JGTL

1.6

Generated by Doxygen 1.5.5

Sun Oct 26 01:25:28 2008

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 JGTL Namespace Reference

The main namespace that contains everything.

Classes

- class Bar

  The Bar Class handles a max/current value system (e.g. a Progress Bar).

- class CircularBuffer

  The CircularBuffer Class handles a Circular Buffer.

- class CircularBufferInterface

  The CircularBufferInterface Class handles a Circular Buffer.

- struct CCmdParam
- class CommandLineParser
- class DataManager
- class DataPool
- class DynamicCircularBuffer

  The DynamicCircularBuffer Class handles a Circular Buffer.

- class DynamicPoolMap

  The DynamicPoolMap Class is a resizable array-based associative map structure.

- class DynamicPoolSet

- class FloatingUnits
- class HexTreeNode
- class HexTreeStub
- class HexTreeBranch
- class HexTree
- class Index2
- class Index3
- class RectangleIndex3
- struct IF
- struct JGTL::IF< false, Then, Else >
- struct STATIC_MOD
- struct TYPEIF
- struct JGTL::TYPEIF< Type, false, Then, Else >
- struct IntegralUnitsGCD
- struct JGTL::IntegralUnitsGCD< 1, j >
- struct JGTL::IntegralUnitsGCD< i, 1 >
- struct JGTL::IntegralUnitsGCD< 1, 1 >
- struct JGTL::IntegralUnitsGCD< 0, j >
- struct JGTL::IntegralUnitsGCD< i, 0 >
- struct JGTL::IntegralUnitsGCD< 0, 0 >
- class IntegralUnits
- class InterpolatedValue

  The InterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff;.

- class LocatedException

  This class handles throwing exceptions which include the file and line number.

- class MapInterface

  This class acts as a base class for the Map construct.

- class PolyVariant
- class PoolMap
- class QuadraticSolution
- class QuadTreeNode
- class QuadTreeStub
- class QuadTreeBranch
- class QuadTree
- struct ProfileBlock
- class Clock
- class Profiler

  A singleton class that manages timing for a set of profiling blocks.

- class ProfileBlockHandler
- class Ray2

  This class handles 2D Rays and Line Segments.

- class Ray3

  This class handles 3D Rays and Line Segments.

- class Rectangle3
- class SetInterface

  This class acts as a base class for the Set construct.

- class Singleton

  This class handles Singletons (Global Single-Instance Classes).

- class SortedList
- class StackCircularBuffer

  The StackCircularBuffer Class handles a Circular Buffer.

- class StackMap

  The StackMap Class is a fixed, array-based, sorted key structure.

- class StackSet
- class TreeListNode
- class TreeList
- class BinaryTreeNode
- struct STATIC_MAX_SIZE
- struct JGTL::STATIC_MAX_SIZE< One, One, Two, Three, Four, Five, Six, Seven, Eight, Nine >
- struct JGTL::STATIC_MAX_SIZE< One, One, One, Two, Three, Four, Five, Six, Seven, Eight >
- struct JGTL::STATIC_MAX_SIZE< One, One, One, One, Two, Three, Four, Five, Six, Seven >
- struct JGTL::STATIC_MAX_SIZE< One, One, One, One, One, Two, Three, Four, Five, Six >
- struct JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, Two, Three, Four, Five >
- struct JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, Two, Three, Four >
- struct JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, Two, Three >
- struct JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, One, Two >
- class NullVariantClass

- class Variant
- class Vector2

  This class handles 2D Vectors.

- class Vector3
- class Vector4
- class WrappedInterpolatedValue

  The WrappedInterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff; This special instance of an InterpolatedValue is for values which wrap (angles, for example, which wrap around 2∗PI).

- class XorSpaceRect

  This handles a single Xor Rectangle.

- class XorSpace

  This class handles Xor spaces. Think of this as a way to handle things like rectangular doughnuts. A positive space followed by a smaller concentric negative space would represent a doughnut.

Typedefs

- typedef std::map< StringType, CCmdParam > _CommandLineParser
- typedef unsigned long long units_internal_ulong
- typedef unsigned char uchar

Enumerations

- enum TimeFormat { SECONDS, MILLISECONDS, MICROSECONDS, PERCENT }

  A set of ways to represent timing results.

- enum IntersectionState { IS_NONE, IS_ONE, IS_INFINITE }

Functions

- template<class BarValueType>
  std::ostream & operator<< (std::ostream &stream, const Bar< BarValueType > &d)
- template<class BarValueType>
  std::istream & operator>> (std::istream &stream, Bar< BarValueType > &d)

- template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR>
  std::ostream & operator<< (std::ostream &stream, const FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR > &d)
- template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR>
  std::istream & operator>> (std::istream &stream, FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR > &d)
- ostream & operator<< (ostream &stream, const Index3 &d)
- istream & operator>> (istream &stream, Index3 &d)
- ostream & operator<< (ostream &stream, const RectangleIndex3 &d)
- istream & operator>> (istream &stream, RectangleIndex3 &d)
- template<class T>
  units_internal_ulong GCD (T a, T b)
- template<>
  units_internal_ulong GCD (double a, double b)
- template<>
  units_internal_ulong GCD (float a, float b)
- template<class ValueType, ValueType SCALE, bool USEGCD>
  std::ostream & operator<< (std::ostream &stream, const IntegralUnits< ValueType, SCALE, USEGCD > &d)
- template<class ValueType, ValueType SCALE, bool USEGCD>
  std::istream & operator>> (std::istream &stream, IntegralUnits< ValueType, SCALE, USEGCD > &d)
- template<class T>
  std::ostream & operator<< (std::ostream &stream, const InterpolatedValue< T > &d)
- template<class T>
  std::istream & operator>> (std::istream &stream, InterpolatedValue< T > &d)
- template<class T, class TT, class TTT, class TTTT>
  QuadraticSolution< T > solveQuadratic (TT a, TTT b, TTTT c)
- template<class Data>
  void packBuffer (uchar *&buffer, int &bufferSize, const Data &data)
- template<class Data>
  void unpackBuffer (uchar *&buffer, int &bufferSize, Data &data)
- template<class Data>
  void packBufferStack (uchar *&buffer, int &bufferSize, const Data &data)
- template<class Data>
  void unpackBufferStack (uchar *&buffer, int &bufferSize, Data &data)
- void packBufferString (uchar *&buffer, int &bufferSize, const char *s)
- void packBufferString (uchar *&buffer, int &bufferSize, const std::string &s)
- void unpackBufferString (uchar *&buffer, int &bufferSize, string &s)
- template<typename T>
  T stringTo (const std::string &s)

- template<typename T>
  void stringTo (const std::string &s, T &x)
- template<typename T>
  std::string toString (const T &x)
- template<class T>
  T getIndexFromName (const char *name, const char **names, T numNames)
- template<class T>
  T getIndexFromName (const std::string &name, const char **names, T numNames)
- template<class T>
  std::ostream & operator<< (std::ostream &stream, const Vector2< T > &d)
- template<class T>
  std::istream & operator>> (std::istream &stream, Vector2< T > &d)
- template<class T, class TT>
  T convertVector2 (const TT &other)
- template<class T, class TT>
  T convertVector3 (const TT &other)
- template<class T>
  std::ostream & operator<< (std::ostream &stream, const Vector3< T > &d)
- template<class T>
  std::istream & operator>> (std::istream &stream, Vector3< T > &d)
- template<class T, class TT>
  T convertVector4 (const TT &other)
- template<class T>
  std::ostream & operator<< (std::ostream &stream, const Vector4< T > &d)
- template<class T>
  std::istream & operator>> (std::istream &stream, Vector4< T > &d)
- template<class T>
  std::ostream & operator<< (std::ostream &stream, const WrappedInterpolatedValue< T > &d)
- template<class T>
  std::istream & operator>> (std::istream &stream, WrappedInterpolatedValue< T > &d)
- template<class Rectangle, class Point>
  ostream & operator<< (ostream &stream, const XorSpace< Rectangle, Point > &d)
- template<class Rectangle, class Point>
  istream & operator>> (istream &stream, XorSpace< Rectangle, Point > &d)

5.1.1 Detailed Description

The main namespace that contains everything.

5.1.2 Typedef Documentation

5.1.2.1 typedef std::map<StringType, CCmdParam> JGTL::_CommandLineParser

5.1.2.2 typedef unsigned char JGTL::uchar

5.1.2.3 typedef unsigned long long JGTL::units_internal_ulong

5.1.3 Enumeration Type Documentation

5.1.3.1 enum JGTL::IntersectionState

Enumerator:

IS_NONE

IS_ONE

IS_INFINITE

5.1.3.2 enum JGTL::TimeFormat

A set of ways to represent timing results.

Enumerator:

SECONDS

MILLISECONDS

MICROSECONDS

PERCENT

5.1.4 Function Documentation

5.1.4.1 **template<class T, class TT> T JGTL::convertVector2 (const TT & *other*)** `[inline]`

5.1.4.2 **template<class T, class TT> T JGTL::convertVector3 (const TT & *other*)** `[inline]`

5.1.4.3 **template<class T, class TT> T JGTL::convertVector4 (const TT & *other*)** `[inline]`

5.1.4.4 **template<> units_internal_ulong JGTL::GCD (float *a*, float *b*)** `[inline]`

5.1.4.5 **template<> units_internal_ulong JGTL::GCD (double *a*, double *b*)** `[inline]`

5.1.4.6 **template<class T> units_internal_ulong JGTL::GCD (T *a*, T *b*)** `[inline]`

5.1.4.7 **template<class T> T JGTL::getIndexFromName (const std::string & *name*, const char ** *names*, T *numNames*)** `[inline]`

5.1.4.8 **template<class T> T JGTL::getIndexFromName (const char * *name*, const char ** *names*, T *numNames*)** `[inline]`

5.1.4.9 **template<class Rectangle, class Point> ostream& JGTL::operator<< (ostream & *stream*, const XorSpace< Rectangle, Point > & *d*)** `[inline]`

5.1.4.10 **template<class T> std::ostream& JGTL::operator<< (std::ostream & *stream*, const WrappedInterpolatedValue< T > & *d*)** `[inline]`

5.1.4.11 **template<class T> std::ostream& JGTL::operator<< (std::ostream & *stream*, const Vector4< T > & *d*)** `[inline]`

5.1.4.12 **template<class T> std::ostream& JGTL::operator<< (std::ostream & *stream*, const Vector3< T > & *d*)** `[inline]`

5.1.4.13 **template<class T> std::ostream& JGTL::operator<< (std::ostream & *stream*, const Vector2< T > & *d*)** `[inline]`

5.1.4.14 **template<class T> std::ostream& JGTL::operator<< (std::ostream & *stream*, const InterpolatedValue< T > & *d*)** `[inline]`

5.1.4.15 **template<class ValueType, ValueType SCALE, bool USEGCD> std::ostream& JGTL::operator<< (std::ostream & *stream*, const IntegralUnits< ValueType, SCALE, USEGCD > & *d*)** `[inline]`

5.1.4.16 **ostream& JGTL::operator<< (ostream & *stream*, const RectangleIndex3 & *d*)** `[inline]`

5.1.4.17 **ostream& JGTL::operator<< (ostream & *stream*, const Index3 & *d*)** `[inline]`

5.1.4.38 template<typename T> std::string JGTL::toString (const T & *x*) [inline]

5.1.4.39 template<class Data> void JGTL::unpackBuffer (uchar *& *buffer*, int & *bufferSize*, Data & *data*) [inline]

5.1.4.40 template<class Data> void JGTL::unpackBufferStack (uchar *& *buffer*, int & *bufferSize*, Data & *data*) [inline]

5.1.4.41 void JGTL::unpackBufferString (uchar *& *buffer*, int & *bufferSize*, string & *s*) [inline]

Chapter 6

Class Documentation

6.1 JGTL::Bar< BarValueType > Class Template Reference

The Bar Class handles a max/current value system (e.g. a Progress Bar).

`#include <JGTL_Bar.h>`

Public Member Functions

- Bar ()
- Bar (const BarValueType &_currentValue, const BarValueType &_maxValue)
- Bar (const BarValueType &_value)

Public Attributes

- BarValueType currentValue
- BarValueType maxValue

6.1.1 Detailed Description

template<class BarValueType> class JGTL::Bar< BarValueType >

The Bar Class handles a max/current value system (e.g. a Progress Bar).

Author:

Jason Gauci 2008

6.1.2 Constructor & Destructor Documentation

6.1.2.1 template<class BarValueType> JGTL::Bar< BarValueType >::Bar () [inline]

6.1.2.2 template<class BarValueType> JGTL::Bar< BarValueType >::Bar (const BarValueType & *_currentValue*, const BarValueType & *_maxValue*) [inline]

6.1.2.3 template<class BarValueType> JGTL::Bar< BarValueType >::Bar (const BarValueType & *_value*) [inline]

6.1.3 Member Data Documentation

6.1.3.1 template<class BarValueType> BarValueType JGTL::Bar< BarValueType >::currentValue

6.1.3.2 template<class BarValueType> BarValueType JGTL::Bar< BarValueType >::maxValue

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

- JGTL_Bar.h

6.2 JGTL::BinaryTreeNode Class Reference

`#include <JGTL_UnorderedMapInterface.h>`

Public Member Functions

- BinaryTreeNode ()
- BinaryTreeNode (BinaryTreeNode *_parent)
- BinaryTreeNode (BinaryTreeNode *_parent, BinaryTreeNode *_left, BinaryTreeNode *_right)

Public Attributes

- BinaryTreeNode * parent
- BinaryTreeNode * left
- BinaryTreeNode * right

Protected Member Functions

- BinaryTreeNode (const BinaryTreeNode &other)
- const BinaryTreeNode & operator= (const BinaryTreeNode &other)

6.2.1 Constructor & Destructor Documentation

6.2.1.1 JGTL::BinaryTreeNode::BinaryTreeNode () [inline]

6.2.1.2 JGTL::BinaryTreeNode::BinaryTreeNode (BinaryTreeNode * *_parent*) [inline]

6.2.1.3 JGTL::BinaryTreeNode::BinaryTreeNode (BinaryTreeNode * *_parent*, BinaryTreeNode * *_left*, BinaryTreeNode * *_right*) [inline]

6.2.1.4 JGTL::BinaryTreeNode::BinaryTreeNode (const BinaryTreeNode & *other*) [protected]

6.2.2 Member Function Documentation

6.2.2.1 const BinaryTreeNode& JGTL::BinaryTreeNode::operator= (const BinaryTreeNode & *other*) [protected]

6.2.3 Member Data Documentation

6.2.3.1 BinaryTreeNode* JGTL::BinaryTreeNode::parent

6.2.3.2 BinaryTreeNode* JGTL::BinaryTreeNode::left

6.2.3.3 BinaryTreeNode* JGTL::BinaryTreeNode::right

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

- JGTL_UnorderedMapInterface.h

6.3 JGTL::CCmdParam Struct Reference

`#include <JGTL_CommandLineParser.h>`

Public Attributes

- std::vector< StringType > m_strings

6.3.1 Member Data Documentation

6.3.1.1 std::vector<StringType> JGTL::CCmdParam::m_strings

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

- JGTL_CommandLineParser.h

6.4 JGTL::CircularBuffer< Data > Class Template Reference

The CircularBuffer Class handles a Circular Buffer.

