

ReMoDy

1.1

Programmer's Guide

Andrei V. Smirnov, May 26, 2009

Contents

1	ReMoDy Source Code Documentation and User's Guide	1
1.1	Contents	1
1.2	Introduction	1
1.3	Installation and Execution	2
1.3.1	Step 1: Extracting the archive	2
1.3.2	Step 2: Compiling	2
1.3.3	Step 3: Running	2
1.4	Setup	2
1.5	Licencing and Support	3
2	Configfile	5
2.1	Configuration File	6
2.1.1	Invocation	6
2.1.2	Sections	6
3	Examples	9
3.1	Examples	10
3.1.1	Syngas Simulation Setup	10
4	Frequently Asked Questions	11
4.1	FAQs	12
5	GUI	15
5.1	Graphical User Interface	16
6	Implementation	19
6.1	Implementation	20
6.1.1	Collision Detection Scheme	20
6.1.1.1	Standard Time-Advancement Scheme	20
6.1.1.2	Space-Time Collision Detection Scheme	20

6.1.2	Linked-Cell Method	21
6.1.3	Multi-Processor Implementation	21
6.1.4	Dynamic Lists	22
7	Model	23
7.1	Physical Model	24
7.1.1	Chemical Reactions	24
7.1.2	Cross-Boundary Species	26
8	Output	27
8.1	Output Procedures	28
9	Namespace Index	29
9.1	Namespace List	29
10	Class Index	31
10.1	Class List	31
11	File Index	33
11.1	File List	33
12	Namespace Documentation	35
12.1	Geom Namespace Reference	35
12.1.1	Detailed Description	35
12.1.2	Function Documentation	35
12.1.2.1	area	35
12.1.2.2	distance	35
12.1.2.3	distance	36
12.1.2.4	distvec	36
12.1.2.5	hash	36
12.1.2.6	length	36
12.1.2.7	normalize	36
12.1.2.8	sclp	36
12.2	GridContainer Namespace Reference	37
12.2.1	Detailed Description	37
12.2.2	Function Documentation	37
12.2.2.1	checkPool	37
12.2.2.2	dimensions	37
12.2.2.3	get	37

12.2.2.4	index	38
12.2.2.5	index	38
12.2.2.6	init	38
12.2.2.7	put	38
12.2.2.8	put	38
12.2.3	Variable Documentation	38
12.2.3.1	cellsize	38
12.2.3.2	mcells	38
12.2.3.3	mcells1	39
12.2.3.4	ncells	39
12.2.3.5	nodes	39
12.2.3.6	pool	39
12.2.3.7	xmax	39
12.2.3.8	xmin	39
12.3	Gui Namespace Reference	40
12.3.1	Detailed Description	43
12.3.2	Enumeration Type Documentation	43
12.3.2.1	AxesParameters	43
12.3.2.2	Color	43
12.3.2.3	ColorSchemes	44
12.3.2.4	Elements	44
12.3.2.5	LineParameters	44
12.3.2.6	Movement	45
12.3.2.7	PointParameters	45
12.3.2.8	ShowPars	45
12.3.2.9	SurfDispType	46
12.3.3	Function Documentation	46
12.3.3.1	animate	46
12.3.3.2	commandMode	46
12.3.3.3	consoleMenu	46
12.3.3.4	display	46
12.3.3.5	displayAxes	47
12.3.3.6	displayMenu	47
12.3.3.7	displaymessage	47
12.3.3.8	drawSegment	47
12.3.3.9	dumpwindow	47

12.3.3.10	Exit	47
12.3.3.11	finish	47
12.3.3.12	getElementColor	47
12.3.3.13	getScaling	48
12.3.3.14	getScaling	48
12.3.3.15	getXLimits	48
12.3.3.16	getXLimits	48
12.3.3.17	helpCommand	48
12.3.3.18	helpDisplay	48
12.3.3.19	init	48
12.3.3.20	initdisp	48
12.3.3.21	InitMaterials	49
12.3.3.22	keyboard	49
12.3.3.23	keyboard	49
12.3.3.24	Materials	49
12.3.3.25	menu	49
12.3.3.26	motion	49
12.3.3.27	mouse	49
12.3.3.28	printGridVecLimits	50
12.3.3.29	printGridXLimits	50
12.3.3.30	printParticleVariables	50
12.3.3.31	printPartVelLimits	50
12.3.3.32	printVariables	50
12.3.3.33	query_extension	50
12.3.3.34	Quit	50
12.3.3.35	readconf	50
12.3.3.36	readparam	50
12.3.3.37	readparam	50
12.3.3.38	refresh	51
12.3.3.39	reshape	51
12.3.3.40	run	51
12.3.3.41	runmany	51
12.3.3.42	selectVariable	51
12.3.3.43	setBackgroundRun	51
12.3.3.44	setElementColor	51
12.3.3.45	setForegroundRun	51

12.3.3.46	showGridElements	52
12.3.3.47	showParticle	52
12.3.3.48	showSphere	52
12.3.3.49	showVector	52
12.3.3.50	showVector	52
12.3.3.51	showVectorComponent	52
12.3.3.52	switchColorScheme	52
12.3.3.53	toggleSpheres	52
12.3.3.54	toggleWindowDump	52
12.3.3.55	writeData	53
12.3.4	Variable Documentation	53
12.3.4.1	animation	53
12.3.4.2	attributeList	53
12.3.4.3	axes	53
12.3.4.4	bface_root	53
12.3.4.5	colorscale	53
12.3.4.6	coms	53
12.3.4.7	configfile	53
12.3.4.8	disp	54
12.3.4.9	domain	54
12.3.4.10	dpy	54
12.3.4.11	dx	54
12.3.4.12	dy	54
12.3.4.13	dz	54
12.3.4.14	finished	54
12.3.4.15	firstR	54
12.3.4.16	iwdump	54
12.3.4.17	lastx	55
12.3.4.18	lasty	55
12.3.4.19	lastz	55
12.3.4.20	lmax	55
12.3.4.21	lmin	55
12.3.4.22	lx	55
12.3.4.23	ly	55
12.3.4.24	lz	55
12.3.4.25	mouseButtons	55

12.3.4.26	movement	56
12.3.4.27	nwdump	56
12.3.4.28	rgbcolor	56
12.3.4.29	rotx	56
12.3.4.30	roty	56
12.3.4.31	scene	56
12.3.4.32	showpar	56
12.3.4.33	step	56
12.3.4.34	tx	56
12.3.4.35	ty	57
12.3.4.36	vars	57
12.3.4.37	vecval	57
12.3.4.38	wdtime	57
12.3.4.39	win	57
12.3.4.40	window	57
12.3.4.41	windowname	57
12.3.4.42	xmax	57
12.3.4.43	xmin	57
12.3.4.44	xo	58
12.3.4.45	zoom	58
12.4	IO Namespace Reference	59
12.4.1	Detailed Description	59
12.4.2	Function Documentation	59
12.4.2.1	getCharAttr	59
12.4.2.2	getIntAttr	59
12.4.2.3	getIntAttr	59
12.4.2.4	getIter	60
12.4.2.5	getTime	60
12.4.2.6	getTimeXML	60
12.4.2.7	initxwd	60
12.4.2.8	parseFloat	60
12.4.2.9	parseFloat	60
12.4.2.10	parseInt	60
12.4.2.11	parseWord	61
12.4.2.12	xwd	61
12.4.3	Variable Documentation	61

12.4.3.1	ioutput	61
12.4.3.2	nxwdump	61
12.5	Palette Namespace Reference	62
12.5.1	Detailed Description	62
12.5.2	Function Documentation	62
12.5.2.1	init	62
12.5.2.2	pickcolor	62
12.5.3	Variable Documentation	62
12.5.3.1	dvi	62
12.5.3.2	vav	62
12.5.3.3	vmn	63
12.5.3.4	vmx	63
12.6	Potential Namespace Reference	64
12.6.1	Detailed Description	64
12.6.2	Function Documentation	64
12.6.2.1	Cutoff	64
12.6.2.2	Cutoff	64
12.6.2.3	derivative	65
12.6.2.4	force	65
12.6.2.5	invdist	65
12.6.2.6	lengthscale	65
12.6.2.7	lengthscale	65
12.6.2.8	strength	65
12.6.2.9	strength	65
12.6.2.10	value	65
12.6.3	Variable Documentation	66
12.6.3.1	cutoff	66
12.6.3.2	eta	66
12.6.3.3	large	66
12.6.3.4	sigma	66
12.6.3.5	small	66
12.7	Run Namespace Reference	67
12.7.1	Detailed Description	68
12.7.2	Function Documentation	68
12.7.2.1	elapsed	68
12.7.2.2	gauss	68

12.7.2.3	init	68
12.7.2.4	readcmdline	68
12.7.2.5	rnd	68
12.7.2.6	seed	69
12.7.2.7	testrnd	69
12.7.2.8	usage	69
12.7.3	Variable Documentation	69
12.7.3.1	configfile	69
12.7.3.2	inputfile	69
12.7.3.3	nthreads	69
12.7.3.4	option	69
12.7.3.5	outputfile	70
12.7.3.6	outputname	70
12.7.3.7	programname	70
12.7.3.8	time	70
12.7.3.9	usage	70
12.7.3.10	worldtime	70
12.8	Species Namespace Reference	71
12.8.1	Detailed Description	71
12.8.2	Enumeration Type Documentation	71
12.8.2.1	Interaction	71
12.8.3	Variable Documentation	71
12.8.3.1	nspecies	71
12.8.3.2	reactions	72
12.8.3.3	species	72
13	Class Documentation	73
13.1	Boundary Class Reference	73
13.1.1	Detailed Description	74
13.1.2	Constructor & Destructor Documentation	74
13.1.2.1	Boundary	74
13.1.2.2	~Boundary	74
13.1.3	Member Function Documentation	75
13.1.3.1	Area	75
13.1.3.2	Area	75
13.1.3.3	init	75
13.1.3.4	Inject	75

13.1.3.5	Temperature	75
13.1.3.6	Temperature	75
13.1.4	Member Data Documentation	76
13.1.4.1	adiabatic	76
13.1.4.2	area	76
13.1.4.3	gases	76
13.1.4.4	idir	76
13.1.4.5	iside	76
13.1.4.6	reactions	76
13.1.4.7	temperature	76
13.1.4.8	type	77
13.1.4.9	xmax	77
13.1.4.10	xmin	77
13.2	Bulk Class Reference	78
13.2.1	Detailed Description	78
13.2.2	Member Function Documentation	79
13.2.2.1	init	79
13.2.2.2	inject	79
13.2.2.3	Temperature	79
13.2.2.4	Temperature	79
13.2.2.5	Volume	79
13.2.2.6	Volume	79
13.2.3	Member Data Documentation	80
13.2.3.1	gases	80
13.2.3.2	temperature	80
13.2.3.3	volume	80
13.2.3.4	xmax	80
13.2.3.5	xmin	80
13.3	Collection< Element > Class Template Reference	81
13.3.1	Detailed Description	82
13.3.2	Constructor & Destructor Documentation	82
13.3.2.1	Collection	82
13.3.2.2	~Collection	83
13.3.3	Member Function Documentation	83
13.3.3.1	append	83
13.3.3.2	append	83

13.3.3.3	append	83
13.3.3.4	append	83
13.3.3.5	append0	83
13.3.3.6	append0	83
13.3.3.7	Current	84
13.3.3.8	Current	84
13.3.3.9	First	84
13.3.3.10	FirstNext	84
13.3.3.11	FirstPrev	84
13.3.3.12	getFirstItem	84
13.3.3.13	getItem	84
13.3.3.14	getItem	84
13.3.3.15	getLastItem	84
13.3.3.16	getNext	84
13.3.3.17	getNext	85
13.3.3.18	getPrev	85
13.3.3.19	getPrev	85
13.3.3.20	go	85
13.3.3.21	goFirst	85
13.3.3.22	goFirst	85
13.3.3.23	goFirstNext	85
13.3.3.24	goFirstNext	85
13.3.3.25	goLast	85
13.3.3.26	goLast	85
13.3.3.27	goLastNext	86
13.3.3.28	goLastNext	86
13.3.3.29	goNext	86
13.3.3.30	goNext	86
13.3.3.31	goPrev	86
13.3.3.32	goPrev	86
13.3.3.33	init	86
13.3.3.34	insert	86
13.3.3.35	insert	86
13.3.3.36	insert	87
13.3.3.37	insert	87
13.3.3.38	insert0	87

13.3.3.39	insert0	87
13.3.3.40	insertAfter	87
13.3.3.41	isFirst	87
13.3.3.42	isFirst	87
13.3.3.43	isFirstLast	87
13.3.3.44	isLast	88
13.3.3.45	isLast	88
13.3.3.46	Last	88
13.3.3.47	LastNext	88
13.3.3.48	LastPrev	88
13.3.3.49	maxCounters	88
13.3.3.50	maxnumber	88
13.3.3.51	Next	88
13.3.3.52	Next	88
13.3.3.53	number	88
13.3.3.54	Prev	89
13.3.3.55	Prev	89
13.3.3.56	remove	89
13.3.3.57	remove	89
13.3.3.58	remove0	89
13.3.3.59	setFirst	89
13.3.3.60	setFirstLast	89
13.3.4	Member Data Documentation	89
13.3.4.1	current	89
13.3.4.2	current1	90
13.3.4.3	dead	90
13.3.4.4	first	90
13.3.4.5	items	90
13.3.4.6	last	91
13.3.4.7	locked	91
13.3.4.8	mitems	91
13.3.4.9	nitems	91
13.4	Gui::ColorScale Struct Reference	92
13.4.1	Detailed Description	92
13.4.2	Member Data Documentation	92
13.4.2.1	relative	92

13.5 Container< Element > Class Template Reference	93
13.5.1 Detailed Description	95
13.5.2 Constructor & Destructor Documentation	95
13.5.2.1 Container	95
13.5.2.2 ~Container	95
13.5.3 Member Function Documentation	95
13.5.3.1 append	95
13.5.3.2 append	95
13.5.3.3 append	95
13.5.3.4 append	96
13.5.3.5 append0	96
13.5.3.6 append0	96
13.5.3.7 append1	96
13.5.3.8 checkPool	96
13.5.3.9 Current	96
13.5.3.10 Current	96
13.5.3.11 First	96
13.5.3.12 First	97
13.5.3.13 FirstNext	97
13.5.3.14 FirstNext	97
13.5.3.15 FirstPrev	97
13.5.3.16 FirstPrev	97
13.5.3.17 getFirstPtr	97
13.5.3.18 getFirstPtr	97
13.5.3.19 getLastPtr	98
13.5.3.20 getNext	98
13.5.3.21 getNext	98
13.5.3.22 getPrev	98
13.5.3.23 getPrev	98
13.5.3.24 getPtr	98
13.5.3.25 getPtr	98
13.5.3.26 go	98
13.5.3.27 go	99
13.5.3.28 goFirst	99
13.5.3.29 goFirst	99
13.5.3.30 goFirstNext	99

13.5.3.31 goFirstNext	99
13.5.3.32 goLast	99
13.5.3.33 goLast	99
13.5.3.34 goLastNext	99
13.5.3.35 goLastNext	100
13.5.3.36 goNext	100
13.5.3.37 goNext	100
13.5.3.38 goPrev	100
13.5.3.39 goPrev	100
13.5.3.40 insert	100
13.5.3.41 insert	100
13.5.3.42 insert	100
13.5.3.43 insert	101
13.5.3.44 insert0	101
13.5.3.45 insert0	101
13.5.3.46 insert1	101
13.5.3.47 isFirst	101
13.5.3.48 isFirst	101
13.5.3.49 isFirstLast	101
13.5.3.50 isFirstLast	102
13.5.3.51 isLast	102
13.5.3.52 isLast	102
13.5.3.53 islocked	102
13.5.3.54 Last	102
13.5.3.55 LastNext	102
13.5.3.56 LastPrev	102
13.5.3.57 link	102
13.5.3.58 link0	103
13.5.3.59 lock	103
13.5.3.60 Next	103
13.5.3.61 Next	103
13.5.3.62 number	103
13.5.3.63 Prev	103
13.5.3.64 Prev	103
13.5.3.65 remove	103
13.5.3.66 remove	104

13.5.3.67	remove	104
13.5.3.68	remove	104
13.5.3.69	remove0	104
13.5.3.70	remove0	104
13.5.3.71	setFirst	104
13.5.3.72	setFirst	105
13.5.3.73	setFirstLast	105
13.5.3.74	setFirstLast	105
13.5.3.75	unlink	105
13.5.3.76	unlink0	105
13.5.3.77	unlock	105
13.5.4	Member Data Documentation	105
13.5.4.1	current	105
13.5.4.2	current1	106
13.5.4.3	first	106
13.5.4.4	first1	106
13.5.4.5	last	107
13.5.4.6	locked	107
13.5.4.7	nptrs	107
13.6	Domain Class Reference	108
13.6.1	Detailed Description	109
13.6.2	Constructor & Destructor Documentation	109
13.6.2.1	Domain	109
13.6.2.2	~Domain	110
13.6.3	Member Function Documentation	110
13.6.3.1	boundary	110
13.6.3.2	BoundaryType	111
13.6.3.3	BoundaryType	111
13.6.3.4	computeBounds	111
13.6.3.5	init	111
13.6.3.6	injection	111
13.6.3.7	interact	113
13.6.3.8	interaction	113
13.6.3.9	interaction	113
13.6.3.10	load	114
13.6.3.11	maxBound	114

13.6.3.12 minBound	114
13.6.3.13 Molecules	114
13.6.3.14 run	114
13.6.3.15 save	115
13.6.3.16 setMaxBound	115
13.6.3.17 setMinBound	115
13.6.3.18 step	115
13.6.4 Member Data Documentation	115
13.6.4.1 boundaries	115
13.6.4.2 boundaryName	115
13.6.4.3 bounds	116
13.6.4.4 bulk	116
13.6.4.5 distribution	116
13.6.4.6 molecules	116
13.6.4.7 pool	116
13.7 Gui::ElementDisp Struct Reference	117
13.7.1 Detailed Description	117
13.7.2 Member Data Documentation	117
13.7.2.1 lighting	117
13.7.2.2 rgbcolor	117
13.8 Species::Gas Struct Reference	118
13.8.1 Detailed Description	118
13.8.2 Constructor & Destructor Documentation	118
13.8.2.1 Gas	118
13.8.3 Member Data Documentation	118
13.8.3.1 density	118
13.8.3.2 specie	119
13.9 Item< Element > Struct Template Reference	120
13.9.1 Detailed Description	120
13.9.2 Member Data Documentation	120
13.9.2.1 element	120
13.9.2.2 next	120
13.9.2.3 prev	120
13.10List< Element > Class Template Reference	121
13.10.1 Detailed Description	122
13.10.2 Constructor & Destructor Documentation	122

13.10.2.1 List	122
13.10.2.2 List	122
13.10.2.3 ~List	122
13.10.3 Member Function Documentation	122
13.10.3.1 append	122
13.10.3.2 append	123
13.10.3.3 cleanup	123
13.10.3.4 Current	123
13.10.3.5 erase	123
13.10.3.6 erase	123
13.10.3.7 eraseall	123
13.10.3.8 First	123
13.10.3.9 FirstNext	124
13.10.3.10 FirstPrev	124
13.10.3.11 getFirstPointer	124
13.10.3.12 getLastPointer	124
13.10.3.13 getNext	124
13.10.3.14 getPointer	124
13.10.3.15 getPrev	124
13.10.3.16 go	124
13.10.3.17 goFirst	124
13.10.3.18 goFirstNext	124
13.10.3.19 goLast	125
13.10.3.20 goLastNext	125
13.10.3.21 goNext	125
13.10.3.22 goPrev	125
13.10.3.23 init	125
13.10.3.24 insert	125
13.10.3.25 insert	125
13.10.3.26 sFirst	125
13.10.3.27 sFirstLast	125
13.10.3.28 sLast	126
13.10.3.29 Last	126
13.10.3.30 LastNext	126
13.10.3.31 LastPrev	126
13.10.3.32 link	126

13.10.3.33	locate	126
13.10.3.34	locate	126
13.10.3.35	moveAfterFirst	126
13.10.3.36	moveBehindFirst	126
13.10.3.37	Next	127
13.10.3.38	number	127
13.10.3.39	prepend	127
13.10.3.40	Prev	127
13.10.3.41	remove	127
13.10.3.42	remove	127
13.10.3.43	setFirst	127
13.10.3.44	setFirstLast	127
13.10.3.45	swapAfterFirst	127
13.10.3.46	unlink	128
13.10.4	Member Data Documentation	128
13.10.4.1	current	128
13.10.4.2	first	128
13.10.4.3	last	128
13.10.4.4	nelements	129
13.11	Molecule Class Reference	130
13.11.1	Detailed Description	131
13.11.2	Constructor & Destructor Documentation	131
13.11.2.1	Molecule	131
13.11.3	Member Function Documentation	131
13.11.3.1	Coordinate	131
13.11.3.2	Coordinate	131
13.11.3.3	Coordinates	131
13.11.3.4	Coordinates	131
13.11.3.5	copy	131
13.11.3.6	InternalEnergy	131
13.11.3.7	InternalEnergy	132
13.11.3.8	KineticEnergy	132
13.11.3.9	Move	132
13.11.3.10	setCoordinates	132
13.11.3.11	setVelocity	132
13.11.3.12	Temperature	132

13.11.3.13Type	133
13.11.3.14Type	133
13.11.3.15Velocity	133
13.11.3.16Velocity	133
13.11.3.17Velocity	133
13.11.3.18Velocity	133
13.11.4 Member Data Documentation	134
13.11.4.1 energy	134
13.11.4.2 type	134
13.11.4.3 v	134
13.11.4.4 x	134
13.12 Option Struct Reference	135
13.12.1 Detailed Description	135
13.12.2 Member Data Documentation	135
13.12.2.1 debug	135
13.12.2.2 mesh	135
13.12.2.3 restart	135
13.12.2.4 tags	136
13.12.2.5 verbose	136
13.12.2.6 xterm	136
13.13 Pointer< Element > Struct Template Reference	137
13.13.1 Detailed Description	137
13.13.2 Member Data Documentation	137
13.13.2.1 element	137
13.13.2.2 next	137
13.13.2.3 prev	137
13.14 Pool< Element > Struct Template Reference	138
13.14.1 Detailed Description	138
13.14.2 Constructor & Destructor Documentation	138
13.14.2.1 Pool	138
13.14.2.2 ~Pool	138
13.14.3 Member Function Documentation	138
13.14.3.1 check	138
13.14.3.2 get	139
13.14.3.3 put	139
13.14.3.4 size	139

13.14.4 Member Data Documentation	139
13.14.4.1 hook	139
13.14.4.2 mptrs	139
13.14.4.3 nptrs	140
13.15Ptr< Element > Struct Template Reference	141
13.15.1 Detailed Description	141
13.15.2 Member Data Documentation	141
13.15.2.1 element	141
13.15.2.2 next	141
13.15.2.3 prev	141
13.16Species::Reaction Struct Reference	142
13.16.1 Detailed Description	142
13.16.2 Constructor & Destructor Documentation	142
13.16.2.1 Reaction	142
13.16.2.2 ~Reaction	143
13.16.3 Member Function Documentation	143
13.16.3.1 Add	143
13.16.3.2 Erase	143
13.16.3.3 First	143
13.16.3.4 Next	143
13.16.4 Member Data Documentation	143
13.16.4.1 current	143
13.16.4.2 outcomes	144
13.17Species::Reaction::Outcome Struct Reference	145
13.17.1 Detailed Description	145
13.17.2 Constructor & Destructor Documentation	145
13.17.2.1 Outcome	145
13.17.2.2 Outcome	146
13.17.3 Member Function Documentation	146
13.17.3.1 ActivationEnergy	146
13.17.3.2 ActivationEnergy	146
13.17.3.3 Enthalpy	146
13.17.3.4 Enthalpy	146
13.17.3.5 Probability	146
13.17.3.6 Probability	147
13.17.3.7 Product	147

13.17.3.8 Products	147
13.17.3.9 Time	147
13.17.3.10Time	147
13.17.4 Member Data Documentation	147
13.17.4.1 activationEnergy	147
13.17.4.2 enthalpy	147
13.17.4.3 next	147
13.17.4.4 probability	148
13.17.4.5 product	148
13.17.4.6 time	148
13.18Gui::Scene Struct Reference	149
13.18.1 Detailed Description	149
13.18.2 Member Data Documentation	149
13.18.2.1 color	149
13.18.2.2 frame	149
13.18.2.3 mesh	149
13.19Gui::Scene::Color Struct Reference	150
13.19.1 Detailed Description	150
13.19.2 Member Data Documentation	150
13.19.2.1 maxvalue	150
13.19.2.2 minvalue	150
13.19.2.3 scheme	150
13.20Gui::Scene::Frame Struct Reference	151
13.20.1 Detailed Description	151
13.20.2 Member Data Documentation	151
13.20.2.1 line	151
13.21Gui::Scene::Mesh Struct Reference	152
13.21.1 Detailed Description	152
13.21.2 Member Data Documentation	152
13.21.2.1 line	152
13.21.2.2 node	152
13.22Species::Specie Class Reference	153
13.22.1 Detailed Description	153
13.22.2 Constructor & Destructor Documentation	153
13.22.2.1 Specie	153
13.22.3 Member Function Documentation	153

13.22.3.1 Cp	153
13.22.3.2 Cp	153
13.22.3.3 Id	154
13.22.3.4 Mass	154
13.22.3.5 Mass	154
13.22.3.6 Size	154
13.22.3.7 Size	154
13.22.4 Member Data Documentation	154
13.22.4.1 cp	154
13.22.4.2 id	154
13.22.4.3 inertia	155
13.22.4.4 mass	155
13.22.4.5 size	155
13.23 State Union Reference	156
13.23.1 Detailed Description	156
13.23.2 Member Data Documentation	156
13.23.2.1 done	156
13.23.2.2 locked	156
13.24 Run::Time Struct Reference	157
13.24.1 Detailed Description	157
13.24.2 Member Data Documentation	157
13.24.2.1 current	157
13.24.2.2 end	157
13.24.2.3 nextstep	158
13.24.2.4 output	158
13.24.2.5 prev	158
13.24.2.6 start	158
13.24.2.7 step	158
13.24.2.8 step0	158
13.25 Gui::WindowGeom Struct Reference	159
13.25.1 Detailed Description	159
13.25.2 Member Data Documentation	159
13.25.2.1 height	159
13.25.2.2 width	159
14 File Documentation	161
14.1 collection.h File Reference	161

14.2	configfile.dox File Reference	162
14.3	container.h File Reference	163
14.4	def.h File Reference	164
14.4.1	Define Documentation	165
14.4.1.1	ABOUT	165
14.4.1.2	ADD	165
14.4.1.3	ADDC	165
14.4.1.4	ADDVEC	165
14.4.1.5	AtomicMassUnit	166
14.4.1.6	AvogadroNumber	166
14.4.1.7	BoltzmannConstant	166
14.4.1.8	COLLISIONTIME	166
14.4.1.9	COPYVEC	166
14.4.1.10	DIM	166
14.4.1.11	DIV	166
14.4.1.12	DOCTYPE	166
14.4.1.13	DOF	167
14.4.1.14	DOFI	167
14.4.1.15	DOFK	167
14.4.1.16	ERROR	167
14.4.1.17	ESC	167
14.4.1.18	FALSE	167
14.4.1.19	GAUSS	167
14.4.1.20	LARGE	167
14.4.1.21	LENGTH	167
14.4.1.22	LOCAL	168
14.4.1.23	MAX	168
14.4.1.24	MAXLINLEN	168
14.4.1.25	MUL	168
14.4.1.26	MULC	168
14.4.1.27	MULVEC	168
14.4.1.28	OMP	168
14.4.1.29	PI	168
14.4.1.30	REAL	168
14.4.1.31	RND	169
14.4.1.32	SCLP	169

14.4.1.33	SMALL	169
14.4.1.34	SQRTPIo8	169
14.4.1.35	SUB	169
14.4.1.36	SUBC	169
14.4.1.37	TINY	169
14.4.1.38	TRUE	169
14.4.1.39	VECP	169
14.4.1.40	VOIDSPECIE	170
14.4.1.41	WAIT	170
14.4.1.42	WORDLENGTH	170
14.4.1.43	ZERO	170
14.4.1.44	ZEROVEC	170
14.4.2	Variable Documentation	170
14.4.2.1	GasConstant	170
14.4.2.2	GasConstantInv	170
14.4.2.3	maxcounters	171
14.4.2.4	OneOverSqrt2	171
14.4.2.5	OneOverSqrt3	171
14.4.2.6	onethird	171
14.4.2.7	sqrt1p25	171
14.4.2.8	sqrt1p5	171
14.4.2.9	Sqrt2	171
14.4.2.10	Sqrt2Over2	171
14.4.2.11	Sqrt3	171
14.4.2.12	Sqrt3Over2	172
14.4.2.13	twothirds	172
14.5	domain.cc File Reference	173
14.5.1	Detailed Description	173
14.6	domain.h File Reference	174
14.6.1	Detailed Description	174
14.6.2	Define Documentation	174
14.6.2.1	XMAX	174
14.6.2.2	XMIN	175
14.6.3	Enumeration Type Documentation	175
14.6.3.1	BoundaryTypes	175
14.7	examples.dox File Reference	176

14.8	faq.dox File Reference	177
14.9	grid.cc File Reference	178
14.10	grid.h File Reference	180
14.11	gui.cc File Reference	181
14.12	gui.dox File Reference	184
14.13	gui.h File Reference	185
14.13.1	Define Documentation	187
14.13.1.1	LEFT_BUTTON	187
14.13.1.2	MAX_WINDOW_HEIGHT	187
14.13.1.3	MAX_WINDOW_WIDTH	187
14.13.1.4	MIDDLE_BUTTON	187
14.13.1.5	RIGHT_BUTTON	188
14.13.1.6	WINDOW_SIZE	188
14.14	implementation.dox File Reference	189
14.15	io.cc File Reference	190
14.16	io.h File Reference	191
14.17	job.cc File Reference	192
14.17.1	Detailed Description	192
14.17.2	Function Documentation	193
14.17.2.1	main	193
14.17.2.2	parsename	193
14.17.2.3	usage	193
14.18	list.h File Reference	194
14.19	main.dox File Reference	195
14.20	model.cc File Reference	196
14.21	model.dox File Reference	197
14.22	model.h File Reference	198
14.22.1	Define Documentation	199
14.22.1.1	HARDBALLS	199
14.22.2	Enumeration Type Documentation	199
14.22.2.1	AtomType	199
14.23	output.dox File Reference	200
14.24	run.cc File Reference	201
14.24.1	Define Documentation	202
14.24.1.1	RAND_MAX	202
14.25	run.h File Reference	203

14.26species.cc File Reference	204
14.27species.h File Reference	205
14.28test.cc File Reference	206
14.28.1 Function Documentation	206
14.28.1.1 main	206
14.29view.cc File Reference	207
14.29.1 Detailed Description	207
14.29.2 Function Documentation	207
14.29.2.1 main	207
14.29.2.2 usage	208

Chapter 1

ReMoDy Source Code Documentation and User's Guide

Version:

1.0

Author:

Andrei Smirnov andrei.v.smirnov@gmail.com

1.1 Contents

- [Introduction](#)
- [Physical Model](#)
- [Installation and Execution](#)
- [Setup](#)
- [Configuration File](#)
- [Graphical User Interface](#)
- [Output Procedures](#)
- [Implementation](#)
- [Examples](#)
- [Frequently Asked Questions](#)
- [Licencing and Support](#)

1.2 Introduction

This is the documentation of the ReMoDy (Reactive Molecular Dynamics) program, which implements the model of reactive molecular dynamics based on the [Collision Theory](#).

1.3 Installation and Execution

1.3.1 Step 1: Extracting the archive

If you have the archive file, like `remody.tbz` (tar-bzipped) or `remody.tgz` (tar-gzipped) then files can be retrieved into a current directory as:

```
tar cvjf remody.tbz
```

or

```
tar cvzf remody.tgz
```

respectively.

1.3.2 Step 2: Compiling

To compile the executable on Linux run `make` from the `remody` root directory, where the makefile was saved. Note that the subdirectories `src/`, `run/`, and `obj/` should be also present.

1.3.3 Step 3: Running

The executables are saved in the `run/` directory. It should also contain the example `remody.xml` and `demo.xml` files, as well as an initial empty input file `empty.dat.gz`. To run the program with an OpenGL window active (slow, good for debugging), one can use the command:

```
./view -f demo.xml empty.dat.gz
```

This will start the program with the initial parameters read from the `demo.xml` file and the initial data read from `empty.dat.gz` file. On startup the program will open the window. One can point at the window with the mouse and press 'f' key to show the frame and 'r' key to run the simulation. Alternatively, one can use 's' key to run iterations step-by-step. Other key functions are described by pressing the '?' key.

To run the program in a batch mode without OpenGL output, one can use this command:

```
./job -f demo.xml empty.dat.gz &
```

This will start the run. To change the parameters of the job, one should modify the input xml file accordingly (`demo.xml`, `remody.xml`, etc.). A link with the name `remody` is also set to one of the executables, like

```
ln -s ./job remody
```

1.4 Setup

Both `view` (OpenGL based) and `job` (batch mode) executables read the configuration xml file. By default the file will have the same name as the executable, for example: `job.xml` or `view.xml`. This can be overwritten with the '-f' option. Two example files `remody.xml` and `demo.xml` are provided with the distribution.

The format of the xml file is in most cases self-explanatory with explanations provided in the "title" fields and comments. The main sections include: `run`, `species`, `domain`, and `gui`.

The `run` section specifies the number of iterations, time steps, output intervals, and the maximum number of molecules to be used.

The `species` section provides the list of species, and reactions between them.

The **domain** section provides the specifications of the boundary conditions and the surface reactions and species.

The **gui** section specifies the parameters used in the OpenGL output window, such as colors, line width, etc.

See also Section [Configuration File](#)

1.5 Licencing and Support

The rest of this documentation provides the description of the structure and functionality of the classes, namespaces, and files of the code.

The code is provided under the [GPL licence](#). Any suggestions/requests can be directed to the developers team at [NIFT](#).

Chapter 2

Configfile

2.1 Configuration File

2.1.1 Invocation

The config file is in XML format. It is loaded at program startup and contains the specification of all the main parameters, grouped in respective sections. An example `syngas.xml` file is provided which includes the setup of syngas reaction on the anode side of a high-temperature solid-oxide fuel cell.

The default name of the config file is `<progrname>.xml`, where `<progrname>` is the name of the executable, such as `view`, `job`, or `remody`. One can override the default name by using `-f` command line option, like:

remody -f syngas.xml

2.1.2 Sections

The sections are enclosed in XML tag pairs, like `<tag>...</tag>`. where the tags are as follows:

- `iterations`: specifies the integer number of iterations the code should run
- `time`: several parameters of physical execution time in seconds, such as `start`=start time, `end`=end time, and `step`=time step. The code will run until either the number of iterations as specified in the `iterations` section or the end time is reached.
- `molecules`: integer number equal to the maximum number of molecules that the program will be able to handle. It will determine the memory allocated at the beginning of execution.
- `species`: this section lists all chemical species and reactions between them. Correspondingly, each specie is described in `specie` section as:

1. `specie`: includes such parameters as `mass` in Atomic units [au], `size` in nanometers [nm], and specific heat, `cp` in kJ/(mol*K). The `specie` tag should also include an attribute `id`, identifying a specie chemical formula, such as CO₂, H₂O, etc.

2. `reaction`: reaction tag has two attributes: `reactants` and `products`. The `reactants` tag should contain two species identifiers. Since only elementary (binary) reactions are considered, there should be exactly two identifiers for reactants. Each of the identifiers should correspond to one specie identifiers listed in the list of species. The `products` attribute should consist of one or two identifiers of reaction products. Also in the reaction section the following parameters are specified:

`activation` = reaction activation temperature in K,

`probability` = probability of reaction outcome as given by the `products` attribute. This parameter should always be 1.0 if there is only one reaction with the given reactants. In case of several reactions with the same reactants, but with different products, this parameter should indicate the probability of this particular branch with given reaction products.

`enthalpy` = enthalpy of reaction in kJ/mol.

- `domain`: The `domain` section consists of the set of parameters describing the geometry and physical properties of the modeled media inside the computational domain, including:
 1. `type`: specifies the geometry. Currently only `box` type is supported.
 2. `grid`: Specifies parameters of the rectangular grid used in segmentation algorithm for accelerating the interaction scheme. In particular, the `cellsize` parameter determines the size of the grid cell. This size should be selected as small as possible but no less than twice the size of the largest specie.

Decreasing the cell size will speed-up code execution in better than linear proportion of the cell-size (but no better than quadratic). At the same time it will increase memory utilization in proportion to it's 3-rd power. It is not recommended to increase the memory utilization above 90%. One can use Unix 'top' utility to check memory utilization.

3. `energy`: Only used for Lennart-Jones type potentials, currently under development.
 4. `bounds`: specifies spatial bounds of the computational domain as: `xmin xmax ymin ymax zmin zmax`.
 5. `bulk`: this section specifies thermodynamic properties and gas composition inside the bulk of the domain. In particular, `temperature` is given in Kelvin [K], and for each `specie`, its `density` is specified in [kg/m³].
 6. `boundary`: each boundary of the domain contains the description of thermodynamic properties and gas composition on the other side of the boundary in the same format as for the bulk of the domain. In addition to these, the boundary tag should have the boundary identifier `id`-attribute, such as "top", "bottom", "right", "left", "front", and "rear". Also additional boundary tag is `type`, which can be one of: "open", "elastic", and "periodic". In the case of open boundary the molecules can freely cross the boundary, in which case they will be removed from the domain. In case of elastic boundary the molecules will bounce from the boundary like from an elastic wall. For periodic boundaries, the molecules crossing the boundary will reappear from the opposite boundary. [Boundary](#) description can also contain the list reactions, between the boundary species given in the same format as in the `species` section.
 - `gui`: the `gui` section describes parameters related to graphical output used when running an OpenGL based version with a visual window output.
1. `translation`: initial translation of the scene.
 2. `vector`: parameters for displaying vectors.
 3. `frame`: parameters for displaying a domain frame.
 4. `mesh`: parameters for displaying mesh. In particular the `node` tag specifies the parameters for displaying particles, or molecules, such as using points or spheres (`type`), etc. Note that using 'sphere' for `type` may significantly slow-down simulation in GUI mode.
 - `xterm`: if set to 1 this parameter will force terminal dumping of certain parameters, like temperature, energy, number of molecules, and species concentrations after every time-step. This can be useful if the time-dependence of concentrations and other parameters needs to be retrieved. In the last case the output can be redirected to a file, which can be processed with the `readlog` utility (see [How do I produce time-curves ...?](#)).

Chapter 3

Examples

3.1 Examples

3.1.1 Syngas Simulation Setup

The `syngas.xml` file can be used as an input config file to ReMoDy to simulate the molecular dynamics of syngas including its interaction with the anode.

- To setup the `syngas.xml` file read `SecConfigfile`.
- To start the program loading the `syngas.xml` file, use the command line as:

```
./remody -f syngas.xml
```

Note, that the name of your executable can be other than `remody`, such as `view` or `job`, in which case change the command line above accordingly.

- To restart from the previous run data, use the command line:

```
./remody -f syngas.xml remody-<number>.dat.gz
```

where `remody-<number>.dat.gz` is the name of the dump file generated by `remody`.

- To retrieve and post process the data, read [FAQs](#).

Chapter 4

Frequently Asked Questions

4.1 FAQs

- **Q:** "When I run remody with a larger domain the program doesn't run and show me this message:"

CAN'T INSERT ELEMENT: Collection OF 99999 IS FULL

A: Increase the number of molecules above 100000 in `<molecules>....</molecules>` section of the config file (syngas.xml) You can estimate the number of molecules you should use from the ideal gas law, that is how many molecules you can have in the domain size you selected and increase that number by 10% to avoid that error condition above.

- **Q:** "How to make the code run faster?"

