram for D3sorter:



Public Member Functions

- void **sort** (**D3sortedelement** *arr, int size, double epsi)
- void **quicksort** (int l, int r)
- void **exchange** (int i, int j)

Protected Attributes

- double **eps**

Private Attributes

- **D3sortedelement** * **a**
- int **n**

4.12.1 Member Function Documentation

4.12.1.1 `void D3sorter::exchange (int i, int j)` [inline]

4.12.1.2 `void D3sorter::quicksort (int l, int r)` [inline]

4.12.1.3 `void D3sorter::sort (D3sortedelement * arr, int size, double epsi)` [inline]

4.12.2 Member Data Documentation

4.12.2.1 `D3sortedelement* D3sorter::a` [private]

4.12.2.2 `double D3sorter::eps` [protected]

4.12.2.3 `int D3sorter::n` [private]

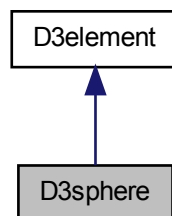
The documentation for this class was generated from the following file:

- `bem.h`

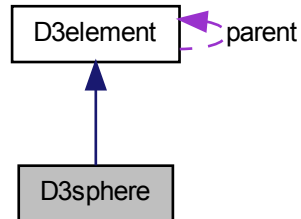
4.13 D3sphere Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3sphere:



Collaboration diagram for D3sphere:



Public Member Functions

- **D3sphere** (double **r**, double **refinement**, double **x**, double **y**, double **z**, double **phi_**, double **theta_**, double **psi_**, bool **inversenorm**=false, **PD3element parent**=NULL)
Creates a sphere with radius r.
- **~D3sphere** ()

Protected Attributes

- double **r**
- double **refinement**

4.13.1 Constructor & Destructor Documentation

4.13.1.1 **D3sphere::D3sphere** (double *r*, double *refinement*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, bool *inversenorm* = false, **PD3element parent** = NULL) [inline]

Creates a sphere with radius r.

The sphere is centered at (x,y,z).

The parameter refinement gives the order of magnitude for the small triangles which assemble the sphere.

Press ctrl+w to see the lattice model and get an impression of the refinement.

Press ctrl+r to see the original model again.

The angles phi_, theta_ and psi_ can be declared in degrees and represent Euler-angles.

The Center (x,y,z) specifies the center of the Euler-rotations

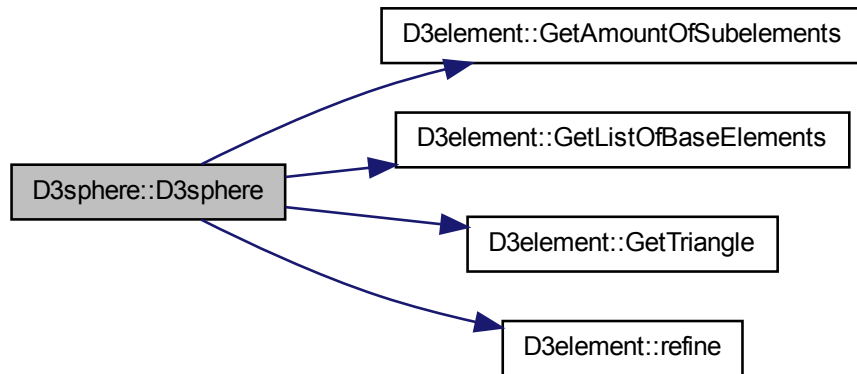
The Euler-rotations occur in the following order:

phi_:.....Rotation around z-axis.....x -> x'.....y -> y'.....z -> z

theta_:.....Rotation around the rotated x-axis (x'):.....x'-> x'.....y' -> y''.....z -> z'

psi_:.....Rotation around z'-axis:.....x'-> x''.....y''-> y'''.....z'-> z'

Here is the call graph for this function:



4.13.1.2 `D3sphere::~~D3sphere () [inline]`

4.13.2 Member Data Documentation

4.13.2.1 `double D3sphere::r [protected]`

4.13.2.2 `double D3sphere::refinement [protected]`

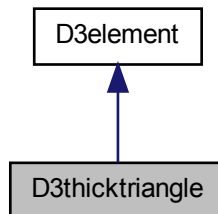
The documentation for this class was generated from the following file:

- **bem.h**

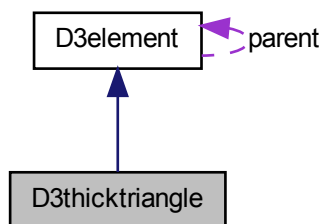
4.14 D3thicktriangle Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3thicktriangle:



Collaboration diagram for D3thicktriangle:



Public Member Functions

- **D3thicktriangle** (double **xa**, double **ya**, double **xb**, double **yb**, double **htriangle**, double **x**, double **y**, double **z**, double **phi**, double **theta**, double **psi**, bool **inversenorm**=false, **PD3element parent**=NULL)

Creates a thick triangle from 2 2-dimensional vectors, 1 3-dimensional center vector and 3 Euler-angles.

- **~D3thicktriangle** ()

Protected Attributes

- double **xa**

- double **ya**
- double **xb**
- double **yb**
- double **htriangle**

4.14.1 Constructor & Destructor Documentation

4.14.1.1 D3thicktriangle::D3thicktriangle (double *xa*, double *ya*, double *xb*, double *yb*, double *htriangle*, double *x*, double *y*, double *z*, double *phi*, double *theta*, double *psi*, bool *inversenorm* = false, PD3element *parent* = NULL) [inline]

Creates a thick triangle from 2 2-dimensional vectors, 1 3-dimensional center vector and 3 Euler-angles.

The parameter *htriangle* determines the thickness of the triangle. For $\phi = \theta = \psi = 0$ the thickness corresponds to the extension in z-direction.

The parameters *x,y,z* determine the 1. corner of the triangle. The Euler-rotations are also performed with respect to this corner.

(*x,y,z*) can also be interpreted as a center of a coordinate system.

The 2. corner is determined by (*xa* + *x*, *ya* + *y*). The 3. corner is determined by (*xb* + *x*, *yb* + *y*).

The ordering does not matter!! The angles ϕ , θ and ψ can be declared in degrees and represent Euler-angles.

The Center (*x,y,z*) specifies the center of the Euler-rotations

The Euler-rotations occur in the following order:

ϕ :.....Rotation around z-axis.....x -> x'.....y -> y'.....z -> z
 θ :.....Rotation around the rotated x-axis (x').....x' -> x''.....y' -> y''.....z -> z'
 ψ :.....Rotation around z'-axis:.....x'' -> x'''.....y'' -> y'''.....z' -> z''

4.14.1.2 D3thicktriangle::~D3thicktriangle () [inline]

4.14.2 Member Data Documentation

4.14.2.1 double D3thicktriangle::htriangle [protected]

4.14.2.2 double D3thicktriangle::xa [protected]

4.14.2.3 double D3thicktriangle::xb [protected]

4.14.2.4 double D3thicktriangle::ya [protected]

4.14.2.5 double D3thicktriangle::yb [protected]

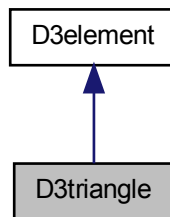
The documentation for this class was generated from the following file:

- `bem.h`

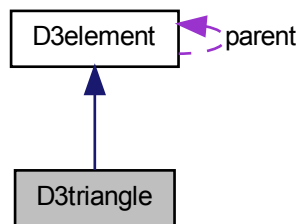
4.15 D3triangle Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3triangle:



Collaboration diagram for D3triangle:



Public Member Functions

- virtual void **InsertTGeoVolume** (TGeoVolume *top, TGeoMedium *matVak, TGeoMedium *mat, TGeoManager *geom)
- virtual bool **IntersectWithRay** (double x0, double y0, double z0, double xdir, double ydir, double zdir, double &mul)

- virtual void **SetNormTowards** (double **x**, double **y**, double **z**, bool towards)
- virtual void **GetReferencePoint** (double &**x**, double &**y**, double &**z**)
- virtual double **GetArea** ()
- virtual void **GetCenter** (double &**x**, double &**y**, double &**z**)
- virtual double **GetSelfPotential** ()
- virtual double **GetPotentialAt** (double **x**, double **y**, double **z**)
- virtual void **GetPotentialAndFieldAt** (double **x**, double **y**, double **z**, double &pot, double &ex, double &ey, double &ez)
- virtual double **GetSelfDoubleLayerPotential** ()
- virtual double **GetDoubleLayerPotentialAt** (double **x**, double **y**, double **z**)
- **D3triangle** (double xa_, double ya_, double xb_, double yb_, double **x**, double **y**, double **z**, double phi_, double theta_, double psi_, bool **inversenorm**=false, **PD3element parent**=NULL, bool **stripeit**=true)

Creates a triangle from 2 2-dimensional vectors, 1 3-dimensional center vector and 3 Euler-angles.

- **D3triangle** (double x1_, double y1_, double z1_, double x2_, double y2_, double z2_, double x3_, double y3_, double z3_, bool **inversenorm**=false, **PD3element parent**=NULL, bool **stripeit**=true)

Creates a triangle from 3 3-dimensional vectors.

