

GEM3D

Generated by Doxygen 1.6.1

Fri Aug 11 16:18:56 2017

Contents

1	Main Page	1
2	Directory Hierarchy	3
2.1	Directories	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	11
6.1	File List	11
7	Directory Documentation	13
7.1	/home/jhasbestan/Morton_Parallel_v0/src/include/ Directory Reference	13
7.2	/home/jhasbestan/Morton_Parallel_v0/src/ Directory Reference	14
8	Namespace Documentation	15
8.1	Abstraction Namespace Reference	15
8.1.1	Enumeration Type Documentation	15
8.1.1.1	DataType	15
8.2	stl Namespace Reference	16
8.2.1	Function Documentation	16
8.2.1.1	operator<<	16
8.2.1.2	operator<<	16
8.2.1.3	parse_float	16
8.2.1.4	parse_point	16

8.2.1.5	parse_stl	16
9	Class Documentation	17
9.1	CenterCoords Struct Reference	17
9.1.1	Detailed Description	17
9.1.2	Member Data Documentation	17
9.1.2.1	x	17
9.1.2.2	y	17
9.1.2.3	z	17
9.2	CommCollective< Type > Class Template Reference	18
9.2.1	Detailed Description	18
9.2.2	Constructor & Destructor Documentation	18
9.2.2.1	CommCollective	18
9.2.2.2	CommCollective	18
9.2.2.3	~CommCollective	19
9.2.3	Member Function Documentation	19
9.2.3.1	bcast	19
9.2.3.2	getTotalNumber	19
9.2.3.3	getTotalNumber	19
9.2.3.4	lbcast	19
9.2.3.5	IgetTotalNumber	19
9.2.3.6	IgetTotalNumber	19
9.2.3.7	waitOnRequest	19
9.2.4	Member Data Documentation	19
9.2.4.1	Com	19
9.2.4.2	Msg	19
9.3	CommPoint2Point< Type > Class Template Reference	21
9.3.1	Detailed Description	21
9.3.2	Constructor & Destructor Documentation	21
9.3.2.1	CommPoint2Point	21
9.3.2.2	CommPoint2Point	22
9.3.2.3	CommPoint2Point	22
9.3.2.4	~CommPoint2Point	22
9.3.3	Member Function Documentation	22
9.3.3.1	assignReciever	22
9.3.3.2	assignSender	22
9.3.3.3	getOffset	22

9.3.3.4	Irecv	22
9.3.3.5	Isend	22
9.3.3.6	myRank	23
9.3.3.7	mySize	23
9.3.3.8	recv	23
9.3.3.9	send	23
9.3.4	Member Data Documentation	23
9.3.4.1	Com	23
9.3.4.2	Msg	23
9.4	DataType Class Reference	24
9.4.1	Detailed Description	24
9.5	Forest< N, Nvalue, M, Mvalue > Class Template Reference	25
9.5.1	Detailed Description	26
9.5.2	Constructor & Destructor Documentation	26
9.5.2.1	Forest	26
9.5.2.2	~Forest	27
9.5.3	Member Function Documentation	27
9.5.3.1	assignGeom	27
9.5.3.2	checkGraphConsistency	27
9.5.3.3	checkNbrsOfNbrsConsistency	27
9.5.3.4	checkWithNbrs	27
9.5.3.5	combine	27
9.5.3.6	combinedLevel	27
9.5.3.7	comPatternConstruct	28
9.5.3.8	constructElementKeyForRcvdMessage	28
9.5.3.9	constructSeedKeyForRcvdMessage	28
9.5.3.10	convertBitsToDouble	28
9.5.3.11	convertDoubleToBits	28
9.5.3.12	createCommGraph	28
9.5.3.13	createNbrsOfNbrs	28
9.5.3.14	debug	28
9.5.3.15	debugDerefine	28
9.5.3.16	encodeGeometry	28
9.5.3.17	findFlipLevel	28
9.5.3.18	findSeedLevelForRcvdMessage	29
9.5.3.19	flipAll	29

9.5.3.20	flipForNbr	29
9.5.3.21	forestsize	29
9.5.3.22	fourToOneBalance	29
9.5.3.23	getDirections	29
9.5.3.24	getElemNbrs	29
9.5.3.25	getListEachTree	29
9.5.3.26	getMaxSeedsLevel	29
9.5.3.27	getNbrSeedLevel	29
9.5.3.28	getTotalSize	29
9.5.3.29	isInSeed	29
9.5.3.30	moveGeom	30
9.5.3.31	pushToDerefineEachTree	30
9.5.3.32	recoverAllZeroSingularity	30
9.5.3.33	refineEachTree	30
9.5.3.34	refineEachTreeVoxel	30
9.5.3.35	refineForestBalanced	31
9.5.3.36	removeAllZeroSingularity	31
9.5.3.37	retainFourToOneBalance	31
9.5.3.38	zoltanGeomrepart	31
9.5.4	Friends And Related Function Documentation	31
9.5.4.1	Phdf5	31
9.5.5	Member Data Documentation	31
9.5.5.1	ancestorcoords	31
9.5.5.2	ancestorlength	31
9.5.5.3	Com	31
9.5.5.4	destination	31
9.5.5.5	geom	31
9.5.5.6	graphComm	31
9.5.5.7	maxseedlevel	32
9.5.5.8	nbrsOfNbrs	32
9.5.5.9	npx	32
9.5.5.10	npv	32
9.5.5.11	npz	32
9.5.5.12	seeds	32
9.5.5.13	trees	32
9.5.5.14	zoltan_out	32

9.5.5.15	zz	32
9.6	GraphData Struct Reference	33
9.6.1	Detailed Description	33
9.6.2	Member Data Documentation	33
9.6.2.1	c	33
9.6.2.2	nbrGID	33
9.6.2.3	nbrIndex	33
9.6.2.4	nbrProc	33
9.6.2.5	numMyVertices	33
9.6.2.6	vertexGID	33
9.7	MeshData Struct Reference	34
9.7.1	Detailed Description	34
9.7.2	Member Data Documentation	34
9.7.2.1	c	34
9.7.2.2	myGlobalIDs	34
9.7.2.3	numGlobalPoints	34
9.7.2.4	numMyPoints	34
9.7.2.5	w	34
9.8	Message Struct Reference	35
9.8.1	Detailed Description	35
9.8.2	Member Function Documentation	35
9.8.2.1	print	35
9.8.3	Member Data Documentation	35
9.8.3.1	buf	35
9.8.3.2	count	35
9.8.3.3	datatype	35
9.8.3.4	reciever	35
9.8.3.5	request	36
9.8.3.6	sender	36
9.8.3.7	status	36
9.8.3.8	tag	36
9.9	MpiCom Struct Reference	37
9.9.1	Detailed Description	37
9.9.2	Constructor & Destructor Documentation	37
9.9.2.1	MpiCom	37
9.9.2.2	MpiCom	37

9.9.3	Member Data Documentation	37
9.9.3.1	comsize	37
9.9.3.2	mpicom	37
9.9.3.3	myrank	37
9.10	Phdf5< N, Nvalue, M, Mvalue > Class Template Reference	38
9.10.1	Detailed Description	38
9.10.2	Constructor & Destructor Documentation	38
9.10.2.1	Phdf5	38
9.10.2.2	~Phdf5	38
9.10.3	Member Function Documentation	38
9.10.3.1	writeMultiBlock	38
9.10.3.2	writePolyvertex	39
9.10.3.3	xdmfMultiBlock	39
9.10.3.4	xdmfPolyvertex	39
9.10.4	Member Data Documentation	39
9.10.4.1	totalnumber	39
9.11	stl::point Struct Reference	40
9.11.1	Detailed Description	40
9.11.2	Constructor & Destructor Documentation	40
9.11.2.1	point	40
9.11.2.2	point	40
9.11.3	Member Data Documentation	40
9.11.3.1	x	40
9.11.3.2	y	40
9.11.3.3	z	40
9.12	stl::stl_data Struct Reference	41
9.12.1	Detailed Description	41
9.12.2	Constructor & Destructor Documentation	41
9.12.2.1	stl_data	41
9.12.3	Member Data Documentation	41
9.12.3.1	name	41
9.12.3.2	triangles	41
9.13	Tree< N, value > Class Template Reference	42
9.13.1	Detailed Description	44
9.13.2	Constructor & Destructor Documentation	44
9.13.2.1	Tree	44

9.13.2.2	Tree	44
9.13.2.3	~Tree	44
9.13.3	Member Function Documentation	44
9.13.3.1	addToDerefineList	44
9.13.3.2	addToList	44
9.13.3.3	begin	44
9.13.3.4	centroid	44
9.13.3.5	clearMesh	45
9.13.3.6	clearMortonSTL	45
9.13.3.7	clearRefineList	45
9.13.3.8	construct	45
9.13.3.9	constructHigherLevelNbrs	45
9.13.3.10	constructNonlocalHigherLevelNbrs	45
9.13.3.11	convertCoordToMorton	45
9.13.3.12	convertStl2Morton	45
9.13.3.13	count	45
9.13.3.14	Dbegin	45
9.13.3.15	Dend	45
9.13.3.16	derefine	45
9.13.3.17	derefineDerefineList	46
9.13.3.18	derefineDerefineList	46
9.13.3.19	enclosingBox	46
9.13.3.20	end	46
9.13.3.21	extractBoundary	46
9.13.3.22	extractBoundaryP	46
9.13.3.23	find	46
9.13.3.24	findFlipLevel	46
9.13.3.25	findInDerefine	46
9.13.3.26	flipForNbr	47
9.13.3.27	flipRefineElemTag	47
9.13.3.28	fourToOne	47
9.13.3.29	fourToOneP	47
9.13.3.30	getDirections	47
9.13.3.31	getKey	47
9.13.3.32	insertKey	47
9.13.3.33	isBoundary	47

9.13.3.34	isBoundary	47
9.13.3.35	isInMeshList	47
9.13.3.36	isInRefineList	47
9.13.3.37	isInsideSolid	47
9.13.3.38	IsInVectorList	48
9.13.3.39	level	48
9.13.3.40	mortonSTLclear	48
9.13.3.41	printMesh	48
9.13.3.42	pushToDerefinelist	48
9.13.3.43	pushToRefinelist	48
9.13.3.44	Rbegin	48
9.13.3.45	readDerefineList	48
9.13.3.46	readRefineList	48
9.13.3.47	refine	48
9.13.3.48	refinelistReset	49
9.13.3.49	refineListSize	49
9.13.3.50	refineRefineList	49
9.13.3.51	refineRefineList	49
9.13.3.52	removeFromDerefineList	49
9.13.3.53	Rend	49
9.13.3.54	reserve	49
9.13.3.55	retainFourToOne	49
9.13.3.56	siblings	49
9.13.3.57	size	49
9.13.4	Friends And Related Function Documentation	49
9.13.4.1	Hdf5Xmf	49
9.13.5	Member Data Documentation	50
9.13.5.1	ancestorcoords	50
9.13.5.2	ancestorkey	50
9.13.5.3	ancestorlength	50
9.13.5.4	boundarylist	50
9.13.5.5	derefinelist	50
9.13.5.6	mesh	50
9.13.5.7	mortonSTL	50
9.13.5.8	npx	50
9.13.5.9	npv	50

9.13.5.10 npz	51
9.13.5.11 refinelist	51
9.14 stl::triangle Struct Reference	52
9.14.1 Detailed Description	52
9.14.2 Constructor & Destructor Documentation	52
9.14.2.1 triangle	52
9.14.3 Member Data Documentation	52
9.14.3.1 normal	52
9.14.3.2 v1	52
9.14.3.3 v2	52
9.14.3.4 v3	52
9.15 Vector3 Class Reference	53
9.15.1 Constructor & Destructor Documentation	53
9.15.1.1 Vector3	53
9.15.1.2 Vector3	53
9.15.1.3 ~Vector3	53
9.15.2 Member Function Documentation	53
9.15.2.1 Length	53
9.15.2.2 Normalize	53
9.15.2.3 Vectors	53
9.15.3 Member Data Documentation	53
9.15.3.1 X	53
9.15.3.2 Y	53
9.15.3.3 Z	53
9.16 Voxel< N, value > Class Template Reference	54
9.16.1 Detailed Description	54
9.16.2 Constructor & Destructor Documentation	54
9.16.2.1 Voxel	54
9.16.2.2 ~Voxel	54
9.16.3 Member Function Documentation	55
9.16.3.1 checkSiblingStatus	55
9.16.3.2 derefineGeomTree	55
9.16.3.3 distributeGeomToLeaves	55
9.16.3.4 generateSearchTree	55
9.16.3.5 IsInsideSegment	55
9.16.3.6 setLevel	55

9.16.4	Friends And Related Function Documentation	55
9.16.4.1	Hdf5XmfV	55
9.16.5	Member Data Documentation	55
9.16.5.1	lookup	55
9.16.5.2	maxlevel	55
9.16.5.3	mesh	56
9.16.5.4	numMax	56
9.17	Zoltan_Out Struct Reference	57
9.17.1	Detailed Description	57
9.17.2	Member Data Documentation	58
9.17.2.1	changes	58
9.17.2.2	exportGlobalGids	58
9.17.2.3	exportLocalGids	58
9.17.2.4	exportProcs	58
9.17.2.5	exportToPart	58
9.17.2.6	importGlobalGids	58
9.17.2.7	importLocalGids	58
9.17.2.8	importProcs	58
9.17.2.9	importToPart	58
9.17.2.10	numExport	58
9.17.2.11	numGidEntries	58
9.17.2.12	numImport	58
9.17.2.13	numLidEntries	58
9.17.2.14	parts	58
10	File Documentation	59
10.1	/home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp File Reference	59
10.1.1	Function Documentation	59
10.1.1.1	ConvertType	59
10.2	/home/jhasbestan/Morton_Parallel_v0/src/forest.cpp File Reference	60
10.2.1	Define Documentation	60
10.2.1.1	CHARACTER	60
10.2.1.2	REORDER	60
10.2.1.3	SENDBOOL	60
10.3	/home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h File Reference	61
10.4	/home/jhasbestan/Morton_Parallel_v0/src/include/datatype.h File Reference	62
10.4.1	Function Documentation	63