A: When you run the code check how much memory the program is using. For example, you can do it running 'top' from a terminal. If it is using less than 80%, then decrease the parameter inside `<grid><cellsize>...</cellsize></grid>` tag. This will make code run a bit faster but it will also increase the memory utilization. It may not even start if you decrease it too much. If the memory utilization is still less than 80%, you can keep decreasing that parameter. Do not decrease it below twice the size of the biggestst molecule in your composition.

- **Q:** "How do I restart the run?"

A: When the code is running it generates output files at regular time interval (physical time, that is) which is set inside `<time><output>...</output></time>` tag. The output is dumped to the zipped-files named like `remody-<timestamp>.dat.gz`. To restart the run from the output file you should start the run as:

```
./remody -f configfile.xml remody-<timestamp>.dat.gz
```

- **Q:** "How do I produce time-curves of various parameters?"

A: You can produce time curves for such parameters as temperature (T), number of molecules in the domain (N), average energy per molecule (E), and various specie concentrations. For T,N, and E follow the procedure as described below, using the syntax with an equal sign (=), for species do not use equal sign as described in the following two questions.

- **Q:** "How do I construct the temperature curve?"

A: **Run** the readlog.py utility with the remody terminal output log file as input. For example, when you start the batch job, use the command like this:

```
./job -f config.xml > job.log &
```

This will dump temperature, concentrations, and other parameters into the job.log file every iteration. Then you can run the readlog.py utility to produce the temperature curve file from the log file, like:

```
python readlog.py T= < job.log > temp.dat
```

where the temp.dat file contains two columns of temperature versus time, which be displayed in gnuplot with a command like:

```
plot 'temp.dat' w l
```

- **Q:** "" **A:**

- **Q:** "How do I construct specie concentration time curves?"

A: Do similarly as for the temperature curve above, except when running the readlog specify the specie chemical formula as provided in the input xml file, and do not use the equal sign (=) after the specie name as in the case above, like this:

```
python readlog.py CO2 < job.log > co2.dat
```

- **Q:** "How can I retrieve the results of simulation?"

A: Here is how you can produce the results from the simulation run. First check if the run is going with top:

top

This will show the table like this:

```
top - 04:29:40 up 42 days, 3:41, 2 users, load average: 1.03,
1.10, 1.20
Tasks: 126 total, 3 running, 123 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.2us, 0.1sy, 0.0ni, 74.7id, 0.0wa, 0.0hi, 0.0si, 0.0st
Mem: 4031520k total, 3878180k used, 153340k free, 336828k buffers
Swap: 10241428k total, 1454736k used, 8786692k free, 552436k
cached
PID USER PR NI VIRT RES SHR S CPU MEM TIME+ COMMAND
29283 rolando 25 0 3353m 2.0g 1336 R 100 52.8 9843:09 job
1 root 15 0 10308 596 544 S 0 0.0 0:04.77 init
...
```

There you can see that job is running. Quit it by pressing 'q'-key. Then check how far the run has gone:

tail -f syngas2.log

This will show the something like:

```
Time=6.103e-01 Step=2.439e-06 Memory=41% N=8279309 T=1039.63K
E=1.188e-13J
H H2 OH H2O CH3 CH4
5.45126 31.48455 0.00104 18.91287 0.00041 13.87211
CO2 CO N2
8.70242 10.84918 10.72622
```

Note that syngas2.log file above was used because the job started with the command:

./job -f syngas2.xml job-4.dat.gz 1> syngas2.log 2> syngas2.err &

If you used a different log file you should use that one above. From the table you can see all the species currently in the simulation. Now you can quit tail -f with pressing ^C (Cntrl.C).

To produce the time sequence do this:

1. Run this command:

```
python readlog.py H2 < syngas2.log > H2-time.dat
```

Note that readlog.py program is in your remedy/util directory. This should create the H2-time.dat file.

2. Check the file with gnuplot:

```
gnuplot> plot 'H2-time.dat' w l
```

And then you do the same for H2O, CO2, etc.

- **Q:** "How to produce a spatial distribution along one axis?"

A:

1. Check that you have the hist program in run directory: `ls hist` if you dont, compile it if necessary in your remody/util directory by running `make` and link it or copy into your run directory, like `ln -s ../util/hist .`
2. Select the time step for which you want the distribution. For example, for 5 nanoseconds, you should use `job-5.dat.gz` file.
3. Run the command:

```
gunzip job-5.dat.gz -c | grep H2 | grep -v H2O | cut -f 3 | ./hist > H2-5.dat
```

This will create the H2-5.dat file, which you can see with `gnuplot` as:

```
plot 'H2-5.dat' w l
```

Note, that the line above has '`grep -v H2O`' part. This is needed to preclude all H2O molecules from the H2 list. Similarly, when you produce CO distribution, you should use:

```
gunzip job-5.dat.gz -c | grep CO | grep -v CO2 | cut -f 3 | ./hist > CO-5.dat
```

But to produce H2O or CO2 distributions, you should use:

```
gunzip job-5.dat.gz -c | grep H2O | cut -f 3 | ./hist > H2O-5.dat gunzip job-5.dat.gz -c | grep CO2 | cut -f 3 | ./hist > CO2-5.dat
```

Then you can produce plot of all of them like:

```
plot 'H2-5.dat' w l, 'H2O-5.dat' w l, 'CO-5.dat' w l, 'CO2-5.dat' w l
```

etc. The same for other nanoseconds. You may need to set the axes labels, like

```
set xlabel 'TIME [ns]' set ylabel 'NUMBER OF MOLECULES'
```

To save the plot into a png-file, use:

```
set term png set output 'concentrations-5ns.png'
```

and then run the above plot command again. This will create the `concentrations-5ns.png` file, which you can include into a presentation or a paper. But don't forget to get the output back to the screen with:

```
set output set term x11
```

before you try other plot commands, or everything will be written to that png file.

Of course if you are proficient with Python, you can use the `matplotlib` module to plot and analyze the results.

Chapter 5

GUI

5.1 Graphical User Interface

ReMoDy can be compiled in two versions: with and without a graphical user interface (GUI). The former is used to visualize the simulation progress on a small set of data, while the latter is to perform larger-scale simulations where the concurrent visualization is not possible.

After the GUI-enabled version (*view*) is started, a display window pops up, showing the computational domain. The initial parameters of the scene can be set in the config file (see Sec. [Configuration File](#)). The configfile can be reloaded by making the display window active and hitting the 'l' key for 'load'. The keyboard control of the simulation has two modes: *keystroke*, and *commandline*. This is similar to vi text editor in UNIX. Initially the keystroke mode is active.

1. In the keystroke model the single-key-strokes can be used while the display window is active. These include:

- **ESC**: exit. The same as 'q';
- **?**: show help.
- **::**: switch to a terminal command mode (similar to vi). The same as '.';
- **a**: show/hide the axes.
- **b**: show/hide boundary faces of the mesh (not used when mesh is absent).
- **c**: switch the color-scheme.
- **f**: show the box frame.
- **G**: show all the mesh edges.
- **g**: show only the boundary mesh edges.
- **I**: read configuration file.
- **m**: display the menu on the console.
- **N**: show mesh nodes.
- **n**: show boundary vertexes.
- **r**: start/stop the continuous run.
- **s**: perform one step of the simulation. Analogous to '+'.
- **v**: show boundary vectors.
- **w**: dump current window into a xwd file.
- **Z**: zoom-in view.
- **z**: zoom-out view.

1. The command mode is switched from the keystroke mode by hitting the ':' or '.' keys. To switch back from the command mode one has to enter an empty command, that is, press ENTER, while the command line is empty. In the command mode, the command line appears in the terminal windows from which the program has started. The commands one can enter in the command line include:

- **bg**: go to background run mode. During this mode the all controls are disabled.

- **cs**: switch the color scheme.
- **fg**: go to foreground mode.
- **?, h, or help**: show help.
- **r**: run a number of iterations.
- **sp**: toggle between spheres and points in particle (molecule) display.
- **wd**: start/stop window dump every fixed number of iterations.
- **dw**: dump current window picture into a xwd file.
- **e**: exit the program. Also saves current data.
- **q**: quit without saving.
- **.**: quit the command mode. Also hitting ENTER with empty command line.

Chapter 6

Implementation

6.1 Implementation

6.1.1 Collision Detection Scheme

The discretization scheme is based on the combination of the linked cell method (Sec.[Linked-Cell Method](#)) with a space-time collision detection scheme (Sec.[Space-Time Collision Detection Scheme](#)), which enable to significantly accelerate the execution of the method. To better understand the principle of the method, it has to be compared with the standard time-advancement scheme.

6.1.1.1 Standard Time-Advancement Scheme

In a standard scheme all molecules are advanced using a single global time step. This time step is selected as the minimum time for a molecule to move at the distance of its radius:

$$dt = \frac{d}{v}$$

where 'd' is on the order of molecule radius and 'v' is the velocity of a molecule. The minimum is sought for all molecules. After each time step the relative positions of all molecules are analyzed to determine if there are any overlaps (collisions) between pairs of molecules.

This choice of time-step, dt, above guarantees that no molecule will miss a collision event. However, since each molecule usually travels much longer distances than its length between the collisions, this scheme leads to unnecessary many steps calculations for an essentially straight trajectory path between the collisions as illustrated in the figure below.

Also, the time step is selected with respect to the smallest molecule, which usually also have the highest velocity. Thus, in the case of a gas mixture, this time-step will be excessively small for bigger and slower moving molecules.

Considering this, and the fact that the molecules travel on straight paths between the collisions, the standard scheme is very inefficient, since it does the majority of computations for molecules undergoing no collisions, but rather traveling on straight paths.

6.1.1.2 Space-Time Collision Detection Scheme

In the new collision detection scheme instead of moving all molecules with the same time step, attempts to move each molecule directly to the site of next collision, avoiding any calculations on the straight path, as shown in the figure.

using a single global time step for all molecules, each molecule has it's own time, and its own time-step. Thus, instead of a 3D position vector and velocity vector, the kinematic parameters for each molecule include a 4D space-time vector (x,y,z,t), velocity vector, (vx,vy,vz) and the time step, dt. The time-step for each molecule is calculated as the time to the next collision. Thus, one can estimate the time of the next collision event, t_{coll} , for each molecule by adding its current time to its current time step:

$$t_{coll} = t + dt$$

The time-steps of molecules are constantly updated within the loop, where all pairs of molecules are analyzed for possible collisions on the basis of their velocities and radii. If the molecules are found to be heading for a collision, then the time of that collision event is compared to the currently estimated next collision time for each molecule. The time-steps for the molecules are updated if the new collision event is found to occur earlier than both estimated collision events for each molecule. This results in selection of the earliest possible collision events. The collision events also include the collisions with the walls.

In the same loop where the molecular time-steps are updated using collision-detection scheme, the actual collision events are processed for those molecules which are found to be within the interaction distance from each other. In the course of collision molecules change their velocities in compliance to momentum and energy conservation, as well as exchange their kinetic and internal energies according to the equipartition principle, that is, the total energy is equally distributed among the combined degrees of freedom of two molecules.

After the time-steps of all molecules are updated and collisions processed in the collision-detection loop, the molecules enter the time-advance loop, where each molecule is advanced by its time-step to its nearest collision event. These two loops are iterated over and over until the termination time for the simulation.

It should be noted, that the described scheme will not work well, if the times of all molecules start deviate from each other by too large a value, causing some molecules to go too far ahead in time compared with other molecules. Thus, it is necessary to periodically synchronize the molecules, by bringing them all to the same time level. This is done by introducing the global time step, such that all the molecules exceeding the next global time level will not move to the next collisions until all the molecules reach that time level. Then the global time level is incremented by the global time-step and the procedure is repeated. The value of the global time step is selected on the order of molecular mean-free-path:

$$dt = \frac{D}{v}$$

where 'D' is on the order of several mean free-path lengths (or inter-molecular distances). This choice guarantees that the molecular collision times will not go out of sync by too much to cause miscalculations in collision events.

Even though the current scheme is still relying on a global time-step, this time step is by far larger than that used in the standard time-stepping scheme discussed above. This is because in the standard scheme the global time-step is based on molecular size, and in the current scheme it is based on inter-molecular distance, which is far larger than the size of the molecule: $D \gg d$. This leads to significant speed-up of calculations compared to the standard time-advance scheme.

6.1.2 Linked-Cell Method

ReMoDy uses interaction acceleration scheme based on linked-cell technique, which is a variant of [Verlet list method](#) [Verlet L (1967). *Phys. Rev.* **159**: 98-103]. In this method the whole computational domain is divided into box-shaped cells. Only interactions between the molecules from the same or adjacent cells are considered (see Figure).

The size of the grid cells is selected by optimizing the execution speed, and is usually on the order of several mean-free-paths. The method enables to achieve near linear dependence of execution time on the number of molecules. In contrast, looping over all molecules makes execution time proportional to the square of the number of molecules, which makes that impractical for large number of molecules.

6.1.3 Multi-Processor Implementation

The code can be made to run in parallel on multi-core workstations. It uses the shared-memory OpenMP library to distribute processing of the molecules time-advancement and interaction loops among the available CPUs.

The two time-critical loops which run in parallel are time-advancement loop, implemented in subroutine [Domain::step\(\)](#), and interaction loop implemented in [Domain::interaction\(\)](#) routine. Before the loops are entered the molecules are indexes sequentially. Each OpenMP thread selects the molecules which has index satisfying the criterion:

$$\text{mod}(i_{mol}, N_{threads}) = i_{thread}$$

where $\text{mod}(*,*)$ is division by modulus. This way all molecules are distributed equally between active threads. The number of threads is selected as the maximum between the available processors, and the pre-defined constant 'nthreads'.

In addition to this, in the interaction loop, the locking mechanism is used to prevent simultaneous processing of the same molecules by more than one processor. This can happen because the interaction procedure considers all pairs of molecules, which is done in a double-looping over all molecules. Thus, the selection of molecules from the primary loop, using the mechanism above will not prevent the possibility of simultaneous processing of the same molecule from the secondary loop by more than one processor. The locking mechanism operates by introducing the 'is-locked' flag for each molecule. Inside the nested loop the molecule is considered only when its state is not locked. If this is the case, then the molecule is temporarily locked while its interaction with the primary molecule from the main loop is being processed.

6.1.4 Dynamic Lists

The main data-structures holding molecules are dynamic lists: **Collection** and **Container**. *Collection* is a double-linked list of fixed number of items, with two pointers for each item, pointing to the next and the previous item. The list consist of two parts: active and dead. Each part forms a loop, such that following next or previous pointer from any element across the list will lead back to the same element. The procedure of moving elements between the active and dead parts is very simple, involving only several pointer reassignment operations. The figure below illustrates the operation of removing the active molecule from the computational space and assigning it to the pool of "dead" molecules.

The reverse procedure of "resurrecting" dead molecules and introducing them into the active pool is done in the same manner. This technique enables to avoid expensive memory allocation/deallocation operations, and save time on looping over the list of molecules, since all dead molecules are not in the active list, and are completely ignored by the looping procedure. Thus, no conditional if-statements are necessary, and shorter list sizes can be used.

The *Collection* class is convenient for the dynamic storage of a single collection of items, and is used to store all the molecules in the domain.

Container is a variable size list of pointers to items. Like a *Collection* it is also a double-lined list with the next and previous pointers for each item. But unlike a *Collection*, the *Container* all container lists share the same pool of dead items, or pointers, which can point to any item. Each container list can acquire items from the pool or return them back to the pool.

The *Container* list is convenient to store multiple lists of items where the number of items constantly change. *Container* lists are used in the implementation of the linked-cell method described below.

Chapter 7

Model

7.1 Physical Model

The physical model of ReMoDy is uses the `Collision_theory` to describe elementary chemical reactions, and `Kinetic theory` to describe molecular gas dynamics. Reactions occur probabilistically during molecular collisions, with the probabilities determined by the activation energies and reaction probabilities, if more than one outcome for the reaction exists.

The thermodynamic properties of the gas species include `heat capacities` which are used to distribute thermal energy among molecular degrees of freedom during the collisions.

7.1.1 Chemical Reactions

The interaction of the two molecules is modeled through the binary collision approximation whereby only two molecules can interact at a time. The interaction between the molecules can be of two kinds: (1) simple mechanical collision, (2) collision with subsequent chemical reaction. The fact of the collision is detected when the distance between the centers of two molecules becomes less than the sum of their radii, and their relative velocities are directed toward each-other.

Collision is modeled in the center-of-mass (CM) frame of reference. The velocities of the two molecules are first recalculated into the CM frame. The total energy of the two molecules is first calculated as their combined internal energy plus their combined kinetic energy in the CM coordinate frame. Chemical reactions are triggered when this energy is in excess of the activation energy for the reaction. In this case the enthalpy of the reaction is added to the total energy. This energy is then redistributed between the degrees of freedom of the product molecules according to the following scheme:

$$\begin{aligned} energy &= KE + IE + enthalpy \\ dof_A &= DOF(Cp_A) \\ dof_B &= DOF(Cp_B) \end{aligned}$$

where dof_X are total degrees of freedom (kinetic + internal) for molecule $X=(A,B)$, and Cp_X is the specific heat of molecule X . The function `DOF(Cp)` of computing the degrees of freedom from the specific heat is defined as:

$$DOF = 3 + (2Cv/R - 3)/2 = 3 + Cv/R - 1.5 = Cv/R + 1.5 = Cp/R + 0.5$$

where $Cv = Cp - R$

The combined kinetic degrees of freedom are 6, then the combined internal degrees of freedom are calculated as:

$$dofi = dof - 6$$

Then for each molecule the ratio of its internal degrees of freedom to the total internal degrees of freedom is computed as:

$$\begin{aligned} rdofi_A &= (dof_A - 3)/dofi \\ rdofi_B &= 1 - rdofi_A \end{aligned}$$

The calculations of energy redistribution between colliding molecules is done in the center-of-mass system (CM). The number of kinetic (translational) degrees of freedom of two colliding molecules in CM system

$(dofk_{CM})$ will be less than that in the laboratory system, since the CM system already has 3 translational degrees of freedom associated with its center of mass. Thus, the total number of kinetic degrees of freedom in CM system will be:

$$dofk_{CM} = 6 - 3 = 3$$

And the total number of dof in the CM system will be reduced accordingly: $dof_{CM} = dof - 3$. The number of internal degrees of freedom in CM system remain the same as in the laboratory system: $dofi_{CM} = dofi$.

The procedure for calculating new velocities and internal energies in CM system during the collision of two molecules ('a' and 'b') is as follows:

- The old kinetic energy of the two molecules are computed as:

$$ke_{old} = \frac{ua_{old}^2}{2} + \frac{ub_{old}^2}{2}$$

- It is combined with the old internal energy to form the new total energy in CM system as:

$$e_{old} = ke_{old} + iea + ieb$$

where iea , ieb are the internal energies of molecules a and b respectively.

- The new combined energy (e) of the two molecules is calculated by adding the enthalpy of reaction (h) to the old total energy:

$$e = e_{old} + h$$

- Kinetic energy in the CM system is computed as the share of the new total energy distributed according to degrees of freedom:

$$ke = \frac{dofk_{CM}}{dof_{CM}} e$$

$$ie = energy - ke$$

- The new velocities of the molecules in the CM frame are then updated from the old ones. The old velocities were computed according the elastic collision scheme between two hard-balls. So, if the energy was released or absorbed during the reaction, and some of it was absorbed into the internal degrees of freedom, the old velocities should be recalculated. The kinetic-energy is redistributed equally between the two molecules:

$$\frac{ma ua^2}{2} = \frac{mb ub^2}{2} = \frac{ke}{2}$$

which gives:

$$ua = \sqrt{ke/ma}$$

$$ub = \sqrt{ke/mb}$$

This is used to build the ratio of new to old velocities (see `interact(...)` function in [domain.cc](#)):

$$uratio_A = ua/ua_{old}$$

$$uratio_B = ub/ub_{old}$$

which are used to update the velocity vector for each molecule as:

$$u_X(i) = u_X(i) * uratio_X$$

where $i=(x,y,z)$ is the Cartesian direction of velocity vector. The remaining internal energy, ie , is distributed among the internal degrees of freedom of each molecule X as:

$$InternalEnergy_X = rdofi_X ie$$

It should be noted, that in the above calculations, the value of kinetic energy was that computed in CM frame of reference. This means that the actual kinetic energy of the molecule will have a somewhat higher value than what would be expected from an equal distribution of energy among the internal and external degrees of freedom. Nevertheless, this scheme is considered accurate, since the redistribution of energy is indeed taking place in the center-of-mass reference frame, and this will inevitably lead to a higher contribution of energy to kinetic degrees of freedom.

7.1.2 Cross-Boundary Species

The code provides the possibility of specifying cross-boundary gases, which can enter the computational domain from the other side of the open boundary (the boundary with the `type="open"` inside the `<boundary>` tag of the XML input file (see [Sec. Configuration File](#)).

The algorithm uses the density and temperature to calculate the frequency of injection of molecules of species, s at the boundary. The injection frequency, f_s per unit area, A is computed as

$$f_s = \frac{N}{\Delta t}$$

where Δt_s is the time interval at which a molecule hits the boundary area A , and N is the number of molecules in a volume with the base A and length Δx as shown in the figure: The time interval, Δt , between the collisions can be related to the component of velocity of the molecule of species s , in direction x , v_{sx} , as follows:

$$\Delta t = \frac{2\Delta x}{v_{sx}}$$

The number of molecules, N can be related to density, ρ , as:

$$N = \frac{\rho}{\mu} \Delta x A$$

where μ is the mass of one molecule and $A = \Delta y \Delta z$. Thus, the frequency is:

$$f_s = \frac{\rho}{\mu} \Delta x A \frac{v_{sx}}{2\Delta x} = \frac{\rho v_{sx}}{2\mu} A$$

Chapter 8

Output

8.1 Output Procedures

There are three data output methods that can be used:

- Compressed snapshots of coordinates, velocities, and internal energies stored in gzipped ASCII format files. The files are dumped at time intervals specified in the input XML file under tag: `<time><output>...</output></time>` with the file names in format: `<task>-<step>.dat.gz`, where `<task>` is the name of the task, which is the same as the name of the executable file (job, view, or any other), and the `<step>` is the sequence number of the dump, which is automatically incremented for each subsequent dump. The output is performed by `save()` function of [Domain](#) class implemented in `domain.cc` file.
- Terminal output of time sequence of average quantities, such as the number of molecules (N), average temperature (T), kinetic (K), internal (I), and total energy (E) per molecule and per degree of freedom. In addition to that is also outputs the physical time (Time) in nanoseconds and memory utilization (Memory). The terminal output can be toggled with `<xterm>0,1</xterm>` flag in XML file. The output can be captured into a file using redirect command, like:

```
./job > log.log &
```

and post-processed using the Python `readlog.py` utility (see [Frequently Asked Questions](#)). The terminal output is performed inside the `Domain::run()` function at every designated time interval.

- The last method is to dump the graphics window in X-Window dump format (XWD) and then convert into one of the common graphics formats, such as png or jpg using standard convert utilities, such as ImageMagic's `convert`. See also Sec. [Graphical User Interface](#).

Chapter 9

Namespace Index

9.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Geom (Provides some trigonometry functions)	35
GridContainer (Domain Segmentation for Interaction Acceleration)	37
Gui (OpenGL window output routines)	40
IO (Some IO routines)	59
Palette (Defines color palette)	62
Potential (Interaction-potential functions, such as LJ)	64
Run (Processing command-line and execution control parameters)	67
Species (Description of Specie and Reaction classes)	71

Chapter 10

Class Index

10.1 Class List

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

Boundary (Domain Boundary Definition)	73
Bulk (Domain Bulk Properties)	78
Collection< Element >	81
Gui::ColorScale	92
Container< Element >	93
Domain (Computational Domain)	108
Gui::ElementDisp	117
Species::Gas (Gas is a specie with some additional parameters)	118
Item< Element >	120
List< Element >	121
Molecule	130
Option (Execution options)	135
Pointer< Element >	137
Pool< Element >	138
Ptr< Element >	141
Species::Reaction (Reaction determines the two products only)	142
Species::Reaction::Outcome (Possible outcomes of a reaction)	145
Gui::Scene	149
Gui::Scene::Color	150
Gui::Scene::Frame	151
Gui::Scene::Mesh	152
Species::Specie	153
State	156
Run::Time (Execution time-control structure)	157
Gui::WindowGeom	159

Chapter 11

File Index

11.1 File List

Here is a list of all files with brief descriptions:

collection.h	161
container.h	163
def.h	164
domain.cc (Implementation of Domain and Boundary classes)	173
domain.h	174
grid.cc	178
grid.h	180
gui.cc	181
gui.h	185
io.cc	190
io.h	191
job.cc (The ReMoDy backend)	192
list.h	194
model.cc	196
model.h	198
run.cc	201
run.h	203
species.cc	204
species.h	205
test.cc	206
view.cc (The frontend of ReMoDy with the OpenGL Visualizer)	207

Chapter 12

Namespace Documentation

12.1 Geom Namespace Reference

Provides some trigeometry functions.

Functions

- void [distvec](#) (REAL a[], REAL b[], REAL c[])
- REAL [distance](#) (REAL a[], REAL b[])
- REAL [distance](#) (int dim, REAL a[], REAL b[])
- REAL [sclp](#) (REAL a[], REAL b[])
- REAL [length](#) (REAL a[])
- REAL [area](#) (REAL a[], REAL b[])
- REAL [normalize](#) (REAL a[])
- int [hash](#) (int i, int j, int n)

12.1.1 Detailed Description

Provides some trigeometry functions.

12.1.2 Function Documentation

12.1.2.1 REAL Geom::area (REAL a[], REAL b[])

Definition at line 148 of file model.cc.

References DIM, LENGTH, REAL, and VECP.

Referenced by Domain::Domain(), and Domain::injection().

12.1.2.2 REAL Geom::distance (int dim, REAL a[], REAL b[])

Definition at line 124 of file model.cc.

References REAL.

12.1.2.3 REAL `Geom::distance` (REAL $a[]$, REAL $b[]$)

Definition at line 116 of file `model.cc`.

References `DIM`, and `REAL`.

12.1.2.4 void `Geom::distvec` (REAL $a[]$, REAL $b[]$, REAL $c[]$)

Definition at line 113 of file `model.cc`.

References `DIM`.

12.1.2.5 int `Geom::hash` (int i , int j , int n)

Definition at line 137 of file `model.cc`.

12.1.2.6 REAL `Geom::length` (REAL $a[]$)

Definition at line 140 of file `model.cc`.

References `DIM`, and `REAL`.

Referenced by `normalize()`.

12.1.2.7 REAL `Geom::normalize` (REAL $a[]$)

Definition at line 108 of file `model.cc`.

References `DIM`, `length()`, and `REAL`.

12.1.2.8 REAL `Geom::sclp` (REAL $a[]$, REAL $b[]$)

Definition at line 132 of file `model.cc`.

References `DIM`, and `REAL`.

12.2 GridContainer Namespace Reference

[Domain](#) Segmentation for Interaction Acceleration.

Functions

- void `init` (REAL ymin[], REAL ymax[], REAL radius, [Pool](#)< [Molecule](#) > *newpool)
- int `index` (int ind[])
- void `index` (int n, int ind[])
- void `put` ([Molecule](#) *node)
- void `put` ([Molecule](#) *node, int icounter)
- [Container](#)< [Molecule](#) > * `get` (int ind[])
- int * `dimensions` ()
- bool `checkPool` (int icell, char *msg)

Variables

- REAL `xmin` [DIM]
- REAL `xmax` [DIM]
- REAL `cellsize`
- int `ncells` [DIM+1]
- int `mcells` = 0
- int `mcells1` = 0
- [Container](#)< [Molecule](#) > * `nodes`
- [Pool](#)< [Molecule](#) > * `pool`

12.2.1 Detailed Description

[Domain](#) Segmentation for Interaction Acceleration.

Splits space into cubic cells to consider local interactions between adjacent cells only

12.2.2 Function Documentation

12.2.2.1 bool GridContainer::checkPool (int icell, char * msg)

12.2.2.2 int * GridContainer::dimensions ()

Definition at line 181 of file grid.cc.

References `ncells`.

12.2.2.3 [Container](#)< [Molecule](#) > * GridContainer::get (int ind[])

Definition at line 178 of file grid.cc.

References `index()`, and `nodes`.

12.2.2.4 void GridContainer::index (int *n*, int *ind*[])

Definition at line 68 of file grid.cc.

References DIM, mcells1, and ncells.

Referenced by Domain::interaction().

12.2.2.5 int GridContainer::index (int *ind*[])

Definition at line 61 of file grid.cc.

References DIM, and ncells.

Referenced by get(), and put().

12.2.2.6 void GridContainer::init (REAL *ymin*[], REAL *ymax*[], REAL *radius*, Pool< Molecule > * *newpool*)

Definition at line 25 of file grid.cc.

References cellsize, DIM, mcells, mcells1, ncells, nodes, Run::option, pool, REAL, xmax, and xmin.

12.2.2.7 void GridContainer::put (Molecule * *node*, int *icounter*)

DDD

DDD

Definition at line 128 of file grid.cc.

References cellsize, Molecule::Coordinates(), DIM, index(), ncells, nodes, REAL, Molecule::Type(), VOIDSPECIE, and xmin.

Referenced by Domain::Domain(), Boundary::Inject(), and Domain::run().

12.2.2.8 void GridContainer::put (Molecule * *node*)

Definition at line 81 of file grid.cc.

References cellsize, Molecule::Coordinates(), DIM, index(), ncells, nodes, REAL, Species::species, Run::time, Molecule::Type(), Molecule::Velocity(), VOIDSPECIE, and xmin.

12.2.3 Variable Documentation

12.2.3.1 REAL GridContainer::cellsize

Definition at line 20 of file grid.cc.

Referenced by Domain::Domain(), init(), put(), and Domain::run().

12.2.3.2 int GridContainer::mcells = 0

Definition at line 22 of file grid.cc.

Referenced by init().

12.2.3.3 int GridContainer::mcells1 = 0

Definition at line 22 of file grid.cc.

Referenced by index(), and init().

12.2.3.4 int GridContainer::ncells

Definition at line 21 of file grid.cc.

Referenced by dimensions(), index(), init(), Domain::interaction(), and put().

12.2.3.5 Container< Molecule > * GridContainer::nodes

Definition at line 23 of file grid.cc.

Referenced by get(), init(), put(), and Domain::run().

12.2.3.6 Pool< Molecule > * GridContainer::pool

Definition at line 24 of file grid.cc.

Referenced by Container< Element >::append0(), init(), Container< Element >::insert0(), and Container< Element >::remove0().

12.2.3.7 REAL GridContainer::xmax[DIM]

Definition at line 19 of file grid.cc.

Referenced by init().

12.2.3.8 REAL GridContainer::xmin[DIM]

Definition at line 19 of file grid.cc.

Referenced by init(), and put().

12.3 Gui Namespace Reference

OpenGL window output routines.

Classes

- struct [ElementDisp](#)
- struct [ColorScale](#)
- struct [Scene](#)
- struct [WindowGeom](#)

Enumerations

- enum [Elements](#) {
[points](#) = 0, [nodes](#), [edges](#), [faces](#),
[cells](#), [frame](#), [mesh](#), [boundary_nodes](#),
[boundary_edges](#), [boundary_faces](#), [boundary_cells](#), [maxelements](#) }
- enum [Color](#) {
[red](#) = 0, [green](#), [blue](#), [skyblue](#),
[brown](#), [magenta](#), [yellow](#), [color6](#),
[color7](#), [color8](#), [color9](#), [color10](#),
[color11](#), [color12](#), [color13](#), [color14](#),
[color15](#), [maxcolor](#) }
- enum [ColorSchemes](#) {
[colorByType](#) = 0, [colorByBoundary](#), [colorByTime](#), [colorByMass](#),
[maxColorSchemes](#) }
- enum [PointParameters](#) {
[variable](#) = [maxcolor](#), [vmin](#), [vmax](#), [size](#),
[sizevar](#), [massvar](#), [maxpointprm](#) }
- enum [LineParameters](#) { [thickness](#) = [maxcolor](#), [length](#), [maxlineprm](#) }
- enum [AxesParameters](#) {
[axeswidth](#) = 0, [xaxislength](#), [yaxislength](#), [zaxislength](#),
[arrowheight](#), [arrowwidth](#), [maxaxesprm](#) }
- enum [SurfDispType](#) { [gridlines](#) = 0, [solidsurface](#), [maxsurfdisptypes](#) }
- enum [Movement](#) { [stay](#) = 0, [rotate](#), [moveuv](#), [movew](#) }
- enum [ShowPars](#) {
[showRun](#) = 0, [showAxes](#), [showSpheres](#), [showBonds](#),
[showGrid](#), [showNodes](#), [showVariables](#), [showBoundaryVertexes](#),
[showBoundaryVectors](#), [showToolVertexes](#), [showBoundaryFaceCenters](#), [showBoundaryFaces](#),
[showBoundaryGrid](#), [showToolGrid](#), [showFrame](#), [showCellCenters](#),
[showFaceCenters](#), [showIsoSurfaces](#), [dumpWindow](#), [maxshowpars](#) }

Functions

- int [query_extension](#) (char *extName)
- void [init](#) (int argc, char *argv[], [Domain](#) *newdomain)
- void [helpDisplay](#) ()
- int [readparam](#) (char *s, char *param[], int maxparam, char *filename, int val[])
- int [readparam](#) (char *s, char *param[], int maxparam, char *filename, REAL val[])
- void [readconf](#) ()
- void [refresh](#) ()
- void [initdisp](#) ()
- void [Materials](#) (int argc, char *argv[])
- void [showVector](#) (double *x, double *v)
- void [showParticle](#) (double vmn, double vmx, double val, double *x)
- void [showSphere](#) (double vmn, double vmx, double val, double rad, double *x)
- void [displayAxes](#) ()
- void [getScaling](#) (double &l0, double &l1)
- void [getXLimits](#) (double *x0, double *x1)
- void [Exit](#) ()
- void [Quit](#) ()
- void [menu](#) (int value)
- void [displayMenu](#) ()
- void [consoleMenu](#) ()
- void [setBackgroundRun](#) ()
- void [setForegroundRun](#) ()
- void [toggleSpheres](#) ()
- void [switchColorScheme](#) ()
- void [runmany](#) ()
- void [toggleWindowDump](#) ()
- void [dumpwindow](#) ()
- void [commandMode](#) ()
- void [display](#) ()
- void [animate](#) ()
- void [helpCommand](#) ()
- void [reshape](#) (int w, int h)
- void [mouse](#) (int button, int state, int x, int y)
- void [motion](#) (int x, int y)
- void [keyboard](#) (unsigned int key)
- void [run](#) ()
- void [displaymessage](#) (char *msg)
- void [setElementColor](#) (int element_type, REAL [rgbcolor](#)[])
- void [getElementColor](#) (int element_type, REAL [rgbcolor](#)[])
- void [writeData](#) ()
- void [printGridXLimits](#) ()
- void [printGridVecLimits](#) ()
- void [getXLimits](#) (REAL *xmin, REAL *xmax)
- void [getScaling](#) (REAL &lmin, REAL &lmax)
- void [printPartVelLimits](#) ()
- void [printParticleVariables](#) ()
- void [showGridElements](#) (int ne, double *X, REAL color[])
- void [showVectorComponent](#) (int ivar, int icomp, double *X, double vmn, double vmx)

- void [showVector](#) (int ivar, double *X)
- void [printVariables](#) (int n)
- void [finish](#) ()
- void [InitMaterials](#) (void)
- void [selectVariable](#) ()
- void [keyboard](#) (unsigned char key, int x, int y)
- void [drawSegment](#) (REAL *x, REAL *y)

Variables

- [Domain](#) * domain
- [ElementDisp](#) disp [maxelements]
- char * [configfile](#) = (char*)"gui.cfg"
- char [windowname](#) [MAXLINLEN]
- int [showpar](#) [maxshowpars]
- int [finished](#)
- int [animation](#)
- int [nwdump](#)
- int [iwdump](#)
- int [attributeList](#) []
- REAL [step](#)
- REAL [vecval](#) [maxlineprm]
- REAL [axes](#) [maxaxesprm]
- REAL [rgbcolor](#) [maxcolor][3]
- REAL [wdtime](#)
- REAL [xo](#) [3]
- REAL [dx](#)
- REAL [dy](#)
- REAL [dz](#)
- REAL [lastx](#)
- REAL [lasty](#)
- REAL [lastz](#)
- int [mouseButtons](#) [3]
- REAL [zoom](#)
- REAL [rotx](#)
- REAL [roty](#)
- REAL [tx](#)
- REAL [ty](#)
- REAL [xmin](#) [DIM]
- REAL [xmax](#) [DIM]
- REAL [lx](#)
- REAL [ly](#)
- REAL [lz](#)
- REAL [lmin](#)
- REAL [lmax](#)
- enum [Movement](#) movement
- struct [ColorScale](#) colorscale
- struct [Scene](#) scene
- struct [WindowGeom](#) window
- struct BFaceList * [bface_root](#)

- Display * [dpy](#)
- Window [win](#)
- int [firstR](#) = 1
- int * [vars](#)
- int * [coms](#)

12.3.1 Detailed Description

OpenGL window output routines.

12.3.2 Enumeration Type Documentation

12.3.2.1 enum Gui::AxesParameters

Enumerator:

axeswidth
xaxislength
yaxislength
zaxislength
arrowheight
arrowwidth
maxaxesprm

Definition at line 71 of file gui.h.

12.3.2.2 enum Gui::Color

Enumerator:

red
green
blue
skyblue
brown
magenta
yellow
color6
color7
color8
color9
color10
color11
color12
color13
color14

color15
maxcolor

Definition at line 31 of file gui.h.

12.3.2.3 enum Gui::ColorSchemes

Enumerator:

colorByType
colorByBoundary
colorByTime
colorByMass
maxColorSchemes

Definition at line 51 of file gui.h.

12.3.2.4 enum Gui::Elements

Enumerator:

points
nodes
edges
faces
cells
frame
mesh
boundary_nodes
boundary_edges
boundary_faces
boundary_cells
maxelements

Definition at line 17 of file gui.h.

12.3.2.5 enum Gui::LineParameters

Enumerator:

thickness
length
maxlineprm

Definition at line 65 of file gui.h.

12.3.2.6 enum Gui::Movement

Enumerator:

stay
rotate
moveuv
movew

Definition at line 123 of file gui.h.

12.3.2.7 enum Gui::PointParameters

Enumerator:

variable
vmin
vmax
size
sizevar
massvar
maxpointprm

Definition at line 58 of file gui.h.

12.3.2.8 enum Gui::ShowPars

Enumerator:

showRun
showAxes
showSpheres
showBonds
showGrid
showNodes
showVariables
showBoundaryVertexes
showBoundaryVectors
showToolVertexes
showBoundaryFaceCenters
showBoundaryFaces
showBoundaryGrid
showToolGrid
showFrame
showCellCenters
showFaceCenters

showIsoSurfaces

dumpWindow

maxshowpars

Definition at line 129 of file gui.h.

12.3.2.9 enum Gui::SurfDispType

Enumerator:

gridlines

solidsurface

maxsurfdisptypes

Definition at line 81 of file gui.h.

12.3.3 Function Documentation

12.3.3.1 void Gui::animate ()

Definition at line 1851 of file gui.cc.

References animation, display(), domain, dumpwindow(), dumpWindow, ESC, finished, iwddump, nwdump, Run::option, Domain::run(), showpar, showRun, and Run::time.

Referenced by menu().

12.3.3.2 void Gui::commandMode ()

Definition at line 1465 of file gui.cc.

References consoleMenu(), dumpwindow(), Exit(), helpCommand(), MAXLINLEN, Quit(), runmany(), setBackgroundRun(), setForegroundRun(), switchColorScheme(), toggleSpheres(), and toggleWindowDump().

Referenced by keyboard().

12.3.3.3 void Gui::consoleMenu ()

Definition at line 1360 of file gui.cc.

References displayMenu(), MAXLINLEN, and menu().

Referenced by commandMode(), and keyboard().

12.3.3.4 void Gui::display ()

Definition at line 1530 of file gui.cc.