`#include <JGTL_CircularBuffer.h>`

Public Member Functions

- CircularBuffer (int _maxElements)
- CircularBuffer (const CircularBuffer &other)
- const CircularBuffer & operator= (const CircularBuffer &other)
- virtual ~CircularBuffer ()
- void enqueue (const Data &data)
- Data front ()
- Data & frontRef ()
- const Data & frontRef () const
- Data * frontPtr ()
- const Data * frontPtr () const
- void dequeue ()
- int size () const
- int capacity () const
- bool empty () const
- bool full () const
- void clear ()
- Data * getIndex (int index)
- const Data * getIndex (int index) const
- Data & getIndexRef (int index)
- const Data & getIndexRef (int index) const

Protected Member Functions

- void copyFrom (const CircularBuffer &other)
- void incCounter (int &counter)

Protected Attributes

- int elementStart
- int elementEnd
- int maxElements
- Data * dataList
- bool enqueueLast

6.4.1 Detailed Description

template<class Data> class JGTL::CircularBuffer< Data >

The CircularBuffer Class handles a Circular Buffer.

Author:

Jason Gauci 2008

6.4.2 Constructor & Destructor Documentation

6.4.2.1 template<class Data> JGTL::CircularBuffer< Data >::CircularBuffer (int *_maxElements*) [inline]

6.4.2.2 template<class Data> JGTL::CircularBuffer< Data >::CircularBuffer (const CircularBuffer< Data > & *other*) [inline]

6.4.2.3 template<class Data> virtual JGTL::CircularBuffer< Data >::~CircularBuffer () [inline, virtual]

6.4.3 Member Function Documentation

6.4.3.1 template<class Data> const CircularBuffer& JGTL::CircularBuffer< Data >::operator= (const CircularBuffer< Data > & *other*) [inline]

6.4.3.2 template<class Data> void JGTL::CircularBuffer< Data >::enqueue (const Data & *data*) [inline]

6.4.3.3 template<class Data> Data JGTL::CircularBuffer< Data >::front () [inline]

6.4.3.4 template<class Data> Data& JGTL::CircularBuffer< Data >::frontRef () [inline]

6.4.3.5 template<class Data> const Data& JGTL::CircularBuffer< Data >::frontRef () const [inline]

6.4.3.6 template<class Data> Data* JGTL::CircularBuffer< Data >::frontPtr () [inline]

6.4.3.7 template<class Data> const Data* JGTL::CircularBuffer< Data >::frontPtr () const [inline]

6.4.3.8 template<class Data> void JGTL::CircularBuffer< Data >::dequeue () [inline]

6.4.3.9 template<class Data> int JGTL::CircularBuffer< Data >::size () const [inline]

6.4.3.10 template<class Data> int JGTL::CircularBuffer< Data >::capacity () const [inline]

6.4.3.11 template<class Data> bool JGTL::CircularBuffer< Data >::empty () const [inline]

6.4.3.12 template<class Data> bool JGTL::CircularBuffer< Data >::full () const [inline]

6.4.3.13 template<class Data> void JGTL::CircularBuffer< Data >::clear () [inline]

6.4.3.14 template<class Data> Data* JGTL::CircularBuffer< Data >::getIndex (int *index*) [inline]

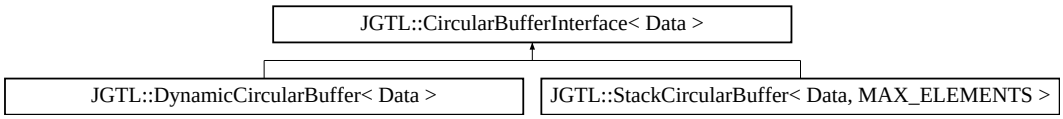
- JGTL_CircularBuffer.h

6.5 JGTL::CircularBufferInterface< Data > Class Template Reference

The CircularBufferInterface Class handles a Circular Buffer.

`#include <JGTL_CircularBufferInterface.h>`

Inheritance diagram for JGTL::CircularBufferInterface< Data >::



Public Member Functions

- CircularBufferInterface (int _maxElements=0)
- virtual ~CircularBufferInterface ()
- void enqueue (const Data &data)
- virtual bool resize (int newSize)=0
- Data front ()
- Data & frontRef ()
- const Data & frontRef () const
- Data * frontPtr ()
- const Data * frontPtr () const
- void dequeue ()
- int size () const
- int capacity () const
- bool empty () const
- bool full () const
- void clear ()
- Data * getIndex (int index)
- const Data * getIndex (int index) const
- Data & getIndexRef (int index)
- const Data & getIndexRef (int index) const

Protected Member Functions

- void incCounter (int &counter)

Protected Attributes

- int elementStart
- int elementEnd
- int maxElements
- Data * dataList
- bool enqueueLast

6.5.1 Detailed Description

template<class Data> class JGTL::CircularBufferInterface< Data >

The CircularBufferInterface Class handles a Circular Buffer.

Author:

Jason Gauci 2008

6.5.2 Constructor & Destructor Documentation

6.5.2.1 template<class Data> JGTL::CircularBufferInterface< Data >::CircularBufferInterface (int *_maxElements* = 0) [inline]

6.5.2.2 template<class Data> virtual JGTL::CircularBufferInterface< Data >::~CircularBufferInterface () [inline, virtual]

6.5.3 Member Function Documentation

6.5.3.1 template<class Data> void JGTL::CircularBufferInterface< Data >::enqueue (const Data & *data*) [inline]

6.5.3.2 template<class Data> virtual bool JGTL::CircularBufferInterface< Data >::resize (int *newSize*) [pure virtual]

Implemented in JGTL::DynamicCircularBuffer< Data >, and JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >.

6.5.3.3 template<class Data> Data JGTL::CircularBufferInterface< Data >::front () [inline]

6.5.3.4 template<class Data> Data& JGTL::CircularBufferInterface< Data >::frontRef () [inline]

6.5.3.5 template<class Data> const Data& JGTL::CircularBufferInterface< Data >::frontRef () const [inline]

6.5.3.6 template<class Data> Data* JGTL::CircularBufferInterface< Data >::frontPtr () [inline]

6.5.3.7 template<class Data> const Data* JGTL::CircularBufferInterface< Data >::frontPtr () const [inline]

6.5.3.8 template<class Data> void JGTL::CircularBufferInterface< Data >::dequeue () [inline]

6.5.3.9 template<class Data> int JGTL::CircularBufferInterface< Data >::size () const [inline]

6.5.3.10 template<class Data> int JGTL::CircularBufferInterface< Data >::capacity () const [inline]

6.5.3.11 template<class Data> bool JGTL::CircularBufferInterface< Data >::empty () const [inline]

6.5.3.12 template<class Data> bool JGTL::CircularBufferInterface< Data >::full () const [inline]

6.5.3.13 template<class Data> void JGTL::CircularBufferInterface< Data >::clear () [inline]

6.5.3.14 template<class Data> Data* JGTL::CircularBufferInterface< Data >::getIndex (int *index*) [inline]

6.5.3.15 template<class Data> const Data* JGTL::CircularBufferInterface< Data >::getIndex (int *index*) const [inline]

6.5.3.16 template<class Data> Data& JGTL::CircularBufferInterface< Data >::getIndexRef (int *index*) [inline]

6.5.3.17 template<class Data> const Data& JGTL::CircularBufferInterface< Data >::getIndexRef (int *index*) const [inline]

6.5.3.18 template<class Data> void JGTL::CircularBufferInterface< Data >::incCounter (int & *counter*) [inline, protected]

6.5.4 Member Data Documentation

6.5.4.1 template<class Data> int JGTL::CircularBufferInterface< Data >::elementStart [protected]

6.5.4.2 template<class Data> int JGTL::CircularBufferInterface< Data >::elementEnd [protected]

- JGTL_CircularBufferInterface.h

6.6 JGTL::Clock Class Reference

`#include <JGTL_QuickProf.h>`

Public Member Functions

- Clock ()
- ~Clock ()
- void reset ()
- unsigned long int getTimeMilliseconds ()
- unsigned long int getTimeMicroseconds ()

Private Attributes

- struct timeval mStartTime

6.6.1 Detailed Description

A cross-platform clock class inspired by the Timer classes in Ogre (`http://www.ogre3d.org`).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 JGTL::Clock::Clock () [inline]

6.6.2.2 JGTL::Clock::~Clock () [inline]

6.6.3 Member Function Documentation

6.6.3.1 void JGTL::Clock::reset () [inline]

Resets the initial reference time.

6.6.3.2 unsigned long int JGTL::Clock::getTimeMilliseconds () [inline]

Returns the time in us since the last call to reset or since the Clock was created.

Returns:

The requested time in milliseconds.

6.6.3.3 unsigned long int JGTL::Clock::getTimeMicroseconds () [inline]

Returns the time in us since the last call to reset or since the Clock was created.

Returns:

The requested time in microseconds.

6.6.4 Member Data Documentation

6.6.4.1 struct timeval JGTL::Clock::mStartTime [read, private]

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

- JGTL_QuickProf.h

6.7 JGTL::CommandLineParser Class Reference

`#include <JGTL_CommandLineParser.h>`

Public Member Functions

- CommandLineParser ()
- CommandLineParser (int argc, char **argv)
- int SplitLine (int argc, char **argv)
- bool HasSwitch (const char *pSwitch)
- StringType GetSafeArgument (const char *pSwitch, int iIdx, const char *pDefault)
- StringType GetArgument (const char *pSwitch, int iIdx)
- int GetArgumentCount (const char *pSwitch)

Protected Member Functions

- bool IsSwitch (const char *pParam)

6.7.1 Constructor & Destructor Documentation

6.7.1.1 JGTL::CommandLineParser::CommandLineParser () [inline]

6.7.1.2 JGTL::CommandLineParser::CommandLineParser (int *argc*, char ** *argv*) [inline]

6.7.2 Member Function Documentation

6.7.2.1 int JGTL::CommandLineParser::SplitLine (int *argc*, char ** *argv*) [inline]

6.7.2.2 bool JGTL::CommandLineParser::HasSwitch (const char * *pSwitch*) [inline]

6.7.2.3 StringType JGTL::CommandLineParser::GetSafeArgument (const char * *pSwitch*, int *iIdx*, const char * *pDefault*) [inline]

6.7.2.4 StringType JGTL::CommandLineParser::GetArgument (const char * *pSwitch*, int *iIdx*) [inline]

6.7.2.5 int JGTL::CommandLineParser::GetArgumentCount (const char * *pSwitch*) [inline]

6.7.2.6 bool JGTL::CommandLineParser::IsSwitch (const char * *pParam*) [inline, protected]

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

- JGTL_CommandLineParser.h

6.8 JGTL::DataManager< Data > Class Template Reference

`#include <JGTL_DataManager.h>`

Public Member Functions

- DataManager (int _maxElements)
- virtual ~DataManager ()
- void addData (Data &data)
- Data & getData (int index) const
- Data & getData (const string &key, bool caseSensitive=true) const
- Data * getDataPtr (int index) const
- Data * begin () const
- Data * end () const
- int getSize () const
- Data * getDataPtr (const std::string &key, bool caseSensitive=true) const
- int getIndex (const Data *data) const
- void refreshNames ()

Protected Attributes

- int numElements
- int maxElements
- Data * dataList
- std::map< std::string, Data * > dataMap
- std::map< std::string, Data * > dataCaseInsensitiveMap

template<class Data> class JGTL::DataManager< Data >

6.8.1 Constructor & Destructor Documentation

6.8.1.1 template<class Data> JGTL::DataManager< Data >::DataManager (int *_maxElements*) [inline]

6.8.1.2 template<class Data> virtual JGTL::DataManager< Data >::~DataManager () [inline, virtual]

6.8.2 Member Function Documentation

6.8.2.1 template<class Data> void JGTL::DataManager< Data >::addData (Data & *data*) [inline]

6.8.2.2 template<class Data> Data& JGTL::DataManager< Data >::getData (int *index*) const [inline]

6.8.2.3 template<class Data> Data& JGTL::DataManager< Data >::getData (const string & *key*, bool *caseSensitive* = true) const [inline]

6.8.2.4 template<class Data> Data* JGTL::DataManager< Data >::getDataPtr (int *index*) const [inline]

6.8.2.5 template<class Data> Data* JGTL::DataManager< Data >::begin () const [inline]

6.8.2.6 template<class Data> Data* JGTL::DataManager< Data >::end () const [inline]

6.8.2.7 template<class Data> int JGTL::DataManager< Data >::getSize () const [inline]

6.8.2.8 template<class Data> Data* JGTL::DataManager< Data >::getDataPtr (const std::string & *key*, bool *caseSensitive* = true) const [inline]

6.8.2.9 template<class Data> int JGTL::DataManager< Data >::getIndex (const Data * *data*) const [inline]

6.8.2.10 template<class Data> void JGTL::DataManager< Data >::refreshNames () [inline]

6.8.3 Member Data Documentation

6.8.3.1 template<class Data> int JGTL::DataManager< Data >::numElements [protected]

6.8.3.2 template<class Data> int JGTL::DataManager< Data >::maxElements [protected]

6.8.3.3 template<class Data> Data* JGTL::DataManager< Data >::dataList [protected]

6.8.3.4 template<class Data> std::map<std::string Data *>

- JGTL_DataManager.h

6.9 JGTL::DataPool< Data > Class Template Reference

`#include <JGTL_DataPool_delete.h>`

Public Member Functions

- DataPool (size_t _maxElements)
- virtual ~DataPool ()
- void addData (const Data &data)
- Data & getData (size_t index) const
- Data & getData (const string &key, bool caseSensitive=true) const
- Data * getDataPtr (size_t index) const
- Data * begin () const
- Data * end () const
- size_t getSize () const
- Data * getDataPtr (const string &key, bool caseSensitive=true)

Protected Attributes

- size_t numElements
- size_t maxElements
- Data * dataList
- bool * used
- map< string, Data * > dataMap
- map< string, Data * > dataCaseInsensitiveMap
- allocator< Data > alloc
- allocator< bool > boolAlloc

template<class Data> class JGTL::DataPool< Data >

6.9.1 Constructor & Destructor Documentation

6.9.1.1 template<class Data> JGTL::DataPool< Data >::DataPool (size_t *_maxElements*) [inline]

6.9.1.2 template<class Data> virtual JGTL::DataPool< Data >::~DataPool () [inline, virtual]

6.9.2 Member Function Documentation

6.9.2.1 template<class Data> void JGTL::DataPool< Data >::addData (const Data & *data*) [inline]

6.9.2.2 template<class Data> Data& JGTL::DataPool< Data >::getData (size_t *index*) const [inline]

6.9.2.3 template<class Data> Data& JGTL::DataPool< Data >::getData (const string & *key*, bool *caseSensitive* = true) const [inline]

6.9.2.4 template<class Data> Data* JGTL::DataPool< Data >::getDataPtr (size_t *index*) const [inline]

6.9.2.5 template<class Data> Data* JGTL::DataPool< Data >::begin () const [inline]

6.9.2.6 template<class Data> Data* JGTL::DataPool< Data >::end () const [inline]

6.9.2.7 template<class Data> size_t JGTL::DataPool< Data >::getSize () const [inline]

6.9.2.8 template<class Data> Data* JGTL::DataPool< Data >::getDataPtr (const string & *key*, bool *caseSensitive* = true) [inline]

6.9.3 Member Data Documentation

6.9.3.1 template<class Data> size_t JGTL::DataPool< Data >::numElements [protected]

6.9.3.2 template<class Data> size_t JGTL::DataPool< Data >::maxElements [protected]

6.9.3.3 template<class Data> Data* JGTL::DataPool< Data >::dataList [protected]

6.9.3.4 template<class Data> bool* JGTL::DataPool< Data >::used [protected]

6.9.3.5 template<class Data> map<string,Data *> JGTL::DataPool< Data >::dataMap [protected]

6.9.3.6 template<class Data> map<string,Data *> JGTL::DataPool< Data >::dataCaseInsensitiveMap [protected]

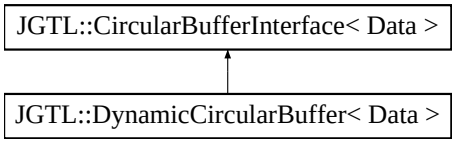
- JGTL_DataPool_delete.h

6.10 JGTL::DynamicCircularBuffer< Data > Class Template Reference

The DynamicCircularBuffer Class handles a Circular Buffer.