10.4.1.1	getAbstractionDataType	63
10.4.1.2	getAbstractionDataType< char >	63
10.4.1.3	getAbstractionDataType< double >	63
10.4.1.4	getAbstractionDataType< float >	63
10.4.1.5	getAbstractionDataType< int >	63
10.4.1.6	getAbstractionDataType< long >	63
10.4.1.7	getAbstractionDataType< nullptr_t >	63
10.4.1.8	getAbstractionDataType< short >	63
10.4.1.9	getAbstractionDataType< unsigned char >	63
10.4.1.10	getAbstractionDataType< unsigned int >	63
10.4.1.11	getAbstractionDataType< unsigned long >	63
10.4.1.12	getAbstractionDataType< unsigned short >	63
10.5	/home/jhasbestan/Morton_Parallel_v0/src/include/definitions.h File Reference	64
10.5.1	Define Documentation	65
10.5.1.1	BLUE	65
10.5.1.2	CYAN	65
10.5.1.3	GREEN	65
10.5.1.4	hash	65
10.5.1.5	MAGENTA	65
10.5.1.6	nonnative	65
10.5.1.7	RED	65
10.5.1.8	RESET	65
10.5.1.9	YELLOW	65
10.6	/home/jhasbestan/Morton_Parallel_v0/src/include/forest.h File Reference	66
10.7	/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h File Reference	67
10.7.1	Function Documentation	67
10.7.1.1	checkMesh	67
10.8	/home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h File Reference	68
10.9	/home/jhasbestan/Morton_Parallel_v0/src/include/tree.h File Reference	69
10.10	/home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h File Reference	70
10.10.1	Define Documentation	71
10.10.1.1	DEBUG	71
10.10.1.2	Rma	71
10.10.2	Typedef Documentation	71
10.10.2.1	Center_coords	71
10.10.3	Function Documentation	71

10.10.3.1 readSTLGeom	71
10.10.3.2 TwoPowN	71
10.10.3.3 zoltanGeometricPartitioner	71
10.11/home/jhasbestan/Morton_Parallel_v0/src/main.cpp File Reference	72
10.11.1 Define Documentation	72
10.11.1.1 PROCSIZE	72
10.11.1.2 TREESIZE	72
10.11.1.3 WEIGHT	72
10.11.1.4 ZOLTAN_GEOMETRIC_PARTITION	72
10.11.1.5 ZOLTAN_ON	72
10.11.2 Function Documentation	72
10.11.2.1 main	72
10.12/home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp File Reference	73
10.12.1 Define Documentation	73
10.12.1.1 CHECK_MESH	73
10.12.1.2 MYSCALE	73
10.12.2 Function Documentation	73
10.12.2.1 checkMesh	73
10.12.2.2 readSTLGeom	73
10.13/home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp File Reference	74
10.13.1 Define Documentation	74
10.13.1.1 H5FILE	74
10.13.1.2 H5FILE_NAME	74
10.13.1.3 offset0	74
10.13.1.4 offset1	74
10.13.1.5 XDMF_NAME	74
10.13.2 Function Documentation	74
10.13.2.1 integer_string	74
10.14/home/jhasbestan/Morton_Parallel_v0/src/tree.cpp File Reference	75
10.15/home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp File Reference	76
10.16/home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp File Reference	77
10.16.1 Define Documentation	78
10.16.1.1 TOL	78
10.16.2 Function Documentation	78
10.16.2.1 get_geometry_list	78
10.16.2.2 get_num_geometry	78

10.16.2.3 get_number_of_objects	78
10.16.2.4 get_object_list	78
10.16.2.5 zoltanGeometricPartitioner	78

Chapter 1

Main Page

"Paralle implementation of Forest of Octrees using Morton Encoding"

Part of NSF project: GEM3D

Required Libraries: HDF5, MPI, Zoltan, CMAKE

If the [.stl](#) file is too large to open with one processor, it is recommended to use a coarser version from a different processor topology, and then each processor can open the big file in parallel and only read the portion related to them

Usage: progName input/geometry *.[stl](#) <params.txt

Authors:

Jaber J. Hasbestan, Inanc Senocak
PI: Inanc Senocak

Date:

12 Feb 2017

Copyright (c) 2017 Mechanical and Bio-medical Engineering Department Boise State University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 2

Directory Hierarchy

2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

src	14
include	13

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Abstraction	15
stl	16

Chapter 4

Class Index

4.1 Class Hierarchy

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

CenterCoords	17
CommCollective< Type >	18
CommPoint2Point< Type >	21
DataType	24
Forest< N, Nvalue, M, Mvalue >	25
GraphData	33
MeshData	34
Message	35
MpiCom	37
Phdf5< N, Nvalue, M, Mvalue >	38
stl::point	40
stl::stl_data	41
Tree< N, value >	42
Voxel< N, value >	54
stl::triangle	52
Vector3	53
Zoltan_Out	57

Chapter 5

Class Index

5.1 Class List

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

CenterCoords (Stores the coordinate of the centroid of the elments)	17
CommCollective< Type > (Template wrapper around MPI functions for collective communica- tios)	18
CommPoint2Point< Type > (A template wrapper around MPI functions for point to point com- munication)	21
DataType (This class abstracts the MPI_Datatypes)	24
Forest< N, Nvalue, M, Mvalue > (Template class that is a forest of octrees with semi-structured process topology)	25
GraphData (Struct to supply information required by zoltan for geometric partitioners)	33
MeshData (Struct to supply information required by zoltan for geometric partitioners)	34
Message (Struct that embeds data related to the message and envelope)	35
MpiCom (Class for embedding data related to the communicator)	37
Phdf5< N, Nvalue, M, Mvalue > (This Writes out Tree data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit)	38
stl::point (Structure to hold coordinates of a point)	40
stl::stl_data (Structure to store vector of triangles read in from *.stl file)	41
Tree< N, value > (This Class Generates a 4:1 balancerd AMR mesh)	42
stl::triangle (Structure to store normals and vertices of a triangle)	52
Vector3	53
Voxel< N, value > (This Class Generates an unbalancerd Voxel to improve search by geometry partitioning)	54
Zoltan_Out (This structure is an interface to store the output from Zoltan)	57

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

/home/jhasbestan/Morton_Parallel_v0/src/ communicate.cpp	59
/home/jhasbestan/Morton_Parallel_v0/src/ forest.cpp	60
/home/jhasbestan/Morton_Parallel_v0/src/ main.cpp	72
/home/jhasbestan/Morton_Parallel_v0/src/ parse_stl.cpp	73
/home/jhasbestan/Morton_Parallel_v0/src/ phdf5.cpp	74
/home/jhasbestan/Morton_Parallel_v0/src/ tree.cpp	75
/home/jhasbestan/Morton_Parallel_v0/src/ voxel.cpp	76
/home/jhasbestan/Morton_Parallel_v0/src/ zoltan.cpp	77
/home/jhasbestan/Morton_Parallel_v0/src/include/ communicate.h	61
/home/jhasbestan/Morton_Parallel_v0/src/include/ datatype.h	62
/home/jhasbestan/Morton_Parallel_v0/src/include/ definitions.h	64
/home/jhasbestan/Morton_Parallel_v0/src/include/ forest.h	66
/home/jhasbestan/Morton_Parallel_v0/src/include/ parse_stl.h	67
/home/jhasbestan/Morton_Parallel_v0/src/include/ phdf5.h	68
/home/jhasbestan/Morton_Parallel_v0/src/include/ tree.h	69
/home/jhasbestan/Morton_Parallel_v0/src/include/ typedefs.h	70

Chapter 7

Directory Documentation

7.1 /home/jhasbestan/Morton_Parallel_v0/src/include/ Directory Reference

Files

- file [communicate.h](#)
- file [datatype.h](#)
- file [definitions.h](#)
- file [forest.h](#)
- file [parse_stl.h](#)
- file [phdf5.h](#)
- file [tree.h](#)
- file [typedefs.h](#)

7.2 /home/jhasbestan/Morton_Parallel_v0/src/ Directory Reference

Directories

- directory [include](#)

Files

- file [communicate.cpp](#)
- file [forest.cpp](#)
- file [main.cpp](#)
- file [parse_stl.cpp](#)
- file [phdf5.cpp](#)
- file [tree.cpp](#)
- file [voxel.cpp](#)
- file [zoltan.cpp](#)

Chapter 8

Namespace Documentation

8.1 Abstraction Namespace Reference

Enumerations

- enum `DataType` {
 `type_byte`, `type_char`, `type_unsigned_char`, `type_short`,
 `type_unsigned_short`, `type_int`, `type_unsigned_int`, `type_long`,
 `type_unsigned_long`, `type_float`, `type_double` }

8.1.1 Enumeration Type Documentation

8.1.1.1 enum `Abstraction::DataType`

Enumerator:

type_byte
type_char
type_unsigned_char
type_short
type_unsigned_short
type_int
type_unsigned_int
type_long
type_unsigned_long
type_float
type_double

8.2 stl Namespace Reference

Classes

- struct [point](#)
Structure to hold coordinates of a [point](#).
- struct [triangle](#)
structure to store normals and vertices of a [triangle](#)
- struct [stl_data](#)
*structure to store vector of triangles read in from *.[stl](#) file*

Functions

- `std::ostream & operator<< (std::ostream &out, const triangle &t)`
- `stl_data parse_stl (const std::string &stl_path)`
- `std::ostream & operator<< (std::ostream &out, const point p)`
- `float parse_float (std::ifstream &s)`
- `point parse_point (std::ifstream &s)`

8.2.1 Function Documentation

8.2.1.1 `std::ostream& stl::operator<< (std::ostream & out, const point p)`

8.2.1.2 `std::ostream & stl::operator<< (std::ostream & out, const triangle & t)`

8.2.1.3 `float stl::parse_float (std::ifstream & s)`

8.2.1.4 `point stl::parse_point (std::ifstream & s)`

8.2.1.5 `stl_data stl::parse_stl (const std::string & stl_path)`

Chapter 9

Class Documentation

9.1 CenterCoords Struct Reference

Stores the coordinate of the centroid of the elements.

```
#include <typedefs.h>
```

Public Attributes

- real [x](#)
- real [y](#)
- real [z](#)

9.1.1 Detailed Description

Stores the coordinate of the centroid of the elements.

9.1.2 Member Data Documentation

9.1.2.1 real CenterCoords::x

9.1.2.2 real CenterCoords::y

9.1.2.3 real CenterCoords::z

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h](#)

9.2 CommCollective< Type > Class Template Reference

is a template wrapper around MPI functions for collective communications

```
#include <communicate.h>
```

Public Member Functions

- [CommCollective](#) (void *buf, uint size, uint root)
- [CommCollective](#) (void *buff, uint size)
- void [bcast](#) ()
- void [lbcast](#) ()
- void [getTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)
- void [lgetTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)
- void [waitOnRequest](#) ()
- [~CommCollective](#) ()
- template<>
void [getTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)
- template<>
void [lgetTotalNumber](#) (uint *offset, uint *myvalue, uint *totalvalue)

Private Attributes

- [MpiCom Com](#)
- [Message Msg](#)

9.2.1 Detailed Description

```
template<class Type> class CommCollective< Type >
```

is a template wrapper around MPI functions for collective communications This class is specialized for collective communications and functions

9.2.2 Constructor & Destructor Documentation

9.2.2.1 `template<class Type > CommCollective< Type >::CommCollective (void * buf, uint size, uint root) [inline]`

constructor

Parameters:

root another constructor for Communicator class

9.2.2.2 `template<class Type > CommCollective< Type >::CommCollective (void * buff, uint size) [inline]`

constructor specialized for root=comsize-1

Parameters:

size another constructor for Communicator class, this one sets the root as comsize-1 for hdf5

9.2.2.3 `template<class Type > CommCollective< Type >::~~CommCollective () [inline]`

9.2.3 Member Function Documentation

9.2.3.1 `template<class Type > void CommCollective< Type >::bcast () [inline]`

blockin broadcast

9.2.3.2 `template<> void CommCollective< uint >::getTotalNumber (uint * offset, uint * myvalue, uint * totalvalue) [inline]`

< the largest offset belongs to the processor with highest rank, add this to its number of cubes will give us the total value

9.2.3.3 `template<class Type > void CommCollective< Type >::getTotalNumber (uint * offset, uint * myvalue, uint * totalvalue)`

calculates the world population for a given variable

9.2.3.4 `template<class Type > void CommCollective< Type >::Ibcast () [inline]`

non-blocking broadcast

I speozialized this function since this is only needed for unsigned ints

9.2.3.5 `template<> void CommCollective< uint >::IgetTotalNumber (uint * offset, uint * myvalue, uint * totalvalue) [inline]`

< the largest offset belongs to the processor with highest rank, add this to its number of cubes will give us the total value

9.2.3.6 `template<class Type > void CommCollective< Type >::IgetTotalNumber (uint * offset, uint * myvalue, uint * totalvalue)`

calculates the world population for a given variable in a non-blocking fashion

9.2.3.7 `template<class Type > void CommCollective< Type >::waitOnRequest () [inline]`

9.2.4 Member Data Documentation

9.2.4.1 `template<class Type > MPICom CommCollective< Type >::Com [private]`

9.2.4.2 `template<class Type > Message CommCollective< Type >::Msg [private]`

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp](#)

9.3 CommPoint2Point< Type > Class Template Reference

A template wrapper around MPI functions for point to point communication.

```
#include <communicate.h>
```

Public Member Functions

- [CommPoint2Point](#) (void *buff, uint size, uint *tg, uint snd, uint rcv, uint type, MPI_Comm com)
- [CommPoint2Point](#) (void *buff, uint size, uint snd, uint rcv)
- [CommPoint2Point](#) (void *buff, uint size)
- void [assignSender](#) (uint sndr)
- void [assignReceiver](#) (uint rcv)
- integer [myRank](#) ()
- integer [mySize](#) ()
- void [Irecv](#) ()
- void [Isend](#) ()
- void [recv](#) ()
- void [send](#) ()
- void [getOffset](#) (uint myvalue, uint *offset)
- [~CommPoint2Point](#) ()

Private Attributes

- [MpiCom Com](#)
- [Message Msg](#)

9.3.1 Detailed Description

`template<class Type> class CommPoint2Point< Type >`

A template wrapper around MPI functions for point to point communication. class [CommPoint2Point](#) abstracts point to point communications and functions