References blue, Gui::Scene::color, colorByBoundary, colorByMass, colorByTime, colorByType, Molecule::Coordinates(), Collection< Element >::Current(), displayAxes(), domain, dpy, faces, Gui::Scene::frame, frame, getElementColor(), Collection< Element >::goFirst(), Collection<

Element >::goNext(), green, init(), Collection< Element >::isFirst(), Potential::lengthscale(), Gui::Scene::Frame::line, Species::Specie::Mass(), Gui::Scene::Color::maxvalue, Gui::Scene::mesh, Gui::Scene::Color::minvalue, Domain::Molecules(), Gui::Scene::Mesh::node, nodes, Collection< Element >::number(), Run::option, Palette::pickcolor(), REAL, red, refresh(), rotx, roty, scene, Gui::Scene::Color::scheme, showBoundaryGrid, showBoundaryVertexes, showFrame, showGrid, showNodes, showpar, showSpheres, Species::Specie::Size(), size, Species::species, thickness, Run::time, tx, ty, Molecule::Type(), win, xmax, xmin, and zoom.

Referenced by animate().

12.3.3.5 void Gui::displayAxes ()

Definition at line 1181 of file gui.cc.

References arrowheight, arrowwidth, axes, axeswidth, lx, ly, lz, showAxes, showpar, xaxislength, xo, yaxislenght, and zaxislength.

Referenced by display().

12.3.3.6 void Gui::displayMenu ()

Definition at line 1334 of file gui.cc.

Referenced by consoleMenu().

12.3.3.7 void Gui::displaymessage (char * msg)

Definition at line 2162 of file gui.cc.

12.3.3.8 void Gui::drawSegment (REAL * x, REAL * y)

12.3.3.9 void Gui::dumpwindow ()

Definition at line 1461 of file gui.cc.

References windowname, and IO::xwd().

Referenced by animate(), and commandMode().

12.3.3.10 void Gui::Exit ()

Definition at line 1229 of file gui.cc.

References animation, and MAXLINLEN.

Referenced by commandMode(), and keyboard().

12.3.3.11 void Gui::finish ()

12.3.3.12 void Gui::getElementColor (int element_type, REAL rgbcolor[])

Definition at line 2172 of file gui.cc.

References disp, and maxelements.

Referenced by `display()`.

12.3.3.13 `void Gui::getScaling (REAL & lmin, REAL & lmax)`

12.3.3.14 `void Gui::getScaling (double & l0, double & l1)`

Definition at line 1212 of file `gui.cc`.

References `lmax`, and `lmin`.

Referenced by `reshape()`.

12.3.3.15 `void Gui::getXLimits (REAL * xmin, REAL * xmax)`

12.3.3.16 `void Gui::getXLimits (double * x0, double * x1)`

Definition at line 1220 of file `gui.cc`.

References `xmax`, and `xmin`.

12.3.3.17 `void Gui::helpCommand ()`

Definition at line 1871 of file `gui.cc`.

References `Run::outputname`.

Referenced by `commandMode()`.

12.3.3.18 `void Gui::helpDisplay ()`

Definition at line 175 of file `gui.cc`.

References `configfile`, and `Run::outputname`.

Referenced by `init()`, `keyboard()`, and `menu()`.

12.3.3.19 `void Gui::init (int argc, char * argv[], Domain * newdomain)`

Definition at line 91 of file `gui.cc`.

References `animation`, `attributeList`, `Gui::Scene::color`, `colorByMass`, `domain`, `dpy`, `dumpWindow`, `ERROR`, `finished`, `Gui::WindowGeom::height`, `helpDisplay()`, `initdisp()`, `lastx`, `lasty`, `lastz`, `Materials()`, `mouseButtons`, `Run::option`, `query_extension()`, `rotx`, `roty`, `scene`, `Gui::Scene::Color::scheme`, `showAxes`, `showBonds`, `showBoundaryFaceCenters`, `showBoundaryFaces`, `showBoundaryGrid`, `showBoundaryVectors`, `showBoundaryVertexes`, `showCellCenters`, `showFaceCenters`, `showFrame`, `showGrid`, `showNodes`, `showpar`, `showRun`, `showSpheres`, `showToolGrid`, `showToolVertexes`, `showVariables`, `tx`, `ty`, `wtime`, `Gui::WindowGeom::width`, `win`, `window`, `windowname`, and `zoom`.

Referenced by `display()`, and `main()`.

12.3.3.20 `void Gui::initdisp ()`

Definition at line 701 of file `gui.cc`.

References arrowheight, arrowwidth, axes, axeswidth, blue, brown, cells, colorscale, DIM, domain, dx, dy, dz, edges, faces, frame, Gui::Scene::frame, green, Gui::WindowGeom::height, LARGE, length, Gui::Scene::Mesh::line, Gui::Scene::Frame::line, lmax, lmin, lx, ly, lz, magenta, MAX_WINDOW_HEIGHT, MAX_WINDOW_WIDTH, maxcolor, maxelements, mesh, Gui::Scene::mesh, Domain::Molecules(), Gui::Scene::Mesh::node, nodes, Collection< Element >::number(), Run::option, readconf(), REAL, red, Gui::ColorScale::relative, rgbcolor, scene, setElementColor(), Domain::setMaxBound(), Domain::setMinBound(), size, skyblue, step, thickness, vecval, Gui::WindowGeom::width, window, WINDOW_SIZE, xaxislength, xmax, xmin, xo, yaxislength, yellow, zaxislength, and zoom.

Referenced by `init()`.

12.3.3.21 void Gui::InitMaterials (void)

12.3.3.22 void Gui::keyboard (unsigned char key, int x, int y)

12.3.3.23 void Gui::keyboard (unsigned int key)

`refresh()`;

Definition at line 1978 of file `gui.cc`.

References `animation`, `commandMode()`, `consoleMenu()`, `domain`, `ESC`, `Exit()`, `finished`, `helpDisplay()`, `MAXLINLEN`, `Run::option`, `Run::outputname`, `readconf()`, `Domain::run()`, `Domain::save()`, `showAxes`, `showBoundaryFaces`, `showBoundaryGrid`, `showBoundaryVectors`, `showBoundaryVertexes`, `showFrame`, `showGrid`, `showNodes`, `showpar`, `switchColorScheme()`, `Run::time`, and `zoom`.

12.3.3.24 void Gui::Materials (int argc, char * argv[])

Definition at line 818 of file `gui.cc`.

References `lz`, `Run::option`, and `xo`.

Referenced by `init()`.

12.3.3.25 void Gui::menu (int value)

Definition at line 1253 of file `gui.cc`.

References `ABOUT`, `animate()`, `animation`, `finished`, `helpDisplay()`, `showAxes`, `showBonds`, `showBoundaryFaceCenters`, `showBoundaryGrid`, `showBoundaryVectors`, `showBoundaryVertexes`, `showCellCenters`, `showFaceCenters`, `showGrid`, `showNodes`, `showpar`, and `showVariables`.

Referenced by `consoleMenu()`.

12.3.3.26 void Gui::motion (int x, int y)

Definition at line 1954 of file `gui.cc`.

References `lastx`, `lasty`, `mouseButtons`, `REAL`, `rotx`, `roty`, `tx`, `ty`, and `zoom`.

12.3.3.27 void Gui::mouse (int button, int state, int x, int y)

Definition at line 1927 of file `gui.cc`.

References `lastx`, `lasty`, `LEFT_BUTTON`, `MIDDLE_BUTTON`, `mouseButtons`, and `RIGHT_BUTTON`.

12.3.3.28 void Gui::printGridVecLimits ()

12.3.3.29 void Gui::printGridXLimits ()

12.3.3.30 void Gui::printParticleVariables ()

12.3.3.31 void Gui::printPartVelLimits ()

12.3.3.32 void Gui::printVariables (int *n*)

12.3.3.33 int Gui::query_extension (char * *extName*)

Definition at line 78 of file `gui.cc`.

Referenced by `init()`.

12.3.3.34 void Gui::Quit ()

Definition at line 1241 of file `gui.cc`.

References `animation`, and `MAXLINLEN`.

Referenced by `commandMode()`.

12.3.3.35 void Gui::readconf ()

Definition at line 259 of file `gui.cc`.

References `arrowheight`, `arrowwidth`, `axeswidth`, `blue`, `brown`, `Gui::Scene::color`, `colorByMass`, `colorByType`, `Run::configfile`, `configfile`, `DIM`, `DOCTYPE`, `domain`, `dx`, `dy`, `dz`, `frame`, `Gui::Scene::frame`, `green`, `length`, `Gui::Scene::Frame::line`, `Gui::Scene::Mesh::line`, `lx`, `ly`, `lz`, `magenta`, `massvar`, `maxaxesprm`, `Domain::maxBound()`, `maxcolor`, `maxlineprm`, `MAXLINLEN`, `maxpointprm`, `maxshowpars`, `Gui::Scene::Color::maxvalue`, `mesh`, `Gui::Scene::mesh`, `Domain::minBound()`, `Gui::Scene::Color::minvalue`, `Gui::Scene::Mesh::node`, `nodes`, `Run::option`, `IO::parseWord()`, `REAL`, `red`, `rgbcolor`, `scene`, `Gui::Scene::Color::scheme`, `setElementColor()`, `showpar`, `showSpheres`, `size`, `sizevar`, `skyblue`, `step`, `thickness`, `variable`, `vecval`, `vmax`, `vmin`, `xaxislength`, `xmax`, `xmin`, `yaxislength`, and `zaxislength`.

Referenced by `initdisp()`, and `keyboard()`.

12.3.3.36 int Gui::readparam (char * *s*, char * *param*[], int *maxparam*, char * *filename*, REAL *val*[])

Definition at line 230 of file `gui.cc`.

12.3.3.37 int Gui::readparam (char * *s*, char * *param*[], int *maxparam*, char * *filename*, int *val*[])

Definition at line 198 of file `gui.cc`.

12.3.3.38 void Gui::refresh ()

Definition at line 689 of file gui.cc.

Referenced by display().

12.3.3.39 void Gui::reshape (int w, int h)

Definition at line 1890 of file gui.cc.

References getScaling(), lmax, and lmin.

12.3.3.40 void Gui::run ()

Definition at line 2159 of file gui.cc.

Referenced by main().

12.3.3.41 void Gui::runmany ()

Definition at line 1392 of file gui.cc.

References domain, and Domain::run().

Referenced by commandMode().

12.3.3.42 void Gui::selectVariable ()**12.3.3.43 void Gui::setBackgroundRun ()**

Definition at line 1376 of file gui.cc.

References showpar, and showRun.

Referenced by commandMode().

12.3.3.44 void Gui::setElementColor (int element_type, REAL rgbcolor[])

Definition at line 2166 of file gui.cc.

References disp, and maxelements.

Referenced by initdisp(), and readconf().

12.3.3.45 void Gui::setForegroundRun ()

Definition at line 1380 of file gui.cc.

References showpar, and showRun.

Referenced by commandMode().

12.3.3.46 void Gui::showGridElements (int *ne*, double * *X*, REAL *color*[])

12.3.3.47 void Gui::showParticle (double *vmn*, double *vmx*, double *val*, double * *x*)

Definition at line 952 of file gui.cc.

References PI, REAL, and Palette::vav.

12.3.3.48 void Gui::showSphere (double *vmn*, double *vmx*, double *val*, double *rad*, double * *x*)

= { 0.6, 0.8, 0.3, 1.0 },

Definition at line 982 of file gui.cc.

References PI, REAL, and Palette::vav.

12.3.3.49 void Gui::showVector (int *ivar*, double * *X*)

12.3.3.50 void Gui::showVector (double * *x*, double * *v*)

Definition at line 874 of file gui.cc.

References length, and vecval.

12.3.3.51 void Gui::showVectorComponent (int *ivar*, int *icomp*, double * *X*, double *vmn*, double *vmx*)

12.3.3.52 void Gui::switchColorScheme ()

Definition at line 1388 of file gui.cc.

References Gui::Scene::color, maxColorSchemes, scene, and Gui::Scene::Color::scheme.

Referenced by commandMode(), and keyboard().

12.3.3.53 void Gui::toggleSpheres ()

Definition at line 1384 of file gui.cc.

References showpar, and showSpheres.

Referenced by commandMode().

12.3.3.54 void Gui::toggleWindowDump ()

Definition at line 1448 of file gui.cc.

References dumpWindow, iwdump, nwdump, showpar, step, Run::time, and wdtime.

Referenced by commandMode().

12.3.3.55 void Gui::writeData ()

12.3.4 Variable Documentation

12.3.4.1 int Gui::animation

Definition at line 29 of file gui.cc.

Referenced by animate(), Exit(), init(), keyboard(), menu(), and Quit().

12.3.4.2 int Gui::attributeList[]

Initial value:

```
{
    GLX_RGBA,
    GLX_RED_SIZE, 1,
    GLX_GREEN_SIZE, 1,
    GLX_BLUE_SIZE, 1,
    GLX_DOUBLEBUFFER,
    GLX_DEPTH_SIZE, 1,
    None
}
```

Definition at line 33 of file gui.cc.

Referenced by init().

12.3.4.3 REAL Gui::axes ()

Definition at line 43 of file gui.cc.

Referenced by displayAxes(), and initdisp().

12.3.4.4 struct BFaceList* Gui::bface_root

Definition at line 70 of file gui.cc.

12.3.4.5 struct ColorScale Gui::colorscale

Definition at line 66 of file gui.cc.

Referenced by initdisp().

12.3.4.6 int * Gui::coms

Definition at line 76 of file gui.cc.

12.3.4.7 char * Gui::configfile = (char*)"gui.cfg"

Definition at line 26 of file gui.cc.

Referenced by helpDisplay(), and readconf().

12.3.4.8 ElementDisp Gui::disp

Definition at line 25 of file gui.cc.

Referenced by getElementColor(), and setElementColor().

12.3.4.9 Domain* Gui::domain

Definition at line 24 of file gui.cc.

Referenced by animate(), display(), init(), initdisp(), keyboard(), main(), readconf(), and runmany().

12.3.4.10 Display* Gui::dpy

Definition at line 72 of file gui.cc.

Referenced by display(), and init().

12.3.4.11 REAL Gui::dx

Definition at line 43 of file gui.cc.

Referenced by Domain::boundary(), initdisp(), Boundary::Inject(), Domain::interact(), Domain::interaction(), and readconf().

12.3.4.12 REAL Gui::dy

Definition at line 43 of file gui.cc.

Referenced by initdisp(), and readconf().

12.3.4.13 REAL Gui::dz

Definition at line 43 of file gui.cc.

Referenced by initdisp(), and readconf().

12.3.4.14 int Gui::finished

Definition at line 29 of file gui.cc.

Referenced by animate(), init(), keyboard(), and menu().

12.3.4.15 int Gui::firstR = 1

Definition at line 75 of file gui.cc.

12.3.4.16 int Gui::iwdump

Definition at line 29 of file gui.cc.

Referenced by animate(), and toggleWindowDump().

12.3.4.17 REAL Gui::lastx

Definition at line 43 of file gui.cc.

Referenced by `init()`, `motion()`, and `mouse()`.

12.3.4.18 REAL Gui::lasty

Definition at line 43 of file gui.cc.

Referenced by `init()`, `motion()`, and `mouse()`.

12.3.4.19 REAL Gui::lastz

Definition at line 43 of file gui.cc.

Referenced by `init()`.

12.3.4.20 REAL Gui::lmax

Definition at line 61 of file gui.cc.

Referenced by `getScaling()`, `initdisp()`, and `reshape()`.

12.3.4.21 REAL Gui::lmin

Definition at line 61 of file gui.cc.

Referenced by `getScaling()`, `initdisp()`, and `reshape()`.

12.3.4.22 REAL Gui::lx

Definition at line 61 of file gui.cc.

Referenced by `displayAxes()`, `initdisp()`, and `readconf()`.

12.3.4.23 REAL Gui::ly

Definition at line 61 of file gui.cc.

Referenced by `displayAxes()`, `initdisp()`, and `readconf()`.

12.3.4.24 REAL Gui::lz

Definition at line 61 of file gui.cc.

Referenced by `displayAxes()`, `initdisp()`, `Materials()`, and `readconf()`.

12.3.4.25 int Gui::mouseButtons

Definition at line 53 of file gui.cc.

Referenced by `init()`, `motion()`, and `mouse()`.

12.3.4.26 enum Movement Gui::movement

Definition at line 64 of file gui.cc.

12.3.4.27 int Gui::nwdump

Definition at line 29 of file gui.cc.

Referenced by `animate()`, and `toggleWindowDump()`.

12.3.4.28 REAL Gui::rgbcolor

Definition at line 43 of file gui.cc.

Referenced by `initdisp()`, and `readconf()`.

12.3.4.29 REAL Gui::rotx

Definition at line 55 of file gui.cc.

Referenced by `display()`, `init()`, and `motion()`.

12.3.4.30 REAL Gui::roty

Definition at line 56 of file gui.cc.

Referenced by `display()`, `init()`, and `motion()`.

12.3.4.31 struct Scene Gui::scene

Definition at line 67 of file gui.cc.

Referenced by `display()`, `init()`, `initdisp()`, `readconf()`, and `switchColorScheme()`.

12.3.4.32 int Gui::showpar

Definition at line 29 of file gui.cc.

Referenced by `animate()`, `display()`, `displayAxes()`, `init()`, `keyboard()`, `menu()`, `readconf()`, `setBackgroundRun()`, `setForegroundRun()`, `toggleSpheres()`, and `toggleWindowDump()`.

12.3.4.33 REAL Gui::step

Definition at line 43 of file gui.cc.

Referenced by `initdisp()`, `readconf()`, and `toggleWindowDump()`.

12.3.4.34 REAL Gui::tx

Definition at line 57 of file gui.cc.

Referenced by `display()`, `init()`, and `motion()`.

12.3.4.35 REAL Gui::ty

Definition at line 58 of file gui.cc.

Referenced by display(), init(), and motion().

12.3.4.36 int* Gui::vars

Definition at line 76 of file gui.cc.

12.3.4.37 REAL Gui::vecval ()

Definition at line 43 of file gui.cc.

Referenced by initdisp(), readconf(), and showVector().

12.3.4.38 REAL Gui::wdtime

Definition at line 43 of file gui.cc.

Referenced by init(), and toggleWindowDump().

12.3.4.39 Window Gui::win

Definition at line 73 of file gui.cc.

Referenced by display(), and init().

12.3.4.40 struct WindowGeom Gui::window

Definition at line 68 of file gui.cc.

Referenced by init(), and initdisp().

12.3.4.41 char Gui::windowname[MAXLINLEN]

Definition at line 26 of file gui.cc.

Referenced by dumpwindow(), and init().

12.3.4.42 REAL Gui::xmax ()

Definition at line 61 of file gui.cc.

Referenced by display(), Domain::Domain(), getXLimits(), initdisp(), and readconf().

12.3.4.43 REAL Gui::xmin

Definition at line 61 of file gui.cc.

Referenced by display(), Domain::Domain(), getXLimits(), initdisp(), and readconf().

12.3.4.44 REAL Gui::xo

Definition at line 43 of file gui.cc.

Referenced by displayAxes(), initdisp(), and Materials().

12.3.4.45 REAL Gui::zoom

Definition at line 54 of file gui.cc.

Referenced by display(), init(), initdisp(), keyboard(), and motion().

12.4 IO Namespace Reference

Some [IO](#) routines.

Functions

- void [initxwd](#) (int i)
- void [xwd](#) (char *windowname)
- int [getCharAttr](#) (xmlNodePtr cur, char *attr, char *result)
- int [getIntAttr](#) (xmlNodePtr cur, char *attr)
- int [getIntAttr](#) (xmlDocPtr doc, char *tag, char *keyname, char *key, char *attr)
- int [parseWord](#) (xmlDocPtr doc, xmlNodePtr cur, char *keyword, char *result)
- int [parseInt](#) (xmlDocPtr doc, xmlNodePtr cur, char *keyword)
- int [parseFloat](#) (xmlDocPtr doc, xmlNodePtr cur, char *keyword, REAL &result)
- double [parseFloat](#) (xmlDocPtr doc, xmlNodePtr cur, char *keyword)
- void [getTimeXML](#) (char infilename[])
- void [getTime](#) (char infilename[])
- int [getIter](#) (char infilename[])

Variables

- int [ioutput](#) = 0
- int [nxwdump](#) = 0

12.4.1 Detailed Description

Some [IO](#) routines.

12.4.2 Function Documentation

12.4.2.1 `int IO::getCharAttr (xmlNodePtr cur, char * attr, char * result)`

Definition at line 40 of file io.cc.

References [MAXLINLEN](#), and [Run::option](#).

Referenced by [Domain::Domain\(\)](#), and [getIntAttr\(\)](#).

12.4.2.2 `int IO::getIntAttr (xmlDocPtr doc, char * tag, char * keyname, char * key, char * attr)`

Definition at line 77 of file io.cc.

References [getCharAttr\(\)](#), [MAXLINLEN](#), [Run::option](#), [parseInt\(\)](#), and [Potential::value\(\)](#).

12.4.2.3 `int IO::getIntAttr (xmlNodePtr cur, char * attr)`

Definition at line 57 of file io.cc.

References [Run::option](#).

12.4.2.4 int IO::getIter (char *inpfilename*[])

Definition at line 249 of file io.cc.

References DOCTYPE, MAXLINLEN, and Run::option.

Referenced by main().

12.4.2.5 void IO::getTime (char *inpfilename*[])

Definition at line 239 of file io.cc.

References getTimeXML(), Run::option, and Run::time.

Referenced by Run::init().

12.4.2.6 void IO::getTimeXML (char *inpfilename*[])

Definition at line 198 of file io.cc.

References DOCTYPE, MAXLINLEN, parseFloat(), and Run::time.

Referenced by getTime().

12.4.2.7 void IO::initxwd (int *i*)

Definition at line 15 of file io.cc.

References nxwdump.

12.4.2.8 double IO::parseFloat (xmlDocPtr *doc*, xmlNodePtr *cur*, char * *keyword*)

Definition at line 178 of file io.cc.

References Run::option.

Referenced by Domain::Domain().

12.4.2.9 int IO::parseFloat (xmlDocPtr *doc*, xmlNodePtr *cur*, char * *keyword*, REAL & *result*)

Definition at line 158 of file io.cc.

References Run::option, and REAL.

Referenced by getTimeXML().

12.4.2.10 int IO::parseInt (xmlDocPtr *doc*, xmlNodePtr *cur*, char * *keyword*)

Definition at line 138 of file io.cc.

References Run::option.

Referenced by getIntAttr().

12.4.2.11 int IO::parseWord (xmlDocPtr *doc*, xmlNodePtr *cur*, char * *keyword*, char * *result*)

Definition at line 117 of file io.cc.

References MAXLINLEN, and Run::option.

Referenced by Domain::Domain(), and Gui::readconf().

12.4.2.12 void IO::xwd (char * *windowname*)

Definition at line 19 of file io.cc.

References MAXLINLEN, nxwdump, and Run::option.

Referenced by Gui::dumpwindow().

12.4.3 Variable Documentation**12.4.3.1 int IO::ioutput = 0**

Definition at line 13 of file io.cc.

Referenced by Domain::load(), Domain::run(), and Domain::save().

12.4.3.2 int IO::nxwdump = 0

Definition at line 14 of file io.cc.

Referenced by initxwd(), and xwd().

12.5 Palette Namespace Reference

Defines color palette.

Functions

- void [init](#) (REAL vmin, REAL vmax)
- void [pickcolor](#) (REAL var, REAL color[])

Variables

- REAL [vmn](#)
- REAL [vmx](#)
- REAL [vav](#)
- REAL [dvi](#)

12.5.1 Detailed Description

Defines color palette.

12.5.2 Function Documentation

12.5.2.1 void Palette::init (REAL *vmin*, REAL *vmax*)

Definition at line 2189 of file gui.cc.

References [dvi](#), [vav](#), [vmn](#), and [vmx](#).

12.5.2.2 void Palette::pickcolor (REAL *var*, REAL *color*[])

Definition at line 2197 of file gui.cc.

References [dvi](#), [REAL](#), [vav](#), [vmn](#), and [vmx](#).

Referenced by [Gui::display\(\)](#).

12.5.3 Variable Documentation

12.5.3.1 REAL Palette::dvi

Definition at line 2185 of file gui.cc.

Referenced by [init\(\)](#), [Domain::interact\(\)](#), and [pickcolor\(\)](#).

12.5.3.2 REAL Palette::vav

Definition at line 2185 of file gui.cc.

Referenced by [init\(\)](#), [pickcolor\(\)](#), [Gui::showParticle\(\)](#), and [Gui::showSphere\(\)](#).

12.5.3.3 REAL Palette::vmn

Definition at line 2185 of file gui.cc.

Referenced by `init()`, and `pickcolor()`.

12.5.3.4 REAL Palette::vmx

Definition at line 2185 of file gui.cc.

Referenced by `init()`, and `pickcolor()`.

12.6 Potential Namespace Reference

Interaction-potential functions, such as LJ.

Functions

- REAL [invdist](#) (REAL x)
- void [strength](#) (REAL strength)
- REAL [strength](#) ()
- void [lengthscale](#) (REAL lengthscale)
- REAL [lengthscale](#) ()
- void [Cutoff](#) (REAL newcutoff)
- REAL [Cutoff](#) ()
- REAL [value](#) (REAL r)
- REAL [force](#) (REAL r)
- REAL [derivative](#) (REAL r)

Variables

- REAL [sigma](#) = 1.0
- REAL [eta](#) = 1.0
- REAL [cutoff](#) = 2.0
- const REAL [small](#) = 1.0e-20
- const REAL [large](#) = 1.0e20

12.6.1 Detailed Description

Interaction-potential functions, such as LJ.

12.6.2 Function Documentation

12.6.2.1 REAL Potential::Cutoff () [inline]

Definition at line 154 of file model.h.

References cutoff.

Referenced by Domain::load().

12.6.2.2 void Potential::Cutoff (REAL newcutoff) [inline]

Definition at line 153 of file model.h.

References cutoff.

12.6.2.3 REAL Potential::derivative (REAL *r*)**12.6.2.4 REAL Potential::force (REAL *r*)** [inline]

Definition at line 166 of file model.h.

References eta, and sigma.

12.6.2.5 REAL Potential::invdist (REAL *x*)

Definition at line 102 of file model.cc.

12.6.2.6 REAL Potential::lengthscale () [inline]

Definition at line 152 of file model.h.

References sigma.

Referenced by Gui::display(), and Domain::Domain().

12.6.2.7 void Potential::lengthscale (REAL *lengthscale*) [inline]

Definition at line 149 of file model.h.

References sigma.

Referenced by Domain::load().

12.6.2.8 REAL Potential::strength () [inline]

Definition at line 148 of file model.h.

References eta.

Referenced by Domain::Domain().

12.6.2.9 void Potential::strength (REAL *strength*) [inline]

Definition at line 145 of file model.h.

References eta.

12.6.2.10 REAL Potential::value (REAL *r*) [inline]

Definition at line 155 of file model.h.

References cutoff, eta, large, REAL, sigma, and small.

Referenced by IO::getIntAttr().

12.6.3 Variable Documentation

12.6.3.1 `REAL Potential::cutoff = 2.0`

Definition at line 85 of file model.cc.

Referenced by `Cutoff()`, and `value()`.

12.6.3.2 `REAL Potential::eta = 1.0`

Definition at line 85 of file model.cc.

Referenced by `force()`, `strength()`, and `value()`.

12.6.3.3 `const REAL Potential::large = 1.0e20`

Definition at line 142 of file model.h.

Referenced by `value()`.

12.6.3.4 `REAL Potential::sigma = 1.0`

Definition at line 85 of file model.cc.

Referenced by `force()`, `lengthscale()`, and `value()`.

12.6.3.5 `const REAL Potential::small = 1.0e-20`

Definition at line 142 of file model.h.

Referenced by `value()`.

12.7 Run Namespace Reference

Processing command-line and execution control parameters.

Classes

- struct [Time](#)
Execution time-control structure.

Functions

- void [testrnd](#) ()
Test random numer generator.
- REAL [rnd](#) ()
Seeds random number.
- REAL [gauss](#) ()
normal random number generator
- void [init](#) (int argc, char *argv[])
Initialize all variables, and execution options.
- void [readcmdline](#) (int argc, char *argv[])
Parses command-line options.
- int [elapsed](#) ()
time (sec) elapsed since the start of the run
- void [seed](#) ()
seed random number generator
- void(*) [usage](#) ()

Variables

- struct [Option option](#) = {0,0,0,0}
command-line options
- void(*) [usage](#) ()
- char [programname](#) [MAXLINLEN]
- char [configfile](#) [MAXLINLEN]
- char [outputfile](#) [MAXLINLEN]
- char [inputfile](#) [MAXLINLEN]
- char [outputname](#) [MAXLINLEN]
- int [nthreads](#)
- struct timeb [worldtime](#)
- struct [Time time](#)

12.7.1 Detailed Description

Processing command-line and execution control parameters.

12.7.2 Function Documentation

12.7.2.1 `int Run::elapsed ()`

time (sec) elapsed since the start of the run

Definition at line 138 of file run.cc.

References worldtime.

Referenced by Domain::save().

12.7.2.2 `REAL Run::gauss ()`

normal random number generator

From Numerical Recipes in C, Ch.7.2 Zero mean and unit variance

Definition at line 57 of file run.cc.

References REAL, and rnd().

12.7.2.3 `void Run::init (int argc, char * argv[])`

Initialize all variables, and execution options.

Definition at line 79 of file run.cc.

References configfile, Option::debug, IO::getTime(), Option::mesh, option, readcmdline(), Option::restart, Option::verbose, worldtime, and Option::xterm.

Referenced by main().

12.7.2.4 `void Run::readcmdline (int argc, char * argv[])`

Parses command-line options.

Definition at line 93 of file run.cc.

References configfile, Option::debug, inputfile, Option::mesh, option, outputfile, outputname, program-name, Option::restart, and Option::verbose.

Referenced by init().

12.7.2.5 `REAL Run::rnd ()`

Seeds random number.

random number generator

Generates random number between 0 and 1

Definition at line 51 of file run.cc.

References RAND_MAX, and REAL.

Referenced by gauss(), and testrnd().

12.7.2.6 void Run::seed ()

seed random number generator

12.7.2.7 void Run::testrnd ()

Test random number generator.

Definition at line 31 of file run.cc.

References RAND_MAX, REAL, and rnd().

12.7.2.8 void(*) Run::usage ()

Definition at line 23 of file job.cc.

References programname.

12.7.3 Variable Documentation

12.7.3.1 char Run::configfile

Definition at line 20 of file run.cc.

Referenced by Domain::Domain(), init(), main(), readcmdline(), and Gui::readconf().

12.7.3.2 char Run::inputfile

Definition at line 20 of file run.cc.

Referenced by main(), and readcmdline().

12.7.3.3 int Run::nthreads

Definition at line 26 of file run.cc.

Referenced by Domain::interaction(), and Domain::run().

12.7.3.4 struct Option Run::option = {0,0,0} [read]

command-line options

Definition at line 17 of file run.cc.

Referenced by Gui::animate(), Gui::display(), Domain::Domain(), IO::getCharAttr(), IO::getIntAttr(), IO::getIter(), IO::getTime(), init(), Gui::init(), GridContainer::init(), Domain::init(), Gui::initdisp(), Gui::keyboard(), Domain::load(), main(), Gui::Materials(), IO::parseFloat(), IO::parseInt(), IO::parseWord(), Pool< Element >::Pool(), readcmdline(), Gui::readconf(), Domain::run(), Domain::save(), IO::xwd(), and Domain::~~Domain().

12.7.3.5 char Run::outputfile

Definition at line 20 of file run.cc.

Referenced by readcmdline().

12.7.3.6 char Run::outputname

Definition at line 20 of file run.cc.

Referenced by Gui::helpCommand(), Gui::helpDisplay(), Gui::keyboard(), readcmdline(), and Domain::run().

12.7.3.7 char Run::programname

Definition at line 20 of file run.cc.

Referenced by main(), readcmdline(), usage(), and usage().

12.7.3.8 struct Time Run::time [read]

Definition at line 29 of file run.cc.

Referenced by Gui::animate(), Domain::boundary(), Gui::display(), IO::getTime(), IO::getTimeXML(), Bulk::inject(), Boundary::Inject(), Domain::injection(), Domain::interaction(), Gui::keyboard(), Domain::load(), Molecule::Molecule(), Molecule::Move(), GridContainer::put(), Domain::run(), Domain::save(), and Gui::toggleWindowDump().

12.7.3.9 void(* Run::usage)() ()

12.7.3.10 struct timeb Run::worldtime

Definition at line 28 of file run.cc.

Referenced by elapsed(), and init().

12.8 Species Namespace Reference

Description of [Specie](#) and [Reaction](#) classes.

Classes

- class [Specie](#)
- struct [Gas](#)
Gas is a specie with some additional parameters.
- struct [Reaction](#)
Reaction determines the two products only.

Enumerations

- enum [Interaction](#) { [missed](#) = 0, [collided](#), [reacted](#), [annihalated](#) }

Variables

- int [nspecies](#)
- [Specie](#) * [species](#)
- [Reaction](#) * [reactions](#)

12.8.1 Detailed Description

Description of [Specie](#) and [Reaction](#) classes.

12.8.2 Enumeration Type Documentation

12.8.2.1 enum Species::Interaction

Enumerator:

missed

collided

reacted two reactants -> two products

annihalated two reactants -> one product

Definition at line 164 of file species.h.

12.8.3 Variable Documentation

12.8.3.1 int Species::nspecies

Definition at line 8 of file species.cc.

Referenced by [Domain::Domain\(\)](#), [Domain::interact\(\)](#), [Domain::load\(\)](#), [Molecule::Move\(\)](#), [Species::Reaction::Outcome::Outcome\(\)](#), and [Domain::run\(\)](#).

12.8.3.2 Reaction * Species::reactions

Definition at line 10 of file species.cc.

Referenced by Domain::Domain(), and Domain::interact().

12.8.3.3 Specie * Species::species

Definition at line 9 of file species.cc.

Referenced by Domain::boundary(), Gui::display(), Domain::Domain(), Domain::init(), Bulk::inject(), Boundary::Inject(), Domain::injection(), Domain::interact(), Molecule::KineticEnergy(), Domain::load(), Molecule::Move(), GridContainer::put(), Domain::run(), Domain::save(), and Molecule::Temperature().

Chapter 13

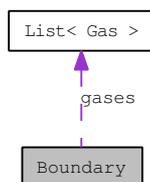
Class Documentation

13.1 Boundary Class Reference

Domain Boundary Definition.

```
#include <domain.h>
```

Collaboration diagram for Boundary:



Public Member Functions

- **Boundary** ()
Boundary constructor: sets defaults.
- **~Boundary** ()
Boundary destructor.
- REAL **Area** ()
- void **Area** (REAL a)
- REAL **Temperature** ()
- void **Temperature** (REAL t)
- void **init** (int i, int j, REAL ymin[], REAL ymax[], int nspecies)
- void **Inject** (int speciotype, REAL vavx, REAL **temperature**, **Collection**< **Molecule** > *molecules)
Injecting a specie from accross the boundary with a specified velocity.

Public Attributes

- int **idir**

boundary orientation in 3D space: 0,1,2

- int [iside](#)
side of the boundary: 0,1
- REAL * [xmin](#)
- REAL * [xmax](#)
min/max bounds
- [BoundaryTypes](#) type
defines boundary type
- bool [adiabatic](#)
flag for adiabatic boundary
- REAL [area](#)
boundary surface area
- REAL [temperature](#)
boundary temperature
- [List< Gas > gases](#)
list of gases across the boundary
- Reaction * [reactions](#)

13.1.1 Detailed Description

[Domain Boundary](#) Definition.

Definition at line 18 of file domain.h.

13.1.2 Constructor & Destructor Documentation

13.1.2.1 [Boundary::Boundary \(\)](#) [inline]

[Boundary](#) constructor: sets defaults.

Definition at line 30 of file domain.h.

References [adiabatic](#), [area](#), [elasticBoundary](#), [idir](#), [iside](#), [reactions](#), [temperature](#), [type](#), [xmax](#), and [xmin](#).

13.1.2.2 [Boundary::~~Boundary \(\)](#) [inline]

[Boundary](#) destructor.

Definition at line 40 of file domain.h.

13.1.3 Member Function Documentation

13.1.3.1 void Boundary::Area (REAL *a*) [inline]

Definition at line 44 of file domain.h.

References area.

13.1.3.2 REAL Boundary::Area () [inline]

Definition at line 43 of file domain.h.

References area.

13.1.3.3 void Boundary::init (int *i*, int *j*, REAL *ymin*[], REAL *ymax*[], int *nspecies*)

Definition at line 27 of file domain.cc.

References idir, iside, reactions, xmax, and xmin.

Referenced by Domain::Domain().

13.1.3.4 void Boundary::Inject (int *specietype*, REAL *vavx*, REAL *temperature*, Collection< Molecule > * *molecules*)

Injecting a specie from across the boundary with a specified velocity.

Parameters:

- specietype* injected specie type
- vavx* average velocity in on direction
- temperature* temperature
- molecules* collection of molecules

Definition at line 32 of file domain.cc.

References Collection< Element >::append(), Molecule::Coordinate(), DIM, Gui::dx, GAUSS, idir, iside, GridContainer::put(), REAL, RND, Species::Specie::Size(), SMALL, Species::species, Molecule::Temperature(), Run::time, Molecule::Type(), Molecule::Velocity(), xmax, and xmin.

Referenced by Domain::injection().

13.1.3.5 void Boundary::Temperature (REAL *t*) [inline]

Definition at line 46 of file domain.h.

References temperature.

13.1.3.6 REAL Boundary::Temperature () [inline]

Definition at line 45 of file domain.h.

References temperature.

13.1.4 Member Data Documentation

13.1.4.1 `bool Boundary::adiabatic`

flag for adiabatic boundary

Definition at line 23 of file domain.h.

Referenced by `Boundary()`, and `Domain::Domain()`.

13.1.4.2 `REAL Boundary::area`

boundary surface area

Definition at line 25 of file domain.h.

Referenced by `Area()`, `Boundary()`, `Domain::Domain()`, and `Domain::injection()`.

13.1.4.3 `List<Gas> Boundary::gases`

list of gases across the boundary

Definition at line 27 of file domain.h.

Referenced by `Domain::Domain()`, and `Domain::injection()`.

13.1.4.4 `int Boundary::idir`

boundary orientation in 3D space: 0,1,2

Definition at line 19 of file domain.h.

Referenced by `Boundary()`, `init()`, and `Inject()`.

13.1.4.5 `int Boundary::iside`

side of the boundary: 0,1

Definition at line 19 of file domain.h.

Referenced by `Boundary()`, `init()`, and `Inject()`.

13.1.4.6 `Reaction* Boundary::reactions`

boundary reactions with all species

Definition at line 28 of file domain.h.

Referenced by `Boundary()`, `Domain::boundary()`, `Domain::Domain()`, and `init()`.

13.1.4.7 `REAL Boundary::temperature`

boundary temperature

Definition at line 25 of file domain.h.

Referenced by `Boundary()`, `Domain::Domain()`, `Domain::injection()`, and `Temperature()`.

13.1.4.8 BoundaryTypes Boundary::type

defines boundary type

Definition at line 22 of file domain.h.

Referenced by Boundary(), Domain::boundary(), Domain::BoundaryType(), and Domain::Domain().

13.1.4.9 REAL * Boundary::xmax

min/max bounds

Definition at line 21 of file domain.h.

Referenced by Boundary(), init(), and Inject().

13.1.4.10 REAL* Boundary::xmin

Definition at line 21 of file domain.h.

Referenced by Boundary(), init(), and Inject().

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

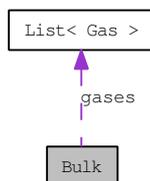
- [domain.h](#)
- [domain.cc](#)

13.2 Bulk Class Reference

Domain Bulk Properties.

```
#include <domain.h>
```

Collaboration diagram for Bulk:



Public Member Functions

- REAL `Temperature` ()
- void `Temperature` (REAL temp)
- REAL `Volume` ()
- void `Volume` (REAL vol)
- void `init` (REAL `xmin`[], REAL `xmax`[])
 - Initializing bulk: setting volume.*
- `Molecule * inject` (int specie, REAL velocity, `Collection< Molecule > *molecules`)
 - Adding a molecule to the pool with not registering on the grid.*

Public Attributes

- `List< Gas > gases`
 - list of gases initially present in the bulk*

Private Attributes

- REAL `temperature`
 - REAL `volume`
 - REAL `xmin` [DIM]
 - REAL `xmax` [DIM]
- box-dimensions*

13.2.1 Detailed Description

Domain Bulk Properties.