`#include <JGTL_DynamicCircularBuffer.h>`

Inheritance diagram for JGTL::DynamicCircularBuffer< Data >::



Public Member Functions

- DynamicCircularBuffer (int initialSize=0)
- DynamicCircularBuffer (const DynamicCircularBuffer &other)
- const DynamicCircularBuffer & operator= (const DynamicCircularBuffer &other)
- virtual ~DynamicCircularBuffer ()
- virtual bool resize (int newSize)

Protected Member Functions

- void copyFrom (const DynamicCircularBuffer &other)

6.10.1 Detailed Description

template<class Data> class JGTL::DynamicCircularBuffer< Data >

The DynamicCircularBuffer Class handles a Circular Buffer.

Author:

Jason Gauci 2008

6.10.2 Constructor & Destructor Documentation

6.10.2.1 template<class Data> JGTL::DynamicCircularBuffer< Data >::DynamicCircularBuffer (int *initialSize* = 0) [inline]

6.10.2.2 template<class Data> JGTL::DynamicCircularBuffer< Data >::DynamicCircularBuffer (const DynamicCircularBuffer< Data > & *other*) [inline]

6.10.2.3 template<class Data> virtual JGTL::DynamicCircularBuffer< Data >::~DynamicCircularBuffer () [inline, virtual]

6.10.3 Member Function Documentation

6.10.3.1 template<class Data> const DynamicCircularBuffer& JGTL::DynamicCircularBuffer< Data >::operator= (const DynamicCircularBuffer< Data > & *other*) [inline]

6.10.3.2 template<class Data> virtual bool JGTL::DynamicCircularBuffer< Data >::resize (int *newSize*) [inline, virtual]

Implements JGTL::CircularBufferInterface< Data >.

6.10.3.3 template<class Data> void JGTL::DynamicCircularBuffer< Data >::copyFrom (const DynamicCircularBuffer< Data > & *other*) [inline, protected]

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

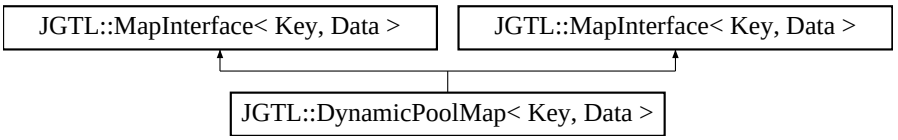
- JGTL_DynamicCircularBuffer.h

6.11 JGTL::DynamicPoolMap< Key, Data > Class Template Reference

The DynamicPoolMap Class is a resizable array-based associative map structure.

`#include <JGTL_DynamicPoolMap.h>`

Inheritance diagram for JGTL::DynamicPoolMap< Key, Data >::



Public Member Functions

- DynamicPoolMap ()
- DynamicPoolMap (int _capacity)
- DynamicPoolMap (const DynamicPoolMap &other)
- const DynamicPoolMap & operator= (const DynamicPoolMap &other)
- virtual ~DynamicPoolMap ()
- virtual bool resize (int newSize)
- DynamicPoolMap ()
- DynamicPoolMap (const DynamicPoolMap &other)
- const DynamicPoolMap & operator= (const DynamicPoolMap &other)
- virtual void copyFrom (const DynamicPoolMap &other)
- virtual ~DynamicPoolMap ()
- virtual bool reserve (int newSize)

Protected Member Functions

- virtual void copyFrom (const DynamicPoolMap &other)

6.11.1 Detailed Description

template<class Key, class Data> class JGTL::DynamicPoolMap< Key, Data >

The DynamicPoolMap Class is a resizable array-based associative map structure.

Author:

Jason Gauci 2008

6.11.2 Constructor & Destructor Documentation

6.11.2.1 template<class Key, class Data> JGTL::DynamicPoolMap< Key, Data >::DynamicPoolMap () [inline]

6.11.2.2 template<class Key, class Data> JGTL::DynamicPoolMap< Key, Data >::DynamicPoolMap (int *_capacity*) [inline]

6.11.2.3 template<class Key, class Data> JGTL::DynamicPoolMap< Key, Data >::DynamicPoolMap (const DynamicPoolMap< Key, Data > & *other*) [inline]

6.11.2.4 template<class Key, class Data> virtual JGTL::DynamicPoolMap< Key, Data >::~DynamicPoolMap () [inline, virtual]

6.11.2.5 template<class Key, class Data> JGTL::DynamicPoolMap< Key, Data >::DynamicPoolMap () [inline]

6.11.2.6 template<class Key, class Data> JGTL::DynamicPoolMap< Key, Data >::DynamicPoolMap (const DynamicPoolMap< Key, Data > & *other*) [inline]

6.11.2.7 template<class Key, class Data> virtual JGTL::DynamicPoolMap< Key, Data >::~DynamicPoolMap () [inline, virtual]

6.11.3 Member Function Documentation

6.11.3.1 template<class Key, class Data> const DynamicPoolMap& JGTL::DynamicPoolMap< Key, Data >::operator= (const DynamicPoolMap< Key, Data > & *other*) [inline]

6.11.3.2 template<class Key, class Data> virtual bool JGTL::DynamicPoolMap< Key, Data >::resize (int *newSize*) [inline, virtual]

Implements JGTL::MapInterface< Key, Data >.

6.11.3.3 template<class Key, class Data> virtual void JGTL::DynamicPoolMap< Key, Data >::copyFrom (const DynamicPoolMap< Key, Data > & *other*) [inline, protected, virtual]

6.11.3.4 template<class Key, class Data> const DynamicPoolMap& JGTL::DynamicPoolMap< Key, Data >::operator= (const DynamicPoolMap< Key, Data > & *other*) [inline]

6.11.3.5 template<class Key, class Data> virtual void JGTL::DynamicPoolMap< Key, Data >::copyFrom (const DynamicPoolMap< Key, Data > & *other*) [inline, virtual]

6.11.3.6 template<class Key, class Data> virtual bool JGTL::DynamicPoolMap< Key, Data >::reserve (int *newSize*) [inline, virtual]

Reimplemented from JGTL::MapInterface< Key, Data >.

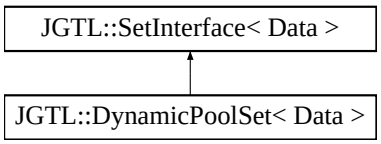
The documentation for this class was generated from the following files:

- JGTL_DynamicPoolMap.h
- JGTL_UnorderedDynamicPoolMap.h

6.12 JGTL::DynamicPoolSet< Data > Class Template Reference

`#include <JGTL_DynamicPoolSet.h>`

Inheritance diagram for JGTL::DynamicPoolSet< Data >::



Public Member Functions

- DynamicPoolSet ()
- DynamicPoolSet (const DynamicPoolSet< Data > &other)
- const DynamicPoolSet & operator= (const DynamicPoolSet< Data > &other)
- void copyFrom (const DynamicPoolSet &other)
- virtual ~DynamicPoolSet ()
- bool operator== (const DynamicPoolSet &other) const
- virtual bool resize (int newSize)

template<class Data> class JGTL::DynamicPoolSet< Data >

6.12.1 Constructor & Destructor Documentation

6.12.1.1 template<class Data> JGTL::DynamicPoolSet< Data >::DynamicPoolSet ()

`[inline]`

6.12.1.2 template<class Data> JGTL::DynamicPoolSet< Data >::DynamicPoolSet (const DynamicPoolSet< Data > & *other*)

`[inline]`

6.12.1.3 template<class Data> virtual JGTL::DynamicPoolSet< Data >::~DynamicPoolSet ()

`[inline, virtual]`

6.12.2 Member Function Documentation

6.12.2.1 template<class Data> const DynamicPoolSet& JGTL::DynamicPoolSet< Data >::operator= (const DynamicPoolSet< Data > & *other*)

`[inline]`

6.12.2.2 template<class Data> void JGTL::DynamicPoolSet< Data >::copyFrom (const DynamicPoolSet< Data > & *other*)

`[inline]`

6.12.2.3 template<class Data> bool JGTL::DynamicPoolSet< Data >::operator== (const DynamicPoolSet< Data > & *other*) const

`[inline]`

6.12.2.4 template<class Data> virtual bool JGTL::DynamicPoolSet< Data >::resize (int *newSize*)

`[inline, virtual]`

Reimplemented from JGTL::SetInterface< Data >.

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

- JGTL_DynamicPoolSet.h

6.13 JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR > Class Template Reference

`#include <JGTL_FloatingUnits.h>`

Public Member Functions

- FloatingUnits (ValueType _value=0)
- template<class OtherValueType, units_internal_ulong OTHERSCALE_NUMERATOR, units_internal_ulong OTHERSCALE_DENOMINATOR>
  FloatingUnits (const FloatingUnits< OtherValueType, OTHERSCALE_NUMERATOR, OTHERSCALE_DENOMINATOR > &t)
- template<class OtherValueType, units_internal_ulong OTHERSCALE_NUMERATOR, units_internal_ulong OTHERSCALE_DENOMINATOR>
  const FloatingUnits & operator= (const FloatingUnits< OtherValueType, OTHERSCALE_NUMERATOR, OTHERSCALE_DENOMINATOR > &t)
- template<class OtherValueType, units_internal_ulong OTHERSCALE_NUMERATOR, units_internal_ulong OTHERSCALE_DENOMINATOR>
  ValueType changeScale (const FloatingUnits< OtherValueType, OTHERSCALE_NUMERATOR, OTHERSCALE_DENOMINATOR > &t)
- virtual ValueType getScale () const
- void setValue (ValueType _value)
- ValueType getValue () const

Protected Attributes

- ValueType value

template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR> class JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >

6.13.1 Constructor & Destructor Documentation

6.13.1.1 template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR> JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >::FloatingUnits (ValueType *_value* = `0`) `[inline]`

6.13.1.2 template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR> template<class OtherValueType, units_internal_ulong OTHERSCALE_NUMERATOR, units_internal_ulong OTHERSCALE_DENOMINATOR> JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >::FloatingUnits (const FloatingUnits< OtherValueType, OTHERSCALE_NUMERATOR, OTHERSCALE_DENOMINATOR > & *t*) `[inline]`

6.13.2 Member Function Documentation

6.13.2.1 template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR> template<class OtherValueType, units_internal_ulong OTHERSCALE_NUMERATOR, units_internal_ulong OTHERSCALE_DENOMINATOR> const FloatingUnits& JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >::operator= (const FloatingUnits< OtherValueType, OTHERSCALE_NUMERATOR, OTHERSCALE_DENOMINATOR > & *t*) `[inline]`

6.13.2.2 template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR> template<class OtherValueType, units_internal_ulong OTHERSCALE_NUMERATOR, units_internal_ulong OTHERSCALE_DENOMINATOR> ValueType JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >::changeScale (const FloatingUnits< OtherValueType, OTHERSCALE_NUMERATOR, OTHERSCALE_DENOMINATOR > & *t*) `[inline]`

6.13.2.3 template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR> virtual ValueType JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >::getScale () const `[inline, virtual]`

6.13.2.4 template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR> void JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >::setValue (ValueType *_value*) `[inline]`

- JGTL_FloatingUnits.h

6.14 JGTL::HexTree< T > Class Template Reference

`#include <JGTL_HexTree.h>`

Public Member Functions

- HexTree (int _size=16, T defaultValue=(T) 0)
- HexTree (const HexTree< T > &other)
- ~HexTree ()
- HexTree< T > & operator= (const HexTree< T > &other)
- void copyFrom (const HexTree< T > &other)
- T getValue (const Vector4< int > &location) const
- T getValue (int x1, int y1, int x2, int y2) const
- void setValue (const Vector4< int > &location, T value)
- void setValue (int x1, int y1, int x2, int y2, T value)
- void setAll (T value)
- void display () const
- T operator() (const Vector4< int > &location) const
- T operator() (int x1, int y1, int x2, int y2) const
- int getMemUsage () const

Private Attributes

- int size
- HexTreeBranch< T > * root
- pool branchPool
- pool stubPool

template<class T> class JGTL::HexTree< T >

6.14.1 Constructor & Destructor Documentation

6.14.1.1 template<class T> JGTL::HexTree< T >::HexTree (int *_size* = 16, T *defaultValue* = (T) 0) [inline]

6.14.1.2 template<class T> JGTL::HexTree< T >::HexTree (const HexTree< T > & *other*) [inline]

6.14.1.3 template<class T> JGTL::HexTree< T >::~HexTree () [inline]

6.14.2 Member Function Documentation

6.14.2.1 template<class T> HexTree<T>& JGTL::HexTree< T >::operator= (const HexTree< T > & *other*) [inline]

6.14.2.2 template<class T> void JGTL::HexTree< T >::copyFrom (const HexTree< T > & *other*) [inline]

6.14.2.3 template<class T> T JGTL::HexTree< T >::getValue (const Vector4< int > & *location*) const [inline]

6.14.2.4 template<class T> T JGTL::HexTree< T >::getValue (int *x1*, int *y1*, int *x2*, int *y2*) const [inline]

6.14.2.5 template<class T> void JGTL::HexTree< T >::setValue (const Vector4< int > & *location*, T *value*) [inline]

6.14.2.6 template<class T> void JGTL::HexTree< T >::setValue (int *x1*, int *y1*, int *x2*, int *y2*, T *value*) [inline]

6.14.2.7 template<class T> void JGTL::HexTree< T >::setAll (T *value*) [inline]

6.14.2.8 template<class T> void JGTL::HexTree< T >::display () const [inline]

6.14.2.9 template<class T> T JGTL::HexTree< T >::operator() (const Vector4< int > & *location*) const [inline]

6.14.2.10 template<class T> T JGTL::HexTree< T >::operator() (int *x1*, int *y1*, int *x2*, int *y2*) const [inline]

6.14.2.11 template<class T> int JGTL::HexTree< T >::getMemUsage () const [inline]

6.14.3 Member Data Documentation

6.14.3.1 template<class T> int JGTL::HexTree< T >::size [private]

6.14.3.2 template<class T> HexTreeBranch<T>* JGTL::HexTree< T >::root [private]

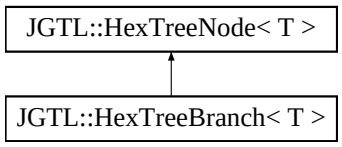
6.14.3.3 template<class T> pool JGTL::HexTree< T >::branchPool

- JGTL_HexTree.h

6.15 JGTL::HexTreeBranch< T > Class Template Reference

`#include <JGTL_HexTree.h>`

Inheritance diagram for JGTL::HexTreeBranch< T >::



Public Member Functions

- HexTreeBranch (pool<> &branchPool, pool<> &stubPool, const T &value)
- virtual ~HexTreeBranch ()
- virtual void destroy (pool<> &branchPool, pool<> &stubPool)
- int getChildIndex (const Vector4< int > &location, const Vector4< int > &center, const Vector4< int > &topLeftVector4, Vector4< int > &newTopLeftVector4) const
- virtual T getValue (Vector4< int > topLeftVector4, int size, const Vector4< int > &location) const
- virtual T getValue () const
- void setAll (pool<> &branchPool, pool<> &stubPool, T &value)
- virtual bool setValue (pool<> &branchPool, pool<> &stubPool, Vector4< int > topLeftVector4, int size, const Vector4< int > &location, const T &value)
- virtual void display (int level) const
- virtual int getMemUsage () const

Private Attributes

- HexTreeNode< T > * children [16]

template<class T> class JGTL::HexTreeBranch< T >

6.15.1 Constructor & Destructor Documentation

6.15.1.1 template<class T> JGTL::HexTreeBranch< T >::HexTreeBranch (pool<> & *branchPool*, pool<> & *stubPool*, const T & *value*) [inline]

6.15.1.2 template<class T> virtual JGTL::HexTreeBranch< T >::~HexTreeBranch () [inline, virtual]

6.15.2 Member Function Documentation

6.15.2.1 template<class T> virtual void JGTL::HexTreeBranch< T >::destroy (pool<> & *branchPool*, pool<> & *stubPool*) [inline, virtual]

Reimplemented from JGTL::HexTreeNode< T >.