9.3.2 Constructor & Destructor Documentation

9.3.2.1 `template<class Type > CommPoint2Point< Type >::CommPoint2Point (void * buff, uint size, uint * tg, uint snd, uint rcv, uint type, MPI_Comm Comm) [inline]`

full constructor

< default tag for message send and receive is "0", MPI_ANY_TAG is only OK for receive command so it can not be used here

Parameters:

Comm If communicator is not specified it will be set as MPI_COMM_WORLD by default

9.3.2.2 `template<class Type > CommPoint2Point< Type >::CommPoint2Point (void * buff, uint size, uint snd, uint rcv) [inline]`

default constructor

Parameters:

rcv another constructor for Communicator class

9.3.2.3 `template<class Type > CommPoint2Point< Type >::CommPoint2Point (void * buff, uint size) [inline]`

deferring the assignment of sender and reciever

Parameters:

size another constructor for Communicator class, deferes assigning sender and reciever

9.3.2.4 `template<class Type > CommPoint2Point< Type >::~~CommPoint2Point () [inline]`

9.3.3 Member Function Documentation

9.3.3.1 `template<class Type > void CommPoint2Point< Type >::assignReciever (uint rcv) [inline]`

assigns the sender of the message

9.3.3.2 `template<class Type > void CommPoint2Point< Type >::assignSender (uint sndr) [inline]`

9.3.3.3 `template<class Type > void CommPoint2Point< Type >::getOffset (uint myvalue, uint * offset) [inline]`

calculates the offset of each processor for variable myvalue

Parameters:

offset gets the number of elements before the current processor, it is needed for globally defining the elements

9.3.3.4 `template<class Type > void CommPoint2Point< Type >::Irecv () [inline]`

non-blocking recieve

9.3.3.5 `template<class Type > void CommPoint2Point< Type >::Isend () [inline]`

non-blocking send

9.3.3.6 template<class Type > int CommPoint2Point< Type >::myRank () [inline]

assigns the destination the message gets the rank of the processor

9.3.3.7 template<class Type > int CommPoint2Point< Type >::mySize () [inline]**9.3.3.8 template<class Type > void CommPoint2Point< Type >::recv () [inline]**

blocking receive

9.3.3.9 template<class Type > void CommPoint2Point< Type >::send () [inline]

blocking send

9.3.4 Member Data Documentation**9.3.4.1 template<class Type > MPI_Comm CommPoint2Point< Type >::Com [private]****9.3.4.2 template<class Type > Message CommPoint2Point< Type >::Msg [private]**

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

- /home/jhasbestan/Morton_Parallel_v0/src/include/[communicate.h](#)
- /home/jhasbestan/Morton_Parallel_v0/src/[communicate.cpp](#)

9.4 DataType Class Reference

this class abstracts the MPI_Datatypes

```
#include <datatype.h>
```

9.4.1 Detailed Description

this class abstracts the MPI_Datatypes using template initializations one can template MPI functions using template specialization for further details see, <https://chuckaknight.wordpress.com/2013/03/13/intrinsic-type-conversion-using-templates/> since BYTE is not a native type in C++, for MPI_TYPE, the nullptr_t (null pointer type) is utilized

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

- /home/jhasbestan/Morton_Parallel_v0/src/include/[datatype.h](#)

9.5 Forest< N, Nvalue, M, Mvalue > Class Template Reference

template class that is a forest of octrees with semi-structured process topology

```
#include <forest.h>
```

Public Member Functions

- [Forest](#) (real *length, real *coords, [Tree](#)< M, Mvalue > &proc, uint nx, uint ny, uint nz)
- uint [getTotalSize](#) ()
- void [refineEachTreeVoxel](#) (uint nlevel)
- void [assignGeom](#) ([Tree](#)< M, Mvalue > &proc, real *geom_xyz, uint geom_nn)
- void [refineEachTree](#) (uint nlevel)
- void [getListEachTree](#) ()
- void [fourToOneBalance](#) ([Tree](#)< M, uint > &proc)
- bool [isInSeed](#) (morton< M > &key, uint *counter)
- void [flipAll](#) (morton< N > &key, uint *mylevel, uint *direction)
- void [findFlipLevel](#) (morton< N+M > key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [flipForNbr](#) (morton< N+M > &key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [combine](#) (const morton< N > &key, const morton< M > &seed, morton< M+N > &combinedkey)
- void [getNbrSeedLevel](#) (morton< N+M > &combinedkey, uint topologylevel, uint *nbrseedleve, [Tree](#)< M, uint > &proc)
- void [debug](#) ([Tree](#)< M, Mvalue > &proc)
- void [getElemNbrs](#) ([Tree](#)< M, Mvalue > &proc, const morton< M > key, bitvector< M > &nbr)
- void [comPatternConstruct](#) ([Tree](#)< M, Mvalue > &proc)
- void [getDirections](#) (morton< N+M > &key, uint combinedlevel, vector< uint > &directions)
- void [encodeGeometry](#) ()
- void [removeAllZeroSingularity](#) (morton< N+M > &key, const uint &combinedlevel)
- void [getMaxSeedsLevel](#) ([Tree](#)< M, Mvalue > &proc)
- void [findSeedLevelForRcvdMessage](#) (const morton< N+M > &key, uint *mylevel, [Tree](#)< M, Mvalue > &proc)
- void [recoverAllZeroSingularity](#) (morton< N+M > &key, const uint &combinedlevel)
- void [constructSeedKeyForRcvdMessage](#) (const morton< N+M > &key, const uint &seedlevel, morton< M > &seedkey)
- void [constructElementKeyForRcvdMessage](#) (const morton< N+M > &key, const uint &seedlevel, morton< N > &elementkey)
- void [refineForestBalanced](#) (uint nlevel, [Tree](#)< M, Mvalue > &proc)
- void [combinedLevel](#) (const morton< N+M > &key, uint *level)
- void [zoltanGeomrepart](#) ([Tree](#)< M, Mvalue > &proc, uint setmethod)
- uint [forestsize](#) ()
- void [retainFourToOneBalance](#) ([Tree](#)< M, uint > &proc)
- void [moveGeom](#) ([Tree](#)< M, Mvalue > &proc, real *geom_xyz, uint n, real x[3])
- void [pushToDerefineEachTree](#) (uint nlevel, [Tree](#)< M, uint > &proc)
- void [convertBitsToDouble](#) (morton< N+M > key, double *val)
- void [convertDoubleToBits](#) (morton< N+M > &key, const double val)
- void [createCommGraph](#) (uint Nnbr)
- void [createNbrsOfNbrs](#) ()
- void [debugDerefine](#) ([Tree](#)< M, Mvalue > &proc)
- void [checkGraphConsistency](#) ()
- void [checkNbrsOfNbrsConsistency](#) ()
- bool [checkWithNbrs](#) (bool *sendbuf, bool *recvbuf)
- [~Forest](#) ()

Protected Attributes

- real [ancestorlength](#) [3]
- real [ancestorcoords](#) [3]
- uint [npx](#)
- uint [npy](#)
- uint [npz](#)

Private Attributes

- [MpiCom Com](#)
- [treelist](#)< N, Nvalue > [trees](#)
- [bitlist](#)< M > [seeds](#)
- uint [maxseedlevel](#)
- [Tree](#)< M, real > [geom](#)
- [vector](#)< uint > [destination](#)
- [vector](#)< uint > [nbrsOfNbrs](#)
- struct Zoltan_Struct * [zz](#) = nullptr
- [Zoltan_Out](#) [zoltan_out](#)
- MPI_Comm [graphComm](#)

Friends

- class [Phdf5](#)

9.5.1 Detailed Description

template<size_t N, typename Nvalue, size_t M, typename Mvalue> class Forest< N, Nvalue, M, Mvalue >

template class that is a forest of octrees with semi-structured process topology includes a list of tree's and functionality for manipulation as well as exchange of the trees with neighboring processes The algorithm starts with a very coarse 16 bit semi-structured processor topology, for now only 16 bits are used but it can be modified according to the need each process will have one forest and trees will be distributed dynamically as the solution porogresses

9.5.2 Constructor & Destructor Documentation

9.5.2.1 **template<size_t N, typename Nvalue , size_t M, typename Mvalue > Forest< N, Nvalue, M, Mvalue >::Forest (real * *length*, real * *coords*, Tree< M, Mvalue > & *proc*, uint *nx*, uint *ny*, uint *nz*) [inline]**

constructor

<part I, initialize the ancestor coords and length, just like we did for class tree

9.5.2.2 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> Forest< N, Nvalue, M, Mvalue >::~~Forest () [inline]`

9.5.3 Member Function Documentation

9.5.3.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::assignGeom (Tree< M, Mvalue > & proc, real * geom_xyz, uint geom_nn) [inline]`

9.5.3.2 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::checkGraphConsistency () [inline]`

9.5.3.3 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::checkNbrsOfNbrsConsistency () [inline]`

9.5.3.4 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > bool Forest< N, Nvalue, M, Mvalue >::checkWithNbrs (bool * sendbuf, bool * recvbuf) [inline]`

this is to enforce broadcast only with first degree neighbors

Parameters:

recvbuf creates distributed (acalable) graph for communication

9.5.3.5 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue >::combine (const morton< N > & key, const morton< M > & seed, morton< M+N > & combinedkey)`

combined seed key with element key

9.5.3.6 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::combinedLevel (const morton< N+M > & key, uint * level) [inline]`

to prevent unnecessary bit operation, the morton code is placed from starting from left hand side

- 9.5.3.7 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::comPatternConstruct (Tree< M, Mvalue> & proc) [inline]`
- 9.5.3.8 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::constructElementKeyForRcvdMessage (const morton< N+M> & key, const uint & seedlevel, morton< N> & elementkey) [inline]`
- 9.5.3.9 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::constructSeedKeyForRcvdMessage (const morton< N+M> & key, const uint & seedlevel, morton< M> & seedkey) [inline]`
- 9.5.3.10 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::convertBitsToDouble (morton< N+M> key, double * val)`
- 9.5.3.11 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::convertDoubleToBits (morton< N+M> & key, const double val)`
- 9.5.3.12 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::createCommGraph (uint Lnbr) [inline]`

level of neighbors

Parameters:

Lnbr creates distributed (scalable) graph for communication

- 9.5.3.13 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::createNbrsOfNbrs () [inline]`
- 9.5.3.14 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::debug (Tree< M, Mvalue> & proc) [inline]`
- 9.5.3.15 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::debugDerefine (Tree< M, Mvalue> & proc) [inline]`
- 9.5.3.16 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::encodeGeometry () [inline]`
- 9.5.3.17 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::findFlipLevel (morton< N+M> key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

same as the function defined in class three except that it workd on (M+N) bits

9.5.3.18 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::findSeedLevelForRcvdMessage (const morton< N+M> & key, uint * mylevel, Tree< M, Mvalue> & proc) [inline]`

9.5.3.19 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::flipAll (morton< N> & key, uint * mylevel, uint * direction) [inline]`

9.5.3.20 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::flipForNbr (morton< N+M> & key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

same as the function defined in class three except that it workd on (M+N) bits

9.5.3.21 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue>::forestsize () [inline]`

9.5.3.22 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::fourToOneBalance (Tree< M, uint> & proc) [inline]`

4:1 balance enforced at forest including the other processors

9.5.3.23 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getDirections (morton< N+M> & key, uint combinedlevel, vector< uint> & directions) [inline]`

9.5.3.24 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getElemNbrs (Tree< M, Mvalue> & proc, const morton< M> key, bitvector< M> & nbr) [inline]`

collects the neighbors of a given element

9.5.3.25 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getListEachTree () [inline]`

9.5.3.26 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getMaxSeedsLevel (Tree< M, Mvalue> & proc) [inline]`

9.5.3.27 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::getNbrSeedLevel (morton< N+M> & combinedkey, uint topologylevel, uint * nbrseedleve, Tree< M, uint> & proc) [inline]`

9.5.3.28 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue>::getTotalSize () [inline]`

9.5.3.29 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> bool Forest< N, Nvalue, M, Mvalue>::isInSeed (morton< M> & key, uint * counter) [inline]`

check and see if the forest incldues the particluar seed

9.5.3.30 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::moveGeom (Tree< M, Mvalue> & proc, real * geom_xyz, uint n, real x[3]) [inline]`

moves the geomerty with displacements specified in *x*[3] in x,y and z directions

9.5.3.31 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::pushToDerefineEachTree (uint nlevel, Tree< M, uint> & proc) [inline]`

9.5.3.32 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::recoverAllZeroSingularity (morton< N+M> & key, const uint & combinedlevel) [inline]`

note that this function operates on the element key, this is done to remove redundant calc

9.5.3.33 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::refineEachTree (uint nlevel) [inline]`

Refines every [Tree](#) in the list, *nlevels*, balance is satisfired for each tree but not in the global scope

9.5.3.34 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> void Forest< N, Nvalue, M, Mvalue>::refineEachTreeVoxel (uint nlevel) [inline]`

Refines every [Tree](#) in the list, *nlevels*, balance is satisfired for each tree but not in the boundaries with other trees

- 9.5.3.35** `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::refineForestBalanced (uint nlevel, Tree< M, Mvalue > & proc) [inline]`
- 9.5.3.36** `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::removeAllZeroSingularity (morton< N+M > & key, const uint & combinedlevel) [inline]`
- 9.5.3.37** `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::retainFourToOneBalance (Tree< M, uint > & proc) [inline]`
- 9.5.3.38** `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Forest< N, Nvalue, M, Mvalue >::zoltanGeomrepart (Tree< M, Mvalue > & proc, uint setmethod) [inline]`

9.5.4 Friends And Related Function Documentation

- 9.5.4.1** `template<size_t N, typename Nvalue, size_t M, typename Mvalue> friend class Phdf5 [friend]`

9.5.5 Member Data Documentation

- 9.5.5.1** `template<size_t N, typename Nvalue, size_t M, typename Mvalue> real Forest< N, Nvalue, M, Mvalue >::ancestorcoords[3] [protected]`

centeroid of the of the first generation (root) element

- 9.5.5.2** `template<size_t N, typename Nvalue, size_t M, typename Mvalue> real Forest< N, Nvalue, M, Mvalue >::ancestorlength[3] [protected]`

original length of the first generation (root) element

- 9.5.5.3** `template<size_t N, typename Nvalue, size_t M, typename Mvalue> MPICom Forest< N, Nvalue, M, Mvalue >::Com [private]`