Holds thermodynamics properties and the list of species in the bulk of the domain. Initialized only when the new run. If `Run::option.restart=1`, it is ignored and the bulk properties and species are retrieved from the input file.

Definition at line 63 of file domain.h.

13.2.2 Member Function Documentation

13.2.2.1 void Bulk::init (REAL *xmin*[], REAL *xmax*[])

Initializing bulk: setting volume.

Definition at line 657 of file domain.cc.

References DIM, volume, xmax, and xmin.

Referenced by Domain::Domain().

13.2.2.2 Molecule * Bulk::inject (int *specie*, REAL *velocity*, Collection< Molecule > * *molecules*)

Adding a molecule to the pool with not registering on the grid.

Injecting a specie into the bulk with a specified velocity (no registering in the grid).

Definition at line 665 of file domain.cc.

References Collection< Element >::append(), Molecule::Coordinate(), DIM, GAUSS, REAL, RND, Species::Specie::Size(), SMALL, Species::species, Run::time, Molecule::Type(), Molecule::Velocity(), xmax, and xmin.

Referenced by Domain::init().

13.2.2.3 void Bulk::Temperature (REAL *temp*) [inline]

Definition at line 69 of file domain.h.

References temperature.

13.2.2.4 REAL Bulk::Temperature () [inline]

Definition at line 68 of file domain.h.

References temperature.

Referenced by Domain::Domain(), and Domain::init().

13.2.2.5 void Bulk::Volume (REAL *vol*) [inline]

Definition at line 71 of file domain.h.

References volume.

13.2.2.6 REAL Bulk::Volume () [inline]

Definition at line 70 of file domain.h.

References volume.

Referenced by Domain::init().

13.2.3 Member Data Documentation

13.2.3.1 List<Gas> Bulk::gases

list of gases initially present in the bulk

Definition at line 67 of file domain.h.

Referenced by Domain::Domain(), and Domain::init().

13.2.3.2 REAL Bulk::temperature [private]

Definition at line 64 of file domain.h.

Referenced by Temperature().

13.2.3.3 REAL Bulk::volume [private]

Definition at line 64 of file domain.h.

Referenced by init(), and Volume().

13.2.3.4 REAL Bulk::xmax[DIM] [private]

box-dimensions

Definition at line 64 of file domain.h.

Referenced by init(), and inject().

13.2.3.5 REAL Bulk::xmin[DIM] [private]

Definition at line 64 of file domain.h.

Referenced by init(), and inject().

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

- [domain.h](#)
- [domain.cc](#)

13.3 Collection< Element > Class Template Reference

```
#include <collection.h>
```

Public Member Functions

- [Collection](#) ()
- [~Collection](#) ()
- [int maxCounters](#) ()
- [int number](#) ()
- [int maxnumber](#) ()
- [void setFirst](#) ()
- [void setFirstLast](#) ()
- [void goFirst](#) ()
- [void goFirst](#) (int i)
- [void goFirstNext](#) ()
- [void goFirstNext](#) (int i)
- [void FirstNext](#) ()
- [void FirstPrev](#) ()
- [bool isFirstLast](#) ()
- [void goLastNext](#) ()
- [void goLastNext](#) (int i)
- [void goLast](#) ()
- [void goLast](#) (int i)
- [void goNext](#) ()
- [void goNext](#) (int i)
- [void goPrev](#) ()
- [void goPrev](#) (int i)
- [bool isFirst](#) ()
- [bool isFirst](#) (int i)
- [bool isLast](#) ()
- [bool isLast](#) (int i)
- [void LastNext](#) ()
- [void LastPrev](#) ()
- [void go](#) (Item< Element > *p)
- [Element * First](#) ()
- [Element * Last](#) ()
- [Element * Next](#) ()
- [Element * Next](#) (int i)
- [Element * getNext](#) ()
- [Element * getNext](#) (int i)
- [Element * Prev](#) ()
- [Element * Prev](#) (int i)
- [Element * getPrev](#) ()
- [Element * getPrev](#) (int i)
- [Element * Current](#) ()
- [Element * Current](#) (int i)
- [Item< Element > * getItem](#) ()
- [Item< Element > * getItem](#) (int i)

- [Item](#)< [Element](#) > * [getFirstItem](#) ()
- [Item](#)< [Element](#) > * [getLastItem](#) ()
- bool [init](#) (int n)
- [Element](#) * [insertAfter](#) ()
- [Element](#) * [insert0](#) ()
- [Element](#) * [insert](#) ()
- [Element](#) * [insert](#) (int icounter)
- bool [insert0](#) ([Element](#) *element)
- bool [insert](#) ([Element](#) *element)
- bool [insert](#) ([Element](#) *element, int icounter)
- [Element](#) * [append0](#) ()
- [Element](#) * [append](#) ()
- [Element](#) * [append](#) (int icounter)
- bool [append0](#) ([Element](#) *element)
- bool [append](#) ([Element](#) *element)
- bool [append](#) ([Element](#) *element, int icounter)
- bool [remove0](#) ()
- bool [remove](#) ()
- bool [remove](#) (int icounter)

Private Attributes

- int [mitems](#)
- int [nitems](#)
- bool [locked](#)
- struct [Item](#)< [Element](#) > * [items](#)
- struct [Item](#)< [Element](#) > * [dead](#)
- struct [Item](#)< [Element](#) > * [first](#)
- struct [Item](#)< [Element](#) > * [last](#)
- struct [Item](#)< [Element](#) > * [current](#)
- struct [Item](#)< [Element](#) > * [current1](#) [[maxcounters](#)]

13.3.1 Detailed Description

template<class [Element](#)> **class** [Collection](#)< [Element](#) >

Definition at line 25 of file [collection.h](#).

13.3.2 Constructor & Destructor Documentation

13.3.2.1 **template**<class [Element](#) > [Collection](#)< [Element](#) >::[Collection](#) () [[inline](#)]

Definition at line 103 of file [collection.h](#).

References [Collection](#)< [Element](#) >::[current](#), [Collection](#)< [Element](#) >::[current1](#), [Collection](#)< [Element](#) >::[dead](#), [Collection](#)< [Element](#) >::[first](#), [Collection](#)< [Element](#) >::[last](#), [Collection](#)< [Element](#) >::[locked](#), [maxcounters](#), [Collection](#)< [Element](#) >::[mitems](#), and [Collection](#)< [Element](#) >::[nitems](#).

13.3.2.2 `template<class Element > Collection< Element >::~~Collection ()` [inline]

Definition at line 110 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::dead`, `Collection< Element >::first`, `Collection< Element >::last`, `maxcounters`, `Collection< Element >::mitems`, and `Collection< Element >::nitems`.

13.3.3 Member Function Documentation**13.3.3.1** `template<class Element > bool Collection< Element >::append (Element * element, int icounter)` [inline]

Definition at line 329 of file collection.h.

References `Collection< Element >::append0()`, `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::locked`, `maxcounters`, and `WAIT`.

13.3.3.2 `template<class Element > bool Collection< Element >::append (Element * element)` [inline]

Definition at line 322 of file collection.h.

References `Collection< Element >::append()`.

13.3.3.3 `template<class Element > Element * Collection< Element >::append (int icounter)` [inline]

Definition at line 308 of file collection.h.

References `Collection< Element >::append0()`, `Collection< Element >::current`, `Collection< Element >::current1`, `Item< Element >::element`, `Collection< Element >::locked`, `maxcounters`, and `WAIT`.

13.3.3.4 `template<class Element > Element * Collection< Element >::append ()` [inline]

Definition at line 297 of file collection.h.

References `Collection< Element >::append0()`, `Item< Element >::element`, `Collection< Element >::locked`, and `WAIT`.

Referenced by `Collection< Element >::append()`, `Bulk::inject()`, `Boundary::Inject()`, and `Domain::load()`.

13.3.3.5 `template<class Element > bool Collection< Element >::append0 (Element * element)`**13.3.3.6** `template<class Element > Element * Collection< Element >::append0 ()` [inline]

Definition at line 263 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::dead`, `Collection< Element >::first`, `Collection< Element >::last`, `maxcounters`, `Item< Element >::next`, `Collection< Element >::nitems`, and `Item< Element >::prev`.

Referenced by `Collection< Element >::append()`.

13.3.3.7 `template<class Element > Element* Collection< Element >::Current (int i)` [inline]

Definition at line 79 of file collection.h.

13.3.3.8 `template<class Element > Element* Collection< Element >::Current ()` [inline]

Definition at line 78 of file collection.h.

Referenced by `Domain::computeBounds()`, `Gui::display()`, `Domain::Domain()`, `Domain::run()`, and `Domain::save()`.

13.3.3.9 `template<class Element > Element* Collection< Element >::First ()` [inline]

Definition at line 68 of file collection.h.

13.3.3.10 `template<class Element > void Collection< Element >::FirstNext ()` [inline]

Definition at line 50 of file collection.h.

13.3.3.11 `template<class Element > void Collection< Element >::FirstPrev ()` [inline]

Definition at line 51 of file collection.h.

13.3.3.12 `template<class Element > Item<Element>* Collection< Element >::getFirstItem ()`
[inline]

Definition at line 82 of file collection.h.

13.3.3.13 `template<class Element > Item<Element>* Collection< Element >::getItem (int i)`
[inline]

Definition at line 81 of file collection.h.

13.3.3.14 `template<class Element > Item<Element>* Collection< Element >::getItem ()`
[inline]

Definition at line 80 of file collection.h.

13.3.3.15 `template<class Element > Item<Element>* Collection< Element >::getLastItem ()`
[inline]

Definition at line 83 of file collection.h.

13.3.3.16 `template<class Element > Element* Collection< Element >::getNext (int i)`
[inline]

Definition at line 73 of file collection.h.

13.3.3.17 `template<class Element > Element* Collection< Element >::getNext ()` [inline]

Definition at line 72 of file collection.h.

13.3.3.18 `template<class Element > Element* Collection< Element >::getPrev (int i)`
[inline]

Definition at line 77 of file collection.h.

13.3.3.19 `template<class Element > Element* Collection< Element >::getPrev ()` [inline]

Definition at line 76 of file collection.h.

13.3.3.20 `template<class Element > void Collection< Element >::go (Item< Element > * p)`
[inline]

Definition at line 67 of file collection.h.

13.3.3.21 `template<class Element > void Collection< Element >::goFirst (int i)` [inline]

Definition at line 47 of file collection.h.

13.3.3.22 `template<class Element > void Collection< Element >::goFirst ()` [inline]

Definition at line 41 of file collection.h.

Referenced by `Domain::computeBounds()`, `Gui::display()`, `Domain::Domain()`, `Domain::run()`, and `Domain::save()`.

13.3.3.23 `template<class Element > void Collection< Element >::goFirstNext (int i)` [inline]

Definition at line 49 of file collection.h.

13.3.3.24 `template<class Element > void Collection< Element >::goFirstNext ()` [inline]

Definition at line 48 of file collection.h.

13.3.3.25 `template<class Element > void Collection< Element >::goLast (int i)` [inline]

Definition at line 56 of file collection.h.

13.3.3.26 `template<class Element > void Collection< Element >::goLast ()` [inline]

Definition at line 55 of file collection.h.

13.3.3.27 `template<class Element > void Collection< Element >::goLastNext (int i)` [inline]

Definition at line 54 of file collection.h.

13.3.3.28 `template<class Element > void Collection< Element >::goLastNext ()` [inline]

Definition at line 53 of file collection.h.

13.3.3.29 `template<class Element > void Collection< Element >::goNext (int i)` [inline]

Definition at line 58 of file collection.h.

13.3.3.30 `template<class Element > void Collection< Element >::goNext ()` [inline]

Definition at line 57 of file collection.h.

Referenced by `Domain::computeBounds()`, `Gui::display()`, `Domain::Domain()`, `Domain::run()`, and `Domain::save()`.

13.3.3.31 `template<class Element > void Collection< Element >::goPrev (int i)` [inline]

Definition at line 60 of file collection.h.

13.3.3.32 `template<class Element > void Collection< Element >::goPrev ()` [inline]

Definition at line 59 of file collection.h.

13.3.3.33 `template<class Element > bool Collection< Element >::init (int n)` [inline]

Definition at line 117 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::dead`, `Collection< Element >::first`, `Collection< Element >::last`, `maxcounters`, `Collection< Element >::mitems`, and `Collection< Element >::nitems`.

Referenced by `Domain::Domain()`.

13.3.3.34 `template<class Element > bool Collection< Element >::insert (Element * element, int icounter)` [inline]

Definition at line 247 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::insert0()`, `Collection< Element >::locked`, `maxcounters`, and `WAIT`.

13.3.3.35 `template<class Element > bool Collection< Element >::insert (Element * element)` [inline]

Definition at line 240 of file collection.h.

References `Collection< Element >::insert()`.

13.3.3.36 `template<class Element > Element * Collection< Element >::insert (int icounter)`
`[inline]`

Definition at line 226 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Item< Element >::element`, `Collection< Element >::insert0()`, `Collection< Element >::locked`, `maxcounters`, and `WAIT`.

13.3.3.37 `template<class Element > Element * Collection< Element >::insert ()` `[inline]`

Definition at line 214 of file collection.h.

References `Collection< Element >::insert0()`, `Collection< Element >::locked`, `Collection< Element >::number()`, and `WAIT`.

Referenced by `Collection< Element >::insert()`.

13.3.3.38 `template<class Element > bool Collection< Element >::insert0 (Element * element)`

13.3.3.39 `template<class Element > Element * Collection< Element >::insert0 ()` `[inline]`

Definition at line 179 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::dead`, `Item< Element >::element`, `Collection< Element >::first`, `Collection< Element >::last`, `maxcounters`, `Item< Element >::next`, `Collection< Element >::nitems`, and `Item< Element >::prev`.

Referenced by `Collection< Element >::insert()`.

13.3.3.40 `template<class Element > Element * Collection< Element >::insertAfter ()`
`[inline]`

Definition at line 144 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::dead`, `Item< Element >::element`, `Collection< Element >::first`, `Collection< Element >::last`, `maxcounters`, `Item< Element >::next`, `Collection< Element >::nitems`, and `Item< Element >::prev`.

Referenced by `Domain::boundary()`.

13.3.3.41 `template<class Element > bool Collection< Element >::isFirst (int i)` `[inline]`

Definition at line 62 of file collection.h.

13.3.3.42 `template<class Element > bool Collection< Element >::isFirst ()` `[inline]`

Definition at line 61 of file collection.h.

Referenced by `Domain::computeBounds()`, `Gui::display()`, `Domain::Domain()`, `Domain::run()`, and `Domain::save()`.

13.3.3.43 `template<class Element > bool Collection< Element >::isFirstLast ()` `[inline]`

Definition at line 52 of file collection.h.

13.3.3.44 `template<class Element > bool Collection< Element >::isLast (int i) [inline]`

Definition at line 64 of file collection.h.

13.3.3.45 `template<class Element > bool Collection< Element >::isLast () [inline]`

Definition at line 63 of file collection.h.

13.3.3.46 `template<class Element > Element* Collection< Element >::Last () [inline]`

Definition at line 69 of file collection.h.

13.3.3.47 `template<class Element > void Collection< Element >::LastNext () [inline]`

Definition at line 65 of file collection.h.

13.3.3.48 `template<class Element > void Collection< Element >::LastPrev () [inline]`

Definition at line 66 of file collection.h.

13.3.3.49 `template<class Element > int Collection< Element >::maxCounters () [inline]`

Definition at line 36 of file collection.h.

Referenced by `Domain::run()`.

13.3.3.50 `template<class Element > int Collection< Element >::maxnumber () [inline]`

Definition at line 38 of file collection.h.

Referenced by `Domain::run()`.

13.3.3.51 `template<class Element > Element* Collection< Element >::Next (int i) [inline]`

Definition at line 71 of file collection.h.

13.3.3.52 `template<class Element > Element* Collection< Element >::Next () [inline]`

Definition at line 70 of file collection.h.

13.3.3.53 `template<class Element > int Collection< Element >::number () [inline]`

Definition at line 37 of file collection.h.

Referenced by `Gui::display()`, `Domain::Domain()`, `Gui::initdisp()`, `Collection< Element >::insert()`, `Domain::load()`, `Domain::run()`, and `Domain::save()`.

13.3.3.54 `template<class Element > Element* Collection< Element >::Prev (int i)` [inline]

Definition at line 75 of file collection.h.

13.3.3.55 `template<class Element > Element* Collection< Element >::Prev ()` [inline]

Definition at line 74 of file collection.h.

13.3.3.56 `template<class Element > bool Collection< Element >::remove (int icounter)`
[inline]

Definition at line 393 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::locked`, `maxcounters`, `Collection< Element >::remove0()`, and `WAIT`.

13.3.3.57 `template<class Element > bool Collection< Element >::remove ()` [inline]

Definition at line 379 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::locked`, `maxcounters`, `Collection< Element >::remove0()`, and `WAIT`.

13.3.3.58 `template<class Element > bool Collection< Element >::remove0 ()` [inline]

Definition at line 344 of file collection.h.

References `Collection< Element >::current`, `Collection< Element >::current1`, `Collection< Element >::dead`, `Collection< Element >::first`, `Collection< Element >::last`, `maxcounters`, `Item< Element >::next`, and `Collection< Element >::nitems`.

Referenced by `Collection< Element >::remove()`, and `Domain::run()`.

13.3.3.59 `template<class Element > void Collection< Element >::setFirst ()` [inline]

Definition at line 39 of file collection.h.

13.3.3.60 `template<class Element > void Collection< Element >::setFirstLast ()` [inline]

Definition at line 40 of file collection.h.

13.3.4 Member Data Documentation

13.3.4.1 `template<class Element > struct Item< Element > * Collection< Element >::current`
[read, private]

Definition at line 28 of file collection.h.

Referenced by `Collection< Element >::append()`, `Collection< Element >::append0()`, `Collection< Element >::Collection()`, `Collection< Molecule >::Current()`, `Collection< Molecule >::getItem()`, `Collection< Molecule >::getNext()`, `Collection< Molecule >::getPrev()`, `Collection< Molecule >::go()`,

Collection< Molecule >::goFirst(), Collection< Molecule >::goFirstNext(), Collection< Molecule >::goLast(), Collection< Molecule >::goLastNext(), Collection< Molecule >::goNext(), Collection< Molecule >::goPrev(), Collection< Element >::init(), Collection< Element >::insert(), Collection< Element >::insert0(), Collection< Element >::insertAfter(), Collection< Molecule >::isFirst(), Collection< Molecule >::isLast(), Collection< Molecule >::Next(), Collection< Molecule >::Prev(), Collection< Element >::remove(), Collection< Element >::remove0(), and Collection< Element >::~~Collection().

13.3.4.2 **template<class Element > struct Item< Element > * Collection< Element >::current1[maxcounters]** [read, private]

Definition at line 28 of file collection.h.

Referenced by Collection< Element >::append(), Collection< Element >::append0(), Collection< Element >::Collection(), Collection< Molecule >::Current(), Collection< Molecule >::getItem(), Collection< Molecule >::getNext(), Collection< Molecule >::getPrev(), Collection< Molecule >::goFirst(), Collection< Molecule >::goFirstNext(), Collection< Molecule >::goLast(), Collection< Molecule >::goLastNext(), Collection< Molecule >::goNext(), Collection< Molecule >::goPrev(), Collection< Element >::init(), Collection< Element >::insert(), Collection< Element >::insert0(), Collection< Element >::insertAfter(), Collection< Molecule >::isFirst(), Collection< Molecule >::isLast(), Collection< Molecule >::Next(), Collection< Molecule >::Prev(), Collection< Element >::remove(), Collection< Element >::remove0(), and Collection< Element >::~~Collection().

13.3.4.3 **template<class Element > struct Item< Element > * Collection< Element >::dead** [read, private]

Definition at line 28 of file collection.h.

Referenced by Collection< Element >::append0(), Collection< Element >::Collection(), Collection< Element >::init(), Collection< Element >::insert0(), Collection< Element >::insertAfter(), Collection< Element >::remove0(), and Collection< Element >::~~Collection().

13.3.4.4 **template<class Element > struct Item< Element > * Collection< Element >::first** [read, private]

Definition at line 28 of file collection.h.

Referenced by Collection< Element >::append0(), Collection< Element >::Collection(), Collection< Molecule >::First(), Collection< Molecule >::FirstNext(), Collection< Molecule >::FirstPrev(), Collection< Molecule >::getFirstItem(), Collection< Molecule >::goFirst(), Collection< Molecule >::goFirstNext(), Collection< Element >::init(), Collection< Element >::insert0(), Collection< Element >::insertAfter(), Collection< Molecule >::isFirst(), Collection< Molecule >::isFirstLast(), Collection< Element >::remove0(), Collection< Molecule >::setFirst(), Collection< Molecule >::setFirstLast(), and Collection< Element >::~~Collection().

13.3.4.5 **template<class Element > struct Item< Element > * Collection< Element >::items** [read, private]

Definition at line 28 of file collection.h.

13.3.4.6 `template<class Element > struct Item< Element > * Collection< Element >::last`
[read, private]

Definition at line 28 of file collection.h.

Referenced by `Collection< Element >::append0()`, `Collection< Element >::Collection()`, `Collection< Molecule >::getLastItem()`, `Collection< Molecule >::goLast()`, `Collection< Molecule >::goLastNext()`, `Collection< Element >::init()`, `Collection< Element >::insert0()`, `Collection< Element >::insertAfter()`, `Collection< Molecule >::isFirstLast()`, `Collection< Molecule >::isLast()`, `Collection< Molecule >::Last()`, `Collection< Molecule >::LastNext()`, `Collection< Molecule >::LastPrev()`, `Collection< Element >::remove0()`, `Collection< Molecule >::setFirst()`, `Collection< Molecule >::setFirstLast()`, and `Collection< Element >::~~Collection()`.

13.3.4.7 `template<class Element > bool Collection< Element >::locked` [private]

Definition at line 27 of file collection.h.

Referenced by `Collection< Element >::append()`, `Collection< Element >::Collection()`, `Collection< Element >::insert()`, and `Collection< Element >::remove()`.

13.3.4.8 `template<class Element > int Collection< Element >::mitems` [private]

Definition at line 26 of file collection.h.

Referenced by `Collection< Element >::Collection()`, `Collection< Element >::init()`, `Collection< Molecule >::maxnumber()`, and `Collection< Element >::~~Collection()`.

13.3.4.9 `template<class Element > int Collection< Element >::nitems` [private]

Definition at line 26 of file collection.h.

Referenced by `Collection< Element >::append0()`, `Collection< Element >::Collection()`, `Collection< Element >::init()`, `Collection< Element >::insert0()`, `Collection< Element >::insertAfter()`, `Collection< Molecule >::number()`, `Collection< Element >::remove0()`, and `Collection< Element >::~~Collection()`.

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

- [collection.h](#)

13.4 Gui::ColorScale Struct Reference

```
#include <gui.h>
```

Public Attributes

- int [relative](#):1

13.4.1 Detailed Description

Definition at line 104 of file gui.h.

13.4.2 Member Data Documentation

13.4.2.1 int Gui::ColorScale::relative

Definition at line 105 of file gui.h.

Referenced by Gui::initdisp().

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

- [gui.h](#)

13.5 Container< Element > Class Template Reference

```
#include <collection.h>
```

Public Member Functions

- [Container](#) ()
- [~Container](#) ()
- [bool islocked](#) ()
- [void lock](#) ()
- [void unlock](#) ()
- [int number](#) ()
- [void setFirst](#) ()
- [void setFirst](#) (int i)
- [void setFirstLast](#) ()
- [void setFirstLast](#) (int i)
- [void goFirst](#) ()
- [void goFirst](#) (int i)
- [void goFirstNext](#) ()
- [void goFirstNext](#) (int i)
- [void FirstNext](#) ()
- [void FirstNext](#) (int i)
- [void FirstPrev](#) ()
- [void FirstPrev](#) (int i)
- [bool isFirstLast](#) ()
- [bool isFirstLast](#) (int i)
- [void goLastNext](#) ()
- [void goLastNext](#) (int i)
- [void goLast](#) ()
- [void goLast](#) (int i)
- [void goNext](#) ()
- [void goNext](#) (int i)
- [void goPrev](#) ()
- [void goPrev](#) (int i)
- [bool isFirst](#) ()
- [bool isFirst](#) (int i)
- [bool isLast](#) ()
- [bool isLast](#) (int i)
- [void LastNext](#) ()
- [void LastPrev](#) ()
- [void go](#) (Ptr< Element > *p)
- [void go](#) (Ptr< Element > *p, int i)
- [Element * First](#) ()
- [Element * First](#) (int i)
- [Element * Last](#) ()
- [Element * Next](#) ()
- [Element * Next](#) (int i)
- [Element * getNext](#) ()
- [Element * getNext](#) (int i)

- Element * [Prev](#) ()
- Element * [Prev](#) (int i)
- Element * [getPrev](#) ()
- Element * [getPrev](#) (int i)
- Element * [Current](#) ()
- Element * [Current](#) (int i)
- [Ptr](#)< Element > * [getPtr](#) ()
- [Ptr](#)< Element > * [getPtr](#) (int i)
- [Ptr](#)< Element > * [getFirstPtr](#) ()
- [Ptr](#)< Element > * [getFirstPtr](#) (int i)
- [Ptr](#)< Element > * [getLastPtr](#) ()
- bool [insert0](#) (Element *element)
- bool [insert](#) (Element *element)
- bool [insert](#) (Element *element, int i)
- [Ptr](#)< Element > * [insert1](#) (Element *element, int i)
- bool [insert0](#) ()
- bool [insert](#) ()
- bool [insert](#) (int i)
- bool [append0](#) (Element *element)
- bool [append](#) (Element *element)
- bool [append](#) (Element *element, int i)
- bool [append0](#) ()
- bool [append](#) ()
- bool [append](#) (int i)
- [Ptr](#)< Element > * [append1](#) (Element *element, int i)
- bool [link0](#) ([Ptr](#)< Element > * &ptr)
- bool [link](#) ([Ptr](#)< Element > * &ptr)
- bool [unlink0](#) ([Ptr](#)< Element > * &ptr)
- bool [unlink](#) ([Ptr](#)< Element > * &ptr)
- bool [remove0](#) ()
- bool [remove](#) ()
- bool [remove](#) (int i)
- bool [remove0](#) ([Ptr](#)< Element > * &ptr)
- bool [remove](#) ([Ptr](#)< Element > * ptr)
- bool [remove](#) ([Ptr](#)< Element > * ptr, int i)
- bool [checkPool](#) (char *msg, [Pool](#)< Element > *pool)

Private Attributes

- int [nptrs](#)
- bool [locked](#)
- [Ptr](#)< Element > * [first](#)
- [Ptr](#)< Element > * [last](#)
- [Ptr](#)< Element > * [current](#)
- [Ptr](#)< Element > * [first1](#) [[maxcounters](#)]
- [Ptr](#)< Element > * [current1](#) [[maxcounters](#)]

13.5.1 Detailed Description

`template<class Element> class Container< Element >`

Definition at line 512 of file collection.h.

13.5.2 Constructor & Destructor Documentation

13.5.2.1 `template<class Element > Container< Element >::Container ()` [inline]

Definition at line 2 of file container.h.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::last`, `Container< Element >::locked`, `maxcounters`, and `Container< Element >::nptrs`.

13.5.2.2 `template<class Element > Container< Element >::~~Container ()` [inline]

Definition at line 11 of file container.h.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::last`, `maxcounters`, and `Container< Element >::nptrs`.

13.5.3 Member Function Documentation

13.5.3.1 `template<class Element > bool Container< Element >::append (int i)` [inline]

Definition at line 82 of file container.h.

References `Container< Element >::append0()`, `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::locked`, `maxcounters`, and `WAIT`.

13.5.3.2 `template<class Element > bool Container< Element >::append ()` [inline]

Definition at line 66 of file container.h.

References `Container< Element >::append0()`, `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::locked`, `maxcounters`, and `WAIT`.

13.5.3.3 `template<class Element > bool Container< Element >::append (Element * element, int i)` [inline]

Definition at line 119 of file container.h.

References `Container< Element >::append0()`, `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::last`, `Container< Element >::locked`, `maxcounters`, and `WAIT`.

13.5.3.4 `template<class Element > bool Container< Element >::append (Element * element)`
`[inline]`

Definition at line 101 of file container.h.

References `Container< Element >::append0()`, `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::last`, `Container< Element >::locked`, `maxcounters`, and `WAIT`.

13.5.3.5 `template<class Element > bool Container< Element >::append0 ()` `[inline]`

Definition at line 30 of file container.h.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Pool< Element >::get()`, `Container< Element >::last`, `maxcounters`, `Container< Element >::nptrs`, `GridContainer::pool`, and `Ptr< Element >::prev`.

Referenced by `Container< Element >::append()`, and `Container< Element >::append1()`.

13.5.3.6 `template<class Element > bool Container< Element >::append0 (Element * element)`

13.5.3.7 `template<class Element > Ptr< Element > * Container< Element >::append1 (Element * element, int i)` `[inline]`

Definition at line 141 of file container.h.

References `Container< Element >::append0()`, `Container< Element >::last`, `Container< Element >::locked`, and `WAIT`.

13.5.3.8 `template<class Element > bool Container< Element >::checkPool (char * msg, Pool< Element > * pool)` `[inline]`

Definition at line 19 of file container.h.

References `Pool< Element >::check()`, `Container< Element >::current`, `Container< Element >::first`, `Pool< Element >::hook`, and `Container< Element >::nptrs`.

13.5.3.9 `template<class Element > Element* Container< Element >::Current (int i)` `[inline]`

Definition at line 573 of file collection.h.

References `Container< Element >::current1`.

13.5.3.10 `template<class Element > Element* Container< Element >::Current ()` `[inline]`

Definition at line 572 of file collection.h.

References `Container< Element >::current`.

Referenced by `Domain::interaction()`.

13.5.3.11 `template<class Element > Element* Container< Element >::First (int i)` `[inline]`

Definition at line 562 of file collection.h.

References Container< Element >::first1.

13.5.3.12 `template<class Element > Element* Container< Element >::First ()` [inline]

Definition at line 561 of file collection.h.

References Container< Element >::first.

Referenced by Domain::interaction().

13.5.3.13 `template<class Element > void Container< Element >::FirstNext (int i)` [inline]

Definition at line 540 of file collection.h.

References Container< Element >::first1.

13.5.3.14 `template<class Element > void Container< Element >::FirstNext ()` [inline]

Definition at line 539 of file collection.h.

References Container< Element >::first.

Referenced by Domain::interaction().

13.5.3.15 `template<class Element > void Container< Element >::FirstPrev (int i)` [inline]

Definition at line 542 of file collection.h.

References Container< Element >::first1.

13.5.3.16 `template<class Element > void Container< Element >::FirstPrev ()` [inline]

Definition at line 541 of file collection.h.

References Container< Element >::first.

13.5.3.17 `template<class Element > Ptr<Element>* Container< Element >::getFirstPtr (int i)` [inline]

Definition at line 577 of file collection.h.

References Container< Element >::first1.

13.5.3.18 `template<class Element > Ptr<Element>* Container< Element >::getFirstPtr ()` [inline]

Definition at line 576 of file collection.h.

References Container< Element >::first.

13.5.3.19 `template<class Element > Ptr<Element>* Container< Element >::getLastPtr ()`
[inline]

Definition at line 578 of file collection.h.

References Container< Element >::last.

13.5.3.20 `template<class Element > Element* Container< Element >::getNext (int i)`
[inline]

Definition at line 567 of file collection.h.

References Container< Element >::current1.

13.5.3.21 `template<class Element > Element* Container< Element >::getNext ()` [inline]

Definition at line 566 of file collection.h.

References Container< Element >::current.

13.5.3.22 `template<class Element > Element* Container< Element >::getPrev (int i)`
[inline]

Definition at line 571 of file collection.h.

References Container< Element >::current1.

13.5.3.23 `template<class Element > Element* Container< Element >::getPrev ()` [inline]

Definition at line 570 of file collection.h.

References Container< Element >::current.

13.5.3.24 `template<class Element > Ptr<Element>* Container< Element >::getPtr (int i)`
[inline]

Definition at line 575 of file collection.h.

References Container< Element >::current1.

13.5.3.25 `template<class Element > Ptr<Element>* Container< Element >::getPtr ()`
[inline]

Definition at line 574 of file collection.h.

References Container< Element >::current.

13.5.3.26 `template<class Element > void Container< Element >::go (Ptr< Element > *p, int i)`
[inline]

Definition at line 560 of file collection.h.

References Container< Element >::current1.

13.5.3.27 `template<class Element > void Container< Element >::go (Ptr< Element > *p)`
[inline]

Definition at line 559 of file collection.h.

References Container< Element >::current.

13.5.3.28 `template<class Element > void Container< Element >::goFirst (int i)` [inline]

Definition at line 536 of file collection.h.

References Container< Element >::current1, and Container< Element >::first1.

13.5.3.29 `template<class Element > void Container< Element >::goFirst ()` [inline]

Definition at line 530 of file collection.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, and maxcounters.

Referenced by Domain::interaction().

13.5.3.30 `template<class Element > void Container< Element >::goFirstNext (int i)` [inline]

Definition at line 538 of file collection.h.

References Container< Element >::current1, and Container< Element >::first1.

13.5.3.31 `template<class Element > void Container< Element >::goFirstNext ()` [inline]

Definition at line 537 of file collection.h.

References Container< Element >::current, and Container< Element >::first.

13.5.3.32 `template<class Element > void Container< Element >::goLast (int i)` [inline]

Definition at line 548 of file collection.h.

References Container< Element >::current1, and Container< Element >::last.

13.5.3.33 `template<class Element > void Container< Element >::goLast ()` [inline]

Definition at line 547 of file collection.h.

References Container< Element >::current, and Container< Element >::last.

13.5.3.34 `template<class Element > void Container< Element >::goLastNext (int i)` [inline]

Definition at line 546 of file collection.h.

References Container< Element >::current1, and Container< Element >::last.

13.5.3.35 `template<class Element > void Container< Element >::goLastNext ()` [inline]

Definition at line 545 of file collection.h.

References `Container< Element >::current`, and `Container< Element >::last`.

13.5.3.36 `template<class Element > void Container< Element >::goNext (int i)` [inline]

Definition at line 550 of file collection.h.

References `Container< Element >::current1`.

13.5.3.37 `template<class Element > void Container< Element >::goNext ()` [inline]

Definition at line 549 of file collection.h.

References `Container< Element >::current`.

Referenced by `Domain::interaction()`.

13.5.3.38 `template<class Element > void Container< Element >::goPrev (int i)` [inline]

Definition at line 552 of file collection.h.

References `Container< Element >::current1`.

13.5.3.39 `template<class Element > void Container< Element >::goPrev ()` [inline]

Definition at line 551 of file collection.h.

References `Container< Element >::current`.

13.5.3.40 `template<class Element > bool Container< Element >::insert (int i)` [inline]

Definition at line 326 of file container.h.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::insert0()`, `Container< Element >::locked`, `maxcounters`, and `WAIT`.

13.5.3.41 `template<class Element > bool Container< Element >::insert ()` [inline]

Definition at line 309 of file container.h.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::insert0()`, `Container< Element >::locked`, `maxcounters`, and `WAIT`.

13.5.3.42 `template<class Element > bool Container< Element >::insert (Element * element, int i)` [inline]

Definition at line 365 of file container.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, Container< Element >::insert0(), Container< Element >::locked, maxcounters, and WAIT.

13.5.3.43 `template<class Element > bool Container< Element >::insert (Element * element)`
`[inline]`

Definition at line 347 of file container.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, Container< Element >::insert0(), Container< Element >::locked, maxcounters, and WAIT.

13.5.3.44 `template<class Element > bool Container< Element >::insert0 ()` `[inline]`

Definition at line 276 of file container.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, Pool< Element >::get(), Container< Element >::last, maxcounters, Ptr< Element >::next, Container< Element >::nptrs, GridContainer::pool, and Ptr< Element >::prev.

Referenced by Container< Element >::insert(), and Container< Element >::insert1().

13.5.3.45 `template<class Element > bool Container< Element >::insert0 (Element * element)`

13.5.3.46 `template<class Element > Ptr< Element > * Container< Element >::insert1 (Element * element, int i)` `[inline]`

Definition at line 387 of file container.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, Container< Element >::insert0(), Container< Element >::locked, maxcounters, and WAIT.

13.5.3.47 `template<class Element > bool Container< Element >::isFirst (int i)` `[inline]`

Definition at line 554 of file collection.h.

References Container< Element >::current1, and Container< Element >::first1.

13.5.3.48 `template<class Element > bool Container< Element >::isFirst ()` `[inline]`

Definition at line 553 of file collection.h.

References Container< Element >::current, and Container< Element >::first.

Referenced by Domain::interaction().

13.5.3.49 `template<class Element > bool Container< Element >::isFirstLast (int i)` `[inline]`

Definition at line 544 of file collection.h.

References Container< Element >::first1, and Container< Element >::last.

13.5.3.50 `template<class Element > bool Container< Element >::isFirstLast () [inline]`

Definition at line 543 of file collection.h.

References `Container< Element >::first`, and `Container< Element >::last`.

Referenced by `Domain::interaction()`.

13.5.3.51 `template<class Element > bool Container< Element >::isLast (int i) [inline]`

Definition at line 556 of file collection.h.

References `Container< Element >::current1`, and `Container< Element >::last`.

13.5.3.52 `template<class Element > bool Container< Element >::isLast () [inline]`

Definition at line 555 of file collection.h.

References `Container< Element >::current`, and `Container< Element >::last`.

Referenced by `Domain::interaction()`.

13.5.3.53 `template<class Element > bool Container< Element >::islocked () [inline]`

Definition at line 522 of file collection.h.

References `Container< Element >::locked`.

13.5.3.54 `template<class Element > Element* Container< Element >::Last () [inline]`

Definition at line 563 of file collection.h.

References `Container< Element >::last`.

13.5.3.55 `template<class Element > void Container< Element >::LastNext () [inline]`

Definition at line 557 of file collection.h.

References `Container< Element >::last`.

13.5.3.56 `template<class Element > void Container< Element >::LastPrev () [inline]`

Definition at line 558 of file collection.h.

References `Container< Element >::last`.

13.5.3.57 `template<class Element > bool Container< Element >::link (Ptr< Element > *& ptr) [inline]`

Definition at line 194 of file container.h.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::link0()`, `Container< Element >::locked`, `maxcounters`, and `WAIT`.

13.5.3.58 `template<class Element > bool Container< Element >::link0 (Ptr< Element > *& ptr)`
[inline]

Definition at line 163 of file container.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, Container< Element >::last, maxcounters, Ptr< Element >::next, Container< Element >::nptrs, and Ptr< Element >::prev.

Referenced by Container< Element >::link().

13.5.3.59 `template<class Element > void Container< Element >::lock ()` [inline]

Definition at line 523 of file collection.h.

References Container< Element >::locked.

13.5.3.60 `template<class Element > Element* Container< Element >::Next (int i)` [inline]

Definition at line 565 of file collection.h.

References Container< Element >::current1.

13.5.3.61 `template<class Element > Element* Container< Element >::Next ()` [inline]

Definition at line 564 of file collection.h.

References Container< Element >::current.

13.5.3.62 `template<class Element > int Container< Element >::number ()` [inline]

Definition at line 525 of file collection.h.

References Container< Element >::nptrs.

Referenced by Domain::interaction().

13.5.3.63 `template<class Element > Element* Container< Element >::Prev (int i)` [inline]

Definition at line 569 of file collection.h.

References Container< Element >::current1.

13.5.3.64 `template<class Element > Element* Container< Element >::Prev ()` [inline]

Definition at line 568 of file collection.h.

