GEM3D

Generated by Doxygen 1.6.1

Fri Aug 11 16:18:56 2017

Contents

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 fro generatin 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

Chapter 2

Directory Hierarchy

2.1 Directories

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

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

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 elments.

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

Public Attributes

- real x
- real y
- real z

9.1.1 Detailed Description

Stores the coordinate of the centroid of the elments.

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 communicatios

`#include <communicate.h>`

Public Member Functions

- CommCollective (void ∗buf, uint size, uint root)
- CommCollective (void ∗buff, uint size)
- void bcast ()
- void Ibcast ()
- void getTotalNumber (uint ∗offset, uint ∗myvalue, uint ∗totalvalue)
- void IgetTotalNumber (uint ∗offset, uint ∗myvalue, uint ∗totalvalue)
- void waitOnRequest ()
- ∼CommCollective ()
- template<>
  void getTotalNumber (uint ∗offset, uint ∗myvalue, uint ∗totalvalue)
- template<>
  void IgetTotalNumber (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 communicatios 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 spoecialized 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 assignReciever (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 recieve is "0", MPI_ANY_TAG is onlu OK for recieve commnad 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 assigment 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 ranl 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 recieve

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 > MpiCom 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-templat
since BYTE is not a native type is 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 unnecesary 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 neghbors 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 tha 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]`

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

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 centeroids 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

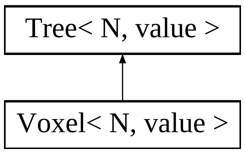
The documentation for this struct was generated from the following file:

- /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 derefining 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 convertStl2Morton (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 npy
- 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 >::convertStl2Morton (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 nonexsiting 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 igonre 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*)

Derefines 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;iminate 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 elments 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 gemoetry 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 unnecesary 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]`

perfomrs 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]`

centeroid 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 0000000000000

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 listi

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]`

discritization 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 Vector3::Normalize ()

9.15.2.3 Vector3 Vector3::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

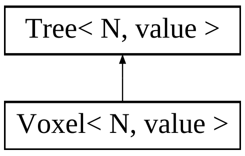
The documentation for this class was generated from the following files:

- /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 unbalancerd 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 unbalancerd 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 elments.

- 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 argcs, 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 *argcs*, 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