- 9.5.5.4** `template<size_t N, typename Nvalue, size_t M, typename Mvalue> vector<uint> Forest< N, Nvalue, M, Mvalue >::destination [private]`

- 9.5.5.5** `template<size_t N, typename Nvalue, size_t M, typename Mvalue> Tree<M, real> Forest< N, Nvalue, M, Mvalue >::geom [private]`

- 9.5.5.6** `template<size_t N, typename Nvalue, size_t M, typename Mvalue> MPI_Comm Forest< N, Nvalue, M, Mvalue >::graphComm [private]`

Each seed will contain its own geometry points to search, this assumption implicitly coincides processor topology with geometry voxelization this way tree's root will be morton code used in geometry voxelization and no extra operation is necessary

9.5.5.7 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::maxseedlevel` `[private]`

finds the maximum level of the seeds

9.5.5.8 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> vector<uint> Forest< N, Nvalue, M, Mvalue >::nbrsOfNbrs` `[private]`

9.5.5.9 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::npx` `[protected]`

discretization in x direction, this value for proc tree is 2, therefore forest needs its own value of npz

9.5.5.10 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::npy` `[protected]`

discretization in y direction

9.5.5.11 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> uint Forest< N, Nvalue, M, Mvalue >::npz` `[protected]`

discretization in z direction

9.5.5.12 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> bitlist<M> Forest< N, Nvalue, M, Mvalue >::seeds` `[private]`

seeds: morton code for boxes to grow tree

9.5.5.13 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> treelist<N, Nvalue> Forest< N, Nvalue, M, Mvalue >::trees` `[private]`

list of trees that each processor includes

9.5.5.14 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> Zoltan_Out Forest< N, Nvalue, M, Mvalue >::zoltan_out` `[private]`

9.5.5.15 `template<size_t N, typename Nvalue, size_t M, typename Mvalue> struct Zoltan_Struct* Forest< N, Nvalue, M, Mvalue >::zz = nullptr` `[read, private]`

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/forest.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/forest.cpp](#)

9.6 GraphData Struct Reference

struct to supply information required by zoltan for geometric partitioners

Public Attributes

- ZOLTAN_ID_TYPE [numMyVertices](#)
- ZOLTAN_ID_PTR [vertexGID](#)
- ZOLTAN_ID_PTR [nbrIndex](#)
- ZOLTAN_ID_PTR [nbrGID](#)
- int * [nbrProc](#)
- float * [c](#)

9.6.1 Detailed Description

struct to supply information required by zoltan for geometric partitioners

9.6.2 Member Data Documentation

9.6.2.1 float* GraphData::c

9.6.2.2 ZOLTAN_ID_PTR GraphData::nbrGID

9.6.2.3 ZOLTAN_ID_PTR GraphData::nbrIndex

9.6.2.4 int* GraphData::nbrProc

9.6.2.5 ZOLTAN_ID_TYPE GraphData::numMyVertices

9.6.2.6 ZOLTAN_ID_PTR GraphData::vertexGID

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

- [/home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp](#)

9.7 MeshData Struct Reference

struct to supply information required by zoltan for geometric partitioners

Public Attributes

- ZOLTAN_ID_TYPE [numGlobalPoints](#)
- ZOLTAN_ID_TYPE [numMyPoints](#)
- ZOLTAN_ID_PTR [myGlobalIDs](#)
- real * [c](#)
- real * [w](#)

9.7.1 Detailed Description

struct to supply information required by zoltan for geometric partitioners

9.7.2 Member Data Documentation

9.7.2.1 real* MeshData::c

9.7.2.2 ZOLTAN_ID_PTR MeshData::myGlobalIDs

9.7.2.3 ZOLTAN_ID_TYPE MeshData::numGlobalPoints

9.7.2.4 ZOLTAN_ID_TYPE MeshData::numMyPoints

9.7.2.5 real* MeshData::w

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

- [/home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp](#)

9.8 Message Struct Reference

struct that embeds data related to the message and envelope

```
#include <communicate.h>
```

Public Member Functions

- void [print](#) ()

Public Attributes

- uint [count](#)
- int [tag](#)
- uint [sender](#)
- uint [reciever](#)
- void * [buf](#)
- MPI_Datatype [datatype](#)
- MPI_Status [status](#)
- MPI_Request [request](#)

9.8.1 Detailed Description

struct that embeds data related to the message and envelope

9.8.2 Member Function Documentation

9.8.2.1 void Message::print () [inline]

9.8.3 Member Data Documentation

9.8.3.1 void* Message::buf

pointer to the buffer of the data

9.8.3.2 uint Message::count

count of the message

9.8.3.3 MPI_Datatype Message::datatype

9.8.3.4 uint Message::reciever

who is the destination

9.8.3.5 MPI_Request Message::request**9.8.3.6 uint Message::sender**

who is sending

9.8.3.7 MPI_Status Message::status**9.8.3.8 int Message::tag**

tag of the message

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h](#)

9.9 MpiCom Struct Reference

class for embedding data related to the communicator

```
#include <communicate.h>
```

Public Member Functions

- [MpiCom \(\)](#)
- [MpiCom \(MPI_Comm Com\)](#)

Public Attributes

- MPI_Comm [mpicom](#)
- integer [myrank](#)
- integer [comsize](#)

9.9.1 Detailed Description

class for embedding data related to the communicator

9.9.2 Constructor & Destructor Documentation

9.9.2.1 MpiCom::MpiCom () [inline]

9.9.2.2 MpiCom::MpiCom (MPI_Comm Com)

constructor second constructor

9.9.3 Member Data Documentation

9.9.3.1 integer MpiCom::comsize

size of the communicator

9.9.3.2 MPI_Comm MpiCom::mpicom

Communicator

9.9.3.3 integer MpiCom::myrank

rank of the processor

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h](#)

9.10 Phdf5< N, Nvalue, M, Mvalue > Class Template Reference

This Writes out [Tree](#) data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit.

```
#include <phdf5.h>
```

Public Member Functions

- [Phdf5](#) ()
- void [writePolyvertex](#) ([Forest](#)< N, Nvalue, M, Mvalue > &F, uint appx)
- void [xdmfPolyvertex](#) (integer my_rank, uint appx)
- void [writeMultiBlock](#) ([Forest](#)< N, Nvalue, M, Mvalue > &F, uint appx)
- void [xdmfMultiBlock](#) ([Forest](#)< N, Nvalue, M, Mvalue > &F, integer comsize, integer my_rank, uint offset, uint appx)
- [~Phdf5](#) ()

Private Attributes

- uint [totalnumber](#)

9.10.1 Detailed Description

```
template<size_t N, typename Nvalue, size_t M, typename Mvalue> class Phdf5< N, Nvalue, M, Mvalue >
```

This Writes out [Tree](#) data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > Phdf5< N, Nvalue, M, Mvalue >::Phdf5 () [inline]`

9.10.2.2 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > Phdf5< N, Nvalue, M, Mvalue >::~~Phdf5 () [inline]`

9.10.3 Member Function Documentation

9.10.3.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::writeMultiBlock (Forest< N, Nvalue, M, Mvalue > &F, uint appx) [inline]`

writes each element as block and mesh is combination of blocks, appx sets the appendix as string for the output file

<the forest size for each processor

9.10.3.2 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::writePolyvertex (Forest< N, Nvalue, M, Mvalue > & F, uint appx) [inline]`

writes only the centroids of the mesh, *appx* sets the appendix as string for the output file
<the forest size for each processor

9.10.3.3 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::xdmfMultiBlock (Forest< N, Nvalue, M, Mvalue > & F, integer comsize, integer my_rank, uint offset, uint appx) [inline]`

9.10.3.4 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > void Phdf5< N, Nvalue, M, Mvalue >::xdmfPolyvertex (integer my_rank, uint appx) [inline]`

9.10.4 Member Data Documentation

9.10.4.1 `template<size_t N, typename Nvalue , size_t M, typename Mvalue > uint Phdf5< N, Nvalue, M, Mvalue >::totalnumber [private]`

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp](#)

9.11 `stl::point` Struct Reference

Structure to hold coordinates of a [point](#).

```
#include <parse_stl.h>
```

Public Member Functions

- [point](#) ()
- [point](#) (float *xp*, float *yp*, float *zp*)

Public Attributes

- float [x](#)
- float [y](#)
- float [z](#)

9.11.1 Detailed Description

Structure to hold coordinates of a [point](#).

9.11.2 Constructor & Destructor Documentation

9.11.2.1 `stl::point::point ()` [`inline`]

z coordinate default constructor

9.11.2.2 `stl::point::point (float xp, float yp, float zp)` [`inline`]

constructor

9.11.3 Member Data Documentation

9.11.3.1 `float stl::point::x`

x coordinate

9.11.3.2 `float stl::point::y`

9.11.3.3 `float stl::point::z`

y coordinate

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h](#)

9.12 stl::stl_data Struct Reference

structure to store vector of triangles read in from *.[stl](#) file

```
#include <parse_stl.h>
```

Public Member Functions

- [stl_data](#) (std::string namep)

Public Attributes

- std::string [name](#)
- std::vector< [triangle](#) > [triangles](#)

9.12.1 Detailed Description

structure to store vector of triangles read in from *.[stl](#) file

9.12.2 Constructor & Destructor Documentation

9.12.2.1 [stl::stl_data::stl_data \(std::string *namep*\)](#) `[inline]`

9.12.3 Member Data Documentation

9.12.3.1 `std::string stl::stl_data::name`

9.12.3.2 `std::vector<triangle> stl::stl_data::triangles`

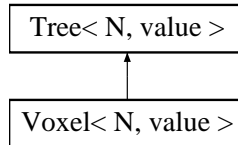
The documentation for this struct was generated from the following file:

- /home/jhasbestan/Morton_Parallel_v0/src/include/[parse_stl.h](#)

9.13 Tree< N, value > Class Template Reference

This Class Generates a 4:1 balancerd AMR mesh.

#include <tree.h>Inheritance diagram for Tree< N, value >::



Public Member Functions

- [Tree](#) (real *length, real *coords, uint nx, uint ny, uint nz)
- [Tree](#) ()
- void [construct](#) (real *length, real *coords, uint nx, uint ny, uint nz)
- void [level](#) (morton< N > key, uint *level)
- void [centroid](#) (morton< N > key, real *xyz)
- void [enclosingBox](#) (morton< N > key, real *X)
- bitmap< N, value >::iterator [begin](#) ()
- bitmap< N, value >::iterator [end](#) ()
- uint [size](#) ()
- void [reserve](#) (uint *reservedsize)
- void [siblings](#) (morton< N > key, uint mylevel, morton< N > *sibkey)
- void [refine](#) (morton< N > key)
- void [derefine](#) (morton< N > key)
- void [refineRefineList](#) ()
- void [fourToOneP](#) (uint istart, uint iend)
- void [refineRefineList](#) (uint istart, uint iend)
- void [fourToOne](#) ()
- void [findFlipLevel](#) (morton< N > key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- void [flipForNbr](#) (morton< N > *key, uint *mylevel, uint *changedirectionlevel, uint *direction)
- uint [IsInVectorList](#) (morton< N > key)
- void [addToList](#) (morton< N > key)
- uint [count](#) (morton< N > key)
- void [addToDerefineList](#) (morton< N > key)

If any of the siblings are listed in the dereffinement do not add to the list as dereffining one child means removing all the siblings.

- void [derefineDerefineList](#) (uint nlevel)
- uint [isInsideSolid](#) (const morton< N > key, const real *geom_xyz, uint n)
- bitmap< N, value >::iterator [find](#) (morton< N > key)
- void [insertKey](#) (morton< N > key)
- void [convertStd2Morton](#) (uint geom_size, real *geom_xyz)
- void [convertCoordToMorton](#) (real *xyz, morton< N > &key)
- void [pushToRefinelist](#) (uint level)
- bool [isBoundary](#) (morton< N > &key)
- void [extractBoundary](#) ()
- void [getDirections](#) (morton< N > &key, vector< uint > &directions)

- uint [refineListSize](#) ()
- void [clearRefineList](#) ()
- void [extractBoundaryP](#) (uint istart, uint iend)
- bool [isInMeshList](#) (const morton< N > &key)
- bool [isInRefineList](#) (const morton< N > &key)
- void [constructHigherLevelNbrs](#) (const morton< N > &key, const uint &keylevel, const uint &direction, morton< N > *nbr)
- void [printMesh](#) ()
- bool [isBoundary](#) (const morton< N > &key, uint direction)
- std::pair< morton< N >, int > [readRefineList](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- morton< N > [readDerefineList](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- void [getKey](#) (uint i, morton< N > &key)
- void [clearMortonSTL](#) ()
- void [retainFourToOne](#) ()
- void [removeFromDerefineList](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- unordered_map< morton< N >, int >::iterator [Dbegin](#) ()
- unordered_map< morton< N >, int >::iterator [Dend](#) ()
- void [derefineDerefineList](#) ()
- void [clearMesh](#) ()
- void [pushToDerefinelist](#) (uint nlevel)
- unordered_map< morton< N >, int >::iterator [Rbegin](#) ()
- unordered_map< morton< N >, int >::iterator [Rend](#) ()
- std::unordered_map< morton< N >, int >::iterator [findInDerefine](#) (morton< N > key)
- void [mortonSTLclear](#) ()
- void [flipRefineElemTag](#) (typename std::unordered_map< morton< N >, int >::iterator it)
- void [refinelistReset](#) ()
- void [constructNonlocalHigherLevelNbrs](#) (const morton< N > &key, const uint &keylevel, const uint &direction, morton< N > *nbr)
- [~Tree](#) ()

Public Attributes

- bitvector< N > [boundarylist](#)

Protected Attributes

- real [ancestorlength](#) [3]
- real [ancestorcoords](#) [3]
- morton< N > [ancestorkey](#) = 0
- uint [npx](#)
- uint [npz](#)
- uint [npz](#)

Private Attributes

- bitmap< N, value > [mesh](#)
- std::unordered_map< morton< N >, int > [refinelist](#)
- bitvector< N > [mortonSTL](#)
- std::unordered_map< morton< N >, int > [derefinelist](#)

Friends

- class [Hdf5Xmf](#)

9.13.1 Detailed Description

template<size_t N, typename value> class Tree< N, value >

This Class Generates a 4:1 balancerd AMR mesh.