References Container< Element >::current.

13.5.3.65 `template<class Element > bool Container< Element >::remove (Ptr< Element > * ptr, int i)` [inline]

Definition at line 524 of file container.h.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::locked`, `maxcounters`, `Container< Element >::remove0()`, and `WAIT`.

13.5.3.66 `template<class Element > bool Container< Element >::remove (Ptr< Element > * ptr)` [inline]

Definition at line 512 of file `container.h`.

References `Container< Element >::locked`, `Container< Element >::remove0()`, and `WAIT`.

13.5.3.67 `template<class Element > bool Container< Element >::remove (int i)` [inline]

Definition at line 451 of file `container.h`.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::locked`, `maxcounters`, `Container< Element >::remove0()`, and `WAIT`.

13.5.3.68 `template<class Element > bool Container< Element >::remove ()` [inline]

Definition at line 431 of file `container.h`.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::locked`, `maxcounters`, `Container< Element >::remove0()`, and `WAIT`.

Referenced by `Domain::interaction()`, and `Domain::run()`.

13.5.3.69 `template<class Element > bool Container< Element >::remove0 (Ptr< Element > *& ptr)` [inline]

Definition at line 473 of file `container.h`.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::last`, `maxcounters`, `Container< Element >::nptrs`, `GridContainer::pool`, and `Pool< Element >::put()`.

13.5.3.70 `template<class Element > bool Container< Element >::remove0 ()` [inline]

Definition at line 409 of file `container.h`.

References `Container< Element >::current`, `Container< Element >::current1`, `Container< Element >::first`, `Container< Element >::first1`, `Container< Element >::last`, `maxcounters`, `Container< Element >::nptrs`, `GridContainer::pool`, and `Pool< Element >::put()`.

Referenced by `Container< Element >::remove()`.

13.5.3.71 `template<class Element > void Container< Element >::setFirst (int i)` [inline]

Definition at line 527 of file `collection.h`.

References `Container< Element >::first1`, and `Container< Element >::last`.

13.5.3.72 `template<class Element > void Container< Element >::setFirst ()` [inline]

Definition at line 526 of file collection.h.

References Container< Element >::first, and Container< Element >::last.

Referenced by Domain::interaction().

13.5.3.73 `template<class Element > void Container< Element >::setFirstLast (int i)` [inline]

Definition at line 529 of file collection.h.

References Container< Element >::first1, and Container< Element >::last.

13.5.3.74 `template<class Element > void Container< Element >::setFirstLast ()` [inline]

Definition at line 528 of file collection.h.

References Container< Element >::first, and Container< Element >::last.

Referenced by Domain::interaction().

13.5.3.75 `template<class Element > bool Container< Element >::unlink (Ptr< Element > *& ptr)` [inline]

Definition at line 261 of file container.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, Container< Element >::locked, maxcounters, Container< Element >::unlink0(), and WAIT.

13.5.3.76 `template<class Element > bool Container< Element >::unlink0 (Ptr< Element > *& ptr)` [inline]

Definition at line 211 of file container.h.

References Container< Element >::current, Container< Element >::current1, Container< Element >::first, Container< Element >::first1, Container< Element >::last, maxcounters, Ptr< Element >::next, Container< Element >::nptrs, and Ptr< Element >::prev.

Referenced by Container< Element >::unlink().

13.5.3.77 `template<class Element > void Container< Element >::unlock ()` [inline]

Definition at line 524 of file collection.h.

References Container< Element >::locked.

13.5.4 Member Data Documentation**13.5.4.1** `template<class Element > Ptr<Element> * Container< Element >::current` [private]

Definition at line 517 of file collection.h.

Referenced by `Container< Element >::append()`, `Container< Element >::append0()`, `Container< Element >::checkPool()`, `Container< Element >::Container()`, `Container< Element >::Current()`, `Container< Element >::getNext()`, `Container< Element >::getPrev()`, `Container< Element >::getPtr()`, `Container< Element >::go()`, `Container< Element >::goFirst()`, `Container< Element >::goFirstNext()`, `Container< Element >::goLast()`, `Container< Element >::goLastNext()`, `Container< Element >::goNext()`, `Container< Element >::goPrev()`, `Container< Element >::insert()`, `Container< Element >::insert0()`, `Container< Element >::insert1()`, `Container< Element >::isFirst()`, `Container< Element >::isLast()`, `Container< Element >::link()`, `Container< Element >::link0()`, `Container< Element >::Next()`, `Container< Element >::Prev()`, `Container< Element >::remove()`, `Container< Element >::remove0()`, `Container< Element >::unlink()`, `Container< Element >::unlink0()`, and `Container< Element >::~~Container()`.

13.5.4.2 `template<class Element > Ptr<Element> * Container< Element >::current1[maxcounters]` [private]

Definition at line 517 of file `collection.h`.

Referenced by `Container< Element >::append()`, `Container< Element >::append0()`, `Container< Element >::Container()`, `Container< Element >::Current()`, `Container< Element >::getNext()`, `Container< Element >::getPrev()`, `Container< Element >::getPtr()`, `Container< Element >::go()`, `Container< Element >::goFirst()`, `Container< Element >::goFirstNext()`, `Container< Element >::goLast()`, `Container< Element >::goLastNext()`, `Container< Element >::goNext()`, `Container< Element >::goPrev()`, `Container< Element >::insert()`, `Container< Element >::insert0()`, `Container< Element >::insert1()`, `Container< Element >::isFirst()`, `Container< Element >::isLast()`, `Container< Element >::link()`, `Container< Element >::link0()`, `Container< Element >::Next()`, `Container< Element >::Prev()`, `Container< Element >::remove()`, `Container< Element >::remove0()`, `Container< Element >::unlink()`, `Container< Element >::unlink0()`, and `Container< Element >::~~Container()`.

13.5.4.3 `template<class Element > Ptr<Element>* Container< Element >::first` [private]

Definition at line 517 of file `collection.h`.

Referenced by `Container< Element >::append()`, `Container< Element >::append0()`, `Container< Element >::checkPool()`, `Container< Element >::Container()`, `Container< Element >::First()`, `Container< Element >::FirstNext()`, `Container< Element >::FirstPrev()`, `Container< Element >::getFirstPtr()`, `Container< Element >::goFirst()`, `Container< Element >::goFirstNext()`, `Container< Element >::insert()`, `Container< Element >::insert0()`, `Container< Element >::insert1()`, `Container< Element >::isFirst()`, `Container< Element >::isFirstLast()`, `Container< Element >::link()`, `Container< Element >::link0()`, `Container< Element >::remove()`, `Container< Element >::remove0()`, `Container< Element >::setFirst()`, `Container< Element >::setFirstLast()`, `Container< Element >::unlink()`, `Container< Element >::unlink0()`, and `Container< Element >::~~Container()`.

13.5.4.4 `template<class Element > Ptr<Element> * Container< Element >::first1[maxcounters]` [private]

Definition at line 517 of file `collection.h`.

Referenced by `Container< Element >::append()`, `Container< Element >::append0()`, `Container< Element >::Container()`, `Container< Element >::First()`, `Container< Element >::FirstNext()`, `Container< Element >::FirstPrev()`, `Container< Element >::getFirstPtr()`, `Container< Element >::goFirst()`, `Container< Element >::goFirstNext()`, `Container< Element >::insert()`, `Container< Element >::insert0()`, `Container< Element >::insert1()`, `Container< Element >::isFirst()`, `Container< Element >::isFirstLast()`, `Container< Element >::link()`, `Container< Element >::link0()`, `Container< Element >::remove()`, `Container< Element >::remove0()`, `Container< Element >::setFirst()`, `Container< Element >::setFirstLast()`, `Container<`

Element >::unlink(), Container< Element >::unlink0(), and Container< Element >::~~Container().

13.5.4.5 `template<class Element > Ptr<Element> * Container< Element >::last` [private]

Definition at line 517 of file collection.h.

Referenced by Container< Element >::append(), Container< Element >::append0(), Container< Element >::append1(), Container< Element >::Container(), Container< Element >::getLastPtr(), Container< Element >::goLast(), Container< Element >::goLastNext(), Container< Element >::insert0(), Container< Element >::isFirstLast(), Container< Element >::isLast(), Container< Element >::Last(), Container< Element >::LastNext(), Container< Element >::LastPrev(), Container< Element >::link0(), Container< Element >::remove0(), Container< Element >::setFirst(), Container< Element >::setFirstLast(), Container< Element >::unlink0(), and Container< Element >::~~Container().

13.5.4.6 `template<class Element > bool Container< Element >::locked` [private]

Definition at line 514 of file collection.h.

Referenced by Container< Element >::append(), Container< Element >::append1(), Container< Element >::Container(), Container< Element >::insert(), Container< Element >::insert1(), Container< Element >::islocked(), Container< Element >::link(), Container< Element >::lock(), Container< Element >::remove(), Container< Element >::unlink(), and Container< Element >::unlock().

13.5.4.7 `template<class Element > int Container< Element >::nptrs` [private]

Definition at line 513 of file collection.h.

Referenced by Container< Element >::append0(), Container< Element >::checkPool(), Container< Element >::Container(), Container< Element >::insert0(), Container< Element >::link0(), Container< Element >::number(), Container< Element >::remove0(), Container< Element >::unlink0(), and Container< Element >::~~Container().

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

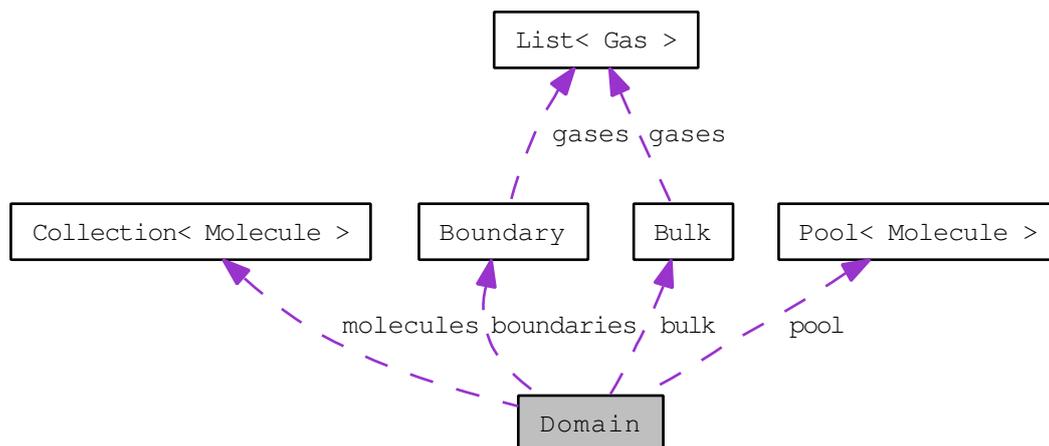
- [collection.h](#)
- [container.h](#)

13.6 Domain Class Reference

Computational [Domain](#).

```
#include <domain.h>
```

Collaboration diagram for Domain:



Public Member Functions

- [Domain](#) (char *filename)
 - Domain* constructor creates an instance of a domain.
- [~Domain](#) ()
- void [BoundaryType](#) (enum [BoundaryTypes](#) b, int idir, int inside)
- enum [BoundaryTypes](#) [BoundaryType](#) (int idir, int inside)
- void [setMinBound](#) (int i, REAL x)
- void [setMaxBound](#) (int i, REAL x)
- REAL [minBound](#) (int i)
- REAL [maxBound](#) (int i)
- int [computeBounds](#) (REAL x0[], REAL x1[])
- [Collection](#)< [Molecule](#) > * [Molecules](#) ()
- int [run](#) (int niter)
 - The Domain solver: running niter iterations.*
- void [step](#) (int iter)
- int [boundary](#) ([Molecule](#) *molecule)
- void [injection](#) ()
 - Inject cross-boundary species:.*
- void [interaction](#) ()
- void [interaction](#) (int icounter)
- [Interaction](#) [interact](#) ([Molecule](#) *a, [Molecule](#) *b)
- void [init](#) ()
- void [load](#) (char *filename)

Initializing molecules according to bulk properties (restart=0).

- void [save](#) (char *taskname)

Private Attributes

- int * [distribution](#)
molecule counts for all species
- REAL [bounds](#) [2][DIM]
- char [boundaryName](#) [maxBoundaryTypes][WORDLENGTH]
to be deprecated
- [Bulk](#) [bulk](#)
bulk of the domain: initialized only if Run::option.restart=0
- [Boundary](#) [boundaries](#) [DIM][2]
six boundaries of a hexahedral domain
- [Collection](#)< [Molecule](#) > [molecules](#)
Collection of molecules.
- [Pool](#)< [Molecule](#) > * [pool](#)
Molecule pointer pool.

13.6.1 Detailed Description

Computational [Domain](#).

Implements main data types inside a computational domain

Definition at line 83 of file domain.h.

13.6.2 Constructor & Destructor Documentation

13.6.2.1 [Domain::Domain](#) (char **filename*)

[Domain](#) constructor creates an instance of a domain.

get domain specs from XML configfile

read XML input file

Loop over file records

set xterm output mode:

Read the number of molecules:

Read species:

load reactions

Read domain specs:

Read domain bounds

Read initial bulk properties

Read boundary specs

Read boundary reactions:

< probability for the formation of these products as opposed to different products If only one pair of products exists the probability = 1

Definition at line 72 of file domain.cc.

References Species::Reaction::Add(), Boundary::adiabatic, List< Element >::append(), Boundary::area, Geom::area(), AtomicMassUnit, AvogadroNumber, BoltzmannConstant, boundaries, boundary(), boundaryName, bounds, bulk, GridContainer::cellsize, Run::configfile, Species::Specie::Cp(), Collection< Element >::Current(), Species::Gas::density, DIM, distribution, DOCTYPE, elasticBoundary, Species::Reaction::Outcome::Enthalpy(), Species::Reaction::First(), GasConstant, Boundary::gases, Bulk::gases, IO::getCharAttr(), Collection< Element >::goFirst(), Collection< Element >::goNext(), Species::Specie::Id(), init(), Bulk::init(), Boundary::init(), Collection< Element >::init(), insideBoundary, Collection< Element >::isFirst(), Potential::lengthscale(), load(), Species::Specie::Mass(), maxBound(), maxBoundaryTypes, MAXLINLEN, minBound(), molecules, Species::Reaction::Next(), Species::nspecies, Collection< Element >::number(), openBoundary, Run::option, IO::parseFloat(), IO::parseWord(), periodicBoundary, pool, Species::Reaction::Outcome::Probability(), Species::Reaction::Outcome::Product(), GridContainer::put(), Boundary::reactions, Species::reactions, REAL, setMaxBound(), setMinBound(), Species::Specie::Size(), Gui::size, Species::species, Potential::strength(), Boundary::temperature, Bulk::Temperature(), TINY, Boundary::type, VOIDSPECIE, WORDLENGTH, XMAX, Gui::xmax, XMIN, Gui::xmin, and Option::xterm.

13.6.2.2 Domain::~~Domain ()

Definition at line 617 of file domain.cc.

References Run::option.

13.6.3 Member Function Documentation

13.6.3.1 int Domain::boundary (Molecule * molecule)

< do reaction: A + B =

< new molecule A type

< new molecule B type

Definition at line 1076 of file domain.cc.

References boundaries, bounds, Molecule::Coordinates(), Species::Specie::Cp(), DIM, DOF, DOFI, DOFK, Gui::dx, elasticBoundary, Species::Reaction::Outcome::Enthalpy(), Species::Reaction::First(), Collection< Element >::insertAfter(), insideBoundary, Molecule::InternalEnergy(), Molecule::KineticEnergy(), LENGTH, Species::Specie::Mass(), molecules, MULC, Species::Reaction::Next(), openBoundary, periodicBoundary, Species::Reaction::Outcome::Probability(), Species::Reaction::Outcome::Product(), Species::reacted, Boundary::reactions, REAL, RND, Species::Specie::Size(), SMALL, Species::species, Run::time, Species::Reaction::Outcome::Time(), Molecule::Type(), Boundary::type, Molecule::Velocity(), VOIDSPECIE, XMAX, and XMIN.

Referenced by Domain(), injection(), and run().

13.6.3.2 enum BoundaryTypes Domain::BoundaryType (int *idir*, int *inside*) [inline]

Definition at line 101 of file domain.h.

References boundaries, DIM, and Boundary::type.

13.6.3.3 void Domain::BoundaryType (enum BoundaryTypes *b*, int *idir*, int *inside*) [inline]

Definition at line 94 of file domain.h.

References boundaries, DIM, and Boundary::type.

13.6.3.4 int Domain::computeBounds (REAL *x0*[], REAL *x1*[])

Definition at line 829 of file domain.cc.

References Molecule::Coordinates(), Collection< Element >::Current(), DIM, Collection< Element >::goFirst(), Collection< Element >::goNext(), Collection< Element >::isFirst(), LARGE, molecules, and REAL.

13.6.3.5 void Domain::init ()

create local gas-array

Go over all gases

point to current gas

select current gas-specie

Estimate the average velocity and the number of molecules of each specie to be injected into the bulk

get molecule's mass in atomic units

convert molecule's mass to kg

average velocity in one direction

get density in kg/m³

get the number of molecules in the bulk

Inject all molecules:

Definition at line 621 of file domain.cc.

References AtomicMassUnit, BoltzmannConstant, bulk, List< Element >::Current(), Species::Gas::density, Bulk::gases, List< Element >::goFirst(), List< Element >::goNext(), Species::Specie::Id(), Bulk::inject(), List< Element >::isFirst(), Species::Specie::Mass(), molecules, List< Element >::number(), Run::option, Species::Gas::specie, Species::species, Molecule::Temperature(), Bulk::Temperature(), and Bulk::Volume().

Referenced by Domain().

13.6.3.6 void Domain::injection ()

Inject cross-boundary species:

Inject cross-boundary species. The algorithm uses the density and temperature to calculate the frequency of injection of molecules of specie, *s* at the boundary. The injection frequency, *f_s* per unit area, *A* is

computed as

$$f_s = \frac{N}{\Delta t}$$

where Δt_s is the time interval at which a molecule hits the boundary area A , and N is the number of molecules in a volume with the base A and length Δx as shown in the figure: The time interval, Δt , between the collisions can be related to the component of velocity of the molecule of species s , in direction x , v_{sx} , as follows:

$$\Delta t = \frac{2\Delta x}{v_{sx}}$$

The number of molecules, N can be related to density, ρ , as:

$$N = \frac{\rho}{\mu} \Delta x A$$

where μ is the mass of one molecule and $A = \Delta y \Delta z$. Thus, the frequency is:

$$f_s = \frac{\rho}{\mu} \Delta x A \frac{v_{sx}}{2\Delta x} = \frac{\rho v_{sx}}{2\mu} A$$

get time-step in sec

Go over all 6 boundaries:

select a boundary

get boundary area in m²

get boundary temperature in K

create local gas-array

Go over all gases

point to current gas

select current gas-specie

Use kinetic theory of gases to estimate collision frequency with the wall and the number of molecules injected. This number can be fractional, so split it into integer and fractional part and inject the fractional part using probability.

get molecule's mass in atomic units

convert molecule's mass to kg

get density in kg/m³

estimate collision frequency per unit area

calculate the number of collisions with the boundary during the past time step

get the whole integer number of molecules injected

calculate the fractional part of a molecule injected

Inject all whole molecules:

Probabilistically inject the fractional molecules:

Definition at line 1260 of file domain.cc.

References Boundary::area, Geom::area(), AtomicMassUnit, BoltzmannConstant, boundaries, boundary(), Species::Gas::density, DIM, Boundary::gases, List< Element >::goFirst(), Boundary::Inject(), Species::Specie::Mass(), molecules, REAL, RND, Species::Gas::specie, Species::species, step(), Boundary::temperature, and Run::time.

Referenced by run().

13.6.3.7 Interaction Domain::interact (Molecule * a, Molecule * b)

Computes interaction between two molecules with the exchange of energy and momentum according to hard-ball collisions, and the possibility of chemical reactions. Also estimates time to the next collision. Initializing molecules according to bulk properties (restart=0) The velocity of each molecule is set from given temperature according to:

$$\frac{3}{2}kT = \frac{mV_{av}^2}{2}$$

Since the average velocity relates to the average velocity in one direction, x, as:

$$V_{av}^2 = 3V_{x,av}^2$$

we obtain for $V_{av,x} = kT$

Definition at line 1486 of file domain.cc.

References Species::Reaction::Outcome::ActivationEnergy(), Species::annihilated, Species::collided, Molecule::Coordinates(), Species::Specie::Cp(), DIM, DOF, DOFI, DOFK, Palette::dvi, Gui::dx, Species::Reaction::Outcome::Enthalpy(), Species::Reaction::First(), Species::Specie::Id(), Molecule::InternalEnergy(), Species::Specie::Mass(), Species::missed, Species::Reaction::Next(), Species::nspecies, Species::Reaction::Outcome::Probability(), Species::Reaction::Outcome::Product(), Species::reacted, Species::reactions, REAL, RND, SCLP, Species::Specie::Size(), SMALL, Species::species, Molecule::Type(), Molecule::Velocity(), and VOIDSPECIE.

Referenced by interaction().

13.6.3.8 void Domain::interaction (int icounter)

DDD: dirty fix, should be as below, but did not work

DDD: end dirty fix

DDD: dirty fix, should be as below, but did not work

DDD

DDD: dirty fix: some interactions may go unaccounted

DDD

Definition at line 1383 of file domain.cc.

References Species::annihilated, Container< Element >::Current(), DIM, Gui::dx, Container< Element >::First(), Container< Element >::FirstNext(), Container< Element >::goFirst(), Container< Element >::goNext(), GridContainer::index(), interact(), Container< Element >::isFirst(), Container< Element >::isFirstLast(), Container< Element >::isLast(), Species::missed, GridContainer::ncells, Gui::nodes, Run::nthreads, Container< Element >::number(), REAL, Container< Element >::setFirst(), Container< Element >::setFirstLast(), Run::time, Molecule::Type(), VOIDSPECIE, XMAX, and XMIN.

13.6.3.9 void Domain::interaction ()

Definition at line 1306 of file domain.cc.

References Species::annihilated, Container< Element >::Current(), DIM, Gui::dx, Container< Element >::First(), Container< Element >::FirstNext(), Container< Element >::goFirst(), Container< Element >::goNext(), GridContainer::index(), interact(), Container< Element >::isFirst(), Container< Element >::isFirstLast(), Container< Element >::isLast(), Species::missed, GridContainer::ncells, Gui::nodes, Container< Element >::number(), REAL, Container< Element >::remove(), Container< Element

>::setFirst(), Container< Element >::setFirstLast(), Molecule::Type(), VOIDSPECIE, XMAX, and XMIN.

Referenced by run().

13.6.3.10 void Domain::load (char *filename)

Initializing molecules according to bulk properties (restart=0).

Definition at line 687 of file domain.cc.

References Collection< Element >::append(), Molecule::Coordinate(), Potential::Cutoff(), DIM, Molecule::InternalEnergy(), IO::ioutput, LARGE, Potential::lengthscale(), MAXLINLEN, molecules, Species::nspecies, Collection< Element >::number(), Run::option, REAL, Species::Specie::Size(), Gui::size, Species::species, Run::time, Molecule::Type(), Molecule::Velocity(), and WORDLENGTH.

Referenced by Domain().

13.6.3.11 REAL Domain::maxBound (int i) [inline]

Definition at line 111 of file domain.h.

References bounds.

Referenced by Domain(), and Gui::readconf().

13.6.3.12 REAL Domain::minBound (int i) [inline]

Definition at line 110 of file domain.h.

References bounds.

Referenced by Domain(), and Gui::readconf().

13.6.3.13 Collection<Molecule>* Domain::Molecules () [inline]

Definition at line 115 of file domain.h.

References molecules.

Referenced by Gui::display(), and Gui::initdisp().

13.6.3.14 int Domain::run (int niter)

The [Domain](#) solver: running **niter** iterations.

The main loop over niter iterations or till the end of time:

Definition at line 849 of file domain.cc.

References AtomicMassUnit, BoltzmannConstant, boundary(), GridContainer::cellsize, Species::Specie::Cp(), Collection< Element >::Current(), DIM, distribution, DOF, DOFI, ESC, Collection< Element >::goFirst(), Collection< Element >::goNext(), injection(), interaction(), Molecule::InternalEnergy(), IO::ioutput, Collection< Element >::isFirst(), Molecule::KineticEnergy(), LARGE, Collection< Element >::maxCounters(), maxcounters, Collection< Element >::maxnumber(), molecules, Molecule::Move(), GridContainer::nodes, Species::nspecies, Run::nthreads, Collection< Element >::number(), onethird, Run::option, Run::outputname, GridContainer::put(), REAL, Container<

Element >::remove(), Collection< Element >::remove0(), save(), SMALL, Species::species, step(), Run::time, twothirds, Molecule::Type(), Molecule::Velocity(), and VOIDSPECIE.

Referenced by Gui::animate(), Gui::keyboard(), main(), and Gui::runmany().

13.6.3.15 void Domain::save (char * *taskname*)

Definition at line 1935 of file domain.cc.

References Molecule::Coordinate(), Collection< Element >::Current(), Run::elapsed(), Collection< Element >::goFirst(), Collection< Element >::goNext(), Species::Specie::Id(), Molecule::InternalEnergy(), IO::ioutput, Collection< Element >::isFirst(), MAXLINLEN, molecules, Collection< Element >::number(), Run::option, Species::species, Run::time, Molecule::Type(), Molecule::Velocity(), and VOIDSPECIE.

Referenced by Gui::keyboard(), main(), and run().

13.6.3.16 void Domain::setMaxBound (int *i*, REAL *x*) [inline]

Definition at line 109 of file domain.h.

References bounds.

Referenced by Domain(), and Gui::initdisp().

13.6.3.17 void Domain::setMinBound (int *i*, REAL *x*) [inline]

Definition at line 108 of file domain.h.

References bounds.

Referenced by Domain(), and Gui::initdisp().

13.6.3.18 void Domain::step (int *iter*)

Referenced by injection(), and run().

13.6.4 Member Data Documentation

13.6.4.1 Boundary Domain::boundaries[DIM][2] [private]

six boundaries of a hexahedral domain

Definition at line 88 of file domain.h.

Referenced by boundary(), BoundaryType(), Domain(), and injection().

13.6.4.2 char Domain::boundaryName[maxBoundaryTypes][WORDLENGTH] [private]

to be deprecated

Definition at line 86 of file domain.h.

Referenced by Domain().

13.6.4.3 REAL Domain::bounds[2][DIM] [private]

Definition at line 85 of file domain.h.

Referenced by boundary(), Domain(), maxBound(), minBound(), setMaxBound(), and setMinBound().

13.6.4.4 Bulk Domain::bulk [private]

bulk of the domain: initialized only if Run::option.restart=0

Definition at line 87 of file domain.h.

Referenced by Domain(), and init().

13.6.4.5 int* Domain::distribution [private]

molecule counts for all species

Definition at line 84 of file domain.h.

Referenced by Domain(), and run().

13.6.4.6 Collection<Molecule> Domain::molecules [private]

[Collection](#) of molecules.

Definition at line 89 of file domain.h.

Referenced by boundary(), computeBounds(), Domain(), init(), injection(), load(), Molecules(), run(), and save().

13.6.4.7 Pool<Molecule>* Domain::pool [private]

[Molecule](#) pointer pool.

Definition at line 90 of file domain.h.

Referenced by Domain().

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

- [domain.h](#)
- [domain.cc](#)

13.7 Gui::ElementDisp Struct Reference

```
#include <gui.h>
```

Public Attributes

- int [lighting](#)
- REAL [rgbcolor](#) [3]

13.7.1 Detailed Description

Definition at line 97 of file gui.h.

13.7.2 Member Data Documentation

13.7.2.1 int Gui::ElementDisp::lighting

Definition at line 99 of file gui.h.

13.7.2.2 REAL Gui::ElementDisp::rgbcolor[3]

Definition at line 101 of file gui.h.

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

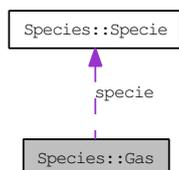
- [gui.h](#)

13.8 Species::Gas Struct Reference

[Gas](#) is a specie with some additional parameters.

```
#include <species.h>
```

Collaboration diagram for Species::Gas:



Public Member Functions

- [Gas](#) ([Specie](#) *s)

Public Attributes

- REAL [density](#)
density in kg/m³
- [Specie](#) * [specie](#)

13.8.1 Detailed Description

[Gas](#) is a specie with some additional parameters.

Definition at line 29 of file species.h.

13.8.2 Constructor & Destructor Documentation

13.8.2.1 Species::Gas::Gas (Specie * s) [inline]

Definition at line 32 of file species.h.

References [specie](#).

13.8.3 Member Data Documentation

13.8.3.1 REAL Species::Gas::density

density in kg/m³

Definition at line 30 of file species.h.

Referenced by [Domain::Domain\(\)](#), [Domain::init\(\)](#), and [Domain::injection\(\)](#).

13.8.3.2 Specie* Species::Gas::specie

Definition at line 31 of file species.h.

Referenced by Gas(), Domain::init(), and Domain::injection().

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

- [species.h](#)

13.9 Item< Element > Struct Template Reference

```
#include <collection.h>
```

Collaboration diagram for Item< Element >:



Public Attributes

- Element [element](#)
- Item * [next](#)
- Item * [prev](#)

13.9.1 Detailed Description

```
template<class Element> struct Item< Element >
```

Definition at line 20 of file collection.h.

13.9.2 Member Data Documentation

13.9.2.1 template<class Element > Element Item< Element >::element

Definition at line 21 of file collection.h.

Referenced by Collection< Element >::append(), Collection< Element >::insert(), Collection< Element >::insert0(), and Collection< Element >::insertAfter().

13.9.2.2 template<class Element > Item* Item< Element >::next

Definition at line 22 of file collection.h.

Referenced by Collection< Element >::append0(), Collection< Element >::insert0(), Collection< Element >::insertAfter(), and Collection< Element >::remove0().

13.9.2.3 template<class Element > Item * Item< Element >::prev

Definition at line 22 of file collection.h.

Referenced by Collection< Element >::append0(), Collection< Element >::insert0(), and Collection< Element >::insertAfter().

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

- [collection.h](#)

13.10 List< Element > Class Template Reference

```
#include <list.h>
```

Public Member Functions

- [List](#) ()
- [List](#) (int n)
- [~List](#) ()
- int [number](#) ()
- void [setFirst](#) ()
- void [setFirstLast](#) ()
- void [goFirst](#) ()
- void [goFirstNext](#) ()
- void [FirstNext](#) ()
- void [FirstPrev](#) ()
- bool [isFirstLast](#) ()
- void [goLastNext](#) ()
- void [goLast](#) ()
- void [goNext](#) ()
- void [goPrev](#) ()
- bool [isFirst](#) ()
- bool [isLast](#) ()
- void [LastNext](#) ()
- void [LastPrev](#) ()
- void [go](#) (Pointer< Element > *p)
- Element * [First](#) ()
- Element * [Last](#) ()
- Element * [Next](#) ()
- Element * [getNext](#) ()
- Element * [Prev](#) ()
- Element * [getPrev](#) ()
- Element * [Current](#) ()
- Pointer< Element > * [getPointer](#) ()
- Pointer< Element > * [getFirstPointer](#) ()
- Pointer< Element > * [getLastPointer](#) ()
- int [init](#) (int n)
- int [insert](#) (Element *element)
- int [insert](#) ()
- int [append](#) (Element *element)
- int [append](#) ()
- int [link](#) (Pointer< Element > *p)
- void [unlink](#) (Pointer< Element > *p)
- int [prepend](#) (Element *element)
- bool [locate](#) (Element *element)
- bool [locate](#) (Pointer< Element > *pointer)
- void [moveAfterFirst](#) ()
- void [swapAfterFirst](#) ()
- void [moveBehindFirst](#) ()

- int `erase` ()
- bool `erase` (Element *element)
- void `eraseall` ()
- bool `remove` ()
- void `cleanup` ()
- void `remove` (Pointer< Element > *pointer)

Private Attributes

- int `nelements`
- struct `Pointer`< Element > * `first`
- struct `Pointer`< Element > * `last`
- struct `Pointer`< Element > * `current`

13.10.1 Detailed Description

`template<class Element> class List< Element >`

Definition at line 24 of file list.h.

13.10.2 Constructor & Destructor Documentation

13.10.2.1 `template<class Element > List< Element >::List ()` [inline]

Definition at line 80 of file list.h.

References `List< Element >::current`, `List< Element >::first`, `List< Element >::last`, and `List< Element >::nelements`.

13.10.2.2 `template<class Element > List< Element >::List (int n)`

13.10.2.3 `template<class Element > List< Element >::~~List ()` [inline]

Definition at line 90 of file list.h.

References `List< Element >::cleanup()`.

13.10.3 Member Function Documentation

13.10.3.1 `template<class Element > int List< Element >::append ()` [inline]

Definition at line 159 of file list.h.

References `Pointer< Element >::element`.

Referenced by `List< Element >::init()`.

13.10.3.2 `template<class Element > int List< Element >::append (Element * element)`
[inline]

Definition at line 136 of file list.h.

References List< Element >::current, List< Element >::first, List< Element >::last, List< Element >::nelements, and Pointer< Element >::prev.

Referenced by Domain::Domain().

13.10.3.3 `template<class Element > void List< Element >::cleanup ()` [inline]

Definition at line 100 of file list.h.

References List< Element >::current, List< Element >::first, List< Element >::last, and List< Element >::nelements.

Referenced by List< Element >::~~List().

13.10.3.4 `template<class Element > Element* List< Element >::Current ()` [inline]

Definition at line 54 of file list.h.

Referenced by Domain::init().

13.10.3.5 `template<class Element > bool List< Element >::erase (Element * element)`
[inline]

Definition at line 287 of file list.h.

References List< Element >::erase(), and List< Element >::locate().

13.10.3.6 `template<class Element > int List< Element >::erase ()` [inline]

Definition at line 235 of file list.h.

References List< Element >::current, Pointer< Element >::element, List< Element >::first, List< Element >::last, and List< Element >::nelements.

Referenced by List< Element >::erase(), and List< Element >::eraseall().

13.10.3.7 `template<class Element > void List< Element >::eraseall ()` [inline]

Definition at line 94 of file list.h.

References List< Element >::current, List< Element >::erase(), List< Element >::first, List< Element >::last, and List< Element >::nelements.

13.10.3.8 `template<class Element > Element* List< Element >::First ()` [inline]

Definition at line 48 of file list.h.

13.10.3.9 `template<class Element > void List< Element >::FirstNext ()` [inline]

Definition at line 36 of file list.h.

13.10.3.10 `template<class Element > void List< Element >::FirstPrev ()` [inline]

Definition at line 37 of file list.h.

13.10.3.11 `template<class Element > Pointer<Element>* List< Element >::getFirstPointer ()`
[inline]

Definition at line 56 of file list.h.

13.10.3.12 `template<class Element > Pointer<Element>* List< Element >::getLastPointer ()`
[inline]

Definition at line 57 of file list.h.

13.10.3.13 `template<class Element > Element* List< Element >::getNext ()` [inline]

Definition at line 51 of file list.h.

13.10.3.14 `template<class Element > Pointer<Element>* List< Element >::getPointer ()`
[inline]

Definition at line 55 of file list.h.

13.10.3.15 `template<class Element > Element* List< Element >::getPrev ()` [inline]

Definition at line 53 of file list.h.

13.10.3.16 `template<class Element > void List< Element >::go (Pointer< Element > * p)`
[inline]

Definition at line 47 of file list.h.

13.10.3.17 `template<class Element > void List< Element >::goFirst ()` [inline]

Definition at line 34 of file list.h.

Referenced by Domain::init(), and Domain::injection().

13.10.3.18 `template<class Element > void List< Element >::goFirstNext ()` [inline]

Definition at line 35 of file list.h.

13.10.3.19 `template<class Element > void List< Element >::goLast () [inline]`

Definition at line 40 of file list.h.

13.10.3.20 `template<class Element > void List< Element >::goLastNext () [inline]`

Definition at line 39 of file list.h.

13.10.3.21 `template<class Element > void List< Element >::goNext () [inline]`

Definition at line 41 of file list.h.

Referenced by Domain::init().

13.10.3.22 `template<class Element > void List< Element >::goPrev () [inline]`

Definition at line 42 of file list.h.

13.10.3.23 `template<class Element > int List< Element >::init (int n) [inline]`

Definition at line 106 of file list.h.

References List< Element >::append(), and List< Element >::nelements.

13.10.3.24 `template<class Element > int List< Element >::insert () [inline]`

Definition at line 209 of file list.h.

References Pointer< Element >::element.

13.10.3.25 `template<class Element > int List< Element >::insert (Element * element) [inline]`

Definition at line 186 of file list.h.

References List< Element >::current, List< Element >::first, List< Element >::last, List< Element >::nelements, and Pointer< Element >::prev.

13.10.3.26 `template<class Element > bool List< Element >::isFirst () [inline]`

Definition at line 43 of file list.h.

Referenced by Domain::init().

13.10.3.27 `template<class Element > bool List< Element >::isFirstLast () [inline]`

Definition at line 38 of file list.h.

13.10.3.28 `template<class Element > bool List< Element >::isLast ()` [inline]

Definition at line 44 of file list.h.

13.10.3.29 `template<class Element > Element* List< Element >::Last ()` [inline]

Definition at line 49 of file list.h.

13.10.3.30 `template<class Element > void List< Element >::LastNext ()` [inline]

Definition at line 45 of file list.h.

13.10.3.31 `template<class Element > void List< Element >::LastPrev ()` [inline]

Definition at line 46 of file list.h.

13.10.3.32 `template<class Element > int List< Element >::link (Pointer< Element > * p)`
[inline]

Definition at line 165 of file list.h.

References `List< Element >::current`, `List< Element >::first`, `List< Element >::last`, `List< Element >::nelements`, and `Pointer< Element >::prev`.

13.10.3.33 `template<class Element > bool List< Element >::locate (Pointer< Element > * pointer)` [inline]

Definition at line 303 of file list.h.

References `List< Element >::current`, and `List< Element >::first`.

13.10.3.34 `template<class Element > bool List< Element >::locate (Element * element)`
[inline]

Definition at line 293 of file list.h.

References `List< Element >::current`, and `List< Element >::first`.

Referenced by `List< Element >::erase()`, and `List< Element >::remove()`.

13.10.3.35 `template<class Element > void List< Element >::moveAfterFirst ()` [inline]

Definition at line 320 of file list.h.

References `List< Element >::current`, `List< Element >::first`, and `Pointer< Element >::prev`.

13.10.3.36 `template<class Element > void List< Element >::moveBehindFirst ()` [inline]

Definition at line 331 of file list.h.

References `List< Element >::current`, `List< Element >::first`, and `Pointer< Element >::next`.

13.10.3.37 `template<class Element > Element* List< Element >::Next ()` [inline]

Definition at line 50 of file list.h.

13.10.3.38 `template<class Element > int List< Element >::number ()` [inline]

Definition at line 31 of file list.h.

Referenced by Domain::init().

13.10.3.39 `template<class Element > int List< Element >::prepend (Element * element)`
[inline]

Definition at line 113 of file list.h.

References List< Element >::current, List< Element >::first, List< Element >::last, List< Element >::nelements, and Pointer< Element >::next.

13.10.3.40 `template<class Element > Element* List< Element >::Prev ()` [inline]

Definition at line 52 of file list.h.

13.10.3.41 `template<class Element > void List< Element >::remove (Pointer< Element > *
pointer)` [inline]

DDD: slow but safe

DDD

Definition at line 272 of file list.h.

References List< Element >::current, and List< Element >::locate().

13.10.3.42 `template<class Element > bool List< Element >::remove ()` [inline]

Definition at line 215 of file list.h.

References List< Element >::current, List< Element >::first, List< Element >::last, and List< Element >::nelements.

13.10.3.43 `template<class Element > void List< Element >::setFirst ()` [inline]

Definition at line 32 of file list.h.

13.10.3.44 `template<class Element > void List< Element >::setFirstLast ()` [inline]

Definition at line 33 of file list.h.