6.15.2.2 template<class T> int JGTL::HexTreeBranch< T >::getChildIndex (const Vector4< int > & *location*, const Vector4< int > & *center*, const Vector4< int > & *topLeftVector4*, Vector4< int > & *newTopLeftVector4*) const [inline]

6.15.2.3 template<class T> virtual T JGTL::HexTreeBranch< T >::getValue (Vector4< int > *topLeftVector4*, int *size*, const Vector4< int > & *location*) const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.15.2.4 template<class T> virtual T JGTL::HexTreeBranch< T >::getValue () const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.15.2.5 template<class T> void JGTL::HexTreeBranch< T >::setAll (pool<> & *branchPool*, pool<> & *stubPool*, T & *value*) [inline]

6.15.2.6 template<class T> virtual bool JGTL::HexTreeBranch< T >::setValue (pool<> & *branchPool*, pool<> & *stubPool*, Vector4< int > *topLeftVector4*, int *size*, const Vector4< int > & *location*, const T & *value*) [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.15.2.7 template<class T> virtual void JGTL::HexTreeBranch< T >::display (int *level*) const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.15.2.8 template<class T> virtual int JGTL::HexTreeBranch< T >::getMemUsage () const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.15.3 Member Data Documentation

6.15.3.1 template<class T> HexTreeNode<T>* JGTL::HexTreeBranch< T >::children[16] [private]

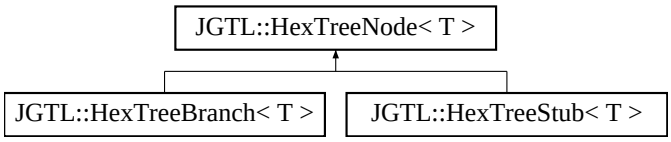
The documentation for this class was generated from the following file:

- JGTL_HexTree.h

6.16 JGTL::HexTreeNode< T > Class Template Reference

`#include <JGTL_HexTree.h>`

Inheritance diagram for JGTL::HexTreeNode< T >::



Public Member Functions

- HexTreeNode ()
- virtual ~HexTreeNode ()
- virtual T getValue (Vector4< int > topLeftVector4, int size, const Vector4< int > &location) const =0
- virtual T getValue () const =0
- virtual bool setValue (pool<> &branchPool, pool<> &stubPool, Vector4< int > topLeftVector4, int size, const Vector4< int > &location, const T &value)=0
- virtual bool isStub () const
- virtual void destroy (pool<> &branchPool, pool<> &stubPool)
- virtual void display (int level) const =0
- virtual int getMemUsage () const =0

template<class T> class JGTL::HexTreeNode< T >

6.16.1 Constructor & Destructor Documentation

6.16.1.1 template<class T> JGTL::HexTreeNode< T >::HexTreeNode () [inline]

6.16.1.2 template<class T> virtual JGTL::HexTreeNode< T >::~HexTreeNode () [inline, virtual]

6.16.2 Member Function Documentation

6.16.2.1 template<class T> virtual T JGTL::HexTreeNode< T >::getValue (Vector4< int > *topLeftVector4*, int *size*, const Vector4< int > & *location*) const [pure virtual]

Implemented in JGTL::HexTreeStub< T >, and JGTL::HexTreeBranch< T >.

6.16.2.2 template<class T> virtual T JGTL::HexTreeNode< T >::getValue () const [pure virtual]

Implemented in JGTL::HexTreeStub< T >, and JGTL::HexTreeBranch< T >.

6.16.2.3 template<class T> virtual bool JGTL::HexTreeNode< T >::setValue (pool<> & *branchPool*, pool<> & *stubPool*, Vector4< int > *topLeftVector4*, int *size*, const Vector4< int > & *location*, const T & *value*) [pure virtual]

Implemented in JGTL::HexTreeStub< T >, and JGTL::HexTreeBranch< T >.

6.16.2.4 template<class T> virtual bool JGTL::HexTreeNode< T >::isStub () const [inline, virtual]

Reimplemented in JGTL::HexTreeStub< T >.

6.16.2.5 template<class T> virtual void JGTL::HexTreeNode< T >::destroy (pool<> & *branchPool*, pool<> & *stubPool*) [inline, virtual]

Reimplemented in JGTL::HexTreeBranch< T >.

6.16.2.6 template<class T> virtual void JGTL::HexTreeNode< T >::display (int *level*) const [pure virtual]

Implemented in JGTL::HexTreeStub< T >, and JGTL::HexTreeBranch< T >.

6.16.2.7 template<class T> virtual int JGTL::HexTreeNode< T >::getMemUsage () const [pure virtual]

Implemented in JGTL::HexTreeStub< T >, and JGTL::HexTreeBranch< T >.

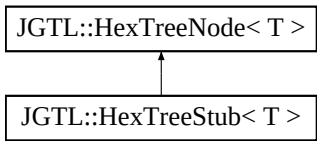
The documentation for this class was generated from the following file:

- JGTL_HexTree.h

6.17 JGTL::HexTreeStub< T > Class Template Reference

`#include <JGTL_HexTree.h>`

Inheritance diagram for JGTL::HexTreeStub< T >::



Public Member Functions

- HexTreeStub (T _value)
- virtual ~HexTreeStub ()
- virtual T getValue (Vector4< int > topLeftVector4, int size, const Vector4< int > &location) const
- virtual T getValue () const
- virtual bool setValue (pool<> &branchPool, pool<> &stubPool, Vector4< int > topLeftVector4, int size, const Vector4< int > &location, const T &value)
- virtual bool setValue (const T &_value)
- virtual bool isStub () const
- virtual void display (int level) const
- virtual int getMemUsage () const

Protected Attributes

- T value

template<class T> class JGTL::HexTreeStub< T >

6.17.1 Constructor & Destructor Documentation

6.17.1.1 template<class T> JGTL::HexTreeStub< T >::HexTreeStub (T *_value*) [inline]

6.17.1.2 template<class T> virtual JGTL::HexTreeStub< T >::~HexTreeStub () [inline, virtual]

6.17.2 Member Function Documentation

6.17.2.1 template<class T> virtual T JGTL::HexTreeStub< T >::getValue (Vector4< int > *topLeftVector4*, int *size*, const Vector4< int > & *location*) const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.17.2.2 template<class T> virtual T JGTL::HexTreeStub< T >::getValue () const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.17.2.3 template<class T> virtual bool JGTL::HexTreeStub< T >::setValue (pool<> & *branchPool*, pool<> & *stubPool*, Vector4< int > *topLeftVector4*, int *size*, const Vector4< int > & *location*, const T & *value*) [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.17.2.4 template<class T> virtual bool JGTL::HexTreeStub< T >::setValue (const T & *_value*) [inline, virtual]

6.17.2.5 template<class T> virtual bool JGTL::HexTreeStub< T >::isStub () const [inline, virtual]

Reimplemented from JGTL::HexTreeNode< T >.

6.17.2.6 template<class T> virtual void JGTL::HexTreeStub< T >::display (int *level*) const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.17.2.7 template<class T> virtual int JGTL::HexTreeStub< T >::getMemUsage () const [inline, virtual]

Implements JGTL::HexTreeNode< T >.

6.17.3 Member Data Documentation

6.17.3.1 template<class T> T JGTL::HexTreeStub< T >::value [protected]

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

- JGTL_HexTree.h

6.18 JGTL::IF< condition, Then, Else > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Public Types

- typedef Then RET

template<bool condition, class Then, class Else> struct JGTL::IF< condition, Then, Else >

6.18.1 Member Typedef Documentation

6.18.1.1 template<bool condition, class Then, class Else> typedef Then JGTL::IF< condition, Then, Else >::RET

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

- JGTL_IntegralUnits.h

6.19 JGTL::JGTL::IF< false, Then, Else > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Public Types

- typedef Else RET

template<class Then, class Else> struct JGTL::JGTL::IF< false, Then, Else >

6.19.1 Member Typedef Documentation

6.19.1.1 template<class Then, class Else> typedef Else JGTL::JGTL::IF< false, Then, Else >::RET

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

- JGTL_IntegralUnits.h

6.20 JGTL::Index2 Class Reference

`#include <JGTL_Index2.h>`

Public Member Functions

- Index2 ()
- Index2 (int _x, int _y)
- Index2 (std::string s)
- Ogre::Vector2 getVector2 ()
- std::string toString ()
- bool operator== (const Index2 &index)
- bool operator!= (const Index2 &index)

Public Attributes

- int x
- int y

6.20.1 Constructor & Destructor Documentation

6.20.1.1 JGTL::Index2::Index2 ()

6.20.1.2 JGTL::Index2::Index2 (int *_x*, int *_y*)

6.20.1.3 JGTL::Index2::Index2 (std::string *s*) [inline]

6.20.2 Member Function Documentation

6.20.2.1 Ogre::Vector2 JGTL::Index2::getVector2 () [inline]

6.20.2.2 std::string JGTL::Index2::toString () [inline]

6.20.2.3 bool JGTL::Index2::operator== (const Index2 & *index*) [inline]

6.20.2.4 bool JGTL::Index2::operator!= (const Index2 & *index*) [inline]

6.20.3 Member Data Documentation

6.20.3.1 int JGTL::Index2::x

6.20.3.2 int JGTL::Index2::y

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

- JGTL_Index2.h

6.21 JGTL::Index3 Class Reference

`#include <JGTL_Index3.h>`

Public Member Functions

- Index3 ()
- Index3 (int _x, int _y, int _z)
- Index3 (std::string s)
- bool operator< (const Index3 &index) const
- bool operator== (const Index3 &index) const
- bool operator!= (const Index3 &index) const
- Index3 operator* (const Index3 &index) const
- template<class TT>
  Index3 operator* (TT i) const
- Index3 operator/ (int i) const
- Index3 operator- (const Index3 &index) const
- void operator-= (const Index3 &index)
- Index3 operator+ (const Index3 &index) const
- void operator+= (const Index3 &index)
- double magnitude () const
- int magnitudeSquared () const
- int distanceSquared (const Index3 &other) const
- int distanceSquared (int _x, int _y, int _z) const
- int chessDistance (const Index3 &other) const
- int manhatDistance (const Index3 &other) const

Public Attributes

- int x
- int y
- int z

6.21.1 Constructor & Destructor Documentation

6.21.1.1 JGTL::Index3::Index3 () [inline]

6.21.1.2 JGTL::Index3::Index3 (int *_x*, int *_y*, int *_z*) [inline]

6.21.1.3 JGTL::Index3::Index3 (std::string *s*) [inline]

6.21.2 Member Function Documentation

6.21.2.1 bool JGTL::Index3::operator< (const Index3 & *index*) const [inline]

6.21.2.2 bool JGTL::Index3::operator== (const Index3 & *index*) const [inline]

6.21.2.3 bool JGTL::Index3::operator!= (const Index3 & *index*) const [inline]

6.21.2.4 Index3 JGTL::Index3::operator* (const Index3 & *index*) const [inline]

6.21.2.5 template<class TT> Index3 JGTL::Index3::operator* (TT *i*) const [inline]

6.21.2.6 Index3 JGTL::Index3::operator/ (int *i*) const [inline]

6.21.2.7 Index3 JGTL::Index3::operator- (const Index3 & *index*) const [inline]

6.21.2.8 void JGTL::Index3::operator-= (const Index3 & *index*) [inline]

6.21.2.9 Index3 JGTL::Index3::operator+ (const Index3 & *index*) const [inline]

6.21.2.10 void JGTL::Index3::operator+= (const Index3 & *index*) [inline]

6.21.2.11 double JGTL::Index3::magnitude () const [inline]

6.21.2.12 int JGTL::Index3::magnitudeSquared () const [inline]

6.21.2.13 int JGTL::Index3::distanceSquared (const Index3 & *other*) const [inline]

6.21.2.14 int JGTL::Index3::distanceSquared (int *_x*, int *_y*, int *_z*) const [inline]

6.21.2.15 int JGTL::Index3::chessDistance (const Index3 & *other*) const [inline]

6.21.2.16 int JGTL::Index3::manhatDistance (const Index3 & *other*) const [inline]

6.21.3 Member Data Documentation

- JGTL_Index3.h

6.22 JGTL::IntegralUnits< ValueType, SCALE, USEGCD > Class Template Reference

`#include <JGTL_IntegralUnits.h>`

Public Member Functions

- IntegralUnits (ValueType _value=0)
- template<class OtherValueType, OtherValueType OTHERSCALE, bool OTHERUSEGCD> IntegralUnits (const IntegralUnits< OtherValueType, OTHERSCALE, OTHERUSEGCD > &t)
- template<class OtherValueType, OtherValueType OTHERSCALE, bool OTHERUSEGCD> const IntegralUnits & operator= (const IntegralUnits< OtherValueType, OTHERSCALE, OTHERUSEGCD > &t)
- template<class OtherValueType, OtherValueType OTHERSCALE, bool OTHERUSEGCD> ValueType changeScale (const IntegralUnits< OtherValueType, OTHERSCALE, OTHERUSEGCD > &t)
- virtual units_internal_ulong getScale () const
- void setValue (ValueType _value)
- ValueType getValue () const

Protected Attributes

- ValueType value

template<class ValueType, ValueType SCALE, bool USEGCD> class JGTL::IntegralUnits< ValueType, SCALE, USEGCD >

6.22.1 Constructor & Destructor Documentation

6.22.1.1 template<class ValueType, ValueType SCALE, bool USEGCD> JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::IntegralUnits (ValueType *_value* = 0) [inline]

6.22.1.2 template<class ValueType, ValueType SCALE, bool USEGCD> template<class OtherValueType, OtherValueType OTHERSCALE, bool OTHERUSEGCD> JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::IntegralUnits (const IntegralUnits< OtherValueType, OTHERSCALE, OTHERUSEGCD > & *t*) [inline]

6.22.2 Member Function Documentation

6.22.2.1 template<class ValueType, ValueType SCALE, bool USEGCD> template<class OtherValueType, OtherValueType OTHERSCALE, bool OTHERUSEGCD> const IntegralUnits& JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::operator= (const IntegralUnits< OtherValueType, OTHERSCALE, OTHERUSEGCD > & *t*) [inline]

6.22.2.2 template<class ValueType, ValueType SCALE, bool USEGCD> template<class OtherValueType, OtherValueType OTHERSCALE, bool OTHERUSEGCD> ValueType JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::changeScale (const IntegralUnits< OtherValueType, OTHERSCALE, OTHERUSEGCD > & *t*) [inline]

6.22.2.3 template<class ValueType, ValueType SCALE, bool USEGCD> virtual units_internal_ulong JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::getScale () const [inline, virtual]

6.22.2.4 template<class ValueType, ValueType SCALE, bool USEGCD> void JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::setValue (ValueType *_value*) [inline]