9.13.2 Constructor & Destructor Documentation

9.13.2.1 template<size_t N, typename value > Tree< N, value >::Tree (real * *length*, real * *coords*, uint *nx*, uint *ny*, uint *nz*) [inline]

constructor

9.13.2.2 template<size_t N, typename value> Tree< N, value >::Tree () [inline]

9.13.2.3 template<size_t N, typename value > Tree< N, value >::~~Tree () [inline]

Destructor of the class, it frees the memeory pointed by pointer in the hashmap value if allocated

9.13.3 Member Function Documentation

9.13.3.1 template<size_t N, typename value > void Tree< N, value >::addToDerefineList (morton< N > *key*) [inline]

If any of the siblings are listed in the dereffinement do not add to the list as derefining one child means removing all the siblings. adds the element to derefinelist

9.13.3.2 template<size_t N, typename value > void Tree< N, value >::addToList (morton< N > *key*) [inline]

adds element to refinelist

9.13.3.3 template<size_t N, typename value > bitmap< N, value >::iterator Tree< N, value >::begin () [inline]

iterator returning the first object

9.13.3.4 template<size_t N, typename value > void Tree< N, value >::centroid (morton< N > *key*, real * *xyz*) [inline]

calculates the centroid of the cube given the morton key of the element

9.13.3.5 `template<size_t N, typename value > void Tree< N, value >::clearMesh () [inline]`

9.13.3.6 `template<size_t N, typename value > void Tree< N, value >::clearMortonSTL () [inline]`

9.13.3.7 `template<size_t N, typename value > void Tree< N, value >::clearRefineList () [inline]`

9.13.3.8 `template<size_t N, typename value > void Tree< N, value >::construct (real * length, real * coords, uint nx, uint ny, uint nz) [inline]`

need to initialize inside forest

9.13.3.9 `template<size_t N, typename value > void Tree< N, value >::constructHigherLevelNbrs (const morton< N > & key, const uint & keylevel, const uint & direction, morton< N > * nbr) [inline]`

9.13.3.10 `template<size_t N, typename value > void Tree< N, value >::constructNonlocalHigherLevelNbrs (const morton< N > & key, const uint & keylevel, const uint & direction, morton< N > * nbr) [inline]`

9.13.3.11 `template<size_t N, typename value > void Tree< N, value >::convertCoordToMorton (real * xyz, morton< N > & key) [inline]`

converts coordinates of a point to morton code

!< this function is to find a value given the key

9.13.3.12 `template<size_t N, typename value > void Tree< N, value >::convertStd2Morton (uint geom_size, real * geom_xyz) [inline]`

converts [stl](#) coordinates to morton and puts them in morton STL

!< this function is to find a value given the key

9.13.3.13 `template<size_t N, typename value > uint Tree< N, value >::count (morton< N > key) [inline]`

counts the number of elements

9.13.3.14 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator Tree< N, value >::Dbegin () [inline]`

9.13.3.15 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator Tree< N, value >::Dend () [inline]`

9.13.3.16 `template<size_t N, typename value > void Tree< N, value >::derefine (morton< N > key) [inline]`

if the morton code does not exist in mesh, refinement is not permitted (derefining a nonexisting element not permitted) Also, if any of the siblings have a higher level of refinement, derefinement is ignored

< if the key does not exist simply ignore doing anything

9.13.3.17 `template<size_t N, typename value > void Tree< N, value >::derefineDerefineList ()
[inline]`

9.13.3.18 `template<size_t N, typename value> void Tree< N, value >::derefineDerefineList (uint
nlevel)`

Derefiners the mesh

9.13.3.19 `template<size_t N, typename value > void Tree< N, value >::enclosingBox (morton< N
> key, real * X) [inline]`

calculates the range that an element occupies in 3D space for a given Element

9.13.3.20 `template<size_t N, typename value > bitmap< N, value >::iterator Tree< N, value
>::end () [inline]`

iterator returning the last object

9.13.3.21 `template<size_t N, typename value > void Tree< N, value >::extractBoundary ()
[inline]`

9.13.3.22 `template<size_t N, typename value > void Tree< N, value >::extractBoundaryP (uint
istart, uint iend) [inline]`

9.13.3.23 `template<size_t N, typename value > bitmap< N, value >::iterator Tree< N, value
>::find (morton< N > key) [inline]`

this function is to find a value given the key

!< this function is to find a value given the key

9.13.3.24 `template<size_t N, typename value > void Tree< N, value >::findFlipLevel (morton< N
> key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

detects the flip level, this info used in finding nonlocal neighbors

9.13.3.25 `template<size_t N, typename value > std::unordered_map< morton< N >, int
>::iterator Tree< N, value >::findInDerefine (morton< N > key) [inline]`

Get the iterator from derefinelist

Parameters:

key E;imate from derefinelist

9.13.3.26 `template<size_t N, typename value > void Tree< N, value >::flipForNbr (morton< N > * key, uint * mylevel, uint * changedirectionlevel, uint * direction) [inline]`

perform the actual operation to identify the nonlocal nbr

9.13.3.27 `template<size_t N, typename value > void Tree< N, value >::flipRefineElemTag (typename std::unordered_map< morton< N >, int >::iterator it) [inline]`

9.13.3.28 `template<size_t N, typename value > void Tree< N, value >::fourToOne () [inline]`

imposes 4:1 balance given the list of elements to be refined in the vector refine list

< all we are interested is the nonlocal neighbors, i.e. the neighbors of the parents as siblings will have same level

< this approach eliminates search algorithm as now we do not have the restrictions on cutting the cube that we had in the previous approach

9.13.3.29 `template<size_t N, typename value> void Tree< N, value >::fourToOneP (uint istart, uint iend)`

imposes 4:1 balance locally for each tree, while loop is eliminated due to parallel implementation

9.13.3.30 `template<size_t N, typename value > void Tree< N, value >::getDirections (morton< N > & key, vector< uint > & directions) [inline]`

9.13.3.31 `template<size_t N, typename value > void Tree< N, value >::getKey (uint i, morton< N > & key) [inline]`

9.13.3.32 `template<size_t N, typename value > void Tree< N, value >::insertKey (morton< N > key) [inline]`

9.13.3.33 `template<size_t N, typename value > bool Tree< N, value >::isBoundary (const morton< N > & key, uint direction) [inline]`

9.13.3.34 `template<size_t N, typename value > bool Tree< N, value >::isBoundary (morton< N > & key) [inline]`

9.13.3.35 `template<size_t N, typename value > bool Tree< N, value >::isInMeshList (const morton< N > & key) [inline]`

9.13.3.36 `template<size_t N, typename value > bool Tree< N, value >::isInRefineList (const morton< N > & key) [inline]`

9.13.3.37 `template<size_t N, typename value > uint Tree< N, value >::isInsideSolid (const morton< N > key, const real * geom_xyz, uint n) [inline]`

tags the elements if any points of the geometry resides in the enclosing box

9.13.3.38 `template<size_t N, typename value > uint Tree< N, value >::IsInVectorList (morton< N > key) [inline]`

checks to see if a given code is already in the list

9.13.3.39 `template<size_t N, typename value > void Tree< N, value >::level (morton< N > key, uint * level) [inline]`

obtains the level of the element from morton key

to prevent unnecessary bit operation, the morton code is placed from starting from left hand side

now look and see if any siblings exist

9.13.3.40 `template<size_t N, typename value > void Tree< N, value >::mortonSTLclear () [inline]`

9.13.3.41 `template<size_t N, typename value > void Tree< N, value >::printMesh () [inline]`

9.13.3.42 `template<size_t N, typename value > void Tree< N, value >::pushToDerefinelist (uint nlevel) [inline]`

!< this function is to find a value given the key

9.13.3.43 `template<size_t N, typename value > void Tree< N, value >::pushToRefinelist (uint nlevel) [inline]`

Note that for dynamic mesh we need to make sure the element exists before adding to this list

!< this function is to find a value given the key

9.13.3.44 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator Tree< N, value >::Rbegin () [inline]`

9.13.3.45 `template<size_t N, typename value > morton< N > Tree< N, value >::readDerefineList (typename std::unordered_map< morton< N >, int >::iterator it) [inline]`

9.13.3.46 `template<size_t N, typename value > std::pair< morton< N >, int > Tree< N, value >::readRefineList (typename std::unordered_map< morton< N >, int >::iterator it) [inline]`

9.13.3.47 `template<size_t N, typename value > void Tree< N, value >::refine (morton< N > key) [inline]`

performs refinement for a tagged element given the Morton Key

9.13.3.48 `template<size_t N, typename value > void Tree< N, value >::refinelistReset ()`
`[inline]`

9.13.3.49 `template<size_t N, typename value > uint Tree< N, value >::refineListSize ()`
`[inline]`

9.13.3.50 `template<size_t N, typename value > void Tree< N, value >::refineRefineList (uint`
`istart, uint iend) [inline]`

performs the refinement

9.13.3.51 `template<size_t N, typename value > void Tree< N, value >::refineRefineList ()`
`[inline]`

performs derefinement on a single element given a morton key performs the refinement

9.13.3.52 `template<size_t N, typename value > void Tree< N, value >::removeFromDerefineList`
`(typename std::unordered_map< morton< N >, int >::iterator it) [inline]`

Parameters:

it E;iminate from derefinelist

9.13.3.53 `template<size_t N, typename value > unordered_map< morton< N >, int >::iterator`
`Tree< N, value >::Rend () [inline]`

9.13.3.54 `template<size_t N, typename value > void Tree< N, value >::reserve (uint *`
`reservedsize) [inline]`

this function reserves the memory given the reservedsize of the mesh

9.13.3.55 `template<size_t N, typename value > void Tree< N, value >::retainFourToOne ()`
`[inline]`

9.13.3.56 `template<size_t N, typename value > void Tree< N, value >::siblings (morton< N >`
`key, uint mylevel, morton< N > * sibkey) [inline]`

extracts the siblings from morton code

9.13.3.57 `template<size_t N, typename value > uint Tree< N, value >::size () [inline]`

returns the size of the mesh

9.13.4 Friends And Related Function Documentation

9.13.4.1 `template<size_t N, typename value> friend class Hdf5Xmf [friend]`

this is a friend class to write out in hdf5 format

9.13.5 Member Data Documentation

9.13.5.1 `template<size_t N, typename value> real Tree< N, value >::ancestorcoords[3]`
`[protected]`

centroid of the of the first generation (root) element

9.13.5.2 `template<size_t N, typename value> morton<N> Tree< N, value >::ancestorkey = 0`
`[protected]`

root value is always set as 00000000000000

9.13.5.3 `template<size_t N, typename value> real Tree< N, value >::ancestorlength[3]`
`[protected]`

original length of the first generation (root) element

9.13.5.4 `template<size_t N, typename value> bitvector<N> Tree< N, value >::boundarylist`

list of elements of refinelist that are boundary elements

9.13.5.5 `template<size_t N, typename value> std::unordered_map<morton<N>,int> Tree< N, value >::derefineList` `[private]`

list of elements tagged to be removed, due to 4:1 balance this list

9.13.5.6 `template<size_t N, typename value> bitmap<N, value> Tree< N, value >::mesh`
`[private]`

base main container for hashmap

Reimplemented in [Voxel< N, value >](#).

9.13.5.7 `template<size_t N, typename value> bitvector<N> Tree< N, value >::mortonSTL`
`[private]`

vector to store geometric points in morton code

9.13.5.8 `template<size_t N, typename value> uint Tree< N, value >::npx` `[protected]`

discretization in x direction

9.13.5.9 `template<size_t N, typename value> uint Tree< N, value >::npy` `[protected]`

discretization in y direction

9.13.5.10 `template<size_t N, typename value> uint Tree< N, value >::npz` `[protected]`

discretization in y direction

9.13.5.11 `template<size_t N, typename value> std::unordered_map<morton<N>,int > Tree< N, value >::refinelist` `[private]`

list of elements tagged to be refined

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/tree.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/tree.cpp](#)

9.14 `stl::triangle` Struct Reference

structure to store normals and vertices of a [triangle](#)

```
#include <parse_stl.h>
```

Public Member Functions

- [triangle](#) ([point](#) normalp, [point](#) v1p, [point](#) v2p, [point](#) v3p)

Public Attributes

- [point](#) normal
- [point](#) v1
- [point](#) v2
- [point](#) v3

9.14.1 Detailed Description

structure to store normals and vertices of a [triangle](#)

9.14.2 Constructor & Destructor Documentation

9.14.2.1 `stl::triangle::triangle (point normalp, point v1p, point v2p, point v3p)` [`inline`]

9.14.3 Member Data Documentation

9.14.3.1 `point stl::triangle::normal`

normal

9.14.3.2 `point stl::triangle::v1`

coordinates of the vertex 1

9.14.3.3 `point stl::triangle::v2`

coordinates of the vertex 2

9.14.3.4 `point stl::triangle::v3`

coordinates of the vertex 3

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

- `/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h`

9.15 Vector3 Class Reference

```
#include <parse_stl.h>
```

Public Member Functions

- [Vector3](#) (void)
- [Vector3](#) (float [X](#), float [Y](#), float [Z](#))
- [~Vector3](#) (void)
- float [Length](#) ()
- [Vector3 Normalize](#) ()
- [Vector3 Vectors](#) ()

Public Attributes

- float [X](#)
- float [Y](#)
- float [Z](#)

9.15.1 Constructor & Destructor Documentation

9.15.1.1 [Vector3::Vector3](#) (void)

9.15.1.2 [Vector3::Vector3](#) (float *X*, float *Y*, float *Z*)

9.15.1.3 [Vector3::~~Vector3](#) (void)

9.15.2 Member Function Documentation

9.15.2.1 float [Vector3::Length](#) ()

9.15.2.2 [Vector3 Vectors::Normalize](#) ()

9.15.2.3 [Vector3 Vectors::Vectors](#) ()

9.15.3 Member Data Documentation

9.15.3.1 float [Vector3::X](#)

9.15.3.2 float [Vector3::Y](#)

9.15.3.3 float [Vector3::Z](#)

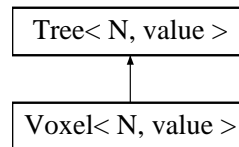
The documentation for this class was generated from the following files:

- [/home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp](#)

9.16 Voxel< N, value > Class Template Reference

This Class Generates an unbalanced [Voxel](#) to improve search by geometry partitioning.