13.10.3.45 `template<class Element > void List< Element >::swapAfterFirst ()` [inline]

Definition at line 313 of file list.h.

References `List< Element >::current`, and `List< Element >::first`.

13.10.3.46 `template<class Element > void List< Element >::unlink (Pointer< Element > *p)`
`[inline]`

Definition at line 257 of file `list.h`.

References `List< Element >::current`, `List< Element >::first`, `List< Element >::last`, `List< Element >::nelements`, `Pointer< Element >::next`, and `Pointer< Element >::prev`.

13.10.4 Member Data Documentation

13.10.4.1 `template<class Element > struct Pointer< Element > * List< Element >::current`
`[read, private]`

Definition at line 26 of file `list.h`.

Referenced by `List< Element >::append()`, `List< Element >::cleanup()`, `List< Gas >::Current()`, `List< Element >::erase()`, `List< Element >::eraseall()`, `List< Gas >::getNext()`, `List< Gas >::getPointer()`, `List< Gas >::getPrev()`, `List< Gas >::go()`, `List< Gas >::goFirst()`, `List< Gas >::goFirstNext()`, `List< Gas >::goLast()`, `List< Gas >::goLastNext()`, `List< Gas >::goNext()`, `List< Gas >::goPrev()`, `List< Element >::insert()`, `List< Gas >::isFirst()`, `List< Gas >::isLast()`, `List< Element >::link()`, `List< Element >::List()`, `List< Element >::locate()`, `List< Element >::moveAfterFirst()`, `List< Element >::moveBehindFirst()`, `List< Gas >::Next()`, `List< Element >::prepend()`, `List< Gas >::Prev()`, `List< Element >::remove()`, `List< Element >::swapAfterFirst()`, and `List< Element >::unlink()`.

13.10.4.2 `template<class Element > struct Pointer< Element > * List< Element >::first`
`[read, private]`

Definition at line 26 of file `list.h`.

Referenced by `List< Element >::append()`, `List< Element >::cleanup()`, `List< Element >::erase()`, `List< Element >::eraseall()`, `List< Gas >::First()`, `List< Gas >::FirstNext()`, `List< Gas >::FirstPrev()`, `List< Gas >::getFirstPointer()`, `List< Gas >::goFirst()`, `List< Gas >::goFirstNext()`, `List< Element >::insert()`, `List< Gas >::isFirst()`, `List< Gas >::isFirstLast()`, `List< Element >::link()`, `List< Element >::List()`, `List< Element >::locate()`, `List< Element >::moveAfterFirst()`, `List< Element >::moveBehindFirst()`, `List< Element >::prepend()`, `List< Element >::remove()`, `List< Gas >::setFirst()`, `List< Gas >::setFirstLast()`, `List< Element >::swapAfterFirst()`, and `List< Element >::unlink()`.

13.10.4.3 `template<class Element > struct Pointer< Element > * List< Element >::last`
`[read, private]`

Definition at line 26 of file `list.h`.

Referenced by `List< Element >::append()`, `List< Element >::cleanup()`, `List< Element >::erase()`, `List< Element >::eraseall()`, `List< Gas >::getLastPointer()`, `List< Gas >::goLast()`, `List< Gas >::goLastNext()`, `List< Element >::insert()`, `List< Gas >::isFirstLast()`, `List< Gas >::isLast()`, `List< Gas >::Last()`, `List< Gas >::LastNext()`, `List< Gas >::LastPrev()`, `List< Element >::link()`, `List< Element >::List()`, `List< Element >::prepend()`, `List< Element >::remove()`, `List< Gas >::setFirst()`, `List< Gas >::setFirstLast()`, and `List< Element >::unlink()`.

13.10.4.4 `template<class Element > int List< Element >::nelements` [private]

Definition at line 25 of file list.h.

Referenced by `List< Element >::append()`, `List< Element >::cleanup()`, `List< Element >::erase()`, `List< Element >::eraseall()`, `List< Element >::init()`, `List< Element >::insert()`, `List< Element >::link()`, `List< Element >::List()`, `List< Gas >::number()`, `List< Element >::prepend()`, `List< Element >::remove()`, and `List< Element >::unlink()`.

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

- [list.h](#)

13.11 Molecule Class Reference

```
#include <model.h>
```

Public Member Functions

- [Molecule](#) ()
- void [Type](#) (int t)
- int [Type](#) ()
- void [InternalEnergy](#) (REAL e)
- REAL [InternalEnergy](#) ()
- REAL [Coordinate](#) (int i)
- void [Coordinate](#) (int i, REAL y)
- REAL * [Coordinates](#) ()
- void [Coordinates](#) (REAL y[])
- void [setCoordinates](#) (REAL y[])
- REAL [Velocity](#) (int i)
- void [Velocity](#) (int i, REAL u)
- REAL * [Velocity](#) ()
- void [Velocity](#) (REAL u[])
- void [setVelocity](#) (REAL u[])
- REAL [KineticEnergy](#) ()
Retrieve kinetic energy.
- void [Temperature](#) (REAL temperature)
Set internal energy according to temperature:.
- void [Move](#) ()
Moving a molecule by one time step.
- void [copy](#) ([Molecule](#) *molecule)

Private Attributes

- int [type](#)
points to the species[type];
- REAL [energy](#)
internal energy
- REAL [x](#) [DIM]
position of center of mass
- REAL [v](#) [DIM]
velocity of center of mass

13.11.1 Detailed Description

Definition at line 23 of file model.h.

13.11.2 Constructor & Destructor Documentation

13.11.2.1 Molecule::Molecule ()

Definition at line 20 of file model.cc.

References energy, Run::time, and type.

13.11.3 Member Function Documentation

13.11.3.1 void Molecule::Coordinate (int *i*, REAL *y*) [inline]

Definition at line 103 of file model.h.

References DIM, and x.

13.11.3.2 REAL Molecule::Coordinate (int *i*) [inline]

Definition at line 95 of file model.h.

References DIM, and x.

Referenced by copy(), Bulk::inject(), Boundary::Inject(), Domain::load(), and Domain::save().

13.11.3.3 void Molecule::Coordinates (REAL *y*[]) [inline]

Definition at line 111 of file model.h.

References DIM, and x.

13.11.3.4 REAL* Molecule::Coordinates () [inline]

Definition at line 110 of file model.h.

References x.

Referenced by Domain::boundary(), Domain::computeBounds(), Gui::display(), Domain::interact(), and GridContainer::put().

13.11.3.5 void Molecule::copy (Molecule * *molecule*)

Definition at line 36 of file model.cc.

References Coordinate(), DIM, Type(), type, v, Velocity(), and x.

13.11.3.6 REAL Molecule::InternalEnergy () [inline]

Definition at line 86 of file model.h.

References energy.

Referenced by Temperature().

13.11.3.7 void Molecule::InternalEnergy (REAL *e*) [inline]

Definition at line 85 of file model.h.

References energy.

Referenced by Domain::boundary(), Domain::interact(), Domain::load(), Domain::run(), and Domain::save().

13.11.3.8 REAL Molecule::KineticEnergy ()

Retrieve kinetic energy.

Definition at line 46 of file model.cc.

References DIM, Species::Specie::Mass(), REAL, Species::species, type, and v.

Referenced by Domain::boundary(), and Domain::run().

13.11.3.9 void Molecule::Move ()

Moving a molecule by one time step.

Definition at line 68 of file model.cc.

References DIM, Species::Specie::Mass(), Species::nspecies, REAL, SMALL, Species::species, Run::time, type, v, and x.

Referenced by Domain::run().

13.11.3.10 void Molecule::setCoordinates (REAL *y*[]) [inline]

Definition at line 112 of file model.h.

References DIM, and x.

13.11.3.11 void Molecule::setVelocity (REAL *u*[]) [inline]

Definition at line 130 of file model.h.

References DIM, and v.

13.11.3.12 void Molecule::Temperature (REAL *temperature*)

Set internal energy according to temperature:.

Set internal energy according to the temperature.

Parameters:

temperature temperature in K

Definition at line 57 of file model.cc.

References AtomicMassUnit, BoltzmannConstant, Species::Specie::Cp(), DOF, DOFI, energy, InternalEnergy(), REAL, Species::species, and Type().

Referenced by Domain::init(), and Boundary::Inject().

13.11.3.13 int Molecule::Type () [inline]

Definition at line 50 of file model.h.

References type.

Referenced by Temperature().

13.11.3.14 void Molecule::Type (int *t*) [inline]

Definition at line 49 of file model.h.

References type.

Referenced by Domain::boundary(), copy(), Gui::display(), Bulk::inject(), Boundary::Inject(), Domain::interact(), Domain::interaction(), Domain::load(), GridContainer::put(), Domain::run(), and Domain::save().

13.11.3.15 void Molecule::Velocity (REAL *u*[]) [inline]

Definition at line 129 of file model.h.

References DIM, and v.

13.11.3.16 REAL* Molecule::Velocity () [inline]

Definition at line 128 of file model.h.

References v.

13.11.3.17 void Molecule::Velocity (int *i*, REAL *u*) [inline]

Definition at line 121 of file model.h.

References DIM, and v.

13.11.3.18 REAL Molecule::Velocity (int *i*) [inline]

Definition at line 113 of file model.h.

References DIM, and v.

Referenced by Domain::boundary(), copy(), Bulk::inject(), Boundary::Inject(), Domain::interact(), Domain::load(), GridContainer::put(), Domain::run(), and Domain::save().

13.11.4 Member Data Documentation

13.11.4.1 REAL Molecule::energy [private]

internal energy

Definition at line 40 of file model.h.

Referenced by InternalEnergy(), Molecule(), and Temperature().

13.11.4.2 int Molecule::type [private]

points to the species[type];

Definition at line 24 of file model.h.

Referenced by copy(), KineticEnergy(), Molecule(), Move(), and Type().

13.11.4.3 REAL Molecule::v[DIM] [private]

velocity of center of mass

Definition at line 40 of file model.h.

Referenced by copy(), KineticEnergy(), Move(), setVelocity(), and Velocity().

13.11.4.4 REAL Molecule::x[DIM] [private]

position of center of mass

Definition at line 40 of file model.h.

Referenced by Coordinate(), Coordinates(), copy(), Move(), and setCoordinates().

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

- [model.h](#)
- [model.cc](#)

13.12 Option Struct Reference

Execution options.

```
#include <run.h>
```

Public Attributes

- unsigned int `restart`:1
- unsigned int `verbose`:1
- unsigned int `debug`:1
run in debugging mode
- unsigned int `mesh`:1
used to start the mesh generator
- unsigned int `tags`:1
used in the trim module to trigger tag assignement
- unsigned int `xterm`:1
used in the trim module to trigger tag assignement

13.12.1 Detailed Description

Execution options.

Definition at line 3 of file run.h.

13.12.2 Member Data Documentation

13.12.2.1 unsigned int Option::debug

run in debugging mode

Definition at line 6 of file run.h.

Referenced by Run::init(), and Run::readcmdline().

13.12.2.2 unsigned int Option::mesh

used to start the mesh generator

Definition at line 7 of file run.h.

Referenced by Run::init(), and Run::readcmdline().

13.12.2.3 unsigned int Option::restart

Definition at line 4 of file run.h.

Referenced by Run::init(), and Run::readcmdline().

13.12.2.4 unsigned int Option::tags

used in the trim module to trigger tag assignement

Definition at line 8 of file run.h.

13.12.2.5 unsigned int Option::verbose

Definition at line 5 of file run.h.

Referenced by Run::init(), and Run::readcmdline().

13.12.2.6 unsigned int Option::xterm

used in the trim module to trigger tag assignement

Definition at line 9 of file run.h.

Referenced by Domain::Domain(), and Run::init().

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

- [run.h](#)

13.13 `Pointer< Element >` Struct Template Reference

```
#include <list.h>
```

Collaboration diagram for `Pointer< Element >`:



Public Attributes

- `Element * element`
- `Pointer * next`
- `Pointer * prev`

13.13.1 Detailed Description

`template<class Element> struct Pointer< Element >`

Definition at line 19 of file `list.h`.

13.13.2 Member Data Documentation

13.13.2.1 `template<class Element > Element* Pointer< Element >::element`

Definition at line 20 of file `list.h`.

Referenced by `List< Element >::append()`, `List< Element >::erase()`, and `List< Element >::insert()`.

13.13.2.2 `template<class Element > Pointer* Pointer< Element >::next`

Definition at line 21 of file `list.h`.

Referenced by `List< Element >::moveBehindFirst()`, `List< Element >::prepend()`, and `List< Element >::unlink()`.

13.13.2.3 `template<class Element > Pointer * Pointer< Element >::prev`

Definition at line 21 of file `list.h`.

Referenced by `List< Element >::append()`, `List< Element >::insert()`, `List< Element >::link()`, `List< Element >::moveAfterFirst()`, and `List< Element >::unlink()`.

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

- [list.h](#)

13.14 Pool< Element > Struct Template Reference

```
#include <collection.h>
```

Public Member Functions

- [Pool](#) (int *nptrs*)
- [~Pool](#) ()
- int [size](#) ()
- [Ptr](#)< Element > * [get](#) ()
- bool [put](#) ([Ptr](#)< Element > **ptr*)
- bool [check](#) (char **msg*)

Public Attributes

- int [mptrs](#)
- int [nptrs](#)
- [Ptr](#)< Element > * [hook](#)

13.14.1 Detailed Description

template<class Element> struct Pool< Element >

Definition at line 414 of file collection.h.

13.14.2 Constructor & Destructor Documentation

13.14.2.1 template<class Element > Pool< Element >::Pool (int *nptrs*) [inline]

Definition at line 425 of file collection.h.

References [Ptr< Element >::element](#), [Pool< Element >::hook](#), [Pool< Element >::mptrs](#), [Ptr< Element >::next](#), [Pool< Element >::nptrs](#), [Run::option](#), and [Ptr< Element >::prev](#).

13.14.2.2 template<class Element > Pool< Element >::~~Pool () [inline]

Definition at line 451 of file collection.h.

References [Pool< Element >::hook](#), [Pool< Element >::mptrs](#), and [Pool< Element >::nptrs](#).

13.14.3 Member Function Documentation

13.14.3.1 template<class Element > bool Pool< Element >::check (char * *msg*) [inline]

DDD

DDD

DDD

DDD

DDD

DDD

DDD

DDD

Definition at line 495 of file collection.h.

References Pool< Element >::hook, Pool< Element >::mptrs, Ptr< Element >::next, and Pool< Element >::nptrs.

Referenced by Container< Element >::checkPool().

13.14.3.2 `template<class Element > Ptr< Element > * Pool< Element >::get ()` [inline]

DDD: probably not needed, but just to be on the safe side

Definition at line 457 of file collection.h.

References Pool< Element >::hook, Ptr< Element >::next, Pool< Element >::nptrs, and Ptr< Element >::prev.

Referenced by Container< Element >::append0(), and Container< Element >::insert0().

13.14.3.3 `template<class Element > bool Pool< Element >::put (Ptr< Element > * ptr)` [inline]

Definition at line 471 of file collection.h.

References Pool< Element >::hook, Pool< Element >::mptrs, Ptr< Element >::next, Pool< Element >::nptrs, and Ptr< Element >::prev.

Referenced by Container< Element >::remove0().

13.14.3.4 `template<class Element > int Pool< Element >::size ()` [inline]

Definition at line 419 of file collection.h.

13.14.4 Member Data Documentation

13.14.4.1 `template<class Element > Ptr<Element>* Pool< Element >::hook`

Definition at line 416 of file collection.h.

Referenced by Pool< Element >::check(), Container< Element >::checkPool(), Pool< Element >::get(), Pool< Element >::Pool(), Pool< Element >::put(), and Pool< Element >::~~Pool().

13.14.4.2 `template<class Element > int Pool< Element >::mptrs`

Definition at line 415 of file collection.h.

Referenced by Pool< Element >::check(), Pool< Element >::Pool(), Pool< Element >::put(), and Pool< Element >::~~Pool().

13.14.4.3 `template<class Element > int Pool< Element >::nptrs`

Definition at line 415 of file `collection.h`.

Referenced by `Pool< Element >::check()`, `Pool< Element >::get()`, `Pool< Element >::Pool()`, `Pool< Element >::put()`, `Pool< Molecule >::size()`, and `Pool< Element >::~~Pool()`.

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

- [collection.h](#)

13.15 Ptr< Element > Struct Template Reference

```
#include <collection.h>
```

Collaboration diagram for Ptr< Element >:



Public Attributes

- Element * [element](#)
- Ptr * [next](#)
- Ptr * [prev](#)

13.15.1 Detailed Description

template<class Element> struct Ptr< Element >

Definition at line 409 of file collection.h.

13.15.2 Member Data Documentation

13.15.2.1 template<class Element > Element* Ptr< Element >::element

Definition at line 410 of file collection.h.

Referenced by Pool< Element >::Pool().

13.15.2.2 template<class Element > Ptr* Ptr< Element >::next

Definition at line 411 of file collection.h.

Referenced by Pool< Element >::check(), Pool< Element >::get(), Container< Element >::insert0(), Container< Element >::link0(), Pool< Element >::Pool(), Pool< Element >::put(), and Container< Element >::unlink0().

13.15.2.3 template<class Element > Ptr * Ptr< Element >::prev

Definition at line 411 of file collection.h.

Referenced by Container< Element >::append0(), Pool< Element >::get(), Container< Element >::insert0(), Container< Element >::link0(), Pool< Element >::Pool(), Pool< Element >::put(), and Container< Element >::unlink0().

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

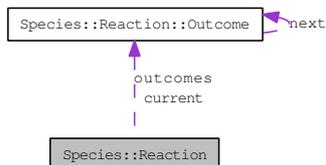
- [collection.h](#)

13.16 Species::Reaction Struct Reference

[Reaction](#) determines the two products only.

```
#include <species.h>
```

Collaboration diagram for Species::Reaction:



Classes

- struct [Outcome](#)
Possible outcomes of a reaction.

Public Member Functions

- [Reaction](#) ()
- [~Reaction](#) ()
- void [Erase](#) ([Outcome](#) *outcome)
- void [Add](#) (int ip0, int ip1, REAL a, REAL p, REAL h)
- [Outcome](#) * [First](#) ()
- [Outcome](#) * [Next](#) ()

Public Attributes

- struct [Species::Reaction::Outcome](#) * [outcomes](#)
Possible outcomes of a reaction.
- struct [Species::Reaction::Outcome](#) * [current](#)
all outcomes, and the current outcome

13.16.1 Detailed Description

[Reaction](#) determines the two products only.

Definition at line 35 of file species.h.

13.16.2 Constructor & Destructor Documentation

13.16.2.1 Species::Reaction::Reaction () [inline]

Definition at line 126 of file species.h.

References current, and outcomes.

13.16.2.2 Species::Reaction::~~Reaction () [inline]

Definition at line 127 of file species.h.

References Erase(), and outcomes.

13.16.3 Member Function Documentation

13.16.3.1 void Species::Reaction::Add (int *ip0*, int *ip1*, REAL *a*, REAL *p*, REAL *h*) [inline]

Add reaction products, probability, and enthalpy:

Definition at line 135 of file species.h.

References Species::Reaction::Outcome::next, and outcomes.

Referenced by Domain::Domain().

13.16.3.2 void Species::Reaction::Erase (Outcome * *outcome*) [inline]

Definition at line 131 of file species.h.

References Species::Reaction::Outcome::next.

Referenced by ~Reaction().

13.16.3.3 Outcome* Species::Reaction::First () [inline]

Definition at line 157 of file species.h.

References current, and outcomes.

Referenced by Domain::boundary(), Domain::Domain(), and Domain::interact().

13.16.3.4 Outcome* Species::Reaction::Next () [inline]

Definition at line 158 of file species.h.

References current, Species::Reaction::Outcome::next, and outcomes.

Referenced by Domain::boundary(), Domain::Domain(), and Domain::interact().

13.16.4 Member Data Documentation

13.16.4.1 struct Species::Reaction::Outcome* Species::Reaction::current

all outcomes, and the current outcome

Referenced by First(), Next(), and Reaction().

13.16.4.2 struct Species::Reaction::Outcome * Species::Reaction::outcomes

Possible outcomes of a reaction.

Referenced by Add(), First(), Next(), Reaction(), and ~Reaction().

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

- [species.h](#)

13.17 Species::Reaction::Outcome Struct Reference

Possible outcomes of a reaction.

```
#include <species.h>
```

Collaboration diagram for Species::Reaction::Outcome:



Public Member Functions

- [Outcome](#) ()
- [Outcome](#) (int ip0, int ip1, REAL a, REAL p, REAL h)
- void [Products](#) (int ip0, int ip1)
- int [Product](#) (int i)
- void [Time](#) (REAL t)
- REAL [Time](#) ()
- void [ActivationEnergy](#) (REAL a)
- REAL [ActivationEnergy](#) ()
- void [Probability](#) (REAL r)
- REAL [Probability](#) ()
- void [Enthalpy](#) (REAL h)
- REAL [Enthalpy](#) ()

Public Attributes

- int [product](#) [2]
two indexes of the products of a reaction
- REAL [activationEnergy](#)
- REAL [probability](#)
- REAL [enthalpy](#)
- REAL [time](#)
- [Outcome](#) * [next](#)
linked list of outcomes

13.17.1 Detailed Description

Possible outcomes of a reaction.

Definition at line 37 of file species.h.

13.17.2 Constructor & Destructor Documentation

13.17.2.1 Species::Reaction::Outcome::Outcome () [inline]

Definition at line 44 of file species.h.

References [enthalpy](#), [next](#), [probability](#), [product](#), [time](#), and [VOIDSPECIE](#).

13.17.2.2 **Species::Reaction::Outcome::Outcome** (int *ip0*, int *ip1*, REAL *a*, REAL *p*, REAL *h*) [inline]

Parameters:

- ip0* index of the first reaction product
- ip1* index of the second reaction product
- a* activation energy
- p* outcome probability
- h* enthalpy of reaction

Definition at line 49 of file species.h.

References activationEnergy, enthalpy, next, Species::nspecies, probability, product, and time.

13.17.3 Member Function Documentation

13.17.3.1 **REAL Species::Reaction::Outcome::ActivationEnergy** () [inline]

Definition at line 120 of file species.h.

References activationEnergy.

13.17.3.2 **void Species::Reaction::Outcome::ActivationEnergy** (REAL *a*) [inline]

Definition at line 119 of file species.h.

References activationEnergy.

Referenced by Domain::interact().

13.17.3.3 **REAL Species::Reaction::Outcome::Enthalpy** () [inline]

Definition at line 124 of file species.h.

References enthalpy.

13.17.3.4 **void Species::Reaction::Outcome::Enthalpy** (REAL *h*) [inline]

Definition at line 123 of file species.h.

References enthalpy.

Referenced by Domain::boundary(), Domain::Domain(), and Domain::interact().

13.17.3.5 **REAL Species::Reaction::Outcome::Probability** () [inline]

Definition at line 122 of file species.h.

References probability.

13.17.3.6 void Species::Reaction::Outcome::Probability (REAL *r*) [inline]

Definition at line 121 of file species.h.

References probability.

Referenced by Domain::boundary(), Domain::Domain(), and Domain::interact().

13.17.3.7 int Species::Reaction::Outcome::Product (int *i*) [inline]

Definition at line 110 of file species.h.

References product.

Referenced by Domain::boundary(), Domain::Domain(), and Domain::interact().

13.17.3.8 void Species::Reaction::Outcome::Products (int *ip0*, int *ip1*) [inline]

Definition at line 98 of file species.h.

References product.

13.17.3.9 REAL Species::Reaction::Outcome::Time () [inline]

Definition at line 118 of file species.h.

References time.

13.17.3.10 void Species::Reaction::Outcome::Time (REAL *t*) [inline]

Definition at line 117 of file species.h.

References time.

Referenced by Domain::boundary().

13.17.4 Member Data Documentation**13.17.4.1 REAL Species::Reaction::Outcome::activationEnergy**

Definition at line 39 of file species.h.

Referenced by ActivationEnergy(), and Outcome().

13.17.4.2 REAL Species::Reaction::Outcome::enthalpy

Definition at line 39 of file species.h.

Referenced by Enthalpy(), and Outcome().

13.17.4.3 Outcome* Species::Reaction::Outcome::next

linked list of outcomes

Definition at line 43 of file species.h.

Referenced by Species::Reaction::Add(), Species::Reaction::Erase(), Species::Reaction::Next(), and Outcome().

13.17.4.4 REAL Species::Reaction::Outcome::probability

Definition at line 39 of file species.h.

Referenced by Outcome(), and Probability().

13.17.4.5 int Species::Reaction::Outcome::product[2]

two indexes of the products of a reaction

Definition at line 38 of file species.h.

Referenced by Outcome(), Product(), and Products().

13.17.4.6 REAL Species::Reaction::Outcome::time

time of reaction: used primarily for surface reactions, where there is a time delay between species adsorption on the surface and product formation.

Definition at line 39 of file species.h.

Referenced by Outcome(), and Time().

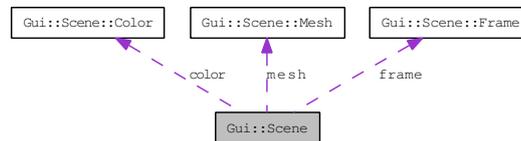
The documentation for this struct was generated from the following file:

- [species.h](#)

13.18 Gui::Scene Struct Reference

```
#include <gui.h>
```

Collaboration diagram for Gui::Scene:



Classes

- struct [Color](#)
- struct [Frame](#)
- struct [Mesh](#)

Public Attributes

- struct `Gui::Scene::Color` [color](#)
- struct `Gui::Scene::Mesh` [mesh](#)
- struct `Gui::Scene::Frame` [frame](#)

13.18.1 Detailed Description

Definition at line 107 of file `gui.h`.

13.18.2 Member Data Documentation

13.18.2.1 struct `Gui::Scene::Color` `Gui::Scene::color`

Referenced by `Gui::display()`, `Gui::init()`, `Gui::readconf()`, and `Gui::switchColorScheme()`.

13.18.2.2 struct `Gui::Scene::Frame` `Gui::Scene::frame`

Referenced by `Gui::display()`, `Gui::initdisp()`, and `Gui::readconf()`.

13.18.2.3 struct `Gui::Scene::Mesh` `Gui::Scene::mesh`

Referenced by `Gui::display()`, `Gui::initdisp()`, and `Gui::readconf()`.

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

- [gui.h](#)

13.19 Gui::Scene::Color Struct Reference

```
#include <gui.h>
```

Public Attributes

- [ColorSchemes scheme](#)
- REAL [minvalue](#)
- REAL [maxvalue](#)

13.19.1 Detailed Description

Definition at line 108 of file `gui.h`.

13.19.2 Member Data Documentation

13.19.2.1 REAL Gui::Scene::Color::maxvalue

Definition at line 110 of file `gui.h`.

Referenced by `Gui::display()`, and `Gui::readconf()`.

13.19.2.2 REAL Gui::Scene::Color::minvalue

Definition at line 110 of file `gui.h`.

Referenced by `Gui::display()`, and `Gui::readconf()`.

13.19.2.3 ColorSchemes Gui::Scene::Color::scheme

Definition at line 109 of file `gui.h`.

Referenced by `Gui::display()`, `Gui::init()`, `Gui::readconf()`, and `Gui::switchColorScheme()`.

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

- [gui.h](#)

13.20 Gui::Scene::Frame Struct Reference

```
#include <gui.h>
```

Public Attributes

- REAL [line](#) [maxlineprm]

13.20.1 Detailed Description

Definition at line 117 of file gui.h.

13.20.2 Member Data Documentation

13.20.2.1 REAL Gui::Scene::Frame::line[maxlineprm]

Definition at line 119 of file gui.h.

Referenced by [Gui::display\(\)](#), [Gui::initdisp\(\)](#), and [Gui::readconf\(\)](#).

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

- [gui.h](#)

13.21 Gui::Scene::Mesh Struct Reference

```
#include <gui.h>
```

Public Attributes

- REAL [node](#) [maxpointprm]
- REAL [line](#) [maxlineprm]

13.21.1 Detailed Description

Definition at line 112 of file gui.h.

13.21.2 Member Data Documentation

13.21.2.1 REAL Gui::Scene::Mesh::line[maxlineprm]

Definition at line 114 of file gui.h.

Referenced by Gui::initdisp(), and Gui::readconf().

13.21.2.2 REAL Gui::Scene::Mesh::node[maxpointprm]

Definition at line 114 of file gui.h.

Referenced by Gui::display(), Gui::initdisp(), and Gui::readconf().

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

- [gui.h](#)

13.22 Species::Specie Class Reference

```
#include <species.h>
```

Public Member Functions

- [Specie](#) ()
- void [Cp](#) (REAL c)
- REAL [Cp](#) ()
- char * [Id](#) ()
- void [Mass](#) (REAL m)
- REAL [Mass](#) ()
- REAL [Size](#) ()
- void [Size](#) (REAL s)

Private Attributes

- char [id](#) [WORDLENGTH]
- REAL [mass](#)
- REAL [size](#)
- REAL [cp](#)
- REAL [inertia](#) [DIM]

13.22.1 Detailed Description

Definition at line 6 of file species.h.

13.22.2 Constructor & Destructor Documentation

13.22.2.1 Species::Specie::Specie () [inline]

Definition at line 13 of file species.h.

References [cp](#), [DIM](#), [inertia](#), [mass](#), and [size](#).

13.22.3 Member Function Documentation

13.22.3.1 REAL Species::Specie::Cp () [inline]

Definition at line 21 of file species.h.

References [cp](#).

13.22.3.2 void Species::Specie::Cp (REAL c) [inline]

Definition at line 20 of file species.h.

References [cp](#).

Referenced by [Domain::boundary\(\)](#), [Domain::Domain\(\)](#), [Domain::interact\(\)](#), [Domain::run\(\)](#), and [Molecule::Temperature\(\)](#).

13.22.3.3 char* Species::Specie::Id () [inline]

Definition at line 22 of file species.h.

References id.

Referenced by Domain::Domain(), Domain::init(), Domain::interact(), and Domain::save().

13.22.3.4 REAL Species::Specie::Mass () [inline]

Definition at line 24 of file species.h.

References mass.

13.22.3.5 void Species::Specie::Mass (REAL m) [inline]

Definition at line 23 of file species.h.

References mass.

Referenced by Domain::boundary(), Gui::display(), Domain::Domain(), Domain::init(), Domain::injection(), Domain::interact(), Molecule::KineticEnergy(), and Molecule::Move().

13.22.3.6 void Species::Specie::Size (REAL s) [inline]

Definition at line 26 of file species.h.

References size.

13.22.3.7 REAL Species::Specie::Size () [inline]

Definition at line 25 of file species.h.

References size.

Referenced by Domain::boundary(), Gui::display(), Domain::Domain(), Bulk::inject(), Boundary::Inject(), Domain::interact(), and Domain::load().

13.22.4 Member Data Documentation**13.22.4.1 REAL Species::Specie::cp [private]**

Definition at line 8 of file species.h.

Referenced by Cp(), and Specie().

13.22.4.2 char Species::Specie::id[WORDLENGTH] [private]

Definition at line 7 of file species.h.

Referenced by Id().

13.22.4.3 REAL Species::Specie::inertia[DIM] [private]

Definition at line 8 of file species.h.

Referenced by Specie().

13.22.4.4 REAL Species::Specie::mass [private]

Definition at line 8 of file species.h.

Referenced by Mass(), and Specie().

13.22.4.5 REAL Species::Specie::size [private]

Definition at line 8 of file species.h.

Referenced by Size(), and Specie().

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

- [species.h](#)

13.23 State Union Reference

```
#include <model.h>
```

Public Attributes

- unsigned int [locked](#): 1
- unsigned int [done](#): 1

13.23.1 Detailed Description

Definition at line 16 of file model.h.

13.23.2 Member Data Documentation

13.23.2.1 unsigned int State::done

Definition at line 21 of file model.h.

13.23.2.2 unsigned int State::locked

Definition at line 20 of file model.h.

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

- [model.h](#)

13.24 Run::Time Struct Reference

Execution time-control structure.

```
#include <run.h>
```

Public Attributes

- double [start](#)
start-time
- double [end](#)
end-time
- double [prev](#)
previous time
- double [current](#)
current time
- double [step](#)
time-step
- double [nextstep](#)
next time-step
- double [step0](#)
previous time-step
- double [output](#)
data dump time interval

13.24.1 Detailed Description

Execution time-control structure.

Definition at line 25 of file run.h.

13.24.2 Member Data Documentation

13.24.2.1 double Run::Time::current

current time

Definition at line 27 of file run.h.

13.24.2.2 double Run::Time::end

end-time

Definition at line 27 of file run.h.

13.24.2.3 double Run::Time::nextstep

next time-step

Definition at line 27 of file run.h.

13.24.2.4 double Run::Time::output

data dump time interval

Definition at line 27 of file run.h.

13.24.2.5 double Run::Time::prev

previous time

Definition at line 27 of file run.h.

13.24.2.6 double Run::Time::start

start-time

Definition at line 27 of file run.h.

13.24.2.7 double Run::Time::step

time-step

Definition at line 27 of file run.h.

13.24.2.8 double Run::Time::step0

previous time-step

Definition at line 27 of file run.h.

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

- [run.h](#)

13.25 Gui::WindowGeom Struct Reference

```
#include <gui.h>
```

Public Attributes

- int [width](#)
- int [height](#)

13.25.1 Detailed Description

Definition at line 169 of file [gui.h](#).

13.25.2 Member Data Documentation

13.25.2.1 int Gui::WindowGeom::height

Definition at line 172 of file [gui.h](#).

Referenced by [Gui::init\(\)](#), and [Gui::initdisp\(\)](#).

13.25.2.2 int Gui::WindowGeom::width

Definition at line 172 of file [gui.h](#).

Referenced by [Gui::init\(\)](#), and [Gui::initdisp\(\)](#).

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

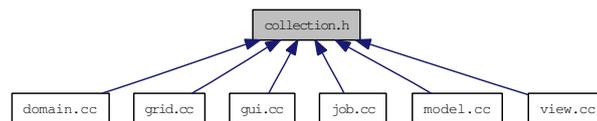
- [gui.h](#)

Chapter 14

File Documentation

14.1 collection.h File Reference

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



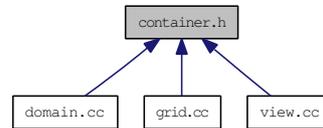
Classes

- struct [Item< Element >](#)
- class [Collection< Element >](#)
- struct [Ptr< Element >](#)
- struct [Pool< Element >](#)
- class [Container< Element >](#)

14.2 configfile.dox File Reference

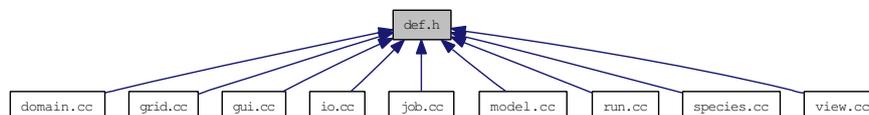
14.3 container.h File Reference

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



14.4 def.h File Reference

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



Defines

- #define ABOUT "MOLECULAR DYNAMICS SIMULATOR BY A.V.SMIRNOV\nandrei.v.smirnov@gmail.com\n"
- #define DOCTYPE "remody"
- #define REAL double
- #define TINY 1e-10
- #define SMALL 1e-30
- #define LARGE 1e30
- #define MAXLINLEN 510
- #define WORDLENGTH 64
- #define ESC 0x1B
- #define PI 3.14159265358979323846
- #define SQRTPIo8 0.62666
- #define ERROR(message) { fprintf(stderr, "\nERROR: %s\n", message); exit(1); }
- #define TRUE 1
- #define FALSE 0
- #define OMP
- #define WAIT for(int i=0; i<999999&&locked; i++)
- #define LOCAL
- #define COLLISIONTIME
- #define RND (REAL)((double)rand()/RAND_MAX)
- #define MAX(a, b) ((a)>(b)?a:b)
- #define ZERO(n, x) for (int i=0; i<(n); i++)x[i]=0.0
- #define ADD(n, x, y) for (int i=0; i<(n); i++) (x)[i] += (y)[i]
- #define SUB(n, x, y) for (int i=0; i<(n); i++) (x)[i] -= (y)[i]
- #define ADDC(n, x, y, d) for (int i=0; i<(n); i++) (x)[i] += (y)[i]*(d)
- #define SUBC(n, x, y, d) for (int i=0; i<(n); i++) (x)[i] -= (y)[i]*(d)
- #define MUL(n, x, y) for (int i=0; i<(n); i++) (x)[i] *= (y)[i]
- #define MULC(n, x, c) for (int i=0; i<(n); i++) (x)[i] *= (c)
- #define DIV(n, x, y) for (int i=0; i<(n); i++) (x)[i] /= (y)[i]
- #define MULVEC(x, s) for (int i=0; i<DIM; i++) (x)[i] *= (s)
- #define ADDVEC(x, y) for (int i=0; i<DIM; i++) (x)[i] += (y)[i]
- #define COPYVEC(x, y) for (int i=0; i<DIM; i++) (x)[i] = (y)[i]
- #define ZEROVEC(x) for (int i=0; i<DIM; i++) (x)[i] = 0.
- #define SCLP(A, B) (A[0]*B[0]+A[1]*B[1]+A[2]*B[2])
- #define VECP(A, B, C)
- #define LENGTH(A) sqrt(SCLP(A,A))
- #define VOIDSPECIE nspecies
- #define GAUSS Run::gauss()

- #define `DIM` 3
- #define `BoltzmannConstant` 1.38066e-23
- #define `AtomicMassUnit` 1.66053886e-27
- #define `AvogadroNumber` 6.02214179e23
Avogadro Number in 1/mol.
- #define `DOFK` 3
- #define `DOF(cp)` MAX(2.0*((cp)*`GasConstantInv`-1.0),DOFK)
- #define `DOFI(dof)` (dof-DOFK)

Variables

- const int `maxcounters` = 4
- const REAL `onethird` = 1.0/3.0
- const REAL `twothirds` = 2.0/3.0
- const REAL `Sqrt2` = sqrt(2.0)
- const REAL `Sqrt2Over2` = 0.5*Sqrt2
- const REAL `sqrt1p5` = sqrt(1.5)
- const REAL `sqrt1p25` = sqrt(1.25)
- const REAL `Sqrt3` = sqrt(3.0)
- const REAL `OneOverSqrt2` = 1.0/Sqrt2
- const REAL `OneOverSqrt3` = 1.0/Sqrt3
- const REAL `Sqrt3Over2` = 0.5*Sqrt3
- const REAL `GasConstant` = BoltzmannConstant*AvogadroNumber
- const REAL `GasConstantInv` = 1.0/GasConstant

14.4.1 Define Documentation

14.4.1.1 #define ABOUT "MOLECULAR DYNAMICS SIMULATOR BY A.V.SMIRNOV\nnandrei.v.smirnov@gmail.com\n"

Definition at line 1 of file def.h.

Referenced by Gui::menu().

14.4.1.2 #define ADD(n, x, y) for (int i=0; i<(n); i++) (x)[i] += (y)[i]

Definition at line 46 of file def.h.

14.4.1.3 #define ADDC(n, x, y, d) for (int i=0; i<(n); i++) (x)[i] += (y)[i]*(d)

Definition at line 48 of file def.h.

14.4.1.4 #define ADDVEC(x, y) for (int i=0; i<DIM; i++) (x)[i] += (y)[i]

Definition at line 54 of file def.h.

14.4.1.5 #define AtomicMassUnit 1.66053886e-27

Definition at line 68 of file def.h.

Referenced by Domain::Domain(), Domain::init(), Domain::injection(), Domain::run(), and Molecule::Temperature().

14.4.1.6 #define AvogadroNumber 6.02214179e23

Avogadro Number in 1/mol.

Definition at line 69 of file def.h.

Referenced by Domain::Domain().

14.4.1.7 #define BoltzmannConstant 1.38066e-23

Definition at line 67 of file def.h.

Referenced by Domain::Domain(), Domain::init(), Domain::injection(), Domain::run(), and Molecule::Temperature().

14.4.1.8 #define COLLISIONTIME

Definition at line 38 of file def.h.