- **~D3triangle** ()
- virtual void **rotate** (double phi_, double theta_, double psi_)
- virtual void **shift** (double xs, double ys, double zs)
- virtual void **GetTriangle** (double *A, double *B, double *C, double *COL)
- virtual void **createNewSubelements** (double length)

Static Public Member Functions

- static bool **IntersectWithRay** (double ax, double ay, double az, double bx, double by, double bz, double cx, double cy, double cz, double xdir, double ydir, double zdir, double &mul)

Public Attributes

- double **x1**
- double **y1**
- double **z1**
- double **x2**
- double **y2**
- double **z2**
- double **x3**
- double **y3**
- double **z3**

Protected Member Functions

- void **Stripeit** (double length)
- void **Newtriangles** (double length)

Protected Attributes

- bool **stripeit**
- double **xa**
- double **ya**
- double **xb**
- double **yb**

4.15.1 Constructor & Destructor Documentation

4.15.1.1 D3triangle::D3triangle (double xa_, double ya_, double xb_, double yb_, double x, double y, double z, double phi_, double theta_, double psi_, bool inversenorm = false, PD3element parent = NULL, bool stripeit = true)

Creates a triangle from 2 2-dimensional vectors, 1 3-dimensional center vector and 3 Euler-angles.

The parameters x,y,z determine the 1. corner of the triangle. The Euler-rotations are also performed with respect to this corner.

(x,y,z) can also be interpreted as a center of a coordinate system.

The 2. corner is determined by (xa_ + x, ya_ + y). The 3. corner is determined by (xb_ + x, yb_ + y). The ordering does not matter!!

The angles phi_, theta_ and psi_ can be declared in degrees and represent Euler-angles.

The Center (x,y,z) specifies the center of the Euler-rotations

They can be used to generate z-coordinates for the triangle

The Euler-rotations occur in the following order:

phi_:.....Rotation around z-axis:.....x -> x'y -> y'z -> z
 theta_:.....Rotation around the rotated x-axis (x'):.....x' -> x''y' -> y''z -> z'
 psi_:.....Rotation around z'-axis:.....x'' -> x'''y'' -> y'''z' -> z''

4.15.1.2 D3triangle::D3triangle (double x1_, double y1_, double z1_, double x2_, double y2_, double z2_, double x3_, double y3_, double z3_, bool inversenorm = false, PD3element parent = NULL, bool stripeit = true)

Creates a triangle from 3 3-dimensional vectors.

(x1_,y1_,z1_), (x2_,y2_,z2_), (x3_,y3_,z3_) define the corners of the triangle. The ordering does not matter!!

4.15.1.3 `D3triangle::~~D3triangle ()`

4.15.2 Member Function Documentation

4.15.2.1 `virtual void D3triangle::createNewSubelements (double length) [virtual]`

Reimplemented from `D3element` (p. 24).

4.15.2.2 `virtual double D3triangle::GetArea () [virtual]`

Reimplemented from `D3element` (p. 25).

4.15.2.3 `virtual void D3triangle::GetCenter (double & x, double & y, double & z) [virtual]`

Reimplemented from `D3element` (p. 25).

4.15.2.4 `virtual double D3triangle::GetDoubleLayerPotentialAt (double x, double y, double z) [inline, virtual]`

Reimplemented from `D3element` (p. 25).

4.15.2.5 `virtual void D3triangle::GetPotentialAndFieldAt (double x, double y, double z, double & pot, double & ex, double & ey, double & ez) [virtual]`

Reimplemented from `D3element` (p. 25).

4.15.2.6 `virtual double D3triangle::GetPotentialAt (double x, double y, double z) [inline, virtual]`

Reimplemented from `D3element` (p. 25).

4.15.2.7 `virtual void D3triangle::GetReferencePoint (double & x, double & y, double & z) [virtual]`

Reimplemented from `D3element` (p. 25).

4.15.2.8 `virtual double D3triangle::GetSelfDoubleLayerPotential () [virtual]`

Reimplemented from `D3element` (p. 26).

4.15.2.9 `virtual double D3triangle::GetSelfPotential () [virtual]`

Reimplemented from `D3element` (p. 26).

4.15.2.10 virtual void D3triangle::GetTriangle (double * *A*, double * *B*, double * *C*, double * *COL*) [virtual]

Reimplemented from **D3element** (p. 26).

4.15.2.11 virtual void D3triangle::InsertTGeoVolume (TGeoVolume * *top*, TGeoMedium * *matVak*, TGeoMedium * *mat*, TGeoManager * *geom*) [virtual]

Reimplemented from **D3element** (p. 26).

4.15.2.12 virtual bool D3triangle::IntersectWithRay (double *x0*, double *y0*, double *z0*, double *xdir*, double *ydir*, double *zdir*, double & *mul*) [virtual]

Reimplemented from **D3element** (p. 26).

4.15.2.13 static bool D3triangle::IntersectWithRay (double *ax*, double *ay*, double *az*, double *bx*, double *by*, double *bz*, double *cx*, double *cy*, double *cz*, double *xdir*, double *ydir*, double *zdir*, double & *mul*) [static]

4.15.2.14 void D3triangle::Newtriangles (double *length*) [protected]

4.15.2.15 virtual void D3triangle::rotate (double *phi_*, double *theta_*, double *psi_*) [virtual]

Reimplemented from **D3element** (p. 27).

4.15.2.16 virtual void D3triangle::SetNormTowards (double *x*, double *y*, double *z*, bool *towards*) [virtual]

Reimplemented from **D3element** (p. 27).

4.15.2.17 virtual void D3triangle::shift (double *xs*, double *ys*, double *zs*) [virtual]

Reimplemented from **D3element** (p. 27).

4.15.2.18 void D3triangle::Stripeit (double *length*) [protected]

4.15.3 Member Data Documentation

4.15.3.1 bool D3triangle::stripeit [protected]

4.15.3.2 double D3triangle::x1

4.15.3.3 double D3triangle::x2

4.15.3.4 double D3triangle::x3

4.15.3.5 double D3triangle::xa [protected]

4.15.3.6 double D3triangle::xb [protected]

4.15.3.7 double D3triangle::y1

4.15.3.8 double D3triangle::y2

4.15.3.9 double D3triangle::y3

4.15.3.10 double D3triangle::ya [protected]

4.15.3.11 double D3triangle::yb [protected]

4.15.3.12 double D3triangle::z1

4.15.3.13 double D3triangle::z2

4.15.3.14 double D3triangle::z3

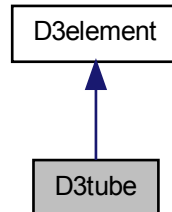
The documentation for this class was generated from the following file:

- **bem.h**

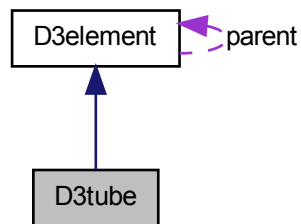
4.16 D3tube Class Reference

```
#include <bem.h>
```

Inheritance diagram for D3tube:



Collaboration diagram for D3tube:



Public Member Functions

- **D3tube** (double r_, double r2_, double l_, double x, double y, double z, double phi_, double theta_, double psi_, int sectors_=12, bool subdivide=true, bool inversenorm=false, **PD3element parent**=NULL)

Creates a tube with outer radius r_, inner radius r2_ and length l_.

- **~D3tube** ()

Protected Attributes

- double **r**
- double **r2**

- double **l**
- double **sectors**

4.16.1 Constructor & Destructor Documentation

4.16.1.1 **D3tube::D3tube** (double *r_*, double *r2_*, double *l_*, double *x*, double *y*, double *z*, double *phi_*, double *theta_*, double *psi_*, int *sectors_* = 12, bool *subdivide* = true, bool *inversenorm* = false, PD3element *parent* = NULL) [inline]

Creates a tube with outer radius *r_*, inner radius *r2_* and length *l_*.

r2_ should be smaller than *r_* for the correct norm.

The bottom of the tube is centered at (*x*,*y*,*z*).

The angles *phi_*, *theta_* and *psi_* can be declared in degrees and represent Euler-angles.

The Center (*x*,*y*,*z*) specifies the center of the Euler-rotations

The Euler-rotations occur in the following order:

phi_:.....Rotation around z-axis.....*x* -> *x'*.....*y* -> *y'*.....*z* -> *z*

theta_:.....Rotation around the rotated x-axis (*x'*):.....*x'* -> *x''*.....*y'* -> *y''*.....*z* -> *z'*

psi_:.....Rotation around *z'*-axis:.....*x''* -> *x'''*.....*y''* -> *y'''*.....*z'* -> *z'*

For *phi_* = *theta_* = *psi_* = 0, the bottom of the tube lies in the *x*,*y*-plane and the *z*-extension correspond to the length *l_*.

The parameter *sectors_* provides an option for refinement. It is set to 12 by initialization but can also be set manually.

A value of *sectors_* = 8 creates a cylinder with 8 edges on the bottom for example. *sectors_* is not limited to powers of two or even numbers!! 5 or 11 works as well.