#include <tree.h> Inheritance diagram for Voxel< N, value >::



Public Member Functions

- [Voxel](#) (real *length, real *coords)
- void [setLevel](#) (uint *l)
- void [generateSearchTree](#) (real *geom_xyz, uint n)
- void [distributeGeomToLeaves](#) (real *geom_xyz, uint n)
- uint [checkSiblingStatus](#) (morton< N > key, morton< N > *sibkey)
- void [derefineGeomTree](#) ()
- bool [IsInsideSegment](#) (morton< N > key, real *xyz)
- [~Voxel](#) ()

Private Attributes

- uint [maxlevel](#)
- uint [numMax](#)
- bitmap< N, value > [mesh](#)
- bitvector< N > [lookup](#)

Friends

- class [Hdf5XmfV](#)

9.16.1 Detailed Description

template<size_t N, typename value> class Voxel< N, value >

This Class Generates an unbalanced [Voxel](#) to improve search by geometry partitioning.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 template<size_t N, typename value> Voxel< N, value >::Voxel (real * *length*, real * *coords*) [inline]

9.16.2.2 template<size_t N, typename value > Voxel< N, value >::~~Voxel () [inline]

Destructor of this class

9.16.3 Member Function Documentation

9.16.3.1 `template<size_t N, typename value> uint Voxel< N, value >::checkSiblingStatus (morton< N > key, morton< N > * sibkey) [inline]`

Parameters:

sibkey checks to see if siblings include any points and whether they have the same level

9.16.3.2 `template<size_t N, typename value> void Voxel< N, value >::derefineGeomTree () [inline]`

checks to see if siblings include any points and whether they have the same level derefines the tree based on geometry

9.16.3.3 `template<size_t N, typename value> void Voxel< N, value >::distributeGeomToLeaves (real * geom_xyz, uint n) [inline]`

distributes geometry to different cells (leaves)

9.16.3.4 `template<size_t N, typename value> void Voxel< N, value >::generateSearchTree (real * geom_xyz, uint n) [inline]`

generates an initial tree

9.16.3.5 `template<size_t N, typename value> bool Voxel< N, value >::IsInsideSegment (morton< N > key, real * xyz) [inline]`

9.16.3.6 `template<size_t N, typename value> void Voxel< N, value >::setLevel (uint * l) [inline]`

constructor sets the maximum level for refinement

9.16.4 Friends And Related Function Documentation

9.16.4.1 `template<size_t N, typename value> friend class Hdf5XmfV [friend]`

this is a friend class to write out in hdf5 format

9.16.5 Member Data Documentation

9.16.5.1 `template<size_t N, typename value> bitvector<N> Voxel< N, value >::lookup [private]`

9.16.5.2 `template<size_t N, typename value> uint Voxel< N, value >::maxlevel [private]`

maximum level of refinement

9.16.5.3 `template<size_t N, typename value> bitmap<N, value> Voxel< N, value >::mesh` `[private]`

base main container for hashmap

Reimplemented from [Tree< N, value >](#).

9.16.5.4 `template<size_t N, typename value> uint Voxel< N, value >::numMax` `[private]`

number of elements having the highest level

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

- [/home/jhasbestan/Morton_Parallel_v0/src/include/tree.h](#)
- [/home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp](#)

9.17 Zoltan_Out Struct Reference

This structure is an interface to store the output from Zoltan.

```
#include <typedefs.h>
```

Public Attributes

- int [changes](#)
- int [numGidEntries](#)
- int [numLidEntries](#)
- int [numImport](#)
- int [numExport](#)
- unsigned int * [importGlobalGids](#)
- unsigned int * [importLocalGids](#)
- unsigned int * [exportGlobalGids](#)
- unsigned int * [exportLocalGids](#)
- int * [importProcs](#)
- int * [importToPart](#)
- int * [exportProcs](#)
- int * [exportToPart](#)
- int * [parts](#)

9.17.1 Detailed Description

This structure is an interface to store the output from Zoltan.

9.17.2 Member Data Documentation

- 9.17.2.1 `int Zoltan_Out::changes`
- 9.17.2.2 `unsigned int* Zoltan_Out::exportGlobalGids`
- 9.17.2.3 `unsigned int* Zoltan_Out::exportLocalGids`
- 9.17.2.4 `int* Zoltan_Out::exportProcs`
- 9.17.2.5 `int* Zoltan_Out::exportToPart`
- 9.17.2.6 `unsigned int* Zoltan_Out::importGlobalGids`
- 9.17.2.7 `unsigned int* Zoltan_Out::importLocalGids`
- 9.17.2.8 `int* Zoltan_Out::importProcs`
- 9.17.2.9 `int* Zoltan_Out::importToPart`
- 9.17.2.10 `int Zoltan_Out::numExport`
- 9.17.2.11 `int Zoltan_Out::numGidEntries`
- 9.17.2.12 `int Zoltan_Out::numImport`
- 9.17.2.13 `int Zoltan_Out::numLidEntries`
- 9.17.2.14 `int* Zoltan_Out::parts`

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

- `/home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h`

Chapter 10

File Documentation

10.1 /home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp File Reference

```
#include "communicate.h"  
#include "datatype.h"
```

Functions

- static MPI_Datatype [ConvertType](#) ([Abstraction::DataType](#) type)

10.1.1 Function Documentation

10.1.1.1 static MPI_Datatype ConvertType ([Abstraction::DataType](#) *type*) [static]

communicate class member functions [CommPoint2Point](#) This Class is a wrapper around MPI functions used in this project

Templating the "Intrinsic Type Conversion Using Template Specialization" message tag and mpi_-communicators are assigned by default if user doesnt assign them default tag is 0 and default Communicator is MPI_COMM_WORLD

10.2 /home/jhasbestan/Morton_Parallel_v0/src/forest.cpp File Reference

```
#include "forest.h"  
#include "definitions.h"
```

Defines

- #define [SENDBOOL](#) 1
- #define [CHARACTER](#) 2
- #define [REORDER](#) 0

10.2.1 Define Documentation

10.2.1.1 #define CHARACTER 2

10.2.1.2 #define REORDER 0

10.2.1.3 #define SENDBOOL 1

10.3 /home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h File Reference

```
#include "definitions.h"
```

Classes

- struct [MpiCom](#)
class for embedding data related to the communicator
- struct [Message](#)
struct that embeds data related to the message and envelope
- class [CommPoint2Point< Type >](#)
A template wrapper around MPI functions for point to point communication.
- class [CommCollective< Type >](#)
is a template wrapper around MPI functions for collective communicatios

10.4 /home/jhasbestan/Morton_Parallel_v0/src/include/datatype.h File Reference

```
#include "definitions.h"
```

Namespaces

- namespace [Abstraction](#)

Enumerations

- enum [Abstraction::DataType](#) {
 [Abstraction::type_byte](#), [Abstraction::type_char](#), [Abstraction::type_unsigned_char](#),
 [Abstraction::type_short](#),
 [Abstraction::type_unsigned_short](#), [Abstraction::type_int](#), [Abstraction::type_unsigned_int](#),
 [Abstraction::type_long](#),
 [Abstraction::type_unsigned_long](#), [Abstraction::type_float](#), [Abstraction::type_double](#) }

Functions

- [template<class T >](#)
 [Abstraction::DataType getAbstractionDataType \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< nullptr_t > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< char > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned char > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< short > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned short > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< int > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned int > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< long > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< unsigned long > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< float > \(\)](#)
- [template<>](#)
 [Abstraction::DataType getAbstractionDataType< double > \(\)](#)

10.4.1 Function Documentation

10.4.1.1 `template<class T> Abstraction::DataType getAbstractionDataType() [inline]`

Specilizations for the template class

10.4.1.2 `template<> Abstraction::DataType getAbstractionDataType<char>() [inline]`

10.4.1.3 `template<> Abstraction::DataType getAbstractionDataType<double>() [inline]`

10.4.1.4 `template<> Abstraction::DataType getAbstractionDataType<float>() [inline]`

10.4.1.5 `template<> Abstraction::DataType getAbstractionDataType<int>() [inline]`

10.4.1.6 `template<> Abstraction::DataType getAbstractionDataType<long>() [inline]`

10.4.1.7 `template<> Abstraction::DataType getAbstractionDataType<nullptr_t>() [inline]`

10.4.1.8 `template<> Abstraction::DataType getAbstractionDataType<short>() [inline]`

10.4.1.9 `template<> Abstraction::DataType getAbstractionDataType<unsigned char>() [inline]`

10.4.1.10 `template<> Abstraction::DataType getAbstractionDataType<unsigned int>() [inline]`

10.4.1.11 `template<> Abstraction::DataType getAbstractionDataType<unsigned long>() [inline]`

10.4.1.12 `template<> Abstraction::DataType getAbstractionDataType<unsigned short>() [inline]`

10.5 /home/jhasbestan/Morton_Parallel_v0/src/include/definitions.h File Reference

```
#include <algorithm>
#include <bitset>
#include <cstdint>
#include <cstdio>
#include <cstdlib>
#include <functional>
#include <iostream>
#include <stack>
#include <unordered_map>
#include <vector>
#include <list>
#include <unistd.h>
#include <mpi.h>
#include <memory>
#include <time.h>
#include <stdexcept>
#include "zoltan.h"
#include <cstddef>
#include <string>
#include <fstream>
#include <sstream>
#include <unordered_set>
#include <utility>
#include <iomanip>
```

Defines

- #define [hash](#) 0
- #define [nonnative](#) 1
- #define [RED](#) "\033[01;31m"
- #define [GREEN](#) "\033[22;32m"
- #define [YELLOW](#) "\033[22;33m"
- #define [BLUE](#) "\033[22;34m"
- #define [MAGENTA](#) "\033[22;35m"
- #define [CYAN](#) "\033[22;36m"
- #define [RESET](#) "\033[22;0m"

10.5.1 Define Documentation

10.5.1.1 `#define BLUE "\033[22;34m"`

10.5.1.2 `#define CYAN "\033[22;36m"`

10.5.1.3 `#define GREEN "\033[22;32m"`

10.5.1.4 `#define hash 0`

10.5.1.5 `#define MAGENTA "\033[22;35m"`

10.5.1.6 `#define nonnative 1`

10.5.1.7 `#define RED "\033[01;31m"`

10.5.1.8 `#define RESET "\033[22;0m"`

10.5.1.9 `#define YELLOW "\033[22;33m"`

10.6 /home/jhasbestan/Morton_Parallel_v0/src/include/forest.h File Reference

```
#include "communicate.h"
#include "definitions.h"
#include "tree.h"
#include "typedefs.h"
```

Classes

- class [Forest< N, Nvalue, M, Mvalue >](#)
template class that is a forest of octrees with semi-structured process topology

10.7 /home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h File Reference

```
#include <string>
#include <vector>
#include <cassert>
#include <fstream>
#include <iostream>
#include <sstream>
#include <streambuf>
```

Classes

- struct [stl::point](#)
Structure to hold coordinates of a [point](#).
- struct [stl::triangle](#)
structure to store normals and vertices of a [triangle](#)
- struct [stl::stl_data](#)
*structure to store vector of triangles read in from *.[stl](#) file*
- class [Vector3](#)

Namespaces

- namespace [stl](#)

Functions

- [std::ostream & stl::operator<<](#) ([std::ostream &out](#), [const triangle &t](#))
- [stl_data stl::parse_stl](#) ([const std::string &stl_path](#))
- void [checkMesh](#) ([std::vector< stl::triangle > &triangles](#))

10.7.1 Function Documentation

10.7.1.1 void checkMesh (std::vector< stl::triangle > & triangles)

10.8 /home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h File Reference

```
#include "definitions.h"
```

Classes

- class [Phdf5](#)< N, Nvalue, M, Mvalue >

*This Writes out [Tree](#) data in hdf5 format in parallel with *.xmf as metadata suitable for paraview and visit.*

10.9 /home/jhasbestan/Morton_Parallel_v0/src/include/tree.h File Reference

```
#include "definitions.h"
```

Classes

- class [Tree< N, value >](#)

This Class Generates a 4:1 balancerd AMR mesh.

- class [Voxel< N, value >](#)

This Class Generates an unbalancerd [Voxel](#) to improve search by geometry partitioning.

10.10 /home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h File Reference

```
#include "definitions.h"  
#include "zoltan.h"
```

Classes

- struct [CenterCoords](#)

Stores the coordinate of the centroid of the elements.

- struct [Zoltan_Out](#)

This structure is an interface to store the output from Zoltan.