6.22.2.5 template<class ValueType, ValueType SCALE, bool USEGCD> ValueType JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::getValue () const [inline]

6.22.3 Member Data Documentation

6.22.3.1 template<class ValueType, ValueType SCALE, bool USEGCD> ValueType JGTL::IntegralUnits< ValueType, SCALE, USEGCD >::value [protected]

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

- JGTL_IntegralUnits.h

6.23 JGTL::IntegralUnitsGCD< i, j > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const units_internal_ulong VALUE

template<units_internal_ulong i, units_internal_ulong j> struct JGTL::IntegralUnitsGCD< i, j >

6.23.1 Member Data Documentation

6.23.1.1 template<units_internal_ulong i, units_internal_ulong j> const units_internal_ulong JGTL::IntegralUnitsGCD< i, j >::VALUE [static]

Initial value:

```
TYPEIF<
                        units_internal_ulong,
                        STATIC_MOD<units_internal_ulong,i,j>::VALUE==0,
                        (j),
                        TYPEIF<
                        units_internal_ulong,
                        STATIC_MOD<units_internal_ulong,j,STATIC_MOD<units_internal_ulong,i,j>::VALUE>::VA
                        STATIC_MOD<units_internal_ulong,i,j>::VALUE,
                        IntegralUnitsGCD<
                        STATIC_MOD<units_internal_ulong,i,j>::VALUE,
                        STATIC_MOD<units_internal_ulong,j,STATIC_MOD<units_internal_ulong,i,j>::VALUE>::VA
                        >::VALUE
                        >::RESULT
                >::RESULT
```

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

- JGTL_IntegralUnits.h

6.24 JGTL::JGTL::IntegralUnitsGCD< 0, 0 > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const units_internal_ulong VALUE = 0

template<> struct JGTL::JGTL::IntegralUnitsGCD< 0, 0 >

6.24.1 Member Data Documentation

6.24.1.1 const units_internal_ulong JGTL::JGTL::IntegralUnitsGCD< 0, 0 >::VALUE = 0 [static]

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

- JGTL_IntegralUnits.h

6.25 JGTL::JGTL::IntegralUnitsGCD< 0, j > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const units_internal_ulong VALUE = 0

template<units_internal_ulong j> struct JGTL::JGTL::IntegralUnitsGCD< 0, j >

6.25.1 Member Data Documentation

6.25.1.1 template<units_internal_ulong j> const units_internal_ulong JGTL::JGTL::IntegralUnitsGCD< 0, j >::VALUE = 0 [static]

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

- JGTL_IntegralUnits.h

6.26 JGTL::JGTL::IntegralUnitsGCD< 1, 1 > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const units_internal_ulong VALUE = 1

template<> struct JGTL::JGTL::IntegralUnitsGCD< 1, 1 >

6.26.1 Member Data Documentation

6.26.1.1 const units_internal_ulong JGTL::JGTL::IntegralUnitsGCD< 1, 1 >::VALUE = 1 [static]

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

- JGTL_IntegralUnits.h

6.27 JGTL::JGTL::IntegralUnitsGCD< 1, j > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const units_internal_ulong VALUE = 1

template<units_internal_ulong j> struct JGTL::JGTL::IntegralUnitsGCD< 1, j >

6.27.1 Member Data Documentation

6.27.1.1 template<units_internal_ulong j> const units_internal_ulong JGTL::JGTL::IntegralUnitsGCD< 1, j >::VALUE = 1 [static]

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

- JGTL_IntegralUnits.h

6.28 JGTL::JGTL::IntegralUnitsGCD< i, 0 > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const units_internal_ulong VALUE = 0

template<units_internal_ulong i> struct JGTL::JGTL::IntegralUnitsGCD< i, 0 >

6.28.1 Member Data Documentation

6.28.1.1 template<units_internal_ulong i> const units_internal_ulong JGTL::JGTL::IntegralUnitsGCD< i, 0 >::VALUE = 0 [static]

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

- JGTL_IntegralUnits.h

6.29 JGTL::JGTL::IntegralUnitsGCD< i, 1 > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const units_internal_ulong VALUE = 1

template<units_internal_ulong i> struct JGTL::JGTL::IntegralUnitsGCD< i, 1 >

6.29.1 Member Data Documentation

6.29.1.1 template<units_internal_ulong i> const units_internal_ulong JGTL::JGTL::IntegralUnitsGCD< i, 1 >::VALUE = 1 [static]

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

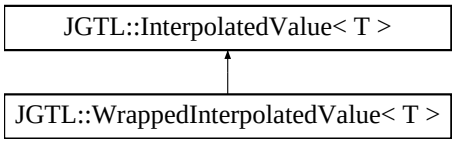
- JGTL_IntegralUnits.h

6.30 JGTL::InterpolatedValue< T > Class Template Reference

The InterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff;.

`#include <JGTL_InterpolatedValue.h>`

Inheritance diagram for JGTL::InterpolatedValue< T >::



Public Member Functions

- InterpolatedValue ()
- InterpolatedValue (float _interpolationCoeff)
- InterpolatedValue (const T &_value, float _interpolationCoeff)
- const InterpolatedValue< T > & operator= (T _potentialValue)
- float getInterpolationCoeff ()
- void setCoeff (float _coeff)
- virtual void setValue (const T &_value)
- void operator+= (const T &_value)
- void operator-= (const T &_value)
- T & getPotentialValue ()
- const T & getPotentialValue () const
- T & getActualValue ()
- const T & getActualValue () const
- void setActualValue (const T &t)
- void forceValue ()
- virtual void update (int times=1)

Protected Attributes

- T potentialValue
- T actualValue
- float interpolationCoeff

6.30.1 Detailed Description

template<class T> class JGTL::InterpolatedValue< T >

The InterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff;.

Author:

Jason Gauci 2008

6.30.2 Constructor & Destructor Documentation

6.30.2.1 template<class T> JGTL::InterpolatedValue< T >::InterpolatedValue ()

`[inline]`

6.30.2.2 template<class T> JGTL::InterpolatedValue< T >::InterpolatedValue (float *_interpolationCoeff*)

`[inline]`

6.30.2.3 template<class T> JGTL::InterpolatedValue< T >::InterpolatedValue (const T & *_value*, float *_interpolationCoeff*)

`[inline]`

6.30.3 Member Function Documentation

6.30.3.1 template<class T> const InterpolatedValue<T>& JGTL::InterpolatedValue< T >::operator= (T *_potentialValue*)

`[inline]`

Reimplemented in JGTL::WrappedInterpolatedValue< T >.

6.30.3.2 template<class T> float JGTL::InterpolatedValue< T >::getInterpolationCoeff ()

`[inline]`

6.30.3.3 template<class T> void JGTL::InterpolatedValue< T >::setCoeff (float *_coeff*)

`[inline]`

6.30.3.4 template<class T> virtual void JGTL::InterpolatedValue< T >::setValue (const T & *_value*)

`[inline, virtual]`

Reimplemented in JGTL::WrappedInterpolatedValue< T >.

6.30.3.5 template<class T> void JGTL::InterpolatedValue< T >::operator+= (const T & *_value*) [inline]

6.30.3.6 template<class T> void JGTL::InterpolatedValue< T >::operator-= (const T & *_value*) [inline]

6.30.3.7 template<class T> T& JGTL::InterpolatedValue< T >::getPotentialValue () [inline]

6.30.3.8 template<class T> const T& JGTL::InterpolatedValue< T >::getPotentialValue () const [inline]

6.30.3.9 template<class T> T& JGTL::InterpolatedValue< T >::getActualValue () [inline]

6.30.3.10 template<class T> const T& JGTL::InterpolatedValue< T >::getActualValue () const [inline]

6.30.3.11 template<class T> void JGTL::InterpolatedValue< T >::setActualValue (const T & *t*) [inline]

6.30.3.12 template<class T> void JGTL::InterpolatedValue< T >::forceValue () [inline]

6.30.3.13 template<class T> virtual void JGTL::InterpolatedValue< T >::update (int *times* = 1) [inline, virtual]

Reimplemented in JGTL::WrappedInterpolatedValue< T >.

6.30.4 Member Data Documentation

6.30.4.1 template<class T> T JGTL::InterpolatedValue< T >::potentialValue [protected]

6.30.4.2 template<class T> T JGTL::InterpolatedValue< T >::actualValue [protected]

This is the current value. This value is approaching the potential value

6.30.4.3 template<class T> float JGTL::InterpolatedValue< T >::interpolationCoeff [protected]

This handles the rate at which the actual value approaches the potential value

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

- JGTL_InterpolatedValue.h

6.31 JGTL::LocatedException Class Reference

This class handles throwing exceptions which include the file and line number.

`#include <JGTL_LocatedException.h>`

Public Member Functions

- LocatedException (const char *_reason, const char *_fileName, const int _lineNumber)
- LocatedException (const std::string &_reason, const char *_fileName, const int _lineNumber)
- virtual const char * what () const throw ()

Private Attributes

- char text [4096]

6.31.1 Detailed Description

This class handles throwing exceptions which include the file and line number.

Example:

throw CREATE_LOCATEDEXCEPTION_INFO("ERROR!");

Author:

Jason Gauci 2008

6.31.2 Constructor & Destructor Documentation

6.31.2.1 JGTL::LocatedException::LocatedException (const char * *_reason*, const char * *_fileName*, const int *_lineNumber*) [inline]

6.31.2.2 JGTL::LocatedException::LocatedException (const std::string & *_reason*, const char * *_fileName*, const int *_lineNumber*) [inline]

6.31.3 Member Function Documentation

6.31.3.1 virtual const char* JGTL::LocatedException::what () const throw () [inline, virtual]

6.31.4 Member Data Documentation

6.31.4.1 char JGTL::LocatedException::text[4096] [private]

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

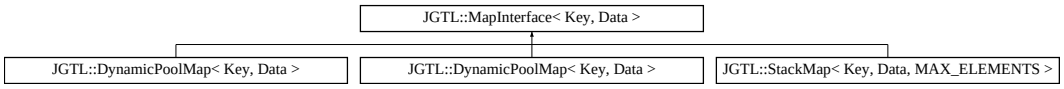
- JGTL_LocatedException.h

6.32 JGTL::MapInterface< Key, Data > Class Template Reference

This class acts as a base class for the Map construct.

`#include <JGTL_MapInterface.h>`

Inheritance diagram for JGTL::MapInterface< Key, Data >::



Public Types

- typedef std::pair< Key, Data > * iterator
- typedef const std::pair< Key, Data > * const_iterator
- typedef std::pair< Key, Data > TreeItem
- typedef BinaryTreeNode TreeNode

Public Member Functions

- MapInterface ()
- virtual ~MapInterface ()
- bool operator== (const MapInterface &other) const
- virtual bool resize (int newSize)=0
- virtual void insert (const Key &key, const Data &data)
- int size () const
- bool empty () const
- void clear ()
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- const bool hasKey (const Key &key) const
- iterator find (const Key &key)
- const_iterator find (const Key &key) const
- Data * getData (const Key &key)
- const Data * getData (const Key &key) const
- Data & getDataRef (const Key &key)
- const Data & getDataRef (const Key &key) const
- void erase (const_iterator iter)

- void eraseIndex (int index)
- const Data & getIndexData (int index) const
- Data * getIndexDataPtr (int index)
- const Data * getIndexDataPtr (int index) const
- const std::pair< Key, Data > & getIndex (int index) const
- iterator getIndexPtr (int index)
- const_iterator getIndexPtr (int index) const
- MapInterface ()
- virtual ~MapInterface ()
- bool operator== (const MapInterface &other) const
- virtual bool reserve (int newSize)
- virtual void insert (const Key &key, const Data &data)
- int size () const
- void clear ()
- const bool hasKey (const Key &key) const
- iterator find (const Key &key)
- const_iterator find (const Key &key) const
- Data * getData (const Key &key)
- const Data * getData (const Key &key) const
- const Data & getDataRef (const Key &key) const
- void erase (const_iterator iter)
- const Data & getIndexData (int index) const
- Data * getIndexDataPtr (int index)
- const Data * getIndexDataPtr (int index) const

Protected Attributes

- int numElements
- int maxElements
- std::pair< Key, Data > * dataList
- TreeItem * dataList
- TreeNode * nodeList
- int rootIndex

6.32.1 Detailed Description

template<class Key, class Data> class JGTL::MapInterface< Key, Data >

This class acts as a base class for the Map construct.

Author:

Jason Gauci 2008

6.32.2 Member Typedef Documentation

6.32.2.1 template<class Key, class Data> typedef std::pair<Key,Data>* JGTL::MapInterface< Key, Data >::iterator

6.32.2.2 template<class Key, class Data> typedef const std::pair<Key,Data>* JGTL::MapInterface< Key, Data >::const_iterator

6.32.2.3 template<class Key, class Data> typedef std::pair<Key,Data> JGTL::MapInterface< Key, Data >::TreeItem

6.32.2.4 template<class Key, class Data> typedef BinaryTreeNode JGTL::MapInterface< Key, Data >::TreeNode

6.32.3 Constructor & Destructor Documentation

6.32.3.1 template<class Key, class Data> JGTL::MapInterface< Key, Data >::MapInterface () [inline]

6.32.3.2 template<class Key, class Data> virtual JGTL::MapInterface< Key, Data >::~MapInterface () [inline, virtual]

6.32.3.3 template<class Key, class Data> JGTL::MapInterface< Key, Data >::MapInterface () [inline]

6.32.3.4 template<class Key, class Data> virtual JGTL::MapInterface< Key, Data >::~MapInterface () [inline, virtual]

6.32.4 Member Function Documentation

6.32.4.1 template<class Key, class Data> bool JGTL::MapInterface< Key, Data >::operator== (const MapInterface< Key, Data > & *other*) const [inline]

6.32.4.2 template<class Key, class Data> virtual bool JGTL::MapInterface< Key, Data >::resize (int *newSize*) [pure virtual]

Implemented in JGTL::DynamicPoolMap< Key, Data >, JGTL::StackMap< Key, Data, MAX_ELEMENTS >, and JGTL::DynamicPoolMap< std::string, JGTL::ProfileBlock * >.