4.16.1.2 **D3tube::~~D3tube** () [inline]

4.16.2 Member Data Documentation

4.16.2.1 double **D3tube::l** [protected]

4.16.2.2 double **D3tube::r** [protected]

4.16.2.3 double **D3tube::r2** [protected]

4.16.2.4 double **D3tube::sectors** [protected]

The documentation for this class was generated from the following file:

- **bem.h**

- **D3world** (const char *_cachefilename, double **tol**=0.001, int **maxit**=32, int **numMom**=2, int **numLev**=4, double **spaceunit**=0.001, int **segmentation**=1000)
- **~D3world** ()
- void **insert** (**D3electrode** *el)
- void **GetListOfBaseElements** (**PD3element** *el, int ***electrodeIndexLimit**, int &cnt)
- void **SymmetrizeCharges** (int axis=1, double **epsilon**=0.00001, bool ignoremirror=false)
- void **solve** ()
- double **calc** (double **x**, double **y**, double **z**)
Computes matrices that are required for the calculations of the potentials.
- void **calc** (double **xxx**, double **y**, double **z**, double &pot, double &feldx, double &feldy, double &feldz)
- void **calc** (double **xmin**, double **xmax**, int **nx**, double **ymin**, double **ymax**, int **ny**, double **zmin**, double **zmax**, int **nz**)
Caching data for access with calc(x,y,z) or calc(x,y,z,&pot,&feldx,&feldy,&feldz). //x.
- void **calc** (int xxxnum, double *xxx, double *Potential)
Calculating only p otentials. //x.
- void **calc** (int xxxnum, double *xxx, double *Potential, double *FieldX, double *FieldY, double *FieldZ)
Calculating potentials and fields. //x.
- double **calc_slow** (double **x**, double **y**, double **z**)
- void **calc_slow** (double **xmin**, double **xmax**, int **nx**, double **ymin**, double **ymax**, int **ny**, double **zmin**, double **zmax**, int **nz**)
- void **calc_slow** (int xxxnum, double *xxx, double *Potential, double *FieldX, double *FieldY, double *FieldZ)
Calculating potentials and fields. //x.
- void **SetScalePostCalc** (double xscale2, double yscale2, double zscale2)
- bool **IsEqualSurfaceElement** (int amountOfVertices, double eps, double *x, double *X)
- void **draw** ()
- void **Dcentroid** (int **shape**, double *pc, double *xcout)
- void **save** (char *fname)
- bool **load** (char *fname)
- int **cut** (int **n**, int max)
- void **savecalc** (char *fname)
- bool **loadcalc** (char *fname)
- void **AssignColors** ()
- unsigned long **update_adler32double** (unsigned long old, double *buf, unsigned long len, int ignorebytes=3)
- void **RefreshChecksum** ()
- void **exportGeometry** (const char *fname)

- void **propagateForwardVerlet** (double x[3], double v[3], double h, double qDivM=eee/mmm, bool onedim=false)
- void **propagateForwardEuler** (double x[3], double v[3], double h, double qDivM=eee/mmm)
- void **propagateForwardVerletRotSymX** (double x[3], double v[3], double h, double qDivM=eee/mmm)
- void **propagateForwardVerletRotSymY** (double x[3], double v[3], double h, double qDivM=eee/mmm)
- void **propagateForwardVerletRotSymZ** (double x[3], double v[3], double h, double qDivM=eee/mmm)
- void **calc2** (int xxxnum, double *xxx, double *Potential, double *FieldX, double *FieldY, double *FieldZ)

Calculating potentials and fields. //x.

- void **calc_slow2** (int xxxnum, double *xxx, double *Potential, double *FieldX, double *FieldY, double *FieldZ)

Calculating potentials and fields. //x.

Public Attributes

- bool **rangeerror**

Protected Member Functions

- void **exch** (double &a, double &b)
- void **RotMirrorSurfaceElement** (int axis, int sym, double *xx)
- unsigned long **update_adler32** (unsigned long old, unsigned char *buf, unsigned long len)

Protected Attributes

- double **xscale**
- double **yscale**
- double **zscale**
- int **totalrot**
- bool **refreshchecksum**
- int **currentel**
- int **segmentation**
- double **spaceunit**
- char **cachefilename** [2000]
- bool **docache**
- double **xmin**
- double **xmax**
- int **nx**
- double **ymin**

- double **ymax**
- int **ny**
- double **zmin**
- double **zmax**
- int **nz**
- double * **potcache**
- double * **feldxcache**
- double * **feldycache**
- double * **feldzcache**
- int **size**
- int **nlhs**
- int **nrhs**
- int **numMom**
- int **numLev**
- int **i**
- int **j**
- char * **shapechar**
- double * **x**
- double * **poten**
- double * **dbydnpotenAll**
- double * **dbydbpoten**
- double * **xcoll**
- double * **xnrm**
- double * **lhsvect**
- double * **rhsvect**
- int * **shape**
- int * **type**
- int * **dtype**
- int * **rhstype**
- int * **lhstype**
- int * **rhsindex**
- int * **lhsindex**
- int **job**
- int **fljob**
- double **error**
- double **max_diri**
- double **ave_diri**
- double **max_neum**
- double **ave_neum**
- double **cnt_diri**
- double **cnt_neum**
- double **tol**
- int **maxit**
- int **numit**
- unsigned long **checksum**
- unsigned long **checksumcalc**

- double * **f**
- double * **dfdnAll**
- int * **electrodeIndexLimit**
- **PD3element** * **el**
- **D3electrode** ** **electrodes**
- int **n**
- int **amountOfElectrodes**

4.17.1 Constructor & Destructor Documentation

4.17.1.1 **D3world::D3world** (unsigned long *num1*, unsigned long *num2*, unsigned long *num3*, unsigned long *num4*, unsigned long *d*, unsigned long *m*, unsigned long *y*)

4.17.1.2 **D3world::D3world** (const char * *_cachefilename*, double *tol* = 0.001, int *maxit* = 32, int *numMom* = 2, int *numLev* = 4, double *spaceunit* = 0.001, int *segmentation* = 1000)

aus der cpp: num is registration key, d is day, m is month, y is year

???

4.17.1.3 **D3world::~D3world** ()

4.17.2 Member Function Documentation

4.17.2.1 void **D3world::AssignColors** ()

4.17.2.2 double **D3world::calc** (double *x*, double *y*, double *z*)

Computes matrices that are required for the calculations of the potentials.

4.17.2.3 void **D3world::calc** (int *xxxnum*, double * *xxx*, double * *Potential*, double * *FieldX*, double * *FieldY*, double * *FieldZ*)

Calculating potentials and fields. //x.

4.17.2.4 void **D3world::calc** (int *xxxnum*, double * *xxx*, double * *Potential*)

Calculating only p otentials. //x.

4.17.2.5 void D3world::calc (double xxx, double y, double z, double & pot, double & feldx, double & feldy, double & feldz)

4.17.2.6 void D3world::calc (double xmin, double xmax, int nx, double ymin, double ymax, int ny, double zmin, double zmax, int nz)

Caching data for access with calc(x,y,z) or calc(x,y,z,&pot,&feldx,&feldy,&feldz). //x.

4.17.2.7 void D3world::calc2 (int xxxnum, double * xxx, double * Potential, double * FieldX, double * FieldY, double * FieldZ)

Calculating potentials and fields. //x.

4.17.2.8 double D3world::calc_slow (double x, double y, double z)

4.17.2.9 void D3world::calc_slow (double xmin, double xmax, int nx, double ymin, double ymax, int ny, double zmin, double zmax, int nz)

4.17.2.10 void D3world::calc_slow (int xxxnum, double * xxx, double * Potential, double * FieldX, double * FieldY, double * FieldZ)

Calculating potentials and fields. //x.

4.17.2.11 void D3world::calc_slow2 (int xxxnum, double * xxx, double * Potential, double * FieldX, double * FieldY, double * FieldZ)

Calculating potentials and fields. //x.

- 4.17.2.12 `int D3world::cut (int n, int max)`
- 4.17.2.13 `void D3world::Dcentroid (int shape, double * pc, double * xcout)`
- 4.17.2.14 `void D3world::draw ()`
- 4.17.2.15 `void D3world::exch (double & a, double & b)` [protected]
- 4.17.2.16 `void D3world::exportGeometry (const char * fname)`
- 4.17.2.17 `void D3world::GetListOfBaseElements (PD3element * el, int * electrodeIndexLimit, int & cnt)`
- 4.17.2.18 `void D3world::insert (D3electrode * el)`
- 4.17.2.19 `bool D3world::IsEqualSurfaceElement (int amountOfVertices, double eps, double * x, double * X)`
- 4.17.2.20 `bool D3world::load (char * fname)`
- 4.17.2.21 `bool D3world::loadcalc (char * fname)`
- 4.17.2.22 `void D3world::propagateForwardEuler (double x[3], double v[3], double h, double qDivM = eee/mmm)`
- 4.17.2.23 `void D3world::propagateForwardVerlet (double x[3], double v[3], double h, double qDivM = eee/mmm, bool onedim = false)`

Propagates particle at position *x[3]* one timestep *h* forward.

Parameters

<i>x[3]</i>	contains x,y,z position at current timestep. After call contains the position at next timestep.
<i>v[3]</i>	contains x,y,z velocity at current timestep.
<i>h</i>	is timestep in seconds.
<i>qDivM</i>	is charge of particle divided by mass.