Defines

- #define [Rma](#) 1
- #define [DEBUG](#) 0

Typedefs

- typedef std::vector< [CenterCoords](#) > [Center_coords](#)

Functions

- bool [zoltanGeometricPartitioner](#) (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct *zz, const [Center_coords](#) &XYZ, real *weight, [Zoltan_Out](#) *zoltan_out)
- void [readSTLGeom](#) (int argc, char *argv[], real **triangle_center, int *nn, const real *xyz)
- void [TwoPowN](#) (uint b, real *result)

10.10.1 Define Documentation

10.10.1.1 `#define DEBUG 0`

10.10.1.2 `#define Rma 1`

10.10.2 Typedef Documentation

10.10.2.1 `typedef std::vector<CenterCoords> Center_coords`

10.10.3 Function Documentation

10.10.3.1 `void readSTLGeom (int argc, char * argv[], real ** triangle_center, int * nn, const real * xyz)`

10.10.3.2 `void TwoPowN (uint b, real * result) [inline]`

10.10.3.3 `bool zoltanGeometricPartitioner (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct * zz, const Center_coords & XYZ, real * weight, Zoltan_Out * zoltan_out)`

10.11 /home/jhasbestan/Morton_Parallel_v0/src/main.cpp File Reference

```
#include "typedefs.h"
#include "communicate.h"
#include "datatype.h"
#include "forest.h"
#include "phdf5.h"
#include "tree.h"
```

Defines

- #define [PROCSIZE](#) 16
- #define [TREESIZE](#) 64
- #define [ZOLTAN_ON](#) 1
- #define [WEIGHT](#) 0
- #define [ZOLTAN_GEOMETRIC_PARTITION](#) 1

Functions

- int [main](#) (int argc, char *pArgs[])

10.11.1 Define Documentation

10.11.1.1 #define [PROCSIZE](#) 16

10.11.1.2 #define [TREESIZE](#) 64

10.11.1.3 #define [WEIGHT](#) 0

10.11.1.4 #define [ZOLTAN_GEOMETRIC_PARTITION](#) 1

10.11.1.5 #define [ZOLTAN_ON](#) 1

10.11.2 Function Documentation

10.11.2.1 int [main](#) (int *argc*, char **pArgs*[])

10.12 /home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp File Reference

```
#include "parse_stl.h"
#include "definitions.h"
```

Namespaces

- namespace [stl](#)

Defines

- #define [MYSCALE](#) 0.5
- #define [CHECK_MESH](#) 0

Functions

- std::ostream & [stl::operator<<](#) (std::ostream &out, const point p)
- std::ostream & [stl::operator<<](#) (std::ostream &out, const triangle &t)
- float [stl::parse_float](#) (std::ifstream &s)
- point [stl::parse_point](#) (std::ifstream &s)
- stl_data [stl::parse_stl](#) (const std::string &stl_path)
- void [readSTLGeom](#) (int argc, char *argv[], real **triangle_center, int *nn, const real *xyz)
- void [checkMesh](#) (std::vector< [stl::triangle](#) > &triangles)

10.12.1 Define Documentation

10.12.1.1 #define [CHECK_MESH](#) 0

10.12.1.2 #define [MYSCALE](#) 0.5

10.12.2 Function Documentation

10.12.2.1 void [checkMesh](#) (std::vector< [stl::triangle](#) > & *triangles*)

10.12.2.2 void [readSTLGeom](#) (int *argc*, char * *argv*[], real ** *triangle_center*, int * *nn*, const real * *xyz*)

10.13 /home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp File Reference

```
#include "forest.h"
#include "hdf5.h"
#include "phdf5.h"
```

Defines

- #define [H5FILE_NAME](#) "soln/Pxdmf3d%u.h5"
- #define [XDMF_NAME](#) "soln/Pxdmf3d%u.xmf"
- #define [H5FILE](#) "Pxdmf3d%u.h5"
- #define [offset0](#) 156
- #define [offset1](#) 2005

Functions

- static void [integer_string](#) (char *strin, int i)

10.13.1 Define Documentation

10.13.1.1 #define [H5FILE](#) "Pxdmf3d%u.h5"

10.13.1.2 #define [H5FILE_NAME](#) "soln/Pxdmf3d%u.h5"

10.13.1.3 #define [offset0](#) 156

10.13.1.4 #define [offset1](#) 2005

10.13.1.5 #define [XDMF_NAME](#) "soln/Pxdmf3d%u.xmf"

10.13.2 Function Documentation

10.13.2.1 static void [integer_string](#) (char * *strin*, int *i*) [[static](#)]

10.14 /home/jhasbestan/Morton_Parallel_v0/src/tree.cpp File Reference

```
#include "tree.h"  
#include "definitions.h"  
#include "typedefs.h"
```

10.15 /home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp File Reference

```
#include "tree.h"
```

10.16 /home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp File Reference

```
#include <stdlib.h>
#include "typedefs.h"
#include "tree.h"
```

Classes

- struct [MeshData](#)

struct to supply information required by zoltan for geometric partitioners

- struct [GraphData](#)

struct to supply information required by zoltan for geometric partitioners

Defines

- #define [TOL](#) "1.1"

Functions

- static int [get_number_of_objects](#) (void *data, int *ierr)
- static int [get_num_geometry](#) (void *data, int *ierr)
- static void [get_object_list](#) (void *data, int sizeGID, int sizeLID, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int wgt_dim, float *obj_wgts, int *ierr)
- static void [get_geometry_list](#) (void *data, int sizeGID, int sizeLID, int num_obj, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int num_dim, double *geom_vec, int *ierr)
- bool [zoltanGeometricPartitioner](#) (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct *zz, const [Center_coords](#) &XYZ, real *weight, [Zoltan_Out](#) *zoltan_out)

10.16.1 Define Documentation

10.16.1.1 `#define TOL "1.1"`

10.16.2 Function Documentation

10.16.2.1 `static void get_geometry_list (void * data, int sizeGID, int sizeLID, int num_obj, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int num_dim, double * geom_vec, int * ierr) [static]`

10.16.2.2 `static int get_num_geometry (void * data, int * ierr) [static]`

10.16.2.3 `static int get_number_of_objects (void * data, int * ierr) [static]`

10.16.2.4 `static void get_object_list (void * data, int sizeGID, int sizeLID, ZOLTAN_ID_PTR globalID, ZOLTAN_ID_PTR localID, int wgt_dim, float * obj_wgts, int * ierr) [static]`

10.16.2.5 `bool zoltanGeometricPartitioner (const uint size, const uint ncube_total, const uint offset, const int method, struct Zoltan_Struct * zz, const Center_coords & XYZ, real * weight, Zoltan_Out * zoltan_out)`

Index

- ~CommCollective
 - CommCollective, [19](#)
- ~CommPoint2Point
 - CommPoint2Point, [22](#)
- ~Forest
 - Forest, [26](#)
- ~Phdf5
 - Phdf5, [38](#)
- ~Tree
 - Tree, [44](#)
- ~Vector3
 - Vector3, [53](#)
- ~Voxel
 - Voxel, [54](#)
- /home/jhasbestan/Morton_Parallel_v0/src/ Directory Reference, [14](#)
- /home/jhasbestan/Morton_Parallel_v0/src/communicate.cpp, [59](#)
- /home/jhasbestan/Morton_Parallel_v0/src/forest.cpp, [60](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/ Directory Reference, [13](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/communicate.h, [61](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/datatype.h, [62](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/definitions.h, [64](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/forest.h, [66](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/parse_stl.h, [67](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/phdf5.h, [68](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/tree.h, [69](#)
- /home/jhasbestan/Morton_Parallel_v0/src/include/typedefs.h, [70](#)
- /home/jhasbestan/Morton_Parallel_v0/src/main.cpp, [72](#)
- /home/jhasbestan/Morton_Parallel_v0/src/parse_stl.cpp, [73](#)
- /home/jhasbestan/Morton_Parallel_v0/src/phdf5.cpp, [74](#)
- /home/jhasbestan/Morton_Parallel_v0/src/tree.cpp, [75](#)
- /home/jhasbestan/Morton_Parallel_v0/src/voxel.cpp, [76](#)
- /home/jhasbestan/Morton_Parallel_v0/src/zoltan.cpp, [77](#)
- Abstraction, [15](#)
 - DataType, [15](#)
 - type_byte, [15](#)
 - type_char, [15](#)
 - type_double, [15](#)
 - type_float, [15](#)
 - type_int, [15](#)
 - type_long, [15](#)
 - type_short, [15](#)
 - type_unsigned_char, [15](#)
 - type_unsigned_int, [15](#)
 - type_unsigned_long, [15](#)
 - type_unsigned_short, [15](#)
- addToDerefineList
 - Tree, [44](#)
- addToList
 - Tree, [44](#)
- ancestorcoords
 - Forest, [31](#)
 - Tree, [50](#)
- ancestorkey
 - Tree, [50](#)
- ancestorlength
 - Forest, [31](#)
 - Tree, [50](#)
- assignGeom
 - Forest, [27](#)
- assignReciever
 - CommPoint2Point, [22](#)
- assignSender
 - CommPoint2Point, [22](#)
- bcast
 - CommCollective, [19](#)
- begin
 - Tree, [44](#)
- BLUE
 - definitions.h, [65](#)

- boundarylist
 - Tree, [50](#)
- buf
 - Message, [35](#)
- c
 - GraphData, [33](#)
 - MeshData, [34](#)
- Center_coords
 - typedefs.h, [71](#)
- CenterCoords, [17](#)
 - x, [17](#)
 - y, [17](#)
 - z, [17](#)
- centroid
 - Tree, [44](#)
- changes
 - Zoltan_Out, [58](#)
- CHARACTER
 - forest.cpp, [60](#)
- CHECK_MESH
 - parse_stl.cpp, [73](#)
- checkGraphConsistency
 - Forest, [27](#)
- checkMesh
 - parse_stl.cpp, [73](#)
 - parse_stl.h, [67](#)
- checkNbrsOfNbrsConsistency
 - Forest, [27](#)
- checkSiblingStatus
 - Voxel, [55](#)
- checkWithNbrs
 - Forest, [27](#)
- clearMesh
 - Tree, [44](#)
- clearMortonSTL
 - Tree, [45](#)
- clearRefineList
 - Tree, [45](#)
- Com
 - CommCollective, [19](#)
 - CommPoint2Point, [23](#)
 - Forest, [31](#)
- combine
 - Forest, [27](#)
- combinedLevel
 - Forest, [27](#)
- CommCollective, [18](#)
 - ~CommCollective, [19](#)
 - bcast, [19](#)
 - Com, [19](#)
 - CommCollective, [18](#)
 - getTotalNumber, [19](#)
 - Ibcast, [19](#)
 - IgetTotalNumber, [19](#)
 - Msg, [19](#)
 - waitOnRequest, [19](#)
- CommPoint2Point, [21](#)
 - ~CommPoint2Point, [22](#)
 - assignReciever, [22](#)
 - assignSender, [22](#)
 - Com, [23](#)
 - CommPoint2Point, [21, 22](#)
 - getOffset, [22](#)
 - Irecv, [22](#)
 - Isend, [22](#)
 - Msg, [23](#)
 - myRank, [22](#)
 - mySize, [23](#)
 - recv, [23](#)
 - send, [23](#)
- communicate.cpp
 - ConvertType, [59](#)
- comPatternConstruct
 - Forest, [27](#)
- comsize
 - MpiCom, [37](#)
- construct
 - Tree, [45](#)
- constructElementKeyForRcvdMessage
 - Forest, [28](#)
- constructHigherLevelNbrs
 - Tree, [45](#)
- constructNonlocalHigherLevelNbrs
 - Tree, [45](#)
- constructSeedKeyForRcvdMessage
 - Forest, [28](#)
- convertBitsToDouble
 - Forest, [28](#)
- convertCoordToMorton
 - Tree, [45](#)
- convertDoubleToBits
 - Forest, [28](#)
- convertStl2Morton
 - Tree, [45](#)
- ConvertType
 - communicate.cpp, [59](#)
- count
 - Message, [35](#)
 - Tree, [45](#)
- createCommGraph
 - Forest, [28](#)
- createNbrsOfNbrs
 - Forest, [28](#)
- CYAN
 - definitions.h, [65](#)
- DataType, [24](#)

- Abstraction, 15
- datatype
 - Message, 35
- datatype.h
 - getAbstractionDataType, 63
 - getAbstractionDataType< char >, 63
 - getAbstractionDataType< double >, 63
 - getAbstractionDataType< float >, 63
 - getAbstractionDataType< int >, 63
 - getAbstractionDataType< long >, 63
 - getAbstractionDataType< nullptr_t >, 63
 - getAbstractionDataType< short >, 63
 - getAbstractionDataType< unsigned char >, 63
 - getAbstractionDataType< unsigned int >, 63
 - getAbstractionDataType< unsigned long >, 63
 - getAbstractionDataType< unsigned short >, 63
- Dbegin
 - Tree, 45
- DEBUG
 - typedefs.h, 71
- debug
 - Forest, 28
- debugDerefine
 - Forest, 28
- definitions.h
 - BLUE, 65
 - CYAN, 65
 - GREEN, 65
 - hash, 65
 - MAGENTA, 65
 - nonnative, 65
 - RED, 65
 - RESET, 65
 - YELLOW, 65
- Dend
 - Tree, 45
- derefine
 - Tree, 45
- derefineDerefineList
 - Tree, 46
- derefineGeomTree
 - Voxel, 55
- derefinelist
 - Tree, 50
- destination
 - Forest, 31
- distributeGeomToLeaves
 - Voxel, 55
- enclosingBox
 - Tree, 46
- encodeGeometry
 - Forest, 28
- end
 - Tree, 46
- exportGlobalGids
 - Zoltan_Out, 58
- exportLocalGids
 - Zoltan_Out, 58
- exportProcs
 - Zoltan_Out, 58
- exportToPart
 - Zoltan_Out, 58
- extractBoundary
 - Tree, 46
- extractBoundaryP
 - Tree, 46
- find
 - Tree, 46
- findFlipLevel
 - Forest, 28
 - Tree, 46
- findInDerefine
 - Tree, 46
- findSeedLevelForRcvdMessage
 - Forest, 28
- flipAll
 - Forest, 29
- flipForNbr
 - Forest, 29
 - Tree, 46
- flipRefineElemTag
 - Tree, 47
- Forest, 25
 - ~Forest, 26
 - ancestorcoords, 31
 - ancestorlength, 31
 - assignGeom, 27
 - checkGraphConsistency, 27
 - checkNbrsOfNbrsConsistency, 27
 - checkWithNbrs, 27
 - Com, 31
 - combine, 27
 - combinedLevel, 27
 - comPatternConstruct, 27
 - constructElementKeyForRcvdMessage, 28
 - constructSeedKeyForRcvdMessage, 28
 - convertBitsToDouble, 28
 - convertDoubleToBits, 28
 - createCommGraph, 28
 - createNbrsOfNbrs, 28
 - debug, 28
 - debugDerefine, 28
 - destination, 31
 - encodeGeometry, 28
 - findFlipLevel, 28