6.32.4.3 template<class Key, class Data> virtual void JGTL::MapInterface< Key, Data >::insert (const Key & *key*, const Data & *data*) `[inline, virtual]`

6.32.4.4 template<class Key, class Data> int JGTL::MapInterface< Key, Data >::size () const `[inline]`

6.32.4.5 template<class Key, class Data> bool JGTL::MapInterface< Key, Data >::empty () const `[inline]`

6.32.4.6 template<class Key, class Data> void JGTL::MapInterface< Key, Data >::clear () `[inline]`

6.32.4.7 template<class Key, class Data> iterator JGTL::MapInterface< Key, Data >::begin () `[inline]`

6.32.4.8 template<class Key, class Data> const_iterator JGTL::MapInterface< Key, Data >::begin () const `[inline]`

6.32.4.9 template<class Key, class Data> iterator JGTL::MapInterface< Key, Data >::end () `[inline]`

6.32.4.10 template<class Key, class Data> const_iterator JGTL::MapInterface< Key, Data >::end () const `[inline]`

6.32.4.11 template<class Key, class Data> const bool JGTL::MapInterface< Key, Data >::hasKey (const Key & *key*) const `[inline]`

6.32.4.12 template<class Key, class Data> iterator JGTL::MapInterface< Key, Data >::find (const Key & *key*) `[inline]`

6.32.4.13 template<class Key, class Data> const_iterator JGTL::MapInterface< Key, Data >::find (const Key & *key*) const `[inline]`

6.32.4.14 template<class Key, class Data> Data* JGTL::MapInterface< Key, Data >::getData (const Key & *key*) `[inline]`

6.32.4.15 template<class Key, class Data> const Data* JGTL::MapInterface< Key, Data >::getData (const Key & *key*) const `[inline]`

6.32.4.16 template<class Key, class Data> Data& JGTL::MapInterface< Key, Data >::getDataRef (const Key & *key*) `[inline]`

6.32.4.17 template<class Key, class Data> const Data& JGTL::MapInterface< Key, Data >::getDataRef (const Key & *key*) const `[inline]`

6.32.4.18 template<class Key, class Data> void JGTL::MapInterface< Key, Data >::erase (const_iterator *iter*) `[inline]`

6.32.4.19 template<class Key, class Data> void JGTL::MapInterface< Key, Data >::eraseIndex (int *index*) `[inline]`

6.32.4.20 template<class Key, class Data> const Data& JGTL::MapInterface< Key, Data >::getIndexData (int *index*) const `[inline]`

6.32.4.28 template<class Key, class Data> virtual void JGTL::MapInterface< Key, Data >::insert (const Key & *key*, const Data & *data*) [inline, virtual]

6.32.4.29 template<class Key, class Data> int JGTL::MapInterface< Key, Data >::size () const [inline]

6.32.4.30 template<class Key, class Data> void JGTL::MapInterface< Key, Data >::clear () [inline]

6.32.4.31 template<class Key, class Data> const bool JGTL::MapInterface< Key, Data >::hasKey (const Key & *key*) const [inline]

6.32.4.32 template<class Key, class Data> iterator JGTL::MapInterface< Key, Data >::find (const Key & *key*) [inline]

6.32.4.33 template<class Key, class Data> const_iterator JGTL::MapInterface< Key, Data >::find (const Key & *key*) const [inline]

6.32.4.34 template<class Key, class Data> Data* JGTL::MapInterface< Key, Data >::getData (const Key & *key*) [inline]

6.32.4.35 template<class Key, class Data> const Data* JGTL::MapInterface< Key, Data >::getData (const Key & *key*) const [inline]

6.32.4.36 template<class Key, class Data> const Data& JGTL::MapInterface< Key, Data >::getDataRef (const Key & *key*) const [inline]

6.32.4.37 template<class Key, class Data> void JGTL::MapInterface< Key, Data >::erase (const_iterator *iter*) [inline]

6.32.4.38 template<class Key, class Data> const Data& JGTL::MapInterface< Key, Data >::getIndexData (int *index*) const [inline]

6.32.4.39 template<class Key, class Data> Data* JGTL::MapInterface< Key, Data >::getIndexDataPtr (int *index*) [inline]

6.32.4.40 template<class Key, class Data> const Data* JGTL::MapInterface< Key, Data >::getIndexDataPtr (int *index*) const [inline]

6.32.5 Member Data Documentation

6.32.5.1 template<class Key, class Data> int JGTL::MapInterface< Key, Data >::numElements [protected]

6.32.5.2 template<class Key, class Data> int JGTL::MapInterface< Key, Data >::maxElements [protected]

6.32.5.3 template<class Key, class Data> std::pair<Key,Data>* JGTL::MapInterface< Key, Data >::dataList [protected]

6.32.5.4 template<class Key, class Data> TreeItem* JGTL::MapInterface< Key, Data >::dataList [protected]

- JGTL_MapInterface.h
- JGTL_UnorderedMapInterface.h

6.33 JGTL::NullVariantClass Class Reference

`#include <JGTL_Variant.h>`

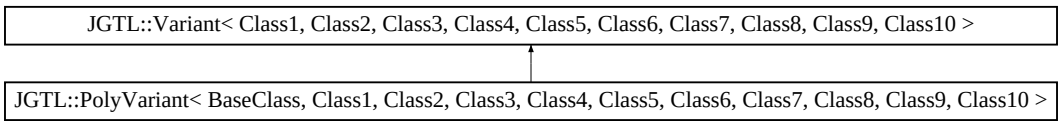
The documentation for this class was generated from the following file:

- JGTL_Variant.h

6.34 JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 > Class Template Reference

`#include <JGTL_PolyVariant.h>`

Inheritance diagram for JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::



Public Member Functions

- PolyVariant ()
- PolyVariant (const PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 > &other)
- BaseClass * operator → () const
- template<class ValueToSet>
  void setValue (const ValueToSet &newValue)

Protected Attributes

- BaseClass * base

template<class BaseClass, class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> class JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >

6.34.1 Constructor & Destructor Documentation

6.34.1.1 template<class BaseClass, class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::PolyVariant () `[inline]`

6.34.1.2 template<class BaseClass, class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::PolyVariant (const PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 > & *other*) `[inline]`

6.34.2 Member Function Documentation

6.34.2.1 template<class BaseClass, class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> BaseClass* JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::operator → () const `[inline]`

6.34.2.2 template<class BaseClass, class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> template<class ValueToSet> void JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::setValue (const ValueToSet & *newValue*) `[inline]`

Reimplemented from JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >.

6.34.3 Member Data Documentation

6.34.3.1 template<class BaseClass, class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> BaseClass* JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::base [protected]

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

- JGTL_PolyVariant.h

6.35 JGTL::PoolMap< Key, Data > Class Template Reference

`#include <JGTL_PoolMap_delete.h>`

Public Types

- typedef std::pair< Key, Data > * iterator
- typedef const std::pair< Key, Data > * const_iterator

Public Member Functions

- PoolMap (int _maxElements)
- PoolMap (const PoolMap< Key, Data > &other)
- const PoolMap & operator= (const PoolMap< Key, Data > &other)
- void copyFrom (const PoolMap< Key, Data > &other)
- virtual ~PoolMap ()
- void insert (const Key &key, const Data &data)
- int size ()
- void clear ()
- iterator begin ()
- iterator end ()
- const bool hasKey (const Key &key) const
- iterator find (const Key &key)
- Data * getData (const Key &key)
- const Data * getData (const Key &key) const
- const Data & getDataRef (const Key &key) const
- const Data & getIndexData (int index) const
- const Data * getIndexDataPtr (int index) const

Protected Attributes

- int numElements
- int maxElements
- std::pair< Key, Data > * dataList

template<class Key, class Data> class JGTL::PoolMap< Key, Data >

6.35.1 Member Typedef Documentation

6.35.1.1 template<class Key, class Data> typedef std::pair<Key,Data>* JGTL::PoolMap< Key, Data >::iterator

6.35.1.2 template<class Key, class Data> typedef const std::pair<Key,Data>* JGTL::PoolMap< Key, Data >::const_iterator

6.35.2 Constructor & Destructor Documentation

6.35.2.1 template<class Key, class Data> JGTL::PoolMap< Key, Data >::PoolMap (int *_maxElements*) [inline]

6.35.2.2 template<class Key, class Data> JGTL::PoolMap< Key, Data >::PoolMap (const PoolMap< Key, Data > & *other*) [inline]

6.35.2.3 template<class Key, class Data> virtual JGTL::PoolMap< Key, Data >::~PoolMap () [inline, virtual]

6.35.3 Member Function Documentation

6.35.3.1 template<class Key, class Data> const PoolMap& JGTL::PoolMap< Key, Data >::operator= (const PoolMap< Key, Data > & *other*) [inline]

6.35.3.2 template<class Key, class Data> void JGTL::PoolMap< Key, Data >::copyFrom (const PoolMap< Key, Data > & *other*) [inline]

6.35.3.3 template<class Key, class Data> void JGTL::PoolMap< Key, Data >::insert (const Key & *key*, const Data & *data*) [inline]

6.35.3.4 template<class Key, class Data> int JGTL::PoolMap< Key, Data >::size () [inline]

6.35.3.5 template<class Key, class Data> void JGTL::PoolMap< Key, Data >::clear () [inline]

6.35.3.6 template<class Key, class Data> iterator JGTL::PoolMap< Key, Data >::begin () [inline]

6.35.3.7 template<class Key, class Data> iterator JGTL::PoolMap< Key, Data >::end () [inline]

6.35.3.8 template<class Key, class Data> const bool JGTL::PoolMap< Key, Data >::hasKey (const Key & *key*) const [inline]

6.35.3.9 template<class Key, class Data> iterator JGTL::PoolMap< Key, Data >::find (const Key & *key*) [inline]

6.35.3.10 template<class Key, class Data> Data* JGTL::PoolMap< Key, Data >::getData (const Key & *key*) [inline]

6.35.3.11 template<class Key, class Data> const Data* JGTL::PoolMap< Key

- JGTL_PoolMap_delete.h

6.36 JGTL::ProfileBlock Struct Reference

`#include <JGTL_QuickProf.h>`

Public Member Functions

- ProfileBlock ()

Public Attributes

- unsigned long int currentBlockStartMicroseconds

  The starting time (in us) of the current block update.

- unsigned long int currentCycleTotalMicroseconds
- double avgCycleTotalMicroseconds
- unsigned long int totalMicroseconds

  The total accumulated time (in us) spent in this block.

- unsigned long int smallestCycleMicroseconds

  The best time for this block.

- double smallestCyclePercent
- unsigned long int largestCycleMicroseconds

  The worst time for this block.

- double largestCyclePercent

6.36.1 Detailed Description

A simple data structure representing a single timed block of code.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 JGTL::ProfileBlock::ProfileBlock () `[inline]`

6.36.3 Member Data Documentation

6.36.3.1 unsigned long int JGTL::ProfileBlock::currentBlockStartMicroseconds

The starting time (in us) of the current block update.

6.36.3.2 unsigned long int JGTL::ProfileBlock::currentCycleTotalMicroseconds

The accumulated time (in us) spent in this block during the current profiling cycle.

6.36.3.3 double JGTL::ProfileBlock::avgCycleTotalMicroseconds

The accumulated time (in us) spent in this block during the past profiling cycle.

6.36.3.4 unsigned long int JGTL::ProfileBlock::totalMicroseconds

The total accumulated time (in us) spent in this block.

6.36.3.5 unsigned long int JGTL::ProfileBlock::smallestCycleMicroseconds

The best time for this block.

6.36.3.6 double JGTL::ProfileBlock::smallestCyclePercent

6.36.3.7 unsigned long int JGTL::ProfileBlock::largestCycleMicroseconds

The worst time for this block.

6.36.3.8 double JGTL::ProfileBlock::largestCyclePercent

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

- JGTL_QuickProf.h

6.37 JGTL::ProfileBlockHandler Class Reference

`#include <JGTL_QuickProf.h>`

Public Member Functions

- ProfileBlockHandler (const char *_blockName)
- ~ProfileBlockHandler ()

Protected Attributes

- const char * blockName

6.37.1 Constructor & Destructor Documentation

6.37.1.1 JGTL::ProfileBlockHandler::ProfileBlockHandler (const char * *_blockName*) [inline]

6.37.1.2 JGTL::ProfileBlockHandler::~ProfileBlockHandler () [inline]

6.37.2 Member Data Documentation

6.37.2.1 const char* JGTL::ProfileBlockHandler::blockName [protected]

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

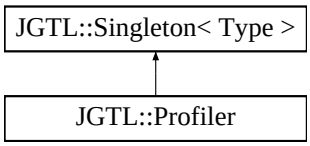
- JGTL_QuickProf.h

6.38 JGTL::Profiler Class Reference

A singleton class that manages timing for a set of profiling blocks.

`#include <JGTL_QuickProf.h>`

Inheritance diagram for JGTL::Profiler::



Public Member Functions

- void init (double smoothing=0.0, const std::string outputFilename="", size_t printPeriod=1, TimeFormat printFormat=MILLISECONDS)
- void reset ()
- void beginBlock (const std::string &name)
- void endBlock (const std::string &name)
- void beginCycle ()
- void endCycle ()
- double getAvgDuration (const std::string &name, TimeFormat format)
- std::string getSummary (TimeFormat format=PERCENT)

Static Public Member Functions

- static Profiler * createInstance ()
- static void destroyInstance ()

Protected Member Functions

- Profiler ()
- virtual ~Profiler ()
- void printError (const std::string &msg)
- ProfileBlock * getProfileBlock (const std::string &name)
- double getBlockMinTime (const std::string &name, TimeFormat format)
- double getBlockTotalTime (const std::string &name, TimeFormat format)
- double getBlockMaxTime (const std::string &name, TimeFormat format)
- double getMicrosecondsSinceInit ()
- std::string getSuffixString (TimeFormat format)

Protected Attributes

- bool mEnabled

  Determines whether the profiler is enabled.

- Clock mClock

  The clock used to time profile blocks.

- unsigned long int mCurrentCycleStartMicroseconds

  The starting time (in us) of the current profiling cycle.

- unsigned long int mLastCycleDurationMicroseconds

  The duration (in us) of the most recent profiling cycle.

- DynamicPoolMap< std::string, ProfileBlock * > mProfileBlocks

  Internal map of named profile blocks.

- std::ofstream mOutputFile

  The data output file used if this feature is enabled in init.

- bool mFirstFileOutput

  Tracks whether we have begun printing data to the output file.

- double mMovingAvgScalar
- size_t mPrintPeriod
- TimeFormat mPrintFormat

  The time format used when printing timing data to the output file.

- size_t mCycleCounter

  Keeps track of how many cycles have elapsed (for printing).

- bool mFirstCycle

  Used to update the initial average cycle times.

- unsigned long microsecondsSinceInit

6.38.1 Detailed Description

A singleton class that manages timing for a set of profiling blocks.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 JGTL::Profiler::Profiler () [inline, protected]

6.38.2.2 JGTL::Profiler::~Profiler () [inline, protected, virtual]

6.38.3 Member Function Documentation

6.38.3.1 static Profiler* JGTL::Profiler::createInstance () [inline, static]

6.38.3.2 static void JGTL::Profiler::destroyInstance () [inline, static]

Reimplemented from JGTL::Singleton< Type >.

6.38.3.3 void JGTL::Profiler::init (double *smoothing* = 0.0, const std::string *outputFilename* = "", size_t *printPeriod* = 1, TimeFormat *printFormat* = MILLISECONDS) [inline]

Initializes the profiler.

This must be called first. If this is never called, the profiler is effectively disabled, and all other functions will return immediately.

Parameters:

smoothing The measured duration for each profile block can be averaged across multiple cycles, and this parameter defines the smoothness of this averaging process. The higher the value, the smoother the resulting average durations will appear. Leaving it at zero will essentially disable the smoothing effect. More specifically, this parameter is a time constant (defined in terms of cycles) that defines an exponentially-weighted moving average. For example, a value of 4.0 means the past four cycles will contribute 63% of the current weighted average. This value must be >= 0.

outputFilename If defined, enables timing data to be printed to a data file for later analysis.

printPeriod Defines how often data is printed to the file, in number of profiling cycles. For example, set this to 1 if you want data printed after each cycle, or 5 if you want it printed every 5 cycles. It is a good idea to increase this if you don't want huge data files. Keep in mind, however, that when you increase this, you might want to increase the smoothing parameter. (A good heuristic is to set the smoothing parameter equal to the print period.) This value must be >= 1.

printFormat Defines the format used when printing data to a file.

6.38.3.4 void JGTL::Profiler::reset () [inline]

6.38.3.5 void JGTL::Profiler::beginBlock (const std::string & *name*) [inline]

Begins timing the named block of code.

Parameters:

name The name of the block.

6.38.3.6 void JGTL::Profiler::endBlock (const std::string & *name*) [inline]

Defines the end of the named timing block.

Parameters:

name The name of the block.

6.38.3.7 void JGTL::Profiler::beginCycle () [inline]

6.38.3.8 void JGTL::Profiler::endCycle () [inline]

Defines the end of a profiling cycle.