- 4.17.2.24 void D3world::propagateForwardVerletRotSymX (double *x[3]*, double *v[3]*, double *h*, double *qDivM = eee/mmm*)
- 4.17.2.25 void D3world::propagateForwardVerletRotSymY (double *x[3]*, double *v[3]*, double *h*, double *qDivM = eee/mmm*)
- 4.17.2.26 void D3world::propagateForwardVerletRotSymZ (double *x[3]*, double *v[3]*, double *h*, double *qDivM = eee/mmm*)
- 4.17.2.27 void D3world::RefreshChecksum ()
- 4.17.2.28 void D3world::RotMirrorSurfaceElement (int *axis*, int *sym*, double * *xx*)
[protected]
- 4.17.2.29 void D3world::save (char * *fname*)
- 4.17.2.30 void D3world::savecalc (char * *fname*)
- 4.17.2.31 void D3world::SetScalePostCalc (double *xscale2*, double *yscale2*, double *zscale2*)
- 4.17.2.32 void D3world::solve ()
- 4.17.2.33 void D3world::SymmetrizeCharges (int *axis = 1*, double *epsilon = 0.00001*, bool *ignoremirror = false*)
- 4.17.2.34 unsigned long D3world::update_adler32 (unsigned long *old*, unsigned char * *buf*, unsigned long *len*) [protected]
- 4.17.2.35 unsigned long D3world::update_adler32double (unsigned long *old*, double * *buf*, unsigned long *len*, int *ignorebytes = 3*)

4.17.3 Member Data Documentation

- 4.17.3.1 int D3world::amountOfElectrodes [protected]
- 4.17.3.2 double D3world::ave_diri [protected]
- 4.17.3.3 double D3world::ave_neum [protected]
- 4.17.3.4 char D3world::cachefilename[2000] [protected]
- 4.17.3.5 unsigned long D3world::checksum [protected]
- 4.17.3.6 unsigned long D3world::checksumcalc [protected]
- 4.17.3.7 double D3world::cnt_diri [protected]
- 4.17.3.8 double D3world::cnt_neum [protected]
- 4.17.3.9 int D3world::currentel [protected]

Generated on Wed Feb 9 2011 15:37:33 for Bemsolver by Doxygen

- 4.17.3.10 double* D3world::dbydbpoten [protected]
- 4.17.3.11 double* D3world::dbydnpotenAll [protected]
- 4.17.3.12 double* D3world::dfdAll [protected]
- 4.17.3.13 bool D3world::docache [protected]

- 4.17.3.58 `double* D3world::xcoll` [protected]
- 4.17.3.59 `double D3world::xmax` [protected]
- 4.17.3.60 `double D3world::xmin` [protected]
- 4.17.3.61 `double* D3world::xnrm` [protected]
- 4.17.3.62 `double D3world::xscale` [protected]
- 4.17.3.63 `double D3world::ymax` [protected]
- 4.17.3.64 `double D3world::ymin` [protected]
- 4.17.3.65 `double D3world::yscale` [protected]
- 4.17.3.66 `double D3world::zmax` [protected]
- 4.17.3.67 `double D3world::zmin` [protected]
- 4.17.3.68 `double D3world::zscale` [protected]

The documentation for this class was generated from the following file:

- `bem.h`

4.18 xv Struct Reference

Public Attributes

- `double x`
- `double y`
- `double z`
- `double vx`
- `double vy`
- `double vz`

4.18.1 Member Data Documentation

4.18.1.1 double `xv::vx`

4.18.1.2 double `xv::vy`

4.18.1.3 double `xv::vz`

4.18.1.4 double `xv::x`

4.18.1.5 double `xv::y`

4.18.1.6 double `xv::z`

The documentation for this struct was generated from the following files:

- `linearPaulTrap_edit.cxx`
- `ringTrap_edit.cxx`

Chapter 5

File Documentation

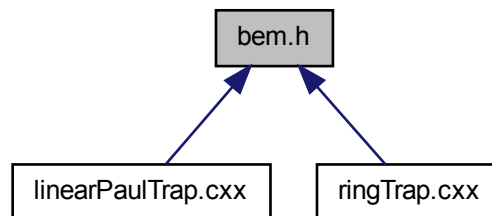
5.1 bem.h File Reference

```
#include <complex>
#include <TGeoMatrix.h>
#include <TGeoManager.h>
#include <TGeoArb8.h>
#include "dl_creationadapter.h"
#include "lapack.h"
#include <float.h>
#include <fftw3.h>
#include <gsl/gsl_sf_bessel.h>
#include <gsl/gsl_sf_ellint.h>
#include <math.h>
#include <gsl/gsl_integration.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_linalg.h>
#include "stdafx.h"
#include <vector>
#include <list>
```

Include dependency graph for bem.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **calcD3triangle**
- class **D3element**
- class **D3electrode**
- class **D3sortedelement**
- class **D3sorter**
- class **D3ImportedElectrodes**
- class **D3triangle**
- class **D3rectangle**
- class **D3box**
- class **D3icosahedron**
- class **D3sphere**
- class **D3disk**
- class **D3cylinder**
- class **D3tube**
- class **D3world**
- class **D3slowworld**
- class **D3thicktriangle**

Defines

- #define **BEMREVISION** 2
- #define **sqr**(x) ((x)*(x))
- #define **size_** 3
- #define **eee** 1.602e-19
- #define **mmm** (40.078*1.66e-27)

Typedefs

- typedef double **trianglefn** (double, double, double, double, double, double)
- typedef **D3element** * **PD3element**
- typedef list< **PD3element** > **PD3elementList**

Functions

- double **testfn** (double x, double y, double z)
- void **q** ()

Variables

- const double **pi** = 3.1415926535897932384626433832795028841971693993751058209749445923078164062862090
- **calcD3triangle calc**

5.1.1 Define Documentation

5.1.1.1 `#define BEMREVISION 2`

5.1.1.2 `#define eee 1.602e-19`

5.1.1.3 `#define mmm (40.078*1.66e-27)`

5.1.1.4 `#define size_ 3`

5.1.1.5 `#define sqr(x)((x)*(x))`

5.1.2 Typedef Documentation

5.1.2.1 `typedef D3element* PD3element`

5.1.2.2 `typedef list<PD3element> PD3elementList`

5.1.2.3 `typedef double trianglefn(double, double, double, double, double, double)`

5.1.3 Function Documentation

5.1.3.1 `void q()`

5.1.3.2 `double testfn(double x, double y, double z)`

5.1.4 Variable Documentation

5.1.4.1 `calcD3triangle calc`

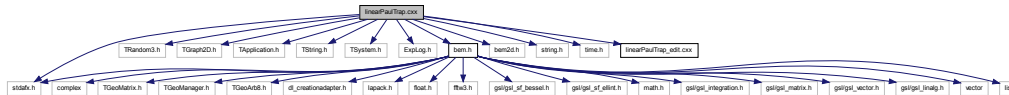
5.1.4.2 `const double pi = 3.1415926535897932384626433832795028841971693993751058209749445923078164062862090`

5.2 linearPaulTrap.cxx File Reference

```
#include "stdafx.h"
#include "TRandom3.h"
#include "TGraph2D.h"
#include <TApplication.h>
#include <TString.h>
#include <TSystem.h>
#include "ExpLog.h"
#include "bem.h"
#include "bem2d.h"
```

```
#include <string.h>
#include <time.h>
#include "linearPaulTrap_edit.cxx"
```

Include dependency graph for linearPaulTrap.cxx:



Defines

- #define **ELCNT** 5

Functions

- int **main** (int argc, char *argv[])

Variables

- **D3world** * wr
- **D3electrode** * eldca [ELCNT]
- **D3electrode** * eldcb [ELCNT]
- **D3electrode** * elrfa
- **D3electrode** * elrfb

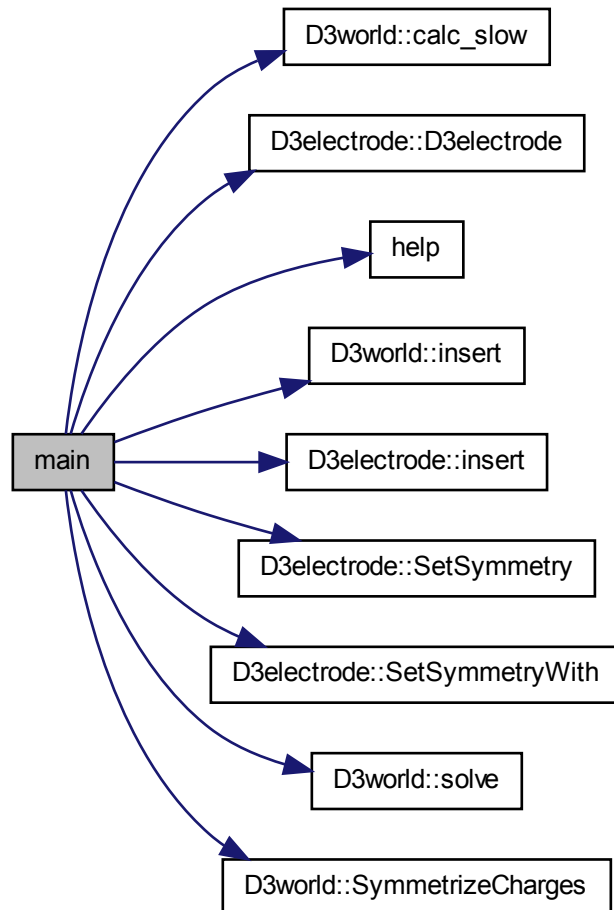
5.2.1 Define Documentation

5.2.1.1 `#define ELCNT 5`

5.2.2 Function Documentation

5.2.2.1 `int main (int argc, char * argv[])`

Here is the call graph for this function:



5.2.3 Variable Documentation

5.2.3.1 D3electrode* eldca[ELCNT]

5.2.3.2 D3electrode * eldcb[ELCNT]

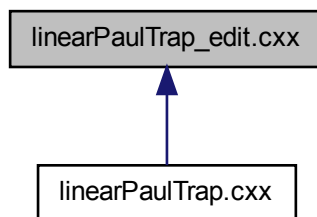
5.2.3.3 D3electrode * elrfa

5.2.3.4 D3electrode * elrfb

5.2.3.5 D3world* wr

5.3 linearPaulTrap_edit.cxx File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct **xv**

Defines

- #define **PI** 3.141592653589793

Functions

- void **help** ()
- void **free** ()
- void **setTrappingVoltage** (double dcac=0, double dcbc=0, double rfac=0, double rfbc=0, double urf=0)