- findSeedLevelForRcvdMessage, 28
- flipAll, 29
- flipForNbr, 29
- Forest, 26
- forestsize, 29
- fourToOneBalance, 29
- geom, 31
- getDirections, 29
- getElemNbrs, 29
- getListEachTree, 29
- getMaxSeedsLevel, 29
- getNbrSeedLevel, 29
- getTotalSize, 29
- graphComm, 31
- isInSeed, 29
- maxseedlevel, 31
- moveGeom, 29
- nbrsOfNbrs, 32
- npx, 32
- npz, 32
- Phdf5, 31
- pushToDerefineEachTree, 30
- recoverAllZeroSingularity, 30
- refineEachTree, 30
- refineEachTreeVoxel, 30
- refineForestBalanced, 30
- removeAllZeroSingularity, 31
- retainFourToOneBalance, 31
- seeds, 32
- trees, 32
- zoltan_out, 32
- zoltanGeomrepart, 31
- zz, 32
- forest.cpp
 - CHARACTER, 60
 - REORDER, 60
 - SENDBOOL, 60
- forestsize
 - Forest, 29
- fourToOne
 - Tree, 47
- fourToOneBalance
 - Forest, 29
- fourToOneP
 - Tree, 47
- generateSearchTree
 - Voxel, 55
- geom
 - Forest, 31
- get_geometry_list
 - zoltan.cpp, 78
- get_num_geometry
 - zoltan.cpp, 78
- get_number_of_objects
 - zoltan.cpp, 78
- get_object_list
 - zoltan.cpp, 78
- getAbstractionDataType
 - datatype.h, 63
- getAbstractionDataType< char >
 - datatype.h, 63
- getAbstractionDataType< double >
 - datatype.h, 63
- getAbstractionDataType< float >
 - datatype.h, 63
- getAbstractionDataType< int >
 - datatype.h, 63
- getAbstractionDataType< long >
 - datatype.h, 63
- getAbstractionDataType< nullptr_t >
 - datatype.h, 63
- getAbstractionDataType< short >
 - datatype.h, 63
- getAbstractionDataType< unsigned char >
 - datatype.h, 63
- getAbstractionDataType< unsigned int >
 - datatype.h, 63
- getAbstractionDataType< unsigned long >
 - datatype.h, 63
- getAbstractionDataType< unsigned short >
 - datatype.h, 63
- getDirections
 - Forest, 29
 - Tree, 47
- getElemNbrs
 - Forest, 29
- getKey
 - Tree, 47
- getListEachTree
 - Forest, 29
- getMaxSeedsLevel
 - Forest, 29
- getNbrSeedLevel
 - Forest, 29
- getOffset
 - CommPoint2Point, 22
- getTotalNumber
 - CommCollective, 19
- getTotalSize
 - Forest, 29
- graphComm
 - Forest, 31
- GraphData, 33
 - c, 33
 - nbrGID, 33
 - nbrIndex, 33

- nbrProc, 33
- numMyVertices, 33
- vertexGID, 33
- GREEN
 - definitions.h, 65
- H5FILE
 - phdf5.cpp, 74
- H5FILE_NAME
 - phdf5.cpp, 74
- hash
 - definitions.h, 65
- Hdf5Xmf
 - Tree, 49
- Hdf5XmfV
 - Voxel, 55
- Ibcast
 - CommCollective, 19
- IgetTotalNumber
 - CommCollective, 19
- importGlobalGids
 - Zoltan_Out, 58
- importLocalGids
 - Zoltan_Out, 58
- importProcs
 - Zoltan_Out, 58
- importToPart
 - Zoltan_Out, 58
- insertKey
 - Tree, 47
- integer_string
 - phdf5.cpp, 74
- Irecv
 - CommPoint2Point, 22
- isBoundary
 - Tree, 47
- Isend
 - CommPoint2Point, 22
- isInMeshList
 - Tree, 47
- isInRefineList
 - Tree, 47
- isInSeed
 - Forest, 29
- IsInsideSegment
 - Voxel, 55
- isInsideSolid
 - Tree, 47
- IsInVectorList
 - Tree, 47
- Length
 - Vector3, 53
- level
 - Tree, 48
- lookup
 - Voxel, 55
- MAGENTA
 - definitions.h, 65
- main
 - main.cpp, 72
- main.cpp
 - main, 72
 - PROCSIZE, 72
 - TREESIZE, 72
 - WEIGHT, 72
 - ZOLTAN_GEOMETRIC_PARTITION, 72
 - ZOLTAN_ON, 72
- maxlevel
 - Voxel, 55
- maxseedlevel
 - Forest, 31
- mesh
 - Tree, 50
 - Voxel, 55
- MeshData, 34
 - c, 34
 - myGlobalIDs, 34
 - numGlobalPoints, 34
 - numMyPoints, 34
 - w, 34
- Message, 35
 - buf, 35
 - count, 35
 - datatype, 35
 - print, 35
 - reciever, 35
 - request, 35
 - sender, 36
 - status, 36
 - tag, 36
- mortonSTL
 - Tree, 50
- mortonSTLclear
 - Tree, 48
- moveGeom
 - Forest, 29
- MpiCom, 37
 - comsize, 37
 - MpiCom, 37
 - mpicom, 37
 - myrank, 37
- mpicom
 - MpiCom, 37
- Msg
 - CommCollective, 19

- CommPoint2Point, 23
- myGlobalIDs
 - MeshData, 34
- myRank
 - CommPoint2Point, 22
- myrank
 - MpiCom, 37
- MYSCALE
 - parse_stl.cpp, 73
- mySize
 - CommPoint2Point, 23
- name
 - stl::stl_data, 41
- nbrGID
 - GraphData, 33
- nbrIndex
 - GraphData, 33
- nbrProc
 - GraphData, 33
- nbrsOfNbrs
 - Forest, 32
- nonnative
 - definitions.h, 65
- normal
 - stl::triangle, 52
- Normalize
 - Vector3, 53
- npx
 - Forest, 32
 - Tree, 50
- npv
 - Forest, 32
 - Tree, 50
- npz
 - Forest, 32
 - Tree, 50
- numExport
 - Zoltan_Out, 58
- numGidEntries
 - Zoltan_Out, 58
- numGlobalPoints
 - MeshData, 34
- numImport
 - Zoltan_Out, 58
- numLidEntries
 - Zoltan_Out, 58
- numMax
 - Voxel, 56
- numMyPoints
 - MeshData, 34
- numMyVertices
 - GraphData, 33
- offset0
 - phdf5.cpp, 74
- offset1
 - phdf5.cpp, 74
- operator<<
 - stl, 16
- parse_float
 - stl, 16
- parse_point
 - stl, 16
- parse_stl
 - stl, 16
- parse_stl.cpp
 - CHECK_MESH, 73
 - checkMesh, 73
 - MYSCALE, 73
 - readSTLGeom, 73
- parse_stl.h
 - checkMesh, 67
- parts
 - Zoltan_Out, 58
- Phdf5, 38
 - ~Phdf5, 38
 - Forest, 31
 - Phdf5, 38
 - totalnumber, 39
 - writeMultiBlock, 38
 - writePolyvertex, 38
 - xdmfMultiBlock, 39
 - xdmfPolyvertex, 39
- phdf5.cpp
 - H5FILE, 74
 - H5FILE_NAME, 74
 - integer_string, 74
 - offset0, 74
 - offset1, 74
 - XDMF_NAME, 74
- point
 - stl::point, 40
- print
 - Message, 35
- printMesh
 - Tree, 48
- PROCSIZE
 - main.cpp, 72
- pushToDerefineEachTree
 - Forest, 30
- pushToDerefinelist
 - Tree, 48
- pushToRefinelist
 - Tree, 48
- Rbegin

- Tree, 48
- readDerefineList
 - Tree, 48
- readRefineList
 - Tree, 48
- readSTLGeom
 - parse_stl.cpp, 73
 - typedefs.h, 71
- reciever
 - Message, 35
- recoverAllZeroSingularity
 - Forest, 30
- recv
 - CommPoint2Point, 23
- RED
 - definitions.h, 65
- refine
 - Tree, 48
- refineEachTree
 - Forest, 30
- refineEachTreeVoxel
 - Forest, 30
- refineForestBalanced
 - Forest, 30
- refinelist
 - Tree, 51
- refinelistReset
 - Tree, 48
- refineListSize
 - Tree, 49
- refineRefineList
 - Tree, 49
- removeAllZeroSingularity
 - Forest, 31
- removeFromDerefineList
 - Tree, 49
- Rend
 - Tree, 49
- REORDER
 - forest.cpp, 60
- request
 - Message, 35
- reserve
 - Tree, 49
- RESET
 - definitions.h, 65
- retainFourToOne
 - Tree, 49
- retainFourToOneBalance
 - Forest, 31
- Rma
 - typedefs.h, 71
- seeds
 - Forest, 32
- send
 - CommPoint2Point, 23
- SENDBOOL
 - forest.cpp, 60
- sender
 - Message, 36
- setLevel
 - Voxel, 55
- siblings
 - Tree, 49
- size
 - Tree, 49
- status
 - Message, 36
- stl, 16
 - operator<<, 16
 - parse_float, 16
 - parse_point, 16
 - parse_stl, 16
- stl::point, 40
 - point, 40
 - x, 40
 - y, 40
 - z, 40
- stl::stl_data, 41
 - name, 41
 - stl_data, 41
 - triangles, 41
- stl::triangle, 52
 - normal, 52
 - triangle, 52
 - v1, 52
 - v2, 52
 - v3, 52
- stl_data
 - stl::stl_data, 41
- tag
 - Message, 36
- TOL
 - zoltan.cpp, 78
- totalnumber
 - Phdf5, 39
- Tree, 42
 - ~Tree, 44
 - addToDerefineList, 44
 - addToList, 44
 - ancestorcoords, 50
 - ancestorkey, 50
 - ancestorlength, 50
 - begin, 44
 - boundarylist, 50
 - centroid, 44

- clearMesh, 44
- clearMortonSTL, 45
- clearRefineList, 45
- construct, 45
- constructHigherLevelNbrs, 45
- constructNonlocalHigherLevelNbrs, 45
- convertCoordToMorton, 45
- convertStl2Morton, 45
- count, 45
- Dbegin, 45
- Dend, 45
- derefine, 45
- derefineDerefineList, 46
- derefinelist, 50
- enclosingBox, 46
- end, 46
- extractBoundary, 46
- extractBoundaryP, 46
- find, 46
- findFlipLevel, 46
- findInDerefine, 46
- flipForNbr, 46
- flipRefineElemTag, 47
- fourToOne, 47
- fourToOneP, 47
- getDirections, 47
- getKey, 47
- Hdf5Xmf, 49
- insertKey, 47
- isBoundary, 47
- isInMeshList, 47
- isInRefineList, 47
- isInsideSolid, 47
- IsInVectorList, 47
- level, 48
- mesh, 50
- mortonSTL, 50
- mortonSTLclear, 48
- npx, 50
- npz, 50
- npz, 50
- printMesh, 48
- pushToDerefinelist, 48
- pushToRefinelist, 48
- Rbegin, 48
- readDerefineList, 48
- readRefineList, 48
- refine, 48
- refinelist, 51
- refinelistReset, 48
- refineListSize, 49
- refineRefineList, 49
- removeFromDerefineList, 49
- Rend, 49
- reserve, 49
- retainFourToOne, 49
- siblings, 49
- size, 49
- Tree, 44
- trees
 - Forest, 32
- TREESIZE
 - main.cpp, 72
- triangle
 - stl::triangle, 52
- triangles
 - stl::stl_data, 41
- TwoPowN
 - typedefs.h, 71
- type_byte
 - Abstraction, 15
- type_char
 - Abstraction, 15
- type_double
 - Abstraction, 15
- type_float
 - Abstraction, 15
- type_int
 - Abstraction, 15
- type_long
 - Abstraction, 15
- type_short
 - Abstraction, 15
- type_unsigned_char
 - Abstraction, 15
- type_unsigned_int
 - Abstraction, 15
- type_unsigned_long
 - Abstraction, 15
- type_unsigned_short
 - Abstraction, 15
- typedefs.h
 - Center_coords, 71
 - DEBUG, 71
 - readSTLGeom, 71
 - Rma, 71
 - TwoPowN, 71
 - zoltanGeometricPartitioner, 71
- v1
 - stl::triangle, 52
- v2
 - stl::triangle, 52
- v3
 - stl::triangle, 52
- Vector3, 53
 - ~Vector3, 53
 - Length, 53

- Normalize, [53](#)
 - Vector3, [53](#)
 - Vectors, [53](#)
 - X, [53](#)
 - Y, [53](#)
 - Z, [53](#)
- Vectors
 - Vector3, [53](#)
- vertexGID
 - GraphData, [33](#)
- Voxel, [54](#)
 - ~Voxel, [54](#)
 - checkSiblingStatus, [55](#)
 - derefineGeomTree, [55](#)
 - distributeGeomToLeaves, [55](#)
 - generateSearchTree, [55](#)
 - Hdf5XmfV, [55](#)
 - IsInsideSegment, [55](#)
 - lookup, [55](#)
 - maxlevel, [55](#)
 - mesh, [55](#)
 - numMax, [56](#)
 - setLevel, [55](#)
 - Voxel, [54](#)
- w
 - MeshData, [34](#)
- waitOnRequest
 - CommCollective, [19](#)
- WEIGHT
 - main.cpp, [72](#)
- writeMultiBlock
 - Phdf5, [38](#)
- writePolyvertex
 - Phdf5, [38](#)
- X
 - Vector3, [53](#)
- x
 - CenterCoords, [17](#)
 - stl::point, [40](#)
- XDMF_NAME
 - phdf5.cpp, [74](#)
- xdmfMultiBlock
 - Phdf5, [39](#)
- xdmfPolyvertex
 - Phdf5, [39](#)
- Y
 - Vector3, [53](#)
- y
 - CenterCoords, [17](#)
 - stl::point, [40](#)
- YELLOW
 - definitions.h, [65](#)
- Z
 - Vector3, [53](#)
- z
 - CenterCoords, [17](#)
 - stl::point, [40](#)
- zoltan.cpp
 - get_geometry_list, [78](#)
 - get_num_geometry, [78](#)
 - get_number_of_objects, [78](#)
 - get_object_list, [78](#)
 - TOL, [78](#)
 - zoltanGeometricPartitioner, [78](#)
- ZOLTAN_GEOMETRIC_PARTITION
 - main.cpp, [72](#)
- ZOLTAN_ON
 - main.cpp, [72](#)
- Zoltan_Out, [57](#)
 - changes, [58](#)
 - exportGlobalGids, [58](#)
 - exportLocalGids, [58](#)
 - exportProcs, [58](#)
 - exportToPart, [58](#)
 - importGlobalGids, [58](#)
 - importLocalGids, [58](#)
 - importProcs, [58](#)
 - importToPart, [58](#)
 - numExport, [58](#)
 - numGidEntries, [58](#)
 - numImport, [58](#)
 - numLidEntries, [58](#)
 - parts, [58](#)
- zoltan_out
 - Forest, [32](#)
- zoltanGeometricPartitioner
 - typedefs.h, [71](#)
 - zoltan.cpp, [78](#)
- zoltanGeomrepart
 - Forest, [31](#)
- zz
 - Forest, [32](#)