Use this regularly by calling it at the end of all timing blocks. This is necessary for smoothing and for file output, but not if you just want a total summary at the end of execution (i.e. from getSummary). This must not be called within a timing block.

6.38.3.9 double JGTL::Profiler::getAvgDuration (const std::string & *name*, TimeFormat *format*) [inline]

Returns the average time used in the named block per profiling cycle.

If smoothing is disabled (see init), this returns the most recent duration measurement.

Parameters:

name The name of the block.

format The desired time format to use for the result.

Returns:

The block's average duration per cycle.

6.38.3.10 std::string JGTL::Profiler::getSummary (TimeFormat *format* = PERCENT) [inline]

Returns a summary of total times in each block.

Parameters:

format The desired time format to use for the results.

Returns:

The timing summary as a string.

6.38.3.11 void JGTL::Profiler::printError (const std::string & *msg*) [inline, protected]

Prints an error message to standard output.

Parameters:

msg The string to print.

6.38.3.12 ProfileBlock * JGTL::Profiler::getProfileBlock (const std::string & *name*) [inline, protected]

Returns a named profile block.

Parameters:

name The name of the block to return.

Returns:

The named ProfileBlock, or NULL if it can't be found.

6.38.3.13 double JGTL::Profiler::getBlockMinTime (const std::string & *name*, TimeFormat *format*) [inline, protected]

Returns the time spent in the named block since the profiler was initialized.

Parameters:

name The name of the block.

format The desired time format to use for the result.

Returns:

The block total time.

6.38.3.14 double JGTL::Profiler::getBlockTotalTime (const std::string & *name*, TimeFormat *format*) [inline, protected]

Returns the time spent in the named block since the profiler was initialized.

Parameters:

name The name of the block.

format The desired time format to use for the result.

Returns:

The block total time.

6.38.3.15 double JGTL::Profiler::getBlockMaxTime (const std::string & *name*, TimeFormat *format*) [inline, protected]

Returns the time spent in the named block since the profiler was initialized.

Parameters:

name The name of the block.

format The desired time format to use for the result.

Returns:

The block total time.

6.38.3.16 double JGTL::Profiler::getMicrosecondsSinceInit () [inline, protected]

Computes the elapsed time since the profiler was initialized.

Returns:

The elapsed time in microseconds.

6.38.3.17 std::string JGTL::Profiler::getSuffixString (TimeFormat *format*) [inline, protected]

Returns the appropriate suffix string for the given time format.

Returns:

The suffix string.

6.38.4 Member Data Documentation

6.38.4.1 bool JGTL::Profiler::mEnabled [protected]

Determines whether the profiler is enabled.

6.38.4.2 Clock JGTL::Profiler::mClock [protected]

The clock used to time profile blocks.

6.38.4.3 unsigned long int JGTL::Profiler::mCurrentCycleStartMicroseconds [protected]

The starting time (in us) of the current profiling cycle.

6.38.4.4 unsigned long int JGTL::Profiler::mLastCycleDurationMicroseconds [protected]

The duration (in us) of the most recent profiling cycle.

6.38.4.5 DynamicPoolMap<std::string, ProfileBlock*> JGTL::Profiler::mProfileBlocks [protected]

Internal map of named profile blocks.

6.38.4.6 std::ofstream JGTL::Profiler::mOutputFile [protected]

The data output file used if this feature is enabled in init.

6.38.4.7 bool JGTL::Profiler::mFirstFileOutput [protected]

Tracks whether we have begun printing data to the output file.

6.38.4.8 double JGTL::Profiler::mMovingAvgScalar [protected]

A pre-computed scalar used to update exponentially-weighted moving averages.

6.38.4.9 size_t JGTL::Profiler::mPrintPeriod [protected]

Determines how often (in number of profiling cycles) timing data is printed to the output file.

6.38.4.10 TimeFormat JGTL::Profiler::mPrintFormat [protected]

The time format used when printing timing data to the output file.

6.38.4.11 size_t JGTL::Profiler::mCycleCounter [protected]

Keeps track of how many cycles have elapsed (for printing).

6.38.4.12 bool JGTL::Profiler::mFirstCycle [protected]

Used to update the initial average cycle times.

6.38.4.13 unsigned long JGTL::Profiler::microsecondsSinceInit [protected]

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

- JGTL_QuickProf.h

6.39 JGTL::QuadraticSolution< T > Class Template Reference

`#include <JGTL_Quadratic.h>`

Public Member Functions

- QuadraticSolution ()

Public Attributes

- int numSolutions
- T t1
- T t2

template<class T> class JGTL::QuadraticSolution< T >

6.39.1 Constructor & Destructor Documentation

6.39.1.1 template<class T> JGTL::QuadraticSolution< T >::QuadraticSolution () [inline]

6.39.2 Member Data Documentation

6.39.2.1 template<class T> int JGTL::QuadraticSolution< T >::numSolutions

6.39.2.2 template<class T> T JGTL::QuadraticSolution< T >::t1

6.39.2.3 template<class T> T JGTL::QuadraticSolution< T >::t2

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

- JGTL_Quadratic.h

6.40 JGTL::QuadTree< T > Class Template Reference

`#include <JGTL_QuadTree.h>`

Public Member Functions

- QuadTree (int _size=16, T defaultValue=(T) 0)
- QuadTree (const QuadTree< T > &other)
- ~QuadTree ()
- QuadTree< T > & operator= (const QuadTree< T > &other)
- void copyFrom (const QuadTree< T > &other)
- T getValue (const Vector2< int > &location) const
- T getValue (int x, int y) const
- void setValue (const Vector2< int > &location, T value)
- void setValue (int x, int y, T value)
- void setAll (T value)
- void display () const
- T operator() (const Vector2< int > &location) const
- T operator() (int x, int y) const

Private Attributes

- int size
- QuadTreeBranch< T > * root
- boost::pool branchPool
- boost::pool stubPool

template<class T> class JGTL::QuadTree< T >

6.40.1 Constructor & Destructor Documentation

6.40.1.1 template<class T> JGTL::QuadTree< T >::QuadTree (int *_size* = 16, T *defaultValue* = (T) 0) [inline]

6.40.1.2 template<class T> JGTL::QuadTree< T >::QuadTree (const QuadTree< T > & *other*) [inline]

6.40.1.3 template<class T> JGTL::QuadTree< T >::~QuadTree () [inline]

6.40.2 Member Function Documentation

6.40.2.1 template<class T> QuadTree<T>& JGTL::QuadTree< T >::operator= (const QuadTree< T > & *other*) [inline]

6.40.2.2 template<class T> void JGTL::QuadTree< T >::copyFrom (const QuadTree< T > & *other*) [inline]

6.40.2.3 template<class T> T JGTL::QuadTree< T >::getValue (const Vector2< int > & *location*) const [inline]

6.40.2.4 template<class T> T JGTL::QuadTree< T >::getValue (int *x*, int *y*) const [inline]

6.40.2.5 template<class T> void JGTL::QuadTree< T >::setValue (const Vector2< int > & *location*, T *value*) [inline]

6.40.2.6 template<class T> void JGTL::QuadTree< T >::setValue (int *x*, int *y*, T *value*) [inline]

6.40.2.7 template<class T> void JGTL::QuadTree< T >::setAll (T *value*) [inline]

6.40.2.8 template<class T> void JGTL::QuadTree< T >::display () const [inline]

6.40.2.9 template<class T> T JGTL::QuadTree< T >::operator() (const Vector2< int > & *location*) const [inline]

6.40.2.10 template<class T> T JGTL::QuadTree< T >::operator() (int *x*, int *y*) const [inline]

6.40.3 Member Data Documentation

6.40.3.1 template<class T> int JGTL::QuadTree< T >::size [private]

6.40.3.2 template<class T> QuadTreeBranch<T>* JGTL::QuadTree< T >::root [private]

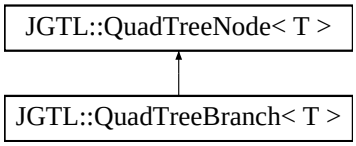
6.40.3.3 template<class T> boost::pool JGTL::QuadTree< T >::branchPool [private]

- JGTL_QuadTree.h

6.41 JGTL::QuadTreeBranch< T > Class Template Reference

`#include <JGTL_QuadTree.h>`

Inheritance diagram for JGTL::QuadTreeBranch< T >::



Public Member Functions

- QuadTreeBranch (boost::pool<> &branchPool, boost::pool<> &stubPool, const T &value)
- virtual ~QuadTreeBranch ()
- virtual void destroy (boost::pool<> &branchPool, boost::pool<> &stubPool)
- virtual T getValue (Vector2< int > topLeftVector2, int size, const Vector2< int > &location) const
- virtual T getValue () const
- void setAll (boost::pool<> &branchPool, boost::pool<> &stubPool, T &value)
- virtual bool setValue (boost::pool<> &branchPool, boost::pool<> &stubPool, Vector2< int > topLeftVector2, int size, const Vector2< int > &location, const T &value)
- virtual void display (int level) const

Private Attributes

- QuadTreeNode< T > * topLeft
- QuadTreeNode< T > * topRight
- QuadTreeNode< T > * bottomLeft
- QuadTreeNode< T > * bottomRight

template<class T> class JGTL::QuadTreeBranch< T >

6.41.1 Constructor & Destructor Documentation

6.41.1.1 template<class T> JGTL::QuadTreeBranch< T >::QuadTreeBranch (boost::pool<> & *branchPool*, boost::pool<> & *stubPool*, const T & *value*) [inline]

6.41.1.2 template<class T> virtual JGTL::QuadTreeBranch< T >::~QuadTreeBranch () [inline, virtual]

6.41.2 Member Function Documentation

6.41.2.1 template<class T> virtual void JGTL::QuadTreeBranch< T >::destroy (boost::pool<> & *branchPool*, boost::pool<> & *stubPool*) [inline, virtual]

Reimplemented from JGTL::QuadTreeNode< T >.

6.41.2.2 template<class T> virtual T JGTL::QuadTreeBranch< T >::getValue (Vector2< int > *topLeftVector2*, int *size*, const Vector2< int > & *location*) const [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.41.2.3 template<class T> virtual T JGTL::QuadTreeBranch< T >::getValue () const [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.41.2.4 template<class T> void JGTL::QuadTreeBranch< T >::setAll (boost::pool<> & *branchPool*, boost::pool<> & *stubPool*, T & *value*) [inline]

6.41.2.5 template<class T> virtual bool JGTL::QuadTreeBranch< T >::setValue (boost::pool<> & *branchPool*, boost::pool<> & *stubPool*, Vector2< int > *topLeftVector2*, int *size*, const Vector2< int > & *location*, const T & *value*) [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.41.2.6 template<class T> virtual void JGTL::QuadTreeBranch< T >::display (int *level*) const [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.41.3 Member Data Documentation

6.41.3.1 template<class T> QuadTreeNode<T>* JGTL::QuadTreeBranch< T >::topLeft [private]

6.41.3.2 template<class T> QuadTreeNode<T> * JGTL::QuadTreeBranch< T >::topRight [private]

6.41.3.3 template<class T> QuadTreeNode<T> * JGTL::QuadTreeBranch< T >::bottomLeft [private]

6.41.3.4 template<class T> QuadTreeNode<T> * JGTL::QuadTreeBranch< T >::bottomRight [private]

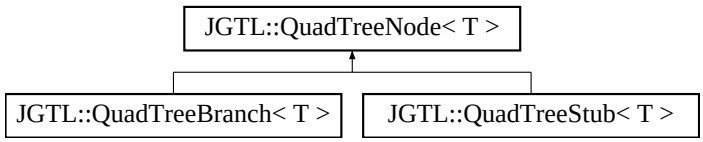
The documentation for this class was generated from the following file:

- JGTL_QuadTree.h

6.42 JGTL::QuadTreeNode< T > Class Template Reference

`#include <JGTL_QuadTree.h>`

Inheritance diagram for JGTL::QuadTreeNode< T >::



Public Member Functions

- QuadTreeNode ()
- virtual ~QuadTreeNode ()
- virtual T getValue (Vector2< int > topLeftVector2, int size, const Vector2< int > &location) const =0
- virtual T getValue () const =0
- virtual bool setValue (boost::pool<> &branchPool, boost::pool<> &stubPool, Vector2< int > topLeftVector2, int size, const Vector2< int > &location, const T &value)=0
- virtual bool isStub () const
- virtual void destroy (boost::pool<> &branchPool, boost::pool<> &stubPool)
- virtual void display (int level) const =0

template<class T> class JGTL::QuadTreeNode< T >

6.42.1 Constructor & Destructor Documentation

6.42.1.1 template<class T> JGTL::QuadTreeNode< T >::QuadTreeNode () [inline]

6.42.1.2 template<class T> virtual JGTL::QuadTreeNode< T >::~QuadTreeNode () [inline, virtual]

6.42.2 Member Function Documentation

6.42.2.1 template<class T> virtual T JGTL::QuadTreeNode< T >::getValue (Vector2< int > *topLeftVector2*, int *size*, const Vector2< int > & *location*) const [pure virtual]

Implemented in JGTL::QuadTreeStub< T >, and JGTL::QuadTreeBranch< T >.

6.42.2.2 template<class T> virtual T JGTL::QuadTreeNode< T >::getValue () const [pure virtual]

Implemented in JGTL::QuadTreeStub< T >, and JGTL::QuadTreeBranch< T >.

6.42.2.3 template<class T> virtual bool JGTL::QuadTreeNode< T >::setValue (boost::pool<> & *branchPool*, boost::pool<> & *stubPool*, Vector2< int > *topLeftVector2*, int *size*, const Vector2< int > & *location*, const T & *value*) [pure virtual]

Implemented in JGTL::QuadTreeStub< T >, and JGTL::QuadTreeBranch< T >.

6.42.2.4 template<class T> virtual bool JGTL::QuadTreeNode< T >::isStub () const [inline, virtual]

Reimplemented in JGTL::QuadTreeStub< T >.

6.42.2.5 template<class T> virtual void JGTL::QuadTreeNode< T >::destroy (boost::pool<> & *branchPool*, boost::pool<> & *stubPool*) [inline, virtual]

Reimplemented in JGTL::QuadTreeBranch< T >.

6.42.2.6 template<class T> virtual void JGTL::QuadTreeNode< T >::display (int *level*) const [pure virtual]

Implemented in JGTL::QuadTreeStub< T >, and JGTL::QuadTreeBranch< T >.

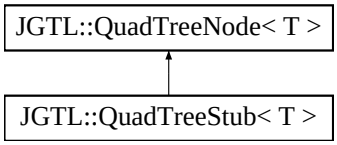
The documentation for this class was generated from the following file:

- JGTL_QuadTree.h

6.43 JGTL::QuadTreeStub< T > Class Template Reference

`#include <JGTL_QuadTree.h>`

Inheritance diagram for JGTL::QuadTreeStub< T >::



Public Member Functions

- QuadTreeStub (T _value)
- virtual ~QuadTreeStub ()
- virtual T getValue (Vector2< int > topLeftVector2, int size, const Vector2< int > &location) const
- virtual T getValue () const
- virtual bool setValue (boost::pool<> &branchPool, boost::pool<> &stubPool, Vector2< int > topLeftVector2, int size, const Vector2< int > &location, const T &value)
- virtual bool setValue (const T &_value)
- virtual bool isStub () const
- virtual void display (int level) const

Protected Attributes

- T value

template<class T> class JGTL::QuadTreeStub< T >

6.43.1 Constructor & Destructor Documentation

6.43.1.1 template<class T> JGTL::QuadTreeStub< T >::QuadTreeStub (T *_value*) [inline]

6.43.1.2 template<class T> virtual JGTL::QuadTreeStub< T >::~QuadTreeStub () [inline, virtual]

6.43.2 Member Function Documentation

6.43.2.1 template<class T> virtual T JGTL::QuadTreeStub< T >::getValue (Vector2< int > *topLeftVector2*, int *size*, const Vector2< int > & *location*) const [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.43.2.2 template<class T> virtual T JGTL::QuadTreeStub< T >::getValue () const [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.43.2.3 template<class T> virtual bool JGTL::QuadTreeStub< T >::setValue (boost::pool<> & *branchPool*, boost::pool<> & *stubPool*, Vector2< int > *topLeftVector2*, int *size*, const Vector2< int > & *location*, const T & *value*) [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.43.2.4 template<class T> virtual bool JGTL::QuadTreeStub< T >::setValue (const T & *_value*) [inline, virtual]

6.43.2.5 template<class T> virtual bool JGTL::QuadTreeStub< T >::isStub () const [inline, virtual]

Reimplemented from JGTL::QuadTreeNode< T >.