- void **setZero** (double dcac=0, double dcbc=0, double rfac=0, double rfbc=0, double urf=0)
- double **calcit** (double x, double y, double z)
- void **ExportAxialPot** (double zstart=-15, double zstop=15, double zcnt=3000)
- void **PlotPot** (double x1=0, double y1=0, double z1=-5, double x2=0, double y2=0, double z2=5, int num=200, bool plotfield=true)
- void **PlotPseudo** ()
- void **Trajectory** (double rfac=0., double rfbc=0., double dcac=0., double dcbc=0., double x=0.00002, double y=0.00002, double z=0.0, double vx=0, double vy=0, double vz=0, double stoptime=1000./12e6, int steps=500000, double RFVoltage=400./2., double RFfreq=12155000, bool verlet=false)

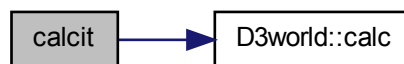
5.3.1 Define Documentation

5.3.1.1 #define PI 3.141592653589793

5.3.2 Function Documentation

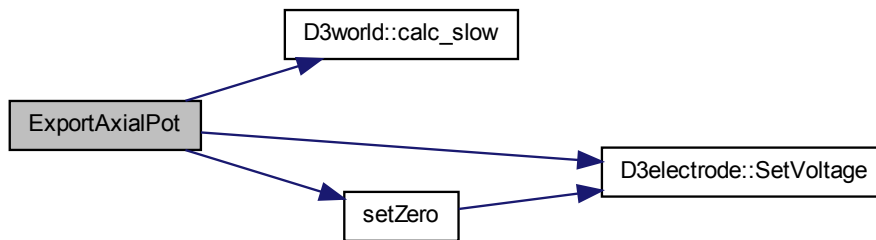
5.3.2.1 double calcit (double x, double y, double z)

Here is the call graph for this function:



5.3.2.2 void ExportAxialPot (double *zstart* = -15, double *zstop* = 15, double *zcnt* = 3000)

Here is the call graph for this function:

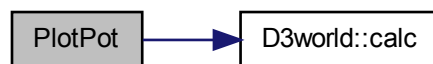


5.3.2.3 void free ()

5.3.2.4 void help ()

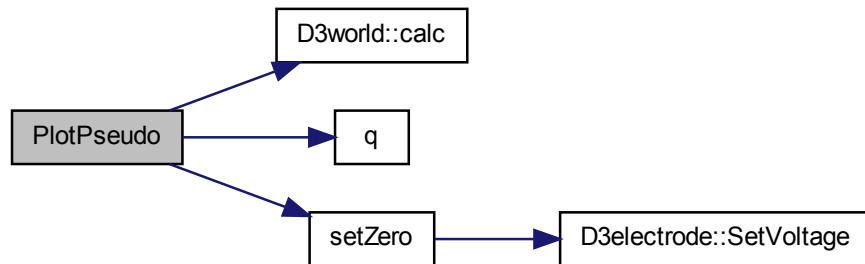
5.3.2.5 void PlotPot (double *x1* = 0, double *y1* = 0, double *z1* = -5, double *x2* = 0, double *y2* = 0, double *z2* = 5, int *num* = 200, bool *plotfield* = true)

Here is the call graph for this function:



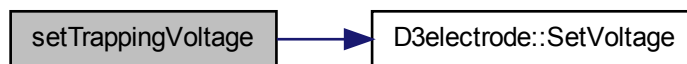
5.3.2.6 void PlotPseudo ()

Here is the call graph for this function:



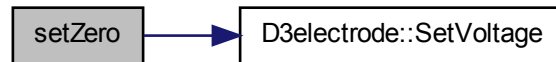
5.3.2.7 void setTrappingVoltage (double *dcac* = 0, double *dcbc* = 0, double *rfac* = 0, double *rfbc* = 0, double *urf* = 0)

Here is the call graph for this function:



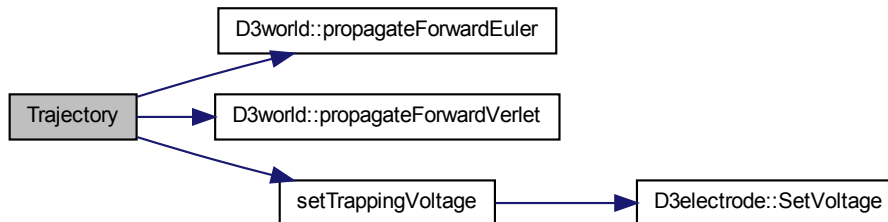
5.3.2.8 void `setZero` (double `dcac` = 0, double `dbcac` = 0, double `rfac` = 0, double `rfbc` = 0, double `urf` = 0)

Here is the call graph for this function:



5.3.2.9 void `Trajectory` (double `rfac` = 0., double `rfbc` = 0., double `dcac` = 0., double `dbcac` = 0., double `x` = 0.00002, double `y` = 0.00002, double `z` = 0.0, double `vx` = 0, double `vy` = 0, double `vz` = 0, double `stoptime` = 1000./12e6, int `steps` = 500000, double `RFVoltage` = 400./2., double `RFfreq` = 12155000, bool `verlet` = false)

Here is the call graph for this function:



5.4 ringTrap.cxx File Reference

```

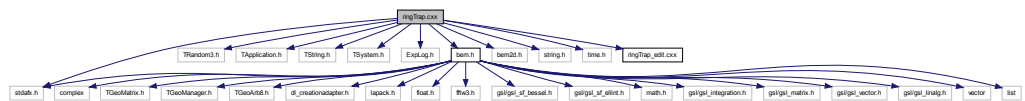
#include "stdafx.h"
#include "TRandom3.h"
#include <TApplication.h>
#include <TString.h>
#include <TSystem.h>
  
```

```

#include "ExpLog.h"
#include "bem.h"
#include "bem2d.h"
#include <string.h>
#include <time.h>
#include "ringTrap_edit.cxx"

```

Include dependency graph for ringTrap.cxx:



Defines

- #define **ELCNT** 5

Functions

- int **main** (int argc, char *argv[])

Variables

- **D3world** * wr
- **D3electrode** * eldrf
- **D3electrode** * eldc1
- **D3electrode** * eldc2

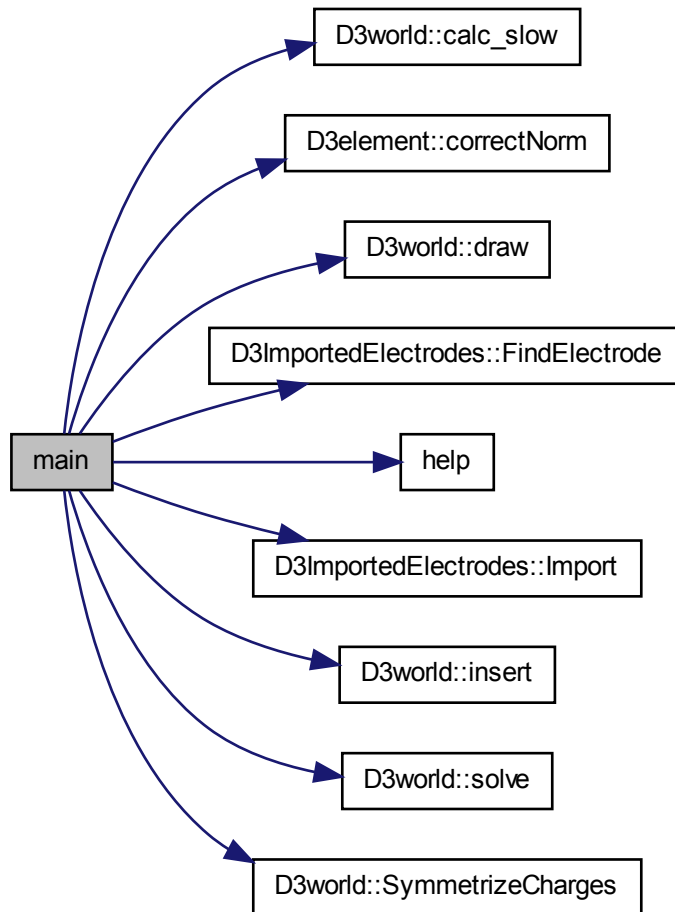
5.4.1 Define Documentation

5.4.1.1 #define ELCNT 5

5.4.2 Function Documentation

5.4.2.1 int main (int argc, char * argv[])

Here is the call graph for this function:



5.4.3 Variable Documentation

5.4.3.1 **D3electrode * eldc1**

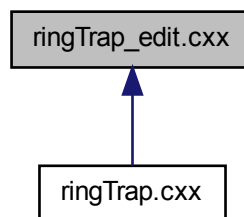
5.4.3.2 **D3electrode * eldc2**

5.4.3.3 **D3electrode* eldrf**

5.4.3.4 **D3world* wr**

5.5 ringTrap_edit.cxx File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct **xv**

Defines

- #define **PI** 3.141592653589793

Functions

- void **help** ()
- void **free** ()
- void **setTrappingVoltage** (double dcac=0, double dcbc=0, double rfac=0, double rfbc=0, double urf=0)
- void **setZero** (double dcac=0, double dcbc=0, double rfac=0, double rfbc=0, double urf=0)
- double **calcit** (double x, double y, double z)

- void **ExportAxialPot** (double zstart=-15, double zstop=15, double zcnt=3000)
- void **PlotPot** (double x1=0, double y1=0, double z1=-5, double x2=0, double y2=0, double z2=5, int num=200, bool plotfield=true)
- void **Trajectory** (double rfac=0., double rfbc=0., double dcac=0., double dcbc=0., double x=0, double y=0, double z=0.0, double vx=0, double vy=0, double vz=0, double stoptime=1000./12e6, int steps=500000, double RFVoltage=400./2., double RFfreq=12155000)