14.4.1.9 #define COPYVEC(x, y) for (int i=0; i<DIM; i++) (x)[i] = (y)[i]

Definition at line 55 of file def.h.

14.4.1.10 #define DIM 3

Definition at line 66 of file def.h.

Referenced by Geom::area(), Domain::boundary(), Domain::BoundaryType(), Domain::computeBounds(), Molecule::Coordinate(), Molecule::Coordinates(), Molecule::copy(), Geom::distance(), Geom::distvec(), Domain::Domain(), GridContainer::index(), GridContainer::init(), Bulk::init(), Gui::initdisp(), Bulk::inject(), Boundary::Inject(), Domain::injection(), Domain::interact(), Domain::interaction(), Molecule::KineticEnergy(), Geom::length(), Domain::load(), Molecule::Move(), Geom::normalize(), GridContainer::put(), Gui::readconf(), Domain::run(), Geom::sclp(), Molecule::setCoordinates(), Molecule::setVelocity(), Species::Specie::Specie(), and Molecule::Velocity().

14.4.1.11 #define DIV(n, x, y) for (int i=0; i<(n); i++) (x)[i] /= (y)[i]

Definition at line 52 of file def.h.

14.4.1.12 #define DOCTYPE "remody"

Definition at line 3 of file def.h.

Referenced by Domain::Domain(), IO::getIter(), IO::getTimeXML(), and Gui::readconf().

14.4.1.13 #define DOF(cp) MAX(2.0*((cp)*GasConstantInv-1.0),DOFK)

Definition at line 77 of file def.h.

Referenced by Domain::boundary(), Domain::interact(), Domain::run(), and Molecule::Temperature().

14.4.1.14 #define DOFI(dof) (dof-DOFK)

Definition at line 78 of file def.h.

Referenced by Domain::boundary(), Domain::interact(), Domain::run(), and Molecule::Temperature().

14.4.1.15 #define DOFK 3

Definition at line 75 of file def.h.

Referenced by Domain::boundary(), and Domain::interact().

14.4.1.16 #define ERROR(message) { fprintf(stderr, "\nERROR: %s\n", message); exit(1); }

Definition at line 13 of file def.h.

Referenced by Gui::init().

14.4.1.17 #define ESC 0x1B

Definition at line 10 of file def.h.

Referenced by Gui::animate(), Gui::keyboard(), main(), and Domain::run().

14.4.1.18 #define FALSE 0

Definition at line 16 of file def.h.

14.4.1.19 #define GAUSS Run::gauss()

Definition at line 63 of file def.h.

Referenced by Bulk::inject(), and Boundary::Inject().

14.4.1.20 #define LARGE 1e30

Definition at line 7 of file def.h.

Referenced by Domain::computeBounds(), Gui::initdisp(), Domain::load(), and Domain::run().

14.4.1.21 #define LENGTH(A) sqrt(SCLP(A,A))

Definition at line 61 of file def.h.

Referenced by Geom::area(), and Domain::boundary().

14.4.1.22 #define LOCAL

Definition at line 37 of file def.h.

14.4.1.23 #define MAX(a, b) ((a)>(b)?a:b)

Definition at line 42 of file def.h.

14.4.1.24 #define MAXLINLEN 510

Definition at line 8 of file def.h.

Referenced by Gui::commandMode(), Gui::consoleMenu(), Domain::Domain(), Gui::Exit(), IO::getCharAttr(), IO::getIntAttr(), IO::getIter(), IO::getTimeXML(), Gui::keyboard(), Domain::load(), IO::parseWord(), Gui::Quit(), Gui::readconf(), Domain::save(), and IO::xwd().

14.4.1.25 #define MUL(n, x, y) for (int i=0; i<(n); i++) (x)[i] *= (y)[i]

Definition at line 50 of file def.h.

14.4.1.26 #define MULC(n, x, c) for (int i=0; i<(n); i++) (x)[i] *= (c)

Definition at line 51 of file def.h.

Referenced by Domain::boundary().

14.4.1.27 #define MULVEC(x, s) for (int i=0; i<DIM; i++) (x)[i] *= (s)

Definition at line 53 of file def.h.

14.4.1.28 #define OMP

Definition at line 19 of file def.h.

14.4.1.29 #define PI 3.14159265358979323846

Definition at line 11 of file def.h.

Referenced by Gui::showParticle(), and Gui::showSphere().

14.4.1.30 #define REAL double

Definition at line 4 of file def.h.

Referenced by Geom::area(), Domain::boundary(), Domain::computeBounds(), Gui::display(), Geom::distance(), Domain::Domain(), Run::gauss(), GridContainer::init(), Gui::initdisp(), Bulk::inject(), Boundary::Inject(), Domain::injection(), Domain::interact(), Domain::interaction(), Molecule::KineticEnergy(), Geom::length(), Domain::load(), Gui::motion(), Molecule::Move(), Geom::normalize(), IO::parseFloat(), Palette::pickcolor(), GridContainer::put(), Gui::readconf(),

Run::rnd(), Domain::run(), Geom::sclp(), Gui::showParticle(), Gui::showSphere(), Molecule::Temperature(), Run::testrnd(), and Potential::value().

14.4.1.31 **#define RND (REAL)((double)rand()/RAND_MAX)**

Definition at line 40 of file def.h.

Referenced by Domain::boundary(), Bulk::inject(), Boundary::Inject(), Domain::injection(), and Domain::interact().

14.4.1.32 **#define SCLP(A, B) (A[0]*B[0]+A[1]*B[1]+A[2]*B[2])**

Definition at line 57 of file def.h.

Referenced by Domain::interact().

14.4.1.33 **#define SMALL 1e-30**

Definition at line 6 of file def.h.

Referenced by Domain::boundary(), Bulk::inject(), Boundary::Inject(), Domain::interact(), Molecule::Move(), and Domain::run().

14.4.1.34 **#define SQRTPIo8 0.62666**

Definition at line 12 of file def.h.

14.4.1.35 **#define SUB(n, x, y) for (int i=0; i<(n); i++) (x)[i] -= (y)[i]**

Definition at line 47 of file def.h.

14.4.1.36 **#define SUBC(n, x, y, d) for (int i=0; i<(n); i++) (x)[i] -= (y)[i]*(d)**

Definition at line 49 of file def.h.

14.4.1.37 **#define TINY 1e-10**

Definition at line 5 of file def.h.

Referenced by Domain::Domain().

14.4.1.38 **#define TRUE 1**

Definition at line 15 of file def.h.

14.4.1.39 **#define VECP(A, B, C)**

Value:

```
A[0]=B[1]*C[2]-B[2]*C[1];\
```

```
A[1]=B[2]*C[0]-B[0]*C[2];\
A[2]=B[0]*C[1]-B[1]*C[0];
```

Definition at line 58 of file def.h.

Referenced by `Geom::area()`.

14.4.1.40 `#define VOIDSPECIE nspecies`

Definition at line 62 of file def.h.

Referenced by `Domain::boundary()`, `Domain::Domain()`, `Domain::interact()`, `Domain::interaction()`, `Species::Reaction::Outcome::Outcome()`, `GridContainer::put()`, `Domain::run()`, and `Domain::save()`.

14.4.1.41 `#define WAIT for(int i=0;i<999999&&locked;i++)`

Definition at line 20 of file def.h.

Referenced by `Container< Element >::append()`, `Collection< Element >::append()`, `Container< Element >::append1()`, `Container< Element >::insert()`, `Collection< Element >::insert()`, `Container< Element >::insert1()`, `Container< Element >::link()`, `Container< Element >::remove()`, `Collection< Element >::remove()`, and `Container< Element >::unlink()`.

14.4.1.42 `#define WORDLENGTH 64`

Definition at line 9 of file def.h.

Referenced by `Domain::Domain()`, `Domain::load()`, and `main()`.

14.4.1.43 `#define ZERO(n, x) for (int i=0; i<(n); i++)x[i]=0.0`

Definition at line 44 of file def.h.

14.4.1.44 `#define ZEROVEC(x) for (int i=0; i<DIM; i++) (x)[i] = 0.`

Definition at line 56 of file def.h.

14.4.2 Variable Documentation

14.4.2.1 `const REAL GasConstant = BoltzmannConstant*AvogadroNumber`

Definition at line 72 of file def.h.

Referenced by `Domain::Domain()`.

14.4.2.2 `const REAL GasConstantInv = 1.0/GasConstant`

Definition at line 73 of file def.h.

14.4.2.3 const int maxcounters = 4

Definition at line 21 of file def.h.

Referenced by Container< Element >::append(), Collection< Element >::append(), Container< Element >::append0(), Collection< Element >::append0(), Collection< Element >::Collection(), Container< Element >::Container(), Container< Element >::goFirst(), Collection< Molecule >::goFirst(), Collection< Element >::init(), Container< Element >::insert(), Collection< Element >::insert(), Container< Element >::insert0(), Collection< Element >::insert0(), Container< Element >::insert1(), Collection< Element >::insertAfter(), Container< Element >::link(), Container< Element >::link0(), Collection< Molecule >::maxCounters(), Container< Element >::remove(), Collection< Element >::remove(), Container< Element >::remove0(), Collection< Element >::remove0(), Domain::run(), Container< Element >::unlink(), Container< Element >::unlink0(), Collection< Element >::~~Collection(), and Container< Element >::~~Container().

14.4.2.4 const REAL OneOverSqrt2 = 1.0/Sqrt2

Definition at line 32 of file def.h.

14.4.2.5 const REAL OneOverSqrt3 = 1.0/Sqrt3

Definition at line 33 of file def.h.

14.4.2.6 const REAL onethird = 1.0/3.0

Definition at line 25 of file def.h.

Referenced by Domain::run().

14.4.2.7 const REAL sqrt1p25 = sqrt(1.25)

Definition at line 30 of file def.h.

14.4.2.8 const REAL sqrt1p5 = sqrt(1.5)

Definition at line 29 of file def.h.

14.4.2.9 const REAL Sqrt2 = sqrt(2.0)

Definition at line 27 of file def.h.

14.4.2.10 const REAL Sqrt2Over2 = 0.5*Sqrt2

Definition at line 28 of file def.h.

14.4.2.11 const REAL Sqrt3 = sqrt(3.0)

Definition at line 31 of file def.h.

14.4.2.12 const REAL Sqrt3Over2 = 0.5*Sqrt3

Definition at line 34 of file def.h.

14.4.2.13 const REAL twothirds = 2.0/3.0

Definition at line 26 of file def.h.

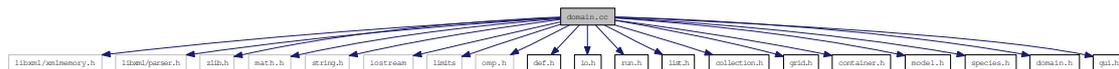
Referenced by Domain::run().

14.5 domain.cc File Reference

Implementation of [Domain](#) and [Boundary](#) classes.

```
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include <zlib.h>
#include <math.h>
#include <string.h>
#include <iostream>
#include <limits>
#include <omp.h>
#include "def.h"
#include "io.h"
#include "run.h"
#include "list.h"
#include "collection.h"
#include "grid.h"
#include "container.h"
#include "model.h"
#include "species.h"
#include "domain.h"
#include "gui.h"
```

Include dependency graph for domain.cc:



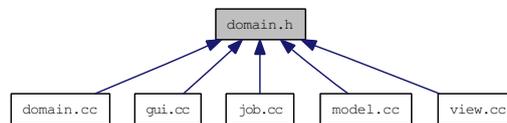
14.5.1 Detailed Description

Implementation of [Domain](#) and [Boundary](#) classes.

Definition in file [domain.cc](#).

14.6 domain.h File Reference

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



Classes

- class [Boundary](#)
Domain Boundary Definition.
- class [Bulk](#)
Domain Bulk Properties.
- class [Domain](#)
Computational Domain.

Defines

- #define [XMIN](#) bounds[0]
- #define [XMAX](#) bounds[1]

Enumerations

- enum [BoundaryTypes](#) {
 [insideBoundary](#) = 0, [periodicBoundary](#), [elasticBoundary](#), [openBoundary](#),
 [maxBoundaryTypes](#) }

14.6.1 Detailed Description

Main definitions of the [Domain](#) class and its associate class [Boundary](#)

Definition in file [domain.h](#).

14.6.2 Define Documentation

14.6.2.1 #define XMAX bounds[1]

Definition at line 2 of file [domain.h](#).

Referenced by [Domain::boundary\(\)](#), [Domain::Domain\(\)](#), and [Domain::interaction\(\)](#).

14.6.2.2 #define XMIN bounds[0]

Definition at line 1 of file domain.h.

Referenced by Domain::boundary(), Domain::Domain(), and Domain::interaction().

14.6.3 Enumeration Type Documentation

14.6.3.1 enum BoundaryTypes

Enumerator:

insideBoundary

periodicBoundary

elasticBoundary

openBoundary

maxBoundaryTypes

Definition at line 7 of file domain.h.

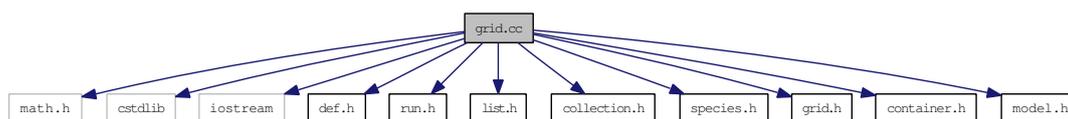
14.7 examples.dox File Reference

14.8 `faq.dox` File Reference

14.9 grid.cc File Reference

```
#include <math.h>
#include <cstdlib>
#include <iostream>
#include "def.h"
#include "run.h"
#include "list.h"
#include "collection.h"
#include "species.h"
#include "grid.h"
#include "container.h"
#include "model.h"
```

Include dependency graph for grid.cc:



Namespaces

- namespace `GridContainer`
Domain Segmentation for Interaction Acceleration.

Functions

- void `GridContainer::init` (REAL ymin[], REAL ymax[], REAL radius, Pool< Molecule > *newpool)
- int `GridContainer::index` (int ind[])
- void `GridContainer::index` (int n, int ind[])
- void `GridContainer::put` (Molecule *node)
- void `GridContainer::put` (Molecule *node, int icounter)
- Container< Molecule > * `GridContainer::get` (int ind[])
- int * `GridContainer::dimensions` ()

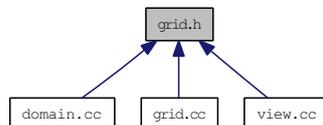
Variables

- REAL `GridContainer::xmin` [DIM]
- REAL `GridContainer::xmax` [DIM]
- REAL `GridContainer::cellsize`
- int `GridContainer::ncells` [DIM+1]
- int `GridContainer::mcells` = 0

- `int GridContainer::mcells1 = 0`
- `Container< Molecule > * GridContainer::nodes`
- `Pool< Molecule > * GridContainer::pool`

14.10 grid.h File Reference

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



Namespaces

- namespace `GridContainer`
Domain Segmentation for Interaction Acceleration.

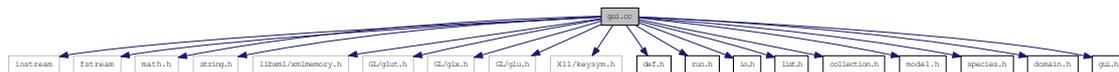
Functions

- `int * GridContainer::dimensions ()`
- `int GridContainer::index (int ind[])`
- `void GridContainer::index (int n, int ind[])`
- `void GridContainer::init (REAL ymin[], REAL ymax[], REAL radius, Pool< Molecule > *newpool)`
- `void GridContainer::put (Molecule *node)`
- `Container< Molecule > * GridContainer::get (int ind[])`
- `bool GridContainer::checkPool (int icell, char *msg)`

14.11 gui.cc File Reference

```
#include <iostream>
#include <fstream>
#include <math.h>
#include <string.h>
#include <libxml/xmlmemory.h>
#include <GL/glut.h>
#include <GL/glx.h>
#include <GL/glu.h>
#include <X11/keysym.h>
#include "def.h"
#include "run.h"
#include "io.h"
#include "list.h"
#include "collection.h"
#include "model.h"
#include "species.h"
#include "domain.h"
#include "gui.h"
```

Include dependency graph for gui.cc:



Namespaces

- namespace [Gui](#)
OpenGL window output routines.
- namespace [Palette](#)
Defines color palette.

Functions

- [int Gui::query_extension](#) (char *extName)
- [void Gui::init](#) (int argc, char *argv[], [Domain](#) *newdomain)
- [void Gui::helpDisplay](#) ()
- [int Gui::readparam](#) (char *s, char *param[], int maxparam, char *filename, int val[])
- [int Gui::readparam](#) (char *s, char *param[], int maxparam, char *filename, REAL val[])

- void [Gui::readconf](#) ()
- void [Gui::refresh](#) ()
- void [Gui::initdisp](#) ()
- void [Gui::Materials](#) (int argc, char *argv[])
- void [Gui::showVector](#) (double *x, double *v)
- void [Gui::showParticle](#) (double vmn, double vmx, double val, double *x)
- void [Gui::showSphere](#) (double vmn, double vmx, double val, double rad, double *x)
- void [Gui::displayAxes](#) ()
- void [Gui::getScaling](#) (double &l0, double &l1)
- void [Gui::getXLimits](#) (double *x0, double *x1)
- void [Gui::Exit](#) ()
- void [Gui::Quit](#) ()
- void [Gui::menu](#) (int value)
- void [Gui::displayMenu](#) ()
- void [Gui::consoleMenu](#) ()
- void [Gui::setBackgroundRun](#) ()
- void [Gui::setForegroundRun](#) ()
- void [Gui::toggleSpheres](#) ()
- void [Gui::switchColorScheme](#) ()
- void [Gui::runmany](#) ()
- void [Gui::toggleWindowDump](#) ()
- void [Gui::dumpwindow](#) ()
- void [Gui::commandMode](#) ()
- void [Gui::display](#) ()
- void [Gui::animate](#) ()
- void [Gui::helpCommand](#) ()
- void [Gui::reshape](#) (int w, int h)
- void [Gui::mouse](#) (int button, int state, int x, int y)
- void [Gui::motion](#) (int x, int y)
- void [Gui::keyboard](#) (unsigned int key)
- void [Gui::run](#) ()
- void [Gui::displaymessage](#) (char *msg)
- void [Gui::setElementColor](#) (int element_type, REAL rgbcolor[])
- void [Gui::getElementColor](#) (int element_type, REAL rgbcolor[])
- void [Palette::init](#) (REAL vmin, REAL vmax)
- void [Palette::pickcolor](#) (REAL var, REAL color[])

Variables

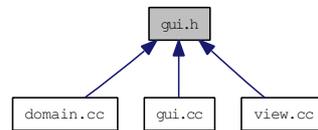
- Domain * [Gui::domain](#)
- ElementDisp [Gui::disp](#) [maxelements]
- char * [Gui::configfile](#) = (char*)"gui.cfg"
- char [Gui::windowname](#) [MAXLINLEN]
- int [Gui::showpar](#) [maxshowpars]
- int [Gui::finished](#)
- int [Gui::animation](#)
- int [Gui::nwdump](#)
- int [Gui::iwdump](#)
- int [Gui::attributeList](#) []
- REAL [Gui::step](#)

- REAL [Gui::vecval](#) [maxlineprm]
- REAL [Gui::axes](#) [maxaxesprm]
- REAL [Gui::rgbcolor](#) [maxcolor][3]
- REAL [Gui::wdtime](#)
- REAL [Gui::xo](#) [3]
- REAL [Gui::dx](#)
- REAL [Gui::dy](#)
- REAL [Gui::dz](#)
- REAL [Gui::lastx](#)
- REAL [Gui::lasty](#)
- REAL [Gui::lastz](#)
- int [Gui::mouseButtons](#) [3]
- REAL [Gui::zoom](#)
- REAL [Gui::rotx](#)
- REAL [Gui::roty](#)
- REAL [Gui::tx](#)
- REAL [Gui::ty](#)
- REAL [Gui::xmin](#) [DIM]
- REAL [Gui::xmax](#) [DIM]
- REAL [Gui::lx](#)
- REAL [Gui::ly](#)
- REAL [Gui::lz](#)
- REAL [Gui::lmin](#)
- REAL [Gui::lmax](#)
- enum Movement [Gui::movement](#)
- struct ColorScale [Gui::colorscale](#)
- struct Scene [Gui::scene](#)
- struct WindowGeom [Gui::window](#)
- struct BFaceList * [Gui::bface_root](#)
- Display * [Gui::dpy](#)
- Window [Gui::win](#)
- int [Gui::firstR](#) = 1
- int * [Gui::vars](#)
- int * [Gui::coms](#)
- REAL [Palette::vmn](#)
- REAL [Palette::vmx](#)
- REAL [Palette::vav](#)
- REAL [Palette::dvi](#)

14.12 `gui.dox` File Reference

14.13 gui.h File Reference

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



Classes

- struct [Gui::ElementDisp](#)
- struct [Gui::ColorScale](#)
- struct [Gui::Scene](#)
- struct [Gui::Scene::Color](#)
- struct [Gui::Scene::Mesh](#)
- struct [Gui::Scene::Frame](#)
- struct [Gui::WindowGeom](#)

Namespaces

- namespace [Gui](#)
OpenGL window output routines.
- namespace [Palette](#)
Defines color palette.

Defines

- #define [WINDOW_SIZE](#) 800
- #define [MAX_WINDOW_WIDTH](#) 800
- #define [MAX_WINDOW_HEIGHT](#) 600
- #define [LEFT_BUTTON](#) 1
- #define [MIDDLE_BUTTON](#) 2
- #define [RIGHT_BUTTON](#) 3

Enumerations

- enum [Gui::Elements](#) {
 [Gui::points](#) = 0, [Gui::nodes](#), [Gui::edges](#), [Gui::faces](#),
 [Gui::cells](#), [Gui::frame](#), [Gui::mesh](#), [Gui::boundary_nodes](#),
 [Gui::boundary_edges](#), [Gui::boundary_faces](#), [Gui::boundary_cells](#), [Gui::maxelements](#) }

- enum `Gui::Color` {
`Gui::red = 0, Gui::green, Gui::blue, Gui::skyblue,`
`Gui::brown, Gui::magenta, Gui::yellow, Gui::color6,`
`Gui::color7, Gui::color8, Gui::color9, Gui::color10,`
`Gui::color11, Gui::color12, Gui::color13, Gui::color14,`
`Gui::color15, Gui::maxcolor` }
- enum `Gui::ColorSchemes` {
`Gui::colorByType = 0, Gui::colorByBoundary, Gui::colorByTime, Gui::colorByMass,`
`Gui::maxColorSchemes` }
- enum `Gui::PointParameters` {
`Gui::variable = maxcolor, Gui::vmin, Gui::vmax, Gui::size,`
`Gui::sizevar, Gui::massvar, Gui::maxpointprm` }
- enum `Gui::LineParameters` { `Gui::thickness = maxcolor, Gui::length, Gui::maxlineprm` }
- enum `Gui::AxesParameters` {
`Gui::axeswidth = 0, Gui::xaxislength, Gui::yaxislength, Gui::zaxislength,`
`Gui::arrowheight, Gui::arrowwidth, Gui::maxaxesprm` }
- enum `Gui::SurfDispType` { `Gui::gridlines = 0, Gui::solidsurface, Gui::maxsurfdisptypes` }
- enum `Gui::Movement` { `Gui::stay = 0, Gui::rotate, Gui::moveuv, Gui::movev` }
- enum `Gui::ShowPars` {
`Gui::showRun = 0, Gui::showAxes, Gui::showSpheres, Gui::showBonds,`
`Gui::showGrid, Gui::showNodes, Gui::showVariables, Gui::showBoundaryVertexes,`
`Gui::showBoundaryVectors, Gui::showToolVertexes, Gui::showBoundaryFaceCenters,`
`Gui::showBoundaryFaces,`
`Gui::showBoundaryGrid, Gui::showToolGrid, Gui::showFrame, Gui::showCellCenters,`
`Gui::showFaceCenters, Gui::showIsoSurfaces, Gui::dumpWindow, Gui::maxshowpars` }

Functions

- void `Gui::Exit` ()
- void `Gui::Quit` ()
- void `Gui::readconf` ()
- void `Gui::helpDisplay` ()
- int `Gui::readparam` (char *s, char *param[], int maxparam, char *filename, REAL val[])
- void `Gui::initdisp` ()
- void `Gui::Materials` (int argc, char *argv[])
- void `Gui::display` ()
- void `Gui::displayAxes` ()
- void `Gui::writeData` ()
- void `Gui::helpCommand` ()
- void `Gui::printGridXLimits` ()
- void `Gui::printGridVecLimits` ()
- void `Gui::getXLimits` (REAL *xmin, REAL *xmax)
- void `Gui::getScaling` (REAL &lmin, REAL &lmax)
- void `Gui::printPartVelLimits` ()
- void `Gui::printParticleVariables` ()
- void `Gui::showGridElements` (int ne, double *X, REAL color[])

- void `Gui::showVectorComponent` (int ivar, int icomp, double *X, double vmn, double vmx)
- void `Gui::showVector` (int ivar, double *X)
- void `Gui::commandMode` ()
- void `Gui::printVariables` (int n)
- void `Gui::init` (int argc, char *argv[], `Domain` *newdomain)
- void `Gui::finish` ()
- void `Gui::InitMaterials` (void)
- void `Gui::getScaling` (double &l0, double &l1)
- void `Gui::getXLimits` (double *x0, double *x1)
- void `Gui::selectVariable` ()
- void `Gui::setBackgroundRun` ()
- void `Gui::setForegroundRun` ()
- void `Gui::animate` ()
- void `Gui::reshape` (int w, int h)
- void `Gui::mouse` (int button, int state, int x, int y)
- void `Gui::motion` (int x, int y)
- void `Gui::keyboard` (unsigned char key, int x, int y)
- void `Gui::menu` (int value)
- void `Gui::run` ()
- void `Gui::drawSegment` (REAL *x, REAL *y)
- void `Gui::displaymessage` (char *msg)
- void `Gui::setElementColor` (int element_type, REAL rgbcolor[])
- void `Gui::getElementColor` (int element_type, REAL rgbcolor[])
- void `Palette::init` (REAL vmin, REAL vmax)
- void `Palette::pickcolor` (REAL var, REAL color[])

14.13.1 Define Documentation

14.13.1.1 #define LEFT_BUTTON 1

Definition at line 10 of file gui.h.

Referenced by `Gui::mouse()`.

14.13.1.2 #define MAX_WINDOW_HEIGHT 600

Definition at line 8 of file gui.h.

Referenced by `Gui::initdisp()`.

14.13.1.3 #define MAX_WINDOW_WIDTH 800

Definition at line 7 of file gui.h.

Referenced by `Gui::initdisp()`.

14.13.1.4 #define MIDDLE_BUTTON 2

Definition at line 11 of file gui.h.

Referenced by `Gui::mouse()`.

14.13.1.5 #define RIGHT_BUTTON 3

Definition at line 12 of file gui.h.

Referenced by Gui::mouse().

14.13.1.6 #define WINDOW_SIZE 800

Definition at line 6 of file gui.h.

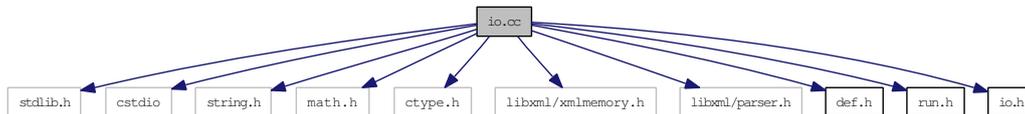
Referenced by Gui::initdisp().

14.14 implementation.dox File Reference

14.15 io.cc File Reference

```
#include <stdlib.h>
#include <cstdio>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include "def.h"
#include "run.h"
#include "io.h"
```

Include dependency graph for io.cc:



Namespaces

- namespace **IO**
Some IO routines.

Functions

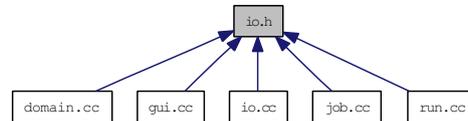
- void **IO::initxwd** (int i)
- void **IO::xwd** (char *windowname)
- int **IO::getCharAttr** (xmlNodePtr cur, char *attr, char *result)
- int **IO::getIntAttr** (xmlNodePtr cur, char *attr)
- int **IO::getIntAttr** (xmlDocPtr doc, char *tag, char *keyname, char *key, char *attr)
- int **IO::parseWord** (xmlDocPtr doc, xmlNodePtr cur, char *keyword, char *result)
- int **IO::parseInt** (xmlDocPtr doc, xmlNodePtr cur, char *keyword)
- int **IO::parseFloat** (xmlDocPtr doc, xmlNodePtr cur, char *keyword, REAL &result)
- double **IO::parseFloat** (xmlDocPtr doc, xmlNodePtr cur, char *keyword)
- void **IO::getTimeXML** (char infilename[])
- void **IO::getTime** (char infilename[])
- int **IO::getIter** (char infilename[])

Variables

- int **IO::ioutput** = 0
- int **IO::nxwdump** = 0

14.16 io.h File Reference

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



Namespaces

- namespace `IO`
Some IO routines.

Functions

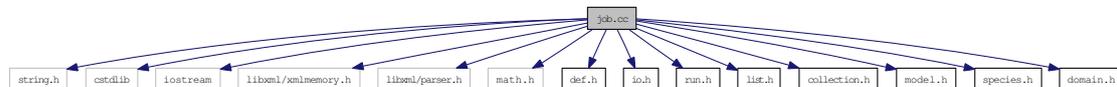
- int `IO::getCharAttr` (xmlNodePtr cur, char *attr, char *result)
- int `IO::getIntAttr` (xmlNodePtr cur, char *attr)
- int `IO::getIntAttr` (xmlDocPtr doc, char *tag, char *keyname, char *key, char *attr)
- int `IO::parseWord` (xmlDocPtr doc, xmlNodePtr cur, char *keyword, char *result)
- int `IO::parseInt` (xmlDocPtr doc, xmlNodePtr cur, char *keyword)
- double `IO::parseFloat` (xmlDocPtr doc, xmlNodePtr cur, char *keyword)
- int `IO::parseFloat` (xmlDocPtr doc, xmlNodePtr cur, char *keyword, REAL &result)
- int `IO::getIter` (char infilename[])
- void `IO::getTime` (char infilename[])
- void `IO::xwd` (char *windowname)

14.17 job.cc File Reference

The ReMoDy backend.

```
#include <string.h>
#include <cstdlib>
#include <iostream>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include <math.h>
#include "def.h"
#include "io.h"
#include "run.h"
#include "list.h"
#include "collection.h"
#include "model.h"
#include "species.h"
#include "domain.h"
```

Include dependency graph for job.cc:



Functions

- void [usage](#) ()
- void [parsename](#) (char *name, char *filename)
Strips the suffix from the filename.
- int [main](#) (int argc, char *argv[])
The Main routine for a batch mode. Executes a background job.

14.17.1 Detailed Description

The ReMoDy backend.

Executes background job.

Definition in file [job.cc](#).

14.17.2 Function Documentation

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

The Main routine for a batch mode. Executes a background job.

Parameters:

argc,argv,: standard command line arguments

Returns:

exit code 0 - success, 1 - failure.

parses command line arguments

creates a domain instance

erases screen

runs the code

retrieves taskname

retrieves taskname

saves data

Definition at line 41 of file job.cc.

References `Run::configfile`, `Gui::domain`, `ESC`, `IO::getIter()`, `Run::init()`, `Run::inputfile`, `Run::option`, `parseiname()`, `Run::programname`, `Domain::run()`, `Domain::save()`, and `WORDLENGTH`.

14.17.2.2 `void parseiname (char * name, char * filename)`

Strips the suffix from the filename.

Parameters:

filename,: file name including suffix

Returns:

name: file name with suffix removed

Definition at line 30 of file job.cc.

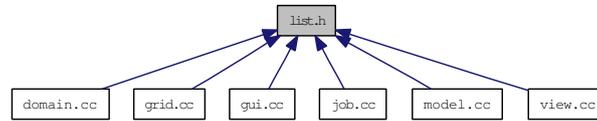
Referenced by `main()`.

14.17.2.3 `void usage ()`

Definition at line 23 of file job.cc.

14.18 list.h File Reference

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



Classes

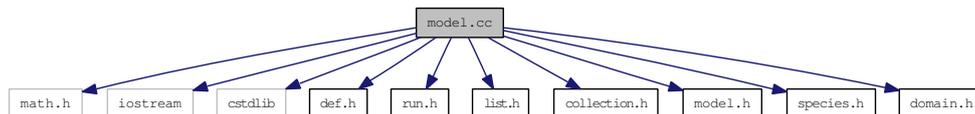
- struct [Pointer< Element >](#)
- class [List< Element >](#)

14.19 main.dox File Reference

14.20 model.cc File Reference

```
#include <math.h>
#include <iostream>
#include <cstdlib>
#include "def.h"
#include "run.h"
#include "list.h"
#include "collection.h"
#include "model.h"
#include "species.h"
#include "domain.h"
```

Include dependency graph for model.cc:



Namespaces

- namespace [Potential](#)
Interaction-potential functions, such as LJ.

Functions

- REAL [Potential::invdist](#) (REAL x)

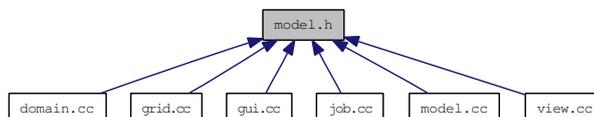
Variables

- REAL [Potential::sigma](#) = 1.0
- REAL [Potential::eta](#) = 1.0
- REAL [Potential::cutoff](#) = 2.0

14.21 model.dox File Reference

14.22 model.h File Reference

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



Classes

- union [State](#)
- class [Molecule](#)

Namespaces

- namespace [Potential](#)
Interaction-potential functions, such as LJ.
- namespace [Geom](#)
Provides some trigonometry functions.

Defines

- #define [HARDBALLS](#)
hardball collision as opposed to interaction via a potential

Enumerations

- enum [AtomType](#) {
 [undefined](#) = 0, [hydrogen](#) = 1, [helium](#) = 2, [carbon](#) = 12,
 [oxygen](#) = 16, [maxAtomTypes](#) }

Functions

- void [Potential::strength](#) (REAL strength)
- REAL [Potential::strength](#) ()
- void [Potential::lengthscale](#) (REAL lengthscale)
- REAL [Potential::lengthscale](#) ()
- void [Potential::Cutoff](#) (REAL newcutoff)
- REAL [Potential::Cutoff](#) ()
- REAL [Potential::value](#) (REAL r)
- REAL [Potential::force](#) (REAL r)
- REAL [Potential::derivative](#) (REAL r)
- void [Geom::distvec](#) (REAL a[], REAL b[], REAL c[])

- REAL `Geom::distance` (REAL a[], REAL b[])
- REAL `Geom::distance` (int dim, REAL a[], REAL b[])
- REAL `Geom::sclp` (REAL a[], REAL b[])
- REAL `Geom::length` (REAL a[])
- REAL `Geom::area` (REAL a[], REAL b[])
- REAL `Geom::normalize` (REAL a[])
- int `Geom::hash` (int i, int j, int n)

Variables

- const REAL `Potential::small` = 1.0e-20
- const REAL `Potential::large` = 1.0e20

14.22.1 Define Documentation

14.22.1.1 #define HARDBALLS

hardball collision as opposed to interaction via a potential

Definition at line 2 of file model.h.

14.22.2 Enumeration Type Documentation

14.22.2.1 enum AtomType

Define the ADSORPTION flag below to enable the time delay for the adsorbed species to be released from the surface

Enumerator:

undefined

hydrogen

helium

carbon

oxygen

maxAtomTypes

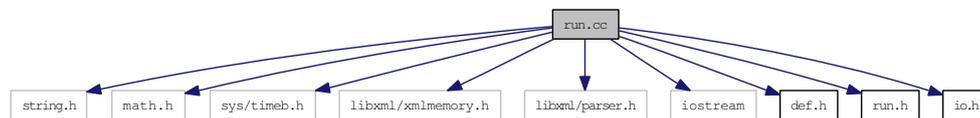
Definition at line 8 of file model.h.

14.23 output.dox File Reference

14.24 run.cc File Reference

```
#include <string.h>
#include <math.h>
#include <sys/timeb.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include <iostream>
#include "def.h"
#include "run.h"
#include "io.h"
```

Include dependency graph for run.cc:



Namespaces

- namespace [Run](#)
Processing command-line and execution control parameters.

Defines

- #define [RAND_MAX](#) 2147483647

Functions

- void [Run::testrnd](#) ()
Test random numer generator.
- REAL [Run::rnd](#) ()
Seeds random number.
- REAL [Run::gauss](#) ()
normal random number generator
- void [Run::init](#) (int argc, char *argv[])
Initialize all variables, and execution options.
- void [Run::readcmdline](#) (int argc, char *argv[])
Parses command-line options.

- int `Run::elapsed` ()
time (sec) elapsed since the start of the run

Variables

- struct `Option Run::option` = {0,0,0,0}
command-line options
- void(* `Run::usage`)()
- char `Run::programname` [MAXLINLEN]
- char `Run::configfile` [MAXLINLEN]
- char `Run::outputfile` [MAXLINLEN]
- char `Run::inputfile` [MAXLINLEN]
- char `Run::outputname` [MAXLINLEN]
- int `Run::nthreads`
- struct timeb `Run::worldtime`
- struct Time `Run::time`

14.24.1 Define Documentation

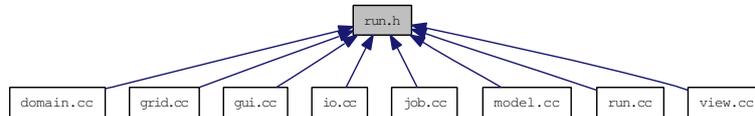
14.24.1.1 #define RAND_MAX 2147483647

Definition at line 14 of file run.cc.

Referenced by `Run::rnd()`, and `Run::testrnd()`.

14.25 run.h File Reference

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



Classes

- struct [Option](#)
Execution options.
- struct [Run::Time](#)
Execution time-control structure.

Namespaces

- namespace [Run](#)
Processing command-line and execution control parameters.

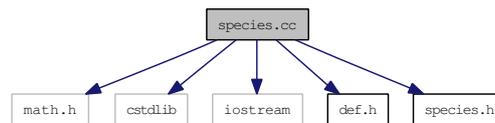
Functions

- void [Run::init](#) (int argc, char *argv[])
Initialize all variables, and execution options.
- void [Run::readcmdline](#) (int argc, char *argv[])
Parses command-line options.
- int [Run::elapsed](#) ()
time (sec) elapsed since the start of the run
- void [Run::seed](#) ()
seed random number generator
- REAL [Run::rnd](#) ()
Seeds random number.
- REAL [Run::gauss](#) ()
normal random number generator
- void(*) [Run::usage](#) ()

14.26 species.cc File Reference

```
#include <math.h>
#include <cstdlib>
#include <iostream>
#include "def.h"
#include "species.h"
```

Include dependency graph for species.cc:



Namespaces

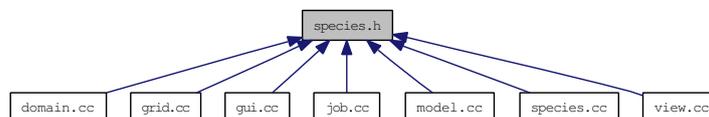
- namespace [Species](#)
Description of [Specie](#) and [Reaction](#) classes.

Variables

- int [Species::nspecies](#)
- [Specie](#) * [Species::species](#)
- [Reaction](#) * [Species::reactions](#)

14.27 species.h File Reference

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



Classes

- class `Species::Specie`
- struct `Species::Gas`
Gas is a specie with some additional parameters.
- struct `Species::Reaction`
Reaction determines the two products only.
- struct `Species::Reaction::Outcome`
Possible outcomes of a reaction.

Namespaces

- namespace `Species`
Description of Specie and Reaction classes.