6.43.2.6 template<class T> virtual void JGTL::QuadTreeStub< T >::display (int *level*) const [inline, virtual]

Implements JGTL::QuadTreeNode< T >.

6.43.3 Member Data Documentation

6.43.3.1 template<class T> T JGTL::QuadTreeStub< T >::value [protected]

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

- JGTL_QuadTree.h

6.44 JGTL::Ray2< T > Class Template Reference

This class handles 2D Rays and Line Segments.

`#include <JGTL_Ray2.h>`

Public Member Functions

- Ray2 ()
- template<class TT, class TTT>
  Ray2 (const Vector2< TT > &_base, const Vector2< TTT > &_direction)
- Vector2< T > getDirection () const
- Vector2< T > getBase () const
- Vector2< T > getEndPoint () const
- template<class TT>
  void setBase (const Vector2< TT > &newBase)
- template<class TT>
  void setDirection (const Vector2< TT > &newDirection)
- template<class TT>
  T getProjectionTVal (const Vector2< TT > &Vector) const
- Vector2< T > getProjectionVector (T t) const
- void normalize ()
- template<class TT>
  std::pair< IntersectionState, float > getIntersection (const Ray2< TT > &other, bool lineSegment=false) const

Protected Member Functions

- bool within (T a, T b, T c) const
- bool putwhere (T x, T y, Vector2< T > &where, bool first) const

Protected Attributes

- Vector2< T > base
- Vector2< T > direction

6.44.1 Detailed Description

template<class T> class JGTL::Ray2< T >

This class handles 2D Rays and Line Segments.

Author:

Jason Gauci 2008

6.44.2 Constructor & Destructor Documentation

6.44.2.1 template<class T> JGTL::Ray2< T >::Ray2 () `[inline]`

6.44.2.2 template<class T> template<class TT, class TTT> JGTL::Ray2< T >::Ray2 (const Vector2< TT > & *_base*, const Vector2< TTT > & *_direction*) `[inline]`

6.44.3 Member Function Documentation

6.44.3.1 template<class T> Vector2<T> JGTL::Ray2< T >::getDirection () const `[inline]`

6.44.3.2 template<class T> Vector2<T> JGTL::Ray2< T >::getBase () const `[inline]`

6.44.3.3 template<class T> Vector2<T> JGTL::Ray2< T >::getEndPoint () const `[inline]`

6.44.3.4 template<class T> template<class TT> void JGTL::Ray2< T >::setBase (const Vector2< TT > & *newBase*) `[inline]`

6.44.3.5 template<class T> template<class TT> void JGTL::Ray2< T >::setDirection (const Vector2< TT > & *newDirection*) `[inline]`

6.44.3.6 template<class T> template<class TT> T JGTL::Ray2< T >::getProjectionTVal (const Vector2< TT > & *Vector*) const `[inline]`

6.44.3.7 template<class T> Vector2<T> JGTL::Ray2< T >::getProjectionVector (T *t*) const `[inline]`

6.44.3.8 template<class T> void JGTL::Ray2< T >::normalize () `[inline]`

6.44.3.9 template<class T> template<class TT> std::pair<IntersectionState,float> JGTL::Ray2< T >::getIntersection (const Ray2< TT > & *other*, bool *lineSegment* = `false`) const `[inline]`

6.44.3.10 template<class T> bool JGTL::Ray2< T >::within (T *a*, T *b*, T *c*) const `[inline, protected]`

6.44.3.11 template<class T> bool JGTL::Ray2< T >::putwhere (T *x*, T *y*, Vector2< T > & *where*, bool *first*) const `[inline, protected]`

6.44.4 Member Data Documentation

6.44.4.1 template<class T> Vector2<T> JGTL::Ray2< T >::base `[protected]`

6.44.4.2 template<class T> Vector2<T> JGTL::Ray2< T >::direction `[protected]`

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

- JGTL_Ray2.h

6.45 JGTL::Ray3< T > Class Template Reference

This class handles 3D Rays and Line Segments.

`#include <JGTL_Ray3.h>`

Public Member Functions

- Ray3 ()
- template<class TT, class TTT>
  Ray3 (const Vector3< TT > &_base, const Vector3< TTT > &_direction)
- Vector3< T > getDirection () const
- Vector3< T > getBase () const
- template<class TT>
  void setBase (Vector3< TT > newBase)
- template<class TT>
  void setDirection (const Vector3< TT > &newDirection)
- template<class TT>
  T getProjectionTVal (const Vector3< TT > &Vector) const
- Vector3< T > getProjectionVector (T t) const
- void normalize ()

Protected Attributes

- Vector3< T > base
- Vector3< T > direction

6.45.1 Detailed Description

template<class T> class JGTL::Ray3< T >

This class handles 3D Rays and Line Segments.

Author:

Jason Gauci 2008

6.45.2 Constructor & Destructor Documentation

6.45.2.1 template<class T> JGTL::Ray3< T >::Ray3 () [inline]

6.45.2.2 template<class T> template<class TT, class TTT> JGTL::Ray3< T >::Ray3 (const Vector3< TT > & *_base*, const Vector3< TTT > & *_direction*) [inline]

6.45.3 Member Function Documentation

6.45.3.1 template<class T> Vector3<T> JGTL::Ray3< T >::getDirection () const [inline]

6.45.3.2 template<class T> Vector3<T> JGTL::Ray3< T >::getBase () const [inline]

6.45.3.3 template<class T> template<class TT> void JGTL::Ray3< T >::setBase (Vector3< TT > *newBase*) [inline]

6.45.3.4 template<class T> template<class TT> void JGTL::Ray3< T >::setDirection (const Vector3< TT > & *newDirection*) [inline]

6.45.3.5 template<class T> template<class TT> T JGTL::Ray3< T >::getProjectionTVal (const Vector3< TT > & *Vector*) const [inline]

6.45.3.6 template<class T> Vector3<T> JGTL::Ray3< T >::getProjectionVector (T *t*) const [inline]

6.45.3.7 template<class T> void JGTL::Ray3< T >::normalize () [inline]

6.45.4 Member Data Documentation

6.45.4.1 template<class T> Vector3<T> JGTL::Ray3< T >::base [protected]

6.45.4.2 template<class T> Vector3<T> JGTL::Ray3< T >::direction [protected]

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

- JGTL_Ray3.h

6.46 JGTL::Rectangle3< T > Class Template Reference

`#include <JGTL_Rectangle3.h>`

Public Member Functions

- Rectangle3 ()
- Rectangle3 (const Vector3< T > &_topLeft, const Vector3< T > &_size)
- bool contains (const Vector3< T > &point) const
- const Vector3< T > & getFirstPoint () const
- bool getNextDiscretePoint (Vector3< T > &currentPoint) const

Public Attributes

- Vector3< T > topLeft
- Vector3< T > size

template<class T> class JGTL::Rectangle3< T >

6.46.1 Constructor & Destructor Documentation

6.46.1.1 template<class T> JGTL::Rectangle3< T >::Rectangle3 () [inline]

6.46.1.2 template<class T> JGTL::Rectangle3< T >::Rectangle3 (const Vector3< T > & *_topLeft*, const Vector3< T > & *_size*) [inline]

6.46.2 Member Function Documentation

6.46.2.1 template<class T> bool JGTL::Rectangle3< T >::contains (const Vector3< T > & *point*) const [inline]

6.46.2.2 template<class T> const Vector3<T>& JGTL::Rectangle3< T >::getFirstPoint () const [inline]

6.46.2.3 template<class T> bool JGTL::Rectangle3< T >::getNextDiscretePoint (Vector3< T > & *currentPoint*) const [inline]

6.46.3 Member Data Documentation

6.46.3.1 template<class T> Vector3<T> JGTL::Rectangle3< T >::topLeft

6.46.3.2 template<class T> Vector3<T> JGTL::Rectangle3< T >::size

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

- JGTL_Rectangle3.h

6.47 JGTL::RectangleIndex3 Class Reference

`#include <JGTL_Index3.h>`

Public Member Functions

- RectangleIndex3 ()
- RectangleIndex3 (const Index3 &_topLeft, const Index3 &_size)
- RectangleIndex3 (int x, int y, int z, int sizex, int sizey, int sizez)
- bool contains (const Index3 &point) const
- bool contains (const RectangleIndex3 &other) const
- const Index3 & getFirstPoint () const
- bool getNextPoint (Index3 &currentPoint) const
- int getArea () const

Public Attributes

- Index3 topLeft
- Index3 size

6.47.1 Constructor & Destructor Documentation

6.47.1.1 JGTL::RectangleIndex3::RectangleIndex3 () [inline]

6.47.1.2 JGTL::RectangleIndex3::RectangleIndex3 (const Index3 & *_topLeft*, const Index3 & *_size*) [inline]

6.47.1.3 JGTL::RectangleIndex3::RectangleIndex3 (int *x*, int *y*, int *z*, int *sizex*, int *sizey*, int *sizez*) [inline]

6.47.2 Member Function Documentation

6.47.2.1 bool JGTL::RectangleIndex3::contains (const Index3 & *point*) const [inline]

6.47.2.2 bool JGTL::RectangleIndex3::contains (const RectangleIndex3 & *other*) const [inline]

6.47.2.3 const Index3& JGTL::RectangleIndex3::getFirstPoint () const [inline]

6.47.2.4 bool JGTL::RectangleIndex3::getNextPoint (Index3 & *currentPoint*) const [inline]

6.47.2.5 int JGTL::RectangleIndex3::getArea () const [inline]

6.47.3 Member Data Documentation

6.47.3.1 Index3 JGTL::RectangleIndex3::topLeft

6.47.3.2 Index3 JGTL::RectangleIndex3::size

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

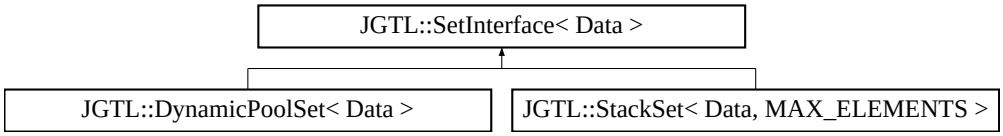
- JGTL_Index3.h

6.48 JGTL::SetInterface< Data > Class Template Reference

This class acts as a base class for the Set construct.

`#include <JGTL_SetInterface.h>`

Inheritance diagram for JGTL::SetInterface< Data >::



Public Types

- typedef Data * iterator
- typedef const Data * const_iterator

Public Member Functions

- SetInterface ()
- virtual ~SetInterface ()
- bool operator== (const SetInterface &other) const
- virtual bool resize (int newSize)
- virtual iterator insert (const Data &data)
- int size () const
- bool empty () const
- void clear ()
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- const bool hasData (const Data &data) const
- iterator find (const Data &data)
- void erase (const Data &data)
- void erase (const_iterator iter)
- void eraseIndex (int index)
- Data getIndex (int index) const
- const Data & getIndexRef (int index) const
- iterator getIndexPtr (int index)
- const_iterator getIndexPtr (int index) const

Protected Attributes

- int numElements
- int maxElements
- Data * dataList

6.48.1 Detailed Description

template<class Data> class JGTL::SetInterface< Data >

This class acts as a base class for the Set construct.

Author:

Jason Gauci 2008

6.48.2 Member Typedef Documentation

6.48.2.1 template<class Data> typedef Data* JGTL::SetInterface< Data >::iterator

6.48.2.2 template<class Data> typedef const Data* JGTL::SetInterface< Data >::const_iterator

6.48.3 Constructor & Destructor Documentation

6.48.3.1 template<class Data> JGTL::SetInterface< Data >::SetInterface () [inline]

6.48.3.2 template<class Data> virtual JGTL::SetInterface< Data >::~SetInterface () [inline, virtual]

6.48.4 Member Function Documentation

6.48.4.1 template<class Data> bool JGTL::SetInterface< Data >::operator== (const SetInterface< Data > & *other*) const [inline]

6.48.4.2 template<class Data> virtual bool JGTL::SetInterface< Data >::resize (int *newSize*) [inline, virtual]

Reimplemented in JGTL::DynamicPoolSet< Data >.

6.48.4.3 template<class Data> virtual iterator JGTL::SetInterface< Data >::insert (const Data & *data*) [inline, virtual]

6.48.4.4 template<class Data> int JGTL::SetInterface< Data >::size () const [inline]

6.48.4.5 template<class Data> bool JGTL::SetInterface< Data >::empty () const [inline]

6.48.4.6 template<class Data> void JGTL::SetInterface< Data >::clear () [inline]

6.48.4.7 template<class Data> iterator JGTL::SetInterface< Data >::begin () [inline]

6.48.4.8 template<class Data> const_iterator JGTL::SetInterface< Data >::begin () const [inline]

6.48.4.9 template<class Data> iterator JGTL::SetInterface< Data >::end () [inline]

6.48.4.10 template<class Data> const_iterator JGTL::SetInterface< Data >::end () const [inline]

6.48.4.11 template<class Data> const bool JGTL::SetInterface< Data >::hasData (const Data & *data*) const [inline]

6.48.4.12 template<class Data> iterator JGTL::SetInterface< Data >::find (const Data & *data*) [inline]

6.48.4.13 template<class Data> void JGTL::SetInterface< Data >::erase (const Data & *data*) [inline]

6.48.4.14 template<class Data> void JGTL::SetInterface< Data >::erase (const_iterator *iter*) [inline]

6.48.4.15 template<class Data> void JGTL::SetInterface< Data >::eraseIndex (int *index*) [inline]

6.48.4.16 template<class Data> Data JGTL::SetInterface< Data >::getIndex (int *index*) const [inline]

6.48.4.17 template<class Data> const Data& JGTL::SetInterface< Data >::getIndexRef (int *index*) const [inline]

6.48.4.18 template<class Data> iterator JGTL::SetInterface< Data >::getIndexPtr (int *index*) [inline]

6.48.4.19 template<class Data> const_iterator JGTL::SetInterface< Data >::getIndexPtr (int *index*) const [inline]

6.48.5 Member Data Documentation

6.48.5.1 template<class Data> int JGTL::SetInterface< Data >::numElements [protected]

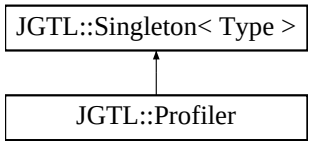
- JGTL_SetInterface.h

6.49 JGTL::Singleton< Type > Class Template Reference

This class handles Singletons (Global Single-Instance Classes).

`#include <JGTL_Singleton.h>`

Inheritance diagram for JGTL::Singleton< Type >::



Static Public Member Functions

- static Type * getInstance (void)
- static void destroyInstance ()