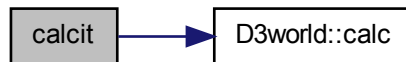
5.5.1 Define Documentation

5.5.1.1 `#define PI 3.141592653589793`

5.5.2 Function Documentation

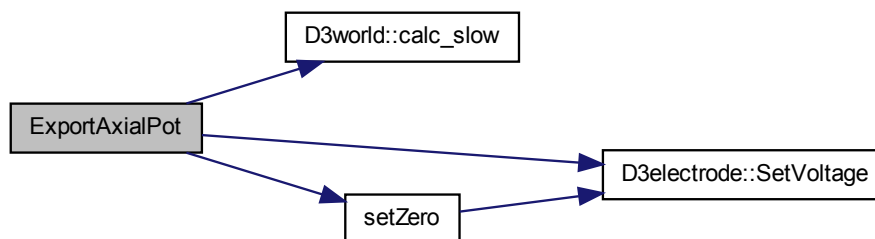
5.5.2.1 `double calcit (double x, double y, double z)`

Here is the call graph for this function:



5.5.2.2 `void ExportAxialPot (double zstart = -15, double zstop = 15, double zcnt = 3000)`

Here is the call graph for this function:



5.5.2.3 `void free ()`

5.5.2.4 `void help ()`

5.5.2.5 `void PlotPot (double x1 = 0, double y1 = 0, double z1 = -5, double x2 = 0, double y2 = 0, double z2 = 5, int num = 200, bool plotfield = true)`

Here is the call graph for this function:



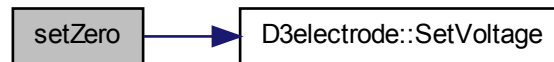
5.5.2.6 `void setTrappingVoltage (double dcac = 0, double dcbc = 0, double rfac = 0, double rfbc = 0, double urf = 0)`

Here is the call graph for this function:



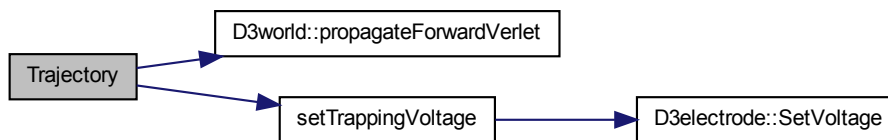
5.5.2.7 void `setZero` (double `dcac` = 0, double `dcbc` = 0, double `rfac` = 0, double `rfbc` = 0, double `urf` = 0)

Here is the call graph for this function:



5.5.2.8 void `Trajectory` (double `rfac` = 0., double `rfbc` = 0., double `dcac` = 0., double `dcbc` = 0., double `x` = 0, double `y` = 0, double `z` = 0.0, double `vx` = 0, double `vy` = 0, double `vz` = 0, double `stoptime` = 1000./12e6, int `steps` = 500000, double `RFVoltage` = 400./2., double `RFfreq` = 12155000)

Here is the call graph for this function:



Index

- ~D3ImportedElectrodes
 - D3ImportedElectrodes, 31
- ~D3box
 - D3box, 11
- ~D3cylinder
 - D3cylinder, 14
- ~D3disk
 - D3disk, 16
- ~D3electrode
 - D3electrode, 18
- ~D3element
 - D3element, 24
- ~D3icosahedron
 - D3icosahedron, 30
- ~D3rectangle
 - D3rectangle, 37
- ~D3slowworld
 - D3slowworld, 42
- ~D3sortedelement
 - D3sortedelement, 44
- ~D3sphere
 - D3sphere, 48
- ~D3thicktriangle
 - D3thicktriangle, 50
- ~D3triangle
 - D3triangle, 53
- ~D3tube
 - D3tube, 58
- ~D3world
 - D3world, 63
- ~calcD3triangle
 - calcD3triangle, 9
- a
 - D3sorter, 46
- Add
 - D3element, 24
- add3DFace
 - D3ImportedElectrodes, 32
- addArc
 - D3ImportedElectrodes, 32
- addBlockRecord
 - D3ImportedElectrodes, 32
- addCircle
 - D3ImportedElectrodes, 32
- addLayer
 - D3ImportedElectrodes, 32
- addLine
 - D3ImportedElectrodes, 32
- addPoint
 - D3ImportedElectrodes, 32
- addPolyline
 - D3ImportedElectrodes, 32
- addVertex
 - D3ImportedElectrodes, 32
- amountOfElectrodes
 - D3slowworld, 42
 - D3world, 66
- AssignColors
 - D3world, 63
- ave_diri
 - D3world, 66
- ave_neum
 - D3world, 66
- axis
 - D3sortedelement, 44
- bem.h, 69
 - BEMREVISION, 72
 - calc, 72
 - eee, 72
 - mmm, 72
 - PD3element, 72
 - PD3elementList, 72
 - pi, 72
 - q, 72
 - size_, 72
 - sqr, 72
 - testfn, 72
 - trianglefn, 72
- BEMREVISION
 - bem.h, 72

-
- cachefilename
 - D3world, 66
 - calc
 - bem.h, 72
 - D3slowworld, 42
 - D3world, 63, 64
 - calc2
 - D3world, 64
 - calc_slow
 - D3world, 64
 - calc_slow2
 - D3world, 64
 - calcD3triangle, 7
 - ~calcD3triangle, 9
 - calcD3triangle, 9
 - dG3Ddx, 9
 - dG3Ddy, 9
 - dG3Ddz, 9
 - EPS, 9
 - eta, 9
 - G3D, 9
 - G3Danalytic, 9
 - G3DdnAnalytic, 9
 - gauleg, 9
 - gauslegsimplextriangle, 9
 - GetIntL1, 9
 - GetIntL1divR3, 9
 - GetIntL1OutOfPlane, 9
 - n, 9
 - numquad, 9
 - triangleint, 9
 - triangleint_pot_xyz, 9
 - w, 9
 - xi, 9
 - calcit
 - linearPaulTrap_edit.cxx, 76
 - ringTrap_edit.cxx, 83
 - cardinalnum
 - D3electrode, 20
 - checksum
 - D3world, 66
 - checksumcalc
 - D3world, 66
 - cnt_diri
 - D3world, 66
 - cnt_neum
 - D3world, 66
 - Color
 - D3element, 28
 - correctNorm
 - D3element, 24
 - D3element, 24
 - createNewSubelements
 - D3element, 24
 - D3rectangle, 37
 - D3triangle, 54
 - currentel
 - D3world, 66
 - cut
 - D3world, 64
 - D3box, 10
 - ~D3box, 11
 - D3box, 11
 - xl, 12
 - yl, 12
 - zl, 12
 - D3cylinder, 12
 - ~D3cylinder, 14
 - D3cylinder, 13
 - l, 14
 - r, 14
 - sectors, 14
 - D3disk, 14
 - ~D3disk, 16
 - D3disk, 15
 - r, 16
 - sectors, 16
 - D3electrode, 16
 - ~D3electrode, 18
 - cardinalnum, 20
 - D3electrode, 18
 - GetCardinalNumber, 18
 - GetSymmetryWith, 18
 - GetTotalrot, 18
 - GetVoltage, 18
 - insert, 18
 - SetCardinalNumber, 18
 - SetSymmetry, 18
 - SetSymmetryWith, 19
 - SetVoltage, 20
 - symel, 20
 - totalrot, 20
 - voltage, 20
 - D3element, 20
 - ~D3element, 24
 - Add, 24
 - Color, 28
 - correctNorm, 24
 - createNewSubelements, 24
 - D3element, 24

- deleteSubelements, 24
- epsilon, 28
- GetAmountOfSubelements, 25
- GetArea, 25
- GetCenter, 25
- GetDoubleLayerPotentialAt, 25
- GetInversenorm, 25
- GetListOfBaseElements, 25
- getName, 25
- GetPotentialAndFieldAt, 25
- GetPotentialAt, 25
- GetRectangle, 25
- GetReferencePoint, 25
- GetSelfDoubleLayerPotential, 25
- GetSelfPotential, 26
- GetTriangle, 26
- h, 28
- init, 26
- InsertTGeoVolume, 26
- IntersectWithRay, 26
- inversenorm, 28
- isBaseElement, 28
- name, 28
- parent, 28
- phi, 28
- PrintAmountOfSubelements, 26
- psi, 28
- refine, 26
- refineable, 28
- rotate, 27
- rotate2, 27
- rotateinv, 27
- setName, 27
- SetNormTowards, 27
- shift, 27
- subelement, 28
- theta, 28
- x, 28
- y, 28
- z, 28
- D3icosahedron, 28
 - ~D3icosahedron, 30
 - D3icosahedron, 30
 - r, 30
 - refinement, 30
 - triangle, 30
 - V, 30
- D3ImportedElectrodes, 30
 - ~D3ImportedElectrodes, 31
 - add3DFace, 32
 - addArc, 32
 - addBlockRecord, 32
 - addCircle, 32
 - addLayer, 32
 - addLine, 32
 - addPoint, 32
 - addPolyline, 32
 - addVertex, 32
 - D3ImportedElectrodes, 31
 - electrode, 33
 - endSequence, 32
 - FindElectrode, 32
 - ignore3DFace, 33
 - ignore_Model_Space_Handle, 33
 - ignorePolyline, 33
 - Import, 32
 - importingPolyline, 33
 - ListElectrodes, 32
 - Model_Space_Handle, 33
 - printAttributes, 33
 - vertexcnt, 33
 - x, 33
 - y, 33
 - z, 33
- D3rectangle, 33
 - ~D3rectangle, 37
 - createNewSubelements, 37
 - D3rectangle, 36, 37
 - GetArea, 37
 - GetCenter, 37
 - GetDoubleLayerPotentialAt, 37
 - GetPotentialAndFieldAt, 37
 - GetPotentialAt, 38
 - GetRectangle, 38
 - GetReferencePoint, 38
 - GetSelfDoubleLayerPotential, 38
 - GetSelfPotential, 38
 - InsertTGeoVolume, 38
 - IntersectWithRay, 38
 - rotate, 38
 - SetNormTowards, 38
 - shift, 39
 - x1, 40
 - x2, 40
 - x3, 40
 - x4, 40
 - xa, 40
 - xb, 40
 - xc, 40
 - y1, 40