Enumerations

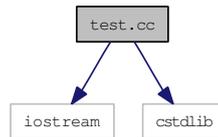
- enum `Species::Interaction` { `Species::missed` = 0, `Species::collided`, `Species::reacted`, `Species::annihilated` }

14.28 test.cc File Reference

```
#include <iostream>
```

```
#include <cstdlib>
```

Include dependency graph for test.cc:



Functions

- [main \(\)](#)

14.28.1 Function Documentation

14.28.1.1 main ()

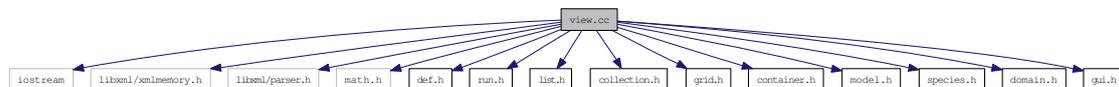
Definition at line 4 of file test.cc.

14.29 view.cc File Reference

The frontend of ReMoDy with the OpenGL Visualizer.

```
#include <iostream>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include <math.h>
#include "def.h"
#include "run.h"
#include "list.h"
#include "collection.h"
#include "grid.h"
#include "container.h"
#include "model.h"
#include "species.h"
#include "domain.h"
#include "gui.h"
```

Include dependency graph for view.cc:



Functions

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

The main routine.

14.29.1 Detailed Description

The frontend of ReMoDy with the OpenGL Visualizer.

Executes a job and displays the computational domain in an OpenGL window.

Definition in file [view.cc](#).

14.29.2 Function Documentation

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

The main routine.

parses command line arguments

creates a domain instance

initializes the domain

executes the job

Definition at line 29 of file view.cc.

References Gui::domain, Gui::init(), Run::init(), Run::inputfile, and Gui::run().

14.29.2.2 void usage ()

Definition at line 25 of file view.cc.

References Run::programname.

Index

- ~Boundary
 - Boundary, [74](#)
- ~Collection
 - Collection, [82](#)
- ~Container
 - Container, [95](#)
- ~Domain
 - Domain, [110](#)
- ~List
 - List, [122](#)
- ~Pool
 - Pool, [138](#)
- ~Reaction
 - Species::Reaction, [143](#)
- ABOUT
 - def.h, [165](#)
- ActivationEnergy
 - Species::Reaction::Outcome, [146](#)
- activationEnergy
 - Species::Reaction::Outcome, [147](#)
- ADD
 - def.h, [165](#)
- Add
 - Species::Reaction, [143](#)
- ADDC
 - def.h, [165](#)
- ADDVEC
 - def.h, [165](#)
- adiabatic
 - Boundary, [76](#)
- animate
 - Gui, [46](#)
- animation
 - Gui, [53](#)
- annihilated
 - Species, [71](#)
- append
 - Collection, [83](#)
 - Container, [95](#)
 - List, [122](#)
- append0
 - Collection, [83](#)
 - Container, [96](#)
- append1
 - Container, [96](#)
- Area
 - Boundary, [75](#)
- area
 - Boundary, [76](#)
 - Geom, [35](#)
- arrowheight
 - Gui, [43](#)
- arrowwidth
 - Gui, [43](#)
- AtomicMassUnit
 - def.h, [165](#)
- AtomType
 - model.h, [199](#)
- attributeList
 - Gui, [53](#)
- AvogadroNumber
 - def.h, [166](#)
- axes
 - Gui, [53](#)
- AxesParameters
 - Gui, [43](#)
- axeswidth
 - Gui, [43](#)
- bface_root
 - Gui, [53](#)
- blue
 - Gui, [43](#)
- BoltzmannConstant
 - def.h, [166](#)
- boundaries
 - Domain, [115](#)
- Boundary, [73](#)
 - ~Boundary, [74](#)
 - adiabatic, [76](#)
 - Area, [75](#)
 - area, [76](#)
 - Boundary, [74](#)
 - gases, [76](#)
 - idir, [76](#)
 - init, [75](#)
 - Inject, [75](#)
 - iside, [76](#)
 - reactions, [76](#)

- Temperature, 75
- temperature, 76
- type, 76
- xmax, 77
- xmin, 77
- boundary
 - Domain, 110
- boundary_cells
 - Gui, 44
- boundary_edges
 - Gui, 44
- boundary_faces
 - Gui, 44
- boundary_nodes
 - Gui, 44
- boundaryName
 - Domain, 115
- BoundaryType
 - Domain, 110, 111
- BoundaryTypes
 - domain.h, 175
- bounds
 - Domain, 115
- brown
 - Gui, 43
- Bulk, 78
 - gases, 80
 - init, 79
 - inject, 79
 - Temperature, 79
 - temperature, 80
 - Volume, 79
 - volume, 80
 - xmax, 80
 - xmin, 80
- bulk
 - Domain, 115
- carbon
 - model.h, 199
- cells
 - Gui, 44
- cellsize
 - GridContainer, 38
- check
 - Pool, 138
- checkPool
 - Container, 96
 - GridContainer, 37
- cleanup
 - List, 123
- Collection, 81
 - ~Collection, 82
 - append, 83
 - append0, 83
 - Collection, 82
 - Current, 83, 84
 - current, 89
 - current1, 89
 - dead, 90
 - First, 84
 - first, 90
 - FirstNext, 84
 - FirstPrev, 84
 - getFirstItem, 84
 - getItem, 84
 - getLastItem, 84
 - getNext, 84
 - getPrev, 85
 - go, 85
 - goFirst, 85
 - goFirstNext, 85
 - goLast, 85
 - goLastNext, 85, 86
 - goNext, 86
 - goPrev, 86
 - init, 86
 - insert, 86, 87
 - insert0, 87
 - isFirst, 87
 - isFirstLast, 87
 - isLast, 87
 - items, 90
 - Last, 87
 - last, 90
 - LastNext, 88
 - LastPrev, 88
 - locked, 90
 - maxCounters, 88
 - maxnumber, 88
 - mitems, 91
 - Next, 88
 - nitems, 91
 - number, 88
 - Prev, 88
 - remove, 88, 89
 - remove0, 89
 - setFirst, 89
 - setFirstLast, 89
- collection.h, 161
- collided
 - Species, 71
- COLLISIONTIME
 - def.h, 166
- Color
 - Gui, 43
- color
 - Gui::Scene, 149

- color10
 - Gui, 43
- color11
 - Gui, 43
- color12
 - Gui, 43
- color13
 - Gui, 43
- color14
 - Gui, 43
- color15
 - Gui, 43
- color6
 - Gui, 43
- color7
 - Gui, 43
- color8
 - Gui, 43
- color9
 - Gui, 43
- colorByBoundary
 - Gui, 44
- colorByMass
 - Gui, 44
- colorByTime
 - Gui, 44
- colorByType
 - Gui, 44
- colorscale
 - Gui, 53
- ColorSchemes
 - Gui, 44
- commandMode
 - Gui, 46
- computeBounds
 - Domain, 111
- coms
 - Gui, 53
- configfile
 - Gui, 53
 - Run, 69
- configfile.dox, 162
- consoleMenu
 - Gui, 46
- Container, 93
 - ~Container, 95
 - append, 95
 - append0, 96
 - append1, 96
 - checkPool, 96
 - Container, 95
 - Current, 96
 - current, 105
 - current1, 106
 - First, 96, 97
 - first, 106
 - first1, 106
 - FirstNext, 97
 - FirstPrev, 97
 - getFirstPtr, 97
 - getLastPtr, 97
 - getNext, 98
 - getPrev, 98
 - getPtr, 98
 - go, 98
 - goFirst, 99
 - goFirstNext, 99
 - goLast, 99
 - goLastNext, 99
 - goNext, 100
 - goPrev, 100
 - insert, 100, 101
 - insert0, 101
 - insert1, 101
 - isFirst, 101
 - isFirstLast, 101
 - isLast, 102
 - islocked, 102
 - Last, 102
 - last, 107
 - LastNext, 102
 - LastPrev, 102
 - link, 102
 - link0, 102
 - lock, 103
 - locked, 107
 - Next, 103
 - nptrs, 107
 - number, 103
 - Prev, 103
 - remove, 103, 104
 - remove0, 104
 - setFirst, 104
 - setFirstLast, 105
 - unlink, 105
 - unlink0, 105
 - unlock, 105
- container.h, 163
- Coordinate
 - Molecule, 131
- Coordinates
 - Molecule, 131
- copy
 - Molecule, 131
- COPYVEC
 - def.h, 166
- Cp
 - Species::Specie, 153

- cp
 - Species::Specie, 154
- Current
 - Collection, 83, 84
 - Container, 96
 - List, 123
- current
 - Collection, 89
 - Container, 105
 - List, 128
 - Run::Time, 157
 - Species::Reaction, 143
- current1
 - Collection, 89
 - Container, 106
- Cutoff
 - Potential, 64
- cutoff
 - Potential, 66
- dead
 - Collection, 90
- DEBUG
 - def.h, 166
- debug
 - Option, 135
- def.h, 164
 - ABOUT, 165
 - ADD, 165
 - ADDC, 165
 - ADDVEC, 165
 - AtomicMassUnit, 165
 - AvogadroNumber, 166
 - BoltzmannConstant, 166
 - COLLISIONTIME, 166
 - COPYVEC, 166
 - DEBUG, 166
 - DIM, 166
 - DIV, 166
 - DOCTYPE, 166
 - DOF, 167
 - DOFI, 167
 - DOFK, 167
 - ERROR, 167
 - ESC, 167
 - FALSE, 167
 - GasConstant, 170
 - GasConstantInv, 171
 - GAUSS, 167
 - LARGE, 167
 - LENGTH, 167
 - LOCAL, 168
 - MAX, 168
 - maxcounters, 171
 - MAXLINLEN, 168
 - MUL, 168
 - MULC, 168
 - MULVEC, 168
 - OMP, 168
 - OneOverSqrt2, 171
 - OneOverSqrt3, 171
 - onethird, 171
 - PI, 168
 - REAL, 168
 - RND, 169
 - SCLP, 169
 - SMALL, 169
 - sqrt1p25, 171
 - sqrt1p5, 171
 - Sqrt2, 171
 - Sqrt2Over2, 171
 - Sqrt3, 171
 - Sqrt3Over2, 172
 - SQRTPIo8, 169
 - SUB, 169
 - SUBC, 169
 - TINY, 169
 - TRUE, 169
 - twothirds, 172
 - VECP, 170
 - VOIDSPECIE, 170
 - WAIT, 170
 - WORDLENGTH, 170
 - ZERO, 170
 - ZEROVEC, 170
- density
 - Species::Gas, 118
- derivative
 - Potential, 64
- DIM
 - def.h, 166
- dimensions
 - GridContainer, 37
- disp
 - Gui, 53
- display
 - Gui, 46
- displayAxes
 - Gui, 47
- displayMenu
 - Gui, 47
- displaymessage
 - Gui, 47
- distance
 - Geom, 35
- distribution
 - Domain, 115
- distvec

- Geom, 36
- DIV
 - def.h, 166
- DOCTYPE
 - def.h, 166
- DOF
 - def.h, 167
- DOFI
 - def.h, 167
- DOFK
 - def.h, 167
- Domain, 108
 - ~Domain, 110
 - boundaries, 115
 - boundary, 110
 - boundaryName, 115
 - BoundaryType, 110, 111
 - bounds, 115
 - bulk, 115
 - computeBounds, 111
 - distribution, 115
 - Domain, 109
 - init, 111
 - injection, 111
 - interact, 112
 - interaction, 113
 - load, 113
 - maxBound, 113
 - minBound, 114
 - Molecules, 114
 - molecules, 115
 - pool, 116
 - run, 114
 - save, 114
 - setMaxBound, 114
 - setMinBound, 114
 - step, 114
- domain
 - Gui, 54
- domain.cc, 173
- domain.h, 174
 - BoundaryTypes, 175
 - elasticBoundary, 175
 - insideBoundary, 175
 - maxBoundaryTypes, 175
 - openBoundary, 175
 - periodicBoundary, 175
 - XMAX, 174
 - XMIN, 174
- done
 - State, 156
- dpy
 - Gui, 54
- drawSegment
 - Gui, 47
- dumpWindow
 - Gui, 46
- dumpwindow
 - Gui, 47
- dvi
 - Palette, 62
- dx
 - Gui, 54
- dy
 - Gui, 54
- dz
 - Gui, 54
- edges
 - Gui, 44
- elapsed
 - Run, 68
- elasticBoundary
 - domain.h, 175
- element
 - Item, 120
 - Pointer, 137
 - Ptr, 141
- Elements
 - Gui, 44
- end
 - Run::Time, 157
- energy
 - Molecule, 134
- Enthalpy
 - Species::Reaction::Outcome, 146
- enthalpy
 - Species::Reaction::Outcome, 147
- Erase
 - Species::Reaction, 143
- erase
 - List, 123
- eraseall
 - List, 123
- ERROR
 - def.h, 167
- ESC
 - def.h, 167
- eta
 - Potential, 66
- examples.dox, 176
- Exit
 - Gui, 47
- faces
 - Gui, 44
- FALSE
 - def.h, 167

- faq.dox, 177
- finish
 - Gui, 47
- finished
 - Gui, 54
- First
 - Collection, 84
 - Container, 96, 97
 - List, 123
 - Species::Reaction, 143
- first
 - Collection, 90
 - Container, 106
 - List, 128
- first1
 - Container, 106
- FirstNext
 - Collection, 84
 - Container, 97
 - List, 123
- FirstPrev
 - Collection, 84
 - Container, 97
 - List, 124
- firstR
 - Gui, 54
- force
 - Potential, 65
- frame
 - Gui, 44
 - Gui::Scene, 149
- Gas
 - Species::Gas, 118
- GasConstant
 - def.h, 170
- GasConstantInv
 - def.h, 171
- gases
 - Boundary, 76
 - Bulk, 80
- GAUSS
 - def.h, 167
- gauss
 - Run, 68
- Geom, 35
 - area, 35
 - distance, 35
 - distvec, 36
 - hash, 36
 - length, 36
 - normalize, 36
 - sclp, 36
- get
 - GridContainer, 37
 - Pool, 139
 - getCharAttr
 - IO, 59
 - getElementColor
 - Gui, 47
 - getFirstItem
 - Collection, 84
 - getFirstPointer
 - List, 124
 - getFirstPtr
 - Container, 97
 - getIntAttr
 - IO, 59
 - getItem
 - Collection, 84
 - getIter
 - IO, 59
 - getLastItem
 - Collection, 84
 - getLastPointer
 - List, 124
 - getLastPtr
 - Container, 97
 - getNext
 - Collection, 84
 - Container, 98
 - List, 124
 - getPointer
 - List, 124
 - getPrev
 - Collection, 85
 - Container, 98
 - List, 124
 - getPtr
 - Container, 98
 - getScaling
 - Gui, 48
 - getTime
 - IO, 60
 - getTimeXML
 - IO, 60
 - getXLimits
 - Gui, 48
 - go
 - Collection, 85
 - Container, 98
 - List, 124
 - goFirst
 - Collection, 85
 - Container, 99
 - List, 124
 - goFirstNext
 - Collection, 85

- Container, 99
- List, 124
- goLast
 - Collection, 85
 - Container, 99
 - List, 124
- goLastNext
 - Collection, 85, 86
 - Container, 99
 - List, 125
- goNext
 - Collection, 86
 - Container, 100
 - List, 125
- goPrev
 - Collection, 86
 - Container, 100
 - List, 125
- green
 - Gui, 43
- grid.cc, 178
- grid.h, 180
- GridContainer, 37
 - cellsize, 38
 - checkPool, 37
 - dimensions, 37
 - get, 37
 - index, 37, 38
 - init, 38
 - mcells, 38
 - mcells1, 39
 - ncells, 39
 - nodes, 39
 - pool, 39
 - put, 38
 - xmax, 39
 - xmin, 39
- gridlines
 - Gui, 46
- Gui, 40
 - animate, 46
 - animation, 53
 - arrowheight, 43
 - arrowwidth, 43
 - attributeList, 53
 - axes, 53
 - AxesParameters, 43
 - axeswidth, 43
 - bface_root, 53
 - blue, 43
 - boundary_cells, 44
 - boundary_edges, 44
 - boundary_faces, 44
 - boundary_nodes, 44
 - brown, 43
 - cells, 44
 - Color, 43
 - color10, 43
 - color11, 43
 - color12, 43
 - color13, 43
 - color14, 43
 - color15, 43
 - color6, 43
 - color7, 43
 - color8, 43
 - color9, 43
 - colorByBoundary, 44
 - colorByMass, 44
 - colorByTime, 44
 - colorByType, 44
 - colorscale, 53
 - ColorSchemes, 44
 - commandMode, 46
 - coms, 53
 - configfile, 53
 - consoleMenu, 46
 - disp, 53
 - display, 46
 - displayAxes, 47
 - displayMenu, 47
 - displaymessage, 47
 - domain, 54
 - dpy, 54
 - drawSegment, 47
 - dumpWindow, 46
 - dumpwindow, 47
 - dx, 54
 - dy, 54
 - dz, 54
 - edges, 44
 - Elements, 44
 - Exit, 47
 - faces, 44
 - finish, 47
 - finished, 54
 - firstR, 54
 - frame, 44
 - getElementColor, 47
 - getScaling, 48
 - getXLimits, 48
 - green, 43
 - gridlines, 46
 - helpCommand, 48
 - helpDisplay, 48
 - init, 48
 - initdisp, 48
 - InitMaterials, 49

- iwdump, 54
- keyboard, 49
- lastx, 54
- lasty, 55
- lastz, 55
- length, 44
- LineParameters, 44
- lmax, 55
- lmin, 55
- lx, 55
- ly, 55
- lz, 55
- magenta, 43
- massvar, 45
- Materials, 49
- maxaxesprm, 43
- maxcolor, 44
- maxColorSchemes, 44
- maxelements, 44
- maxlineprm, 44
- maxpointprm, 45
- maxshowpars, 46
- maxsurfdisptypes, 46
- menu, 49
- mesh, 44
- motion, 49
- mouse, 49
- mouseButtons, 55
- Movement, 44
- movement, 55
- moveuv, 45
- movew, 45
- nodes, 44
- nwdump, 56
- PointParameters, 45
- points, 44
- printGridVecLimits, 50
- printGridXLimits, 50
- printParticleVariables, 50
- printPartVelLimits, 50
- printVariables, 50
- query_extension, 50
- Quit, 50
- readconf, 50
- readparam, 50
- red, 43
- refresh, 50
- reshape, 51
- rgbcolor, 56
- rotate, 45
- rotx, 56
- roty, 56
- run, 51
- runmany, 51
- scene, 56
- selectVariable, 51
- setBackgroundRun, 51
- setElementColor, 51
- setForegroundRun, 51
- showAxes, 45
- showBonds, 45
- showBoundaryFaceCenters, 45
- showBoundaryFaces, 45
- showBoundaryGrid, 45
- showBoundaryVectors, 45
- showBoundaryVertexes, 45
- showCellCenters, 45
- showFaceCenters, 45
- showFrame, 45
- showGrid, 45
- showGridElements, 51
- showIsoSurfaces, 45
- showNodes, 45
- showpar, 56
- ShowPars, 45
- showParticle, 52
- showRun, 45
- showSphere, 52
- showSpheres, 45
- showToolGrid, 45
- showToolVertexes, 45
- showVariables, 45
- showVector, 52
- showVectorComponent, 52
- size, 45
- sizevar, 45
- skyblue, 43
- solidsurface, 46
- stay, 45
- step, 56
- SurfDispType, 46
- switchColorScheme, 52
- thickness, 44
- toggleSpheres, 52
- toggleWindowDump, 52
- tx, 56
- ty, 56
- variable, 45
- vars, 57
- vecval, 57
- vmax, 45
- vmin, 45
- wdtime, 57
- win, 57
- window, 57
- windowname, 57
- writeData, 52
- xaxislength, 43

- xmax, [57](#)
- xmin, [57](#)
- xo, [57](#)
- yaxislength, [43](#)
- yellow, [43](#)
- zaxislength, [43](#)
- zoom, [58](#)
- gui.cc, [181](#)
- gui.dox, [184](#)
- gui.h, [185](#)
 - LEFT_BUTTON, [187](#)
 - MAX_WINDOW_HEIGHT, [187](#)
 - MAX_WINDOW_WIDTH, [187](#)
 - MIDDLE_BUTTON, [187](#)
 - RIGHT_BUTTON, [187](#)
 - WINDOW_SIZE, [188](#)
- Gui::ColorScale, [92](#)
 - relative, [92](#)
- Gui::ElementDisp, [117](#)
 - lighting, [117](#)
 - rgbcolor, [117](#)
- Gui::Scene, [149](#)
 - color, [149](#)
 - frame, [149](#)
 - mesh, [149](#)
- Gui::Scene::Color, [150](#)
 - maxvalue, [150](#)
 - minvalue, [150](#)
 - scheme, [150](#)
- Gui::Scene::Frame, [151](#)
 - line, [151](#)
- Gui::Scene::Mesh, [152](#)
 - line, [152](#)
 - node, [152](#)
- Gui::WindowGeom, [159](#)
 - height, [159](#)
 - width, [159](#)
- HARDBALLS
 - model.h, [199](#)
- hash
 - Geom, [36](#)
- height
 - Gui::WindowGeom, [159](#)
- helium
 - model.h, [199](#)
- helpCommand
 - Gui, [48](#)
- helpDisplay
 - Gui, [48](#)
- hook
 - Pool, [139](#)
- hydrogen
 - model.h, [199](#)
- Id
 - Species::Specie, [153](#)
- id
 - Species::Specie, [154](#)
- idir
 - Boundary, [76](#)
- implementation.dox, [189](#)
- index
 - GridContainer, [37, 38](#)
- inertia
 - Species::Specie, [154](#)
- init
 - Boundary, [75](#)
 - Bulk, [79](#)
 - Collection, [86](#)
 - Domain, [111](#)
 - GridContainer, [38](#)
 - Gui, [48](#)
 - List, [125](#)
 - Palette, [62](#)
 - Run, [68](#)
- initdisp
 - Gui, [48](#)
- InitMaterials
 - Gui, [49](#)
- initxwd
 - IO, [60](#)
- Inject
 - Boundary, [75](#)
- inject
 - Bulk, [79](#)
- injection
 - Domain, [111](#)
- inputfile
 - Run, [69](#)
- insert
 - Collection, [86, 87](#)
 - Container, [100, 101](#)
 - List, [125](#)
- insert0
 - Collection, [87](#)
 - Container, [101](#)
- insert1
 - Container, [101](#)
- insideBoundary
 - domain.h, [175](#)
- interact
 - Domain, [112](#)
- Interaction
 - Species, [71](#)
- interaction
 - Domain, [113](#)
- InternalEnergy
 - Molecule, [131, 132](#)

- invdist
 - Potential, 65
- IO, 59
 - getCharAttr, 59
 - getIntAttr, 59
 - getIter, 59
 - getTime, 60
 - getTimeXML, 60
 - initxwd, 60
 - ioutput, 61
 - nxwdump, 61
 - parseFloat, 60
 - parseInt, 60
 - parseWord, 60
 - xwd, 61
- io.cc, 190
- io.h, 191
- ioutput
 - IO, 61
- isFirst
 - Collection, 87
 - Container, 101
 - List, 125
- isFirstLast
 - Collection, 87
 - Container, 101
 - List, 125
- iside
 - Boundary, 76
- isLast
 - Collection, 87
 - Container, 102
 - List, 125
- islocked
 - Container, 102
- Item, 120
 - element, 120
 - next, 120
 - prev, 120
- items
 - Collection, 90
- iwdump
 - Gui, 54
- job.cc, 192
 - main, 193
 - parsename, 193
 - usage, 193
- keyboard
 - Gui, 49
- KineticEnergy
 - Molecule, 132
- LARGE
 - def.h, 167
- large
 - Potential, 66
- Last
 - Collection, 87
 - Container, 102
 - List, 126
- last
 - Collection, 90
 - Container, 107
 - List, 128
- LastNext
 - Collection, 88
 - Container, 102
 - List, 126
- LastPrev
 - Collection, 88
 - Container, 102
 - List, 126
- lastx
 - Gui, 54
- lasty
 - Gui, 55
- lastz
 - Gui, 55
- LEFT_BUTTON
 - gui.h, 187
- LENGTH
 - def.h, 167
- length
 - Geom, 36
 - Gui, 44
- lengthscale
 - Potential, 65
- lighting
 - Gui::ElementDisp, 117
- line
 - Gui::Scene::Frame, 151
 - Gui::Scene::Mesh, 152
- LineParameters
 - Gui, 44
- link
 - Container, 102
 - List, 126
- link0
 - Container, 102
- List, 121
 - ~List, 122
 - append, 122
 - cleanup, 123
 - Current, 123
 - current, 128
 - erase, 123

- eraseall, 123
- First, 123
- first, 128
- FirstNext, 123
- FirstPrev, 124
- getFirstPointer, 124
- getLastPointer, 124
- getNext, 124
- getPointer, 124
- getPrev, 124
- go, 124
- goFirst, 124
- goFirstNext, 124
- goLast, 124
- goLastNext, 125
- goNext, 125
- goPrev, 125
- init, 125
- insert, 125
- isFirst, 125
- isFirstLast, 125
- isLast, 125
- Last, 126
- last, 128
- LastNext, 126
- LastPrev, 126
- link, 126
- List, 122
- locate, 126
- moveAfterFirst, 126
- moveBehindFirst, 126
- nelements, 128
- Next, 126
- number, 127
- prepend, 127
- Prev, 127
- remove, 127
- setFirst, 127
- setFirstLast, 127
- swapAfterFirst, 127
- unlink, 128
- list.h, 194
- lmax
 - Gui, 55
- lmin
 - Gui, 55
- load
 - Domain, 113
- LOCAL
 - def.h, 168
- locate
 - List, 126
- lock
 - Container, 103
- locked
 - Collection, 90
 - Container, 107
 - State, 156
- lx
 - Gui, 55
- ly
 - Gui, 55
- lz
 - Gui, 55
- magenta
 - Gui, 43
- main
 - job.cc, 193
 - test.cc, 206
 - view.cc, 207
- main.dox, 195
- Mass
 - Species::Specie, 154
- mass
 - Species::Specie, 155
- massvar
 - Gui, 45
- Materials
 - Gui, 49
- MAX
 - def.h, 168
- MAX_WINDOW_HEIGHT
 - gui.h, 187
- MAX_WINDOW_WIDTH
 - gui.h, 187
- maxAtomTypes
 - model.h, 199
- maxaxesprm
 - Gui, 43
- maxBound
 - Domain, 113
- maxBoundaryTypes
 - domain.h, 175
- maxcolor
 - Gui, 44
- maxColorSchemes
 - Gui, 44
- maxCounters
 - Collection, 88
- maxcounters
 - def.h, 171
- maxelements
 - Gui, 44
- maxlineprm
 - Gui, 44
- MAXLINLEN
 - def.h, 168

- maxnumber
 - Collection, 88
- maxpointprm
 - Gui, 45
- maxshowpars
 - Gui, 46
- maxsurfdisptypes
 - Gui, 46
- maxvalue
 - Gui::Scene::Color, 150
- mcells
 - GridContainer, 38
- mcells1
 - GridContainer, 39
- menu
 - Gui, 49
- mesh
 - Gui, 44
 - Gui::Scene, 149
 - Option, 135
- MIDDLE_BUTTON
 - gui.h, 187
- minBound
 - Domain, 114
- minvalue
 - Gui::Scene::Color, 150
- missed
 - Species, 71
- mitems
 - Collection, 91
- model.cc, 196
- model.dox, 197
- model.h, 198
 - AtomType, 199
 - carbon, 199
 - HARDBALLS, 199
 - helium, 199
 - hydrogen, 199
 - maxAtomTypes, 199
 - oxygen, 199
 - undefined, 199
- Molecule, 130
 - Coordinate, 131
 - Coordinates, 131
 - copy, 131
 - energy, 134
 - InternalEnergy, 131, 132
 - KineticEnergy, 132
 - Molecule, 131
 - Move, 132
 - setCoordinates, 132
 - setVelocity, 132
 - Temperature, 132
 - Type, 133
 - type, 134
 - v, 134
 - Velocity, 133
 - x, 134
- Molecules
 - Domain, 114
- molecules
 - Domain, 115
- motion
 - Gui, 49
- mouse
 - Gui, 49
- mouseButtons
 - Gui, 55
- Move
 - Molecule, 132
- moveAfterFirst
 - List, 126
- moveBehindFirst
 - List, 126
- Movement
 - Gui, 44
- movement
 - Gui, 55
- moveuv
 - Gui, 45
- movew
 - Gui, 45
- mptrs
 - Pool, 139
- MUL
 - def.h, 168
- MULC
 - def.h, 168
- MULVEC
 - def.h, 168
- ncells
 - GridContainer, 39
- nelements
 - List, 128
- Next
 - Collection, 88
 - Container, 103
 - List, 126
 - Species::Reaction, 143
- next
 - Item, 120
 - Pointer, 137
 - Ptr, 141
 - Species::Reaction::Outcome, 147
- nextstep
 - Run::Time, 157
- nitems

- Collection, 91
- node
 - Gui::Scene::Mesh, 152
- nodes
 - GridContainer, 39
 - Gui, 44
- normalize
 - Geom, 36
- nptrs
 - Container, 107
 - Pool, 139
- nspecies
 - Species, 71
- nthreads
 - Run, 69
- number
 - Collection, 88
 - Container, 103
 - List, 127
- nwdump
 - Gui, 56
- nxwdump
 - IO, 61
- OMP
 - def.h, 168
- OneOverSqrt2
 - def.h, 171
- OneOverSqrt3
 - def.h, 171
- onethird
 - def.h, 171
- openBoundary
 - domain.h, 175
- Option, 135
 - debug, 135
 - mesh, 135
 - restart, 135
 - tags, 135
 - verbose, 136
 - xterm, 136
- option
 - Run, 69
- Outcome
 - Species::Reaction::Outcome, 145
- outcomes
 - Species::Reaction, 143
- output
 - Run::Time, 158
- output.dox, 200
- outputfile
 - Run, 69
- outputname
 - Run, 70
- oxygen
 - model.h, 199
- Palette, 62
 - dvi, 62
 - init, 62
 - pickcolor, 62
 - vav, 62
 - vmn, 62
 - vmx, 63
- parseFloat
 - IO, 60
- parseInt
 - IO, 60
- parsename
 - job.cc, 193
- parseWord
 - IO, 60
- periodicBoundary
 - domain.h, 175
- PI
 - def.h, 168
- pickcolor
 - Palette, 62
- Pointer, 137
 - element, 137
 - next, 137
 - prev, 137
- PointParameters
 - Gui, 45
- points
 - Gui, 44
- Pool, 138
 - ~Pool, 138
 - check, 138
 - get, 139
 - hook, 139
 - mptrs, 139
 - nptrs, 139
 - Pool, 138
 - put, 139
 - size, 139
- pool
 - Domain, 116
 - GridContainer, 39
- Potential, 64
 - Cutoff, 64
 - cutoff, 66
 - derivative, 64
 - eta, 66
 - force, 65
 - invdist, 65
 - large, 66
 - lengthscale, 65

- sigma, 66
- small, 66
- strength, 65
- value, 65
- prepend
 - List, 127
- Prev
 - Collection, 88
 - Container, 103
 - List, 127
- prev
 - Item, 120
 - Pointer, 137
 - Ptr, 141
 - Run::Time, 158
- printGridVecLimits
 - Gui, 50
- printGridXLimits
 - Gui, 50
- printParticleVariables
 - Gui, 50
- printPartVelLimits
 - Gui, 50
- printVariables
 - Gui, 50
- Probability
 - Species::Reaction::Outcome, 146
- probability
 - Species::Reaction::Outcome, 148
- Product
 - Species::Reaction::Outcome, 147
- product
 - Species::Reaction::Outcome, 148
- Products
 - Species::Reaction::Outcome, 147
- programname
 - Run, 70
- Ptr, 141
 - element, 141
 - next, 141
 - prev, 141
- put
 - GridContainer, 38
 - Pool, 139
- query_extension
 - Gui, 50
- Quit
 - Gui, 50
- RAND_MAX
 - run.cc, 202
- reacted
 - Species, 71
- Reaction
 - Species::Reaction, 142
- reactions
 - Boundary, 76
 - Species, 71
- readcmdline
 - Run, 68
- readconf
 - Gui, 50
- readparam
 - Gui, 50
- REAL
 - def.h, 168
- red
 - Gui, 43
- refresh
 - Gui, 50
- relative
 - Gui::ColorScale, 92
- remove
 - Collection, 88, 89
 - Container, 103, 104
 - List, 127
- remove0
 - Collection, 89
 - Container, 104
- reshape
 - Gui, 51
- restart
 - Option, 135
- rgbcolor
 - Gui, 56
 - Gui::ElementDisp, 117
- RIGHT_BUTTON
 - gui.h, 187
- RND
 - def.h, 169
- rnd
 - Run, 68
- rotate
 - Gui, 45
- rotx
 - Gui, 56
- roty
 - Gui, 56
- Run, 67
 - configfile, 69
 - elapsed, 68
 - gauss, 68
 - init, 68
 - inputfile, 69
 - nthreads, 69
 - option, 69
 - outputfile, 69

- outputname, 70
- programname, 70
- readcmdline, 68
- rnd, 68
- seed, 69
- testrnd, 69
- time, 70
- usage, 69, 70
- worldtime, 70
- run
 - Domain, 114
 - Gui, 51
- run.cc, 201
- RAND_MAX, 202
- run.h, 203
- Run::Time, 157
 - current, 157
 - end, 157
 - nextstep, 157
 - output, 158
 - prev, 158
 - start, 158
 - step, 158
 - step0, 158
- runmany
 - Gui, 51
- save
 - Domain, 114
- scene
 - Gui, 56
- scheme
 - Gui::Scene::Color, 150
- SCLP
 - def.h, 169
- sclp
 - Geom, 36
- seed
 - Run, 69
- selectVariable
 - Gui, 51
- setBackgroundRun
 - Gui, 51
- setCoordinates
 - Molecule, 132
- setElementColor
 - Gui, 51
- setFirst
 - Collection, 89
 - Container, 104
 - List, 127
- setFirstLast
 - Collection, 89
 - Container, 105
 - List, 127
- setForegroundRun
 - Gui, 51
- setMaxBound
 - Domain, 114
- setMinBound
 - Domain, 114
- setVelocity
 - Molecule, 132
- showAxes
 - Gui, 45
- showBonds
 - Gui, 45
- showBoundaryFaceCenters
 - Gui, 45
- showBoundaryFaces
 - Gui, 45
- showBoundaryGrid
 - Gui, 45
- showBoundaryVectors
 - Gui, 45
- showBoundaryVertexes
 - Gui, 45
- showCellCenters
 - Gui, 45
- showFaceCenters
 - Gui, 45
- showFrame
 - Gui, 45
- showGrid
 - Gui, 45
- showGridElements
 - Gui, 51
- showIsoSurfaces
 - Gui, 45
- showNodes
 - Gui, 45
- showpar
 - Gui, 56
- ShowPars
 - Gui, 45
- showParticle
 - Gui, 52
- showRun
 - Gui, 45
- showSphere
 - Gui, 52
- showSpheres
 - Gui, 45
- showToolGrid
 - Gui, 45
- showToolVertexes
 - Gui, 45
- showVariables

- Gui, 45
- showVector
 - Gui, 52
- showVectorComponent
 - Gui, 52
- sigma
 - Potential, 66
- Size
 - Species::Specie, 154
- size
 - Gui, 45
 - Pool, 139
 - Species::Specie, 155
- sizevar
 - Gui, 45
- skyblue
 - Gui, 43
- SMALL
 - def.h, 169
- small
 - Potential, 66
- solidsurface
 - Gui, 46
- Specie
 - Species::Specie, 153
- specie
 - Species::Gas, 118
- Species, 71
 - annihilated, 71
 - collided, 71
 - Interaction, 71
 - missed, 71
 - nspecies, 71
 - reacted, 71
 - reactions, 71
 - species, 72
- species
 - Species, 72
- species.cc, 204
- species.h, 205
- Species::Gas, 118
 - density, 118
 - Gas, 118
 - specie, 118
- Species::Reaction, 142
 - ~Reaction, 143
 - Add, 143
 - current, 143
 - Erase, 143
 - First, 143
 - Next, 143
 - outcomes, 143
 - Reaction, 142
- Species::Reaction::Outcome, 145
 - ActivationEnergy, 146
 - activationEnergy, 147
 - Enthalpy, 146
 - enthalpy, 147
 - next, 147
 - Outcome, 145
 - Probability, 146
 - probability, 148
 - Product, 147
 - product, 148
 - Products, 147
 - Time, 147
 - time, 148
- Species::Specie, 153
 - Cp, 153
 - cp, 154
 - Id, 153
 - id, 154
 - inertia, 154
 - Mass, 154
 - mass, 155
 - Size, 154
 - size, 155
 - Specie, 153
- sqrt1p25
 - def.h, 171
- sqrt1p5
 - def.h, 171
- Sqrt2
 - def.h, 171
- Sqrt2Over2
 - def.h, 171
- Sqrt3
 - def.h, 171
- Sqrt3Over2
 - def.h, 172
- SQRTPIo8
 - def.h, 169
- start
 - Run::Time, 158
- State, 156
 - done, 156
 - locked, 156
- stay
 - Gui, 45
- step
 - Domain, 114
 - Gui, 56
 - Run::Time, 158
- step0
 - Run::Time, 158
- strength
 - Potential, 65
- SUB

- def.h, 169
- SUBC
 - def.h, 169
- SurfDispType
 - Gui, 46
- swapAfterFirst
 - List, 127
- switchColorScheme
 - Gui, 52
- tags
 - Option, 135
- Temperature
 - Boundary, 75
 - Bulk, 79
 - Molecule, 132
- temperature
 - Boundary, 76
 - Bulk, 80
- test.cc, 206
 - main, 206
- testrnd
 - Run, 69
- thickness
 - Gui, 44
- Time
 - Species::Reaction::Outcome, 147
- time
 - Run, 70
 - Species::Reaction::Outcome, 148
- TINY
 - def.h, 169
- toggleSpheres
 - Gui, 52
- toggleWindowDump
 - Gui, 52
- TRUE
 - def.h, 169
- twothirds
 - def.h, 172
- tx
 - Gui, 56
- ty
 - Gui, 56
- Type
 - Molecule, 133
- type
 - Boundary, 76
 - Molecule, 134
- undefined
 - model.h, 199
- unlink
 - Container, 105
- List, 128
- unlink0
 - Container, 105
- unlock
 - Container, 105
- usage
 - job.cc, 193
 - Run, 69, 70
 - view.cc, 208
- v
 - Molecule, 134
- value
 - Potential, 65
- variable
 - Gui, 45
- vars
 - Gui, 57
- vav
 - Palette, 62
- VECP
 - def.h, 170
- vecval
 - Gui, 57
- Velocity
 - Molecule, 133
- verbose
 - Option, 136
- view.cc, 207
 - main, 207
 - usage, 208
- vmax
 - Gui, 45
- vmin
 - Gui, 45
- vmn
 - Palette, 62
- vmx
 - Palette, 63
- VOIDSPECIE
 - def.h, 170
- Volume
 - Bulk, 79
- volume
 - Bulk, 80
- WAIT
 - def.h, 170
- wdtime
 - Gui, 57
- width
 - Gui::WindowGeom, 159
- win
 - Gui, 57

window
 Gui, 57
WINDOW_SIZE
 gui.h, 188
windowname
 Gui, 57
WORDLENGTH
 def.h, 170
worldtime
 Run, 70
writeData
 Gui, 52

x
 Molecule, 134
xaxislength
 Gui, 43
XMAX
 domain.h, 174
xmax
 Boundary, 77
 Bulk, 80
 GridContainer, 39
 Gui, 57
XMIN
 domain.h, 174
xmin
 Boundary, 77
 Bulk, 80
 GridContainer, 39
 Gui, 57
xo
 Gui, 57
xterm
 Option, 136
xwd
 IO, 61

yaxislength
 Gui, 43
yellow
 Gui, 43

zaxislength
 Gui, 43
ZERO
 def.h, 170
ZEROVEC
 def.h, 170
zoom
 Gui, 58