- y2, 40
- y3, 40
- y4, 40
- ya, 40
- yb, 40
- yc, 40
- z1, 40
- z2, 40
- z3, 40
- z4, 40
- D3slowworld, 40
 - ~D3slowworld, 42
 - amountOfElectrodes, 42
 - calc, 42
 - D3slowworld, 42
 - dfdnAll, 42
 - draw, 42
 - el, 42
 - electrodeIndexLimit, 42
 - f, 42
 - GetListOfBaseElements, 42
 - insert, 42
 - n, 42
 - solve, 42
- D3sortedelement, 43
 - ~D3sortedelement, 44
 - axis, 44
 - D3sortedelement, 44
 - done, 44
 - electrodeptr, 44
 - index, 44
 - radius, 44
 - SmallerThan, 44
- D3sorter, 44
 - a, 46
 - eps, 46
 - exchange, 46
 - n, 46
 - quicksort, 46
 - sort, 46
- D3sphere, 46
 - ~D3sphere, 48
 - D3sphere, 47
 - r, 48
 - refinement, 48
- D3thicktriangle, 48
 - ~D3thicktriangle, 50
 - D3thicktriangle, 50
 - htriangle, 50
 - xa, 50
 - xb, 50
 - ya, 50
 - yb, 50
- D3triangle, 51
 - ~D3triangle, 53
 - createNewSubelements, 54
 - D3triangle, 53
 - GetArea, 54
 - GetCenter, 54
 - GetDoubleLayerPotentialAt, 54
 - GetPotentialAndFieldAt, 54
 - GetPotentialAt, 54
 - GetReferencePoint, 54
 - GetSelfDoubleLayerPotential, 54
 - GetSelfPotential, 54
 - GetTriangle, 54
 - InsertTGeoVolume, 55
 - IntersectWithRay, 55
 - Newtriangles, 55
 - rotate, 55
 - SetNormTowards, 55
 - shift, 55
 - Stripeit, 55
 - stripeit, 56
 - x1, 56
 - x2, 56
 - x3, 56
 - xa, 56
 - xb, 56
 - y1, 56
 - y2, 56
 - y3, 56
 - ya, 56
 - yb, 56
 - z1, 56
 - z2, 56
 - z3, 56
- D3tube, 56
 - ~D3tube, 58
 - D3tube, 58
 - l, 58
 - r, 58
 - r2, 58
 - sectors, 58
- D3world, 59
 - ~D3world, 63
 - amountOfElectrodes, 66
 - AssignColors, 63
 - ave_diri, 66
 - ave_neum, 66

cachefilename, 66
 calc, 63, 64
 calc2, 64
 calc_slow, 64
 calc_slow2, 64
 checksum, 66
 checksumcalc, 66
 cnt_diri, 66
 cnt_neum, 66
 currentel, 66
 cut, 64
 D3world, 63
 dbydbpoten, 66
 dbydnpotenAll, 66
 Dcentroid, 65
 dfdnAll, 66
 docache, 66
 draw, 65
 dtype, 66
 el, 66
 electrodeIndexLimit, 66
 electrodes, 66
 error, 66
 exch, 65
 exportGeometry, 65
 f, 66
 feldxcache, 66
 feldycache, 66
 feldzcache, 66
 fljob, 66
 GetListOfBaseElements, 65
 i, 66
 insert, 65
 IsEqualSurfaceElement, 65
 j, 66
 job, 66
 lhsindex, 66
 lhstype, 66
 lhsvect, 66
 load, 65
 loadcalc, 65
 max_diri, 66
 max_neum, 66
 maxit, 66
 n, 66
 nlhs, 66
 nrhs, 66
 numit, 66
 numLev, 66
 numMom, 66
 nx, 66
 ny, 66
 nz, 66
 potcache, 66
 poten, 66
 propagateForwardEuler, 65
 propagateForwardVerlet, 65
 propagateForwardVerletRotSymX, 65
 propagateForwardVerletRotSymY, 66
 propagateForwardVerletRotSymZ, 66
 rangeerror, 66
 RefreshChecksum, 66
 refreshchecksum, 66
 rhsindex, 66
 rhstype, 66
 rhsvect, 66
 RotMirrorSurfaceElement, 66
 save, 66
 savecalc, 66
 segmentation, 66
 SetScalePostCalc, 66
 shape, 66
 shapechar, 66
 size, 66
 solve, 66
 spaceunit, 66
 SymmetrizeCharges, 66
 tol, 66
 totalrot, 66
 type, 66
 update_adler32, 66
 update_adler32double, 66
 x, 66
 xcoll, 66
 xmax, 67
 xmin, 67
 xnrm, 67
 xscale, 67
 ymax, 67
 ymin, 67
 yscale, 67
 zmax, 67
 zmin, 67
 zscale, 67
 dbydbpoten
 D3world, 66
 dbydnpotenAll
 D3world, 66
 Dcentroid
 D3world, 65

- deleteSubelements
 - D3element, 24
- dfdnAll
 - D3slowworld, 42
 - D3world, 66
- dG3Ddx
 - calcD3triangle, 9
- dG3Ddy
 - calcD3triangle, 9
- dG3Ddz
 - calcD3triangle, 9
- docache
 - D3world, 66
- done
 - D3sortedelement, 44
- draw
 - D3slowworld, 42
 - D3world, 65
- dtype
 - D3world, 66
- eee
 - bem.h, 72
- el
 - D3slowworld, 42
 - D3world, 66
- ELCNT
 - linearPaulTrap.cxx, 74
 - ringTrap.cxx, 81
- eldc1
 - ringTrap.cxx, 82
- eldc2
 - ringTrap.cxx, 82
- eldca
 - linearPaulTrap.cxx, 75
- eldcb
 - linearPaulTrap.cxx, 75
- eldrf
 - ringTrap.cxx, 82
- electrode
 - D3ImportedElectrodes, 33
- electrodeIndexLimit
 - D3slowworld, 42
 - D3world, 66
- electrodeptr
 - D3sortedelement, 44
- electrodes
 - D3world, 66
- elrfa
 - linearPaulTrap.cxx, 75
- elrfb
 - linearPaulTrap.cxx, 75
- endSequence
 - D3ImportedElectrodes, 32
- EPS
 - calcD3triangle, 9
- eps
 - D3sorter, 46
- epsilon
 - D3element, 28
- error
 - D3world, 66
- eta
 - calcD3triangle, 9
- exch
 - D3world, 65
- exchange
 - D3sorter, 46
- ExportAxialPot
 - linearPaulTrap_edit.cxx, 76
 - ringTrap_edit.cxx, 83
- exportGeometry
 - D3world, 65
- f
 - D3slowworld, 42
 - D3world, 66
- feldxcache
 - D3world, 66
- feldycache
 - D3world, 66
- feldzcache
 - D3world, 66
- FindElectrode
 - D3ImportedElectrodes, 32
- fljob
 - D3world, 66
- free
 - linearPaulTrap_edit.cxx, 77
 - ringTrap_edit.cxx, 84
- G3D
 - calcD3triangle, 9
- G3Danalytic
 - calcD3triangle, 9
- G3DdnAnalytic
 - calcD3triangle, 9
- gauleg
 - calcD3triangle, 9
- gauslegsimplextriangle

- calcD3triangle, 9
- GetAmountOfSubelements
 - D3element, 25
- GetArea
 - D3element, 25
 - D3rectangle, 37
 - D3triangle, 54
- GetCardinalNumber
 - D3electrode, 18
- GetCenter
 - D3element, 25
 - D3rectangle, 37
 - D3triangle, 54
- GetDoubleLayerPotentialAt
 - D3element, 25
 - D3rectangle, 37
 - D3triangle, 54
- GetIntL1
 - calcD3triangle, 9
- GetIntL1divR3
 - calcD3triangle, 9
- GetIntL1OutOfPlane
 - calcD3triangle, 9
- GetInversenorm
 - D3element, 25
- GetListOfBaseElements
 - D3element, 25
 - D3slowworld, 42
 - D3world, 65
- getName
 - D3element, 25
- GetPotentialAndFieldAt
 - D3element, 25
 - D3rectangle, 37
 - D3triangle, 54
- GetPotentialAt
 - D3element, 25
 - D3rectangle, 38
 - D3triangle, 54
- GetRectangle
 - D3element, 25
 - D3rectangle, 38
- GetReferencePoint
 - D3element, 25
 - D3rectangle, 38
 - D3triangle, 54
- GetSelfDoubleLayerPotential
 - D3element, 25
 - D3rectangle, 38
 - D3triangle, 54
- GetSelfPotential
 - D3element, 26
 - D3rectangle, 38
 - D3triangle, 54
- GetSymmetryWith
 - D3electrode, 18
- GetTotalrot
 - D3electrode, 18
- GetTriangle
 - D3element, 26
 - D3triangle, 54
- GetVoltage
 - D3electrode, 18
- h
 - D3element, 28
- help
 - linearPaulTrap_edit.cxx, 77
 - ringTrap_edit.cxx, 84
- htriangle
 - D3thicktriangle, 50
- i
 - D3world, 66
- ignore3DFace
 - D3ImportedElectrodes, 33
- ignore_Model_Space_Handle
 - D3ImportedElectrodes, 33
- ignorePolyline
 - D3ImportedElectrodes, 33
- Import
 - D3ImportedElectrodes, 32
- importingPolyline
 - D3ImportedElectrodes, 33
- index
 - D3sortedelement, 44
- init
 - D3element, 26
- insert
 - D3electrode, 18
 - D3slowworld, 42
 - D3world, 65
- InsertTGeoVolume
 - D3element, 26
 - D3rectangle, 38
 - D3triangle, 55
- IntersectWithRay
 - D3element, 26
 - D3rectangle, 38
 - D3triangle, 55

-
- inversenorm
 - D3element, 28
 - isBaseElement
 - D3element, 28
 - IsEqualSurfaceElement
 - D3world, 65
 - j
 - D3world, 66
 - job
 - D3world, 66
 - l
 - D3cylinder, 14
 - D3tube, 58
 - lhsindex
 - D3world, 66
 - lhstype
 - D3world, 66
 - lhsvect
 - D3world, 66
 - linearPaulTrap.cxx, 72
 - ELCNT, 74
 - eldca, 75
 - eldcb, 75
 - elrfa, 75
 - elrfb, 75
 - main, 74
 - wr, 75
 - linearPaulTrap_edit.cxx, 75
 - calcit, 76
 - ExportAxialPot, 76
 - free, 77
 - help, 77
 - PI, 76
 - PlotPot, 77
 - PlotPseudo, 77
 - setTrappingVoltage, 78
 - setZero, 78
 - Trajectory, 79
 - ListElectrodes
 - D3ImportedElectrodes, 32
 - load
 - D3world, 65
 - loadcalc
 - D3world, 65
 - main
 - linearPaulTrap.cxx, 74
 - ringTrap.cxx, 81
 - max_diri
 - D3world, 66
 - max_neum
 - D3world, 66
 - maxit
 - D3world, 66
 - mmm
 - bem.h, 72
 - Model_Space_Handle
 - D3ImportedElectrodes, 33
 - n
 - calcD3triangle, 9
 - D3slowworld, 42
 - D3sorter, 46
 - D3world, 66
 - name
 - D3element, 28
 - Newtriangles
 - D3triangle, 55
 - nlhs
 - D3world, 66
 - nrhs
 - D3world, 66
 - numit
 - D3world, 66
 - numLev
 - D3world, 66
 - numMom
 - D3world, 66
 - numquad
 - calcD3triangle, 9
 - nx
 - D3world, 66
 - ny
 - D3world, 66
 - nz
 - D3world, 66
 - parent
 - D3element, 28
 - PD3element
 - bem.h, 72
 - PD3elementList
 - bem.h, 72
 - phi
 - D3element, 28
 - PI
 - linearPaulTrap_edit.cxx, 76
 - ringTrap_edit.cxx, 83

- pi
 - bem.h, 72
- PlotPot
 - linearPaulTrap_edit.cxx, 77
 - ringTrap_edit.cxx, 84
- PlotPseudo
 - linearPaulTrap_edit.cxx, 77
- potcache
 - D3world, 66
- poten
 - D3world, 66
- PrintAmountOfSubelements
 - D3element, 26
- printAttributes
 - D3ImportedElectrodes, 33
- propagateForwardEuler
 - D3world, 65
- propagateForwardVerlet
 - D3world, 65
- propagateForwardVerletRotSymX
 - D3world, 65
- propagateForwardVerletRotSymY
 - D3world, 66
- propagateForwardVerletRotSymZ
 - D3world, 66
- psi
 - D3element, 28
- q
 - bem.h, 72
- quicksort
 - D3sorter, 46
- r
 - D3cylinder, 14
 - D3disk, 16
 - D3icosahedron, 30
 - D3sphere, 48
 - D3tube, 58
- r2
 - D3tube, 58
- radius
 - D3sortedelement, 44
- rangeerror
 - D3world, 66
- refine
 - D3element, 26
- refineable
 - D3element, 28
- refinement
 - D3icosahedron, 30
 - D3sphere, 48
- RefreshChecksum
 - D3world, 66
- refreshchecksum
 - D3world, 66
- rhsindex
 - D3world, 66
- rhstype
 - D3world, 66
- rhsvect
 - D3world, 66
- ringTrap.cxx, 79
 - ELCNT, 81
 - eldc1, 82
 - eldc2, 82
 - eldrf, 82
 - main, 81
 - wr, 82
- ringTrap_edit.cxx, 82
 - calcit, 83
 - ExportAxialPot, 83
 - free, 84
 - help, 84
 - PI, 83
 - PlotPot, 84
 - setTrappingVoltage, 84
 - setZero, 84
 - Trajectory, 85
- rotate
 - D3element, 27
 - D3rectangle, 38
 - D3triangle, 55
- rotate2
 - D3element, 27
- rotateinv
 - D3element, 27
- RotMirrorSurfaceElement
 - D3world, 66
- save
 - D3world, 66
- savecalc
 - D3world, 66
- sectors
 - D3cylinder, 14
 - D3disk, 16
 - D3tube, 58
- segmentation
 - D3world, 66

- SetCardinalNumber
 - D3electrode, 18
- setName
 - D3element, 27
- SetNormTowards
 - D3element, 27
 - D3rectangle, 38
 - D3triangle, 55
- SetScalePostCalc
 - D3world, 66
- SetSymmetry
 - D3electrode, 18
- SetSymmetryWith
 - D3electrode, 19
- setTrappingVoltage
 - linearPaulTrap_edit.cxx, 78
 - ringTrap_edit.cxx, 84
- SetVoltage
 - D3electrode, 20
- setZero
 - linearPaulTrap_edit.cxx, 78
 - ringTrap_edit.cxx, 84
- shape
 - D3world, 66
- shapechar
 - D3world, 66
- shift
 - D3element, 27
 - D3rectangle, 39
 - D3triangle, 55
- size
 - D3world, 66
- size_
 - bem.h, 72
- SmallerThan
 - D3sortedelement, 44
- solve
 - D3slowworld, 42
 - D3world, 66
- sort
 - D3sorter, 46
- spaceunit
 - D3world, 66
- sqr
 - bem.h, 72
- Stripeit
 - D3triangle, 55
- stripeit
 - D3triangle, 56
- subelement
 - D3element, 28
- symel
 - D3electrode, 20
- SymmetrizeCharges
 - D3world, 66
- testfn
 - bem.h, 72
- theta
 - D3element, 28
- tol
 - D3world, 66
- totalrot
 - D3electrode, 20
 - D3world, 66
- Trajectory
 - linearPaulTrap_edit.cxx, 79
 - ringTrap_edit.cxx, 85
- triangle
 - D3icosahedron, 30
- trianglefn
 - bem.h, 72
- triangleint
 - calcD3triangle, 9
- triangleint_pot_xyz
 - calcD3triangle, 9
- type
 - D3world, 66
- update_adler32
 - D3world, 66
- update_adler32double
 - D3world, 66
- V
 - D3icosahedron, 30
- vertexcnt
 - D3ImportedElectrodes, 33
- voltage
 - D3electrode, 20
- vx
 - xv, 68
- vy
 - xv, 68
- vz
 - xv, 68
- w
 - calcD3triangle, 9
- wr

- linearPaulTrap.cxx, 75
- ringTrap.cxx, 82
- x
 - D3element, 28
 - D3ImportedElectrodes, 33
 - D3world, 66
 - xv, 68
- x1
 - D3rectangle, 40
 - D3triangle, 56
- x2
 - D3rectangle, 40
 - D3triangle, 56
- x3
 - D3rectangle, 40
 - D3triangle, 56
- x4
 - D3rectangle, 40
- xa
 - D3rectangle, 40
 - D3thicktriangle, 50
 - D3triangle, 56
- xb
 - D3rectangle, 40
 - D3thicktriangle, 50
 - D3triangle, 56
- xc
 - D3rectangle, 40
- xcoll
 - D3world, 66
- xi
 - calcD3triangle, 9
- xl
 - D3box, 12
- xmax
 - D3world, 67
- xmin
 - D3world, 67
- xnrm
 - D3world, 67
- xscale
 - D3world, 67
- xv, 67
 - vx, 68
 - vy, 68
 - vz, 68
 - x, 68
 - y, 68
 - z, 68
- y
 - D3element, 28
 - D3ImportedElectrodes, 33
 - xv, 68
- y1
 - D3rectangle, 40
 - D3triangle, 56
- y2
 - D3rectangle, 40
 - D3triangle, 56
- y3
 - D3rectangle, 40
 - D3triangle, 56
- y4
 - D3rectangle, 40
- ya
 - D3rectangle, 40
 - D3thicktriangle, 50
 - D3triangle, 56
- yb
 - D3rectangle, 40
 - D3thicktriangle, 50
 - D3triangle, 56
- yc
 - D3rectangle, 40
- yl
 - D3box, 12
- ymax
 - D3world, 67
- ymin
 - D3world, 67
- yscale
 - D3world, 67
- z
 - D3element, 28
 - D3ImportedElectrodes, 33
 - xv, 68
- z1
 - D3rectangle, 40
 - D3triangle, 56
- z2
 - D3rectangle, 40
 - D3triangle, 56
- z3
 - D3rectangle, 40
 - D3triangle, 56
- z4
 - D3rectangle, 40
- zl

D3box, 12
zmax
 D3world, 67
zmin
 D3world, 67
zscale
 D3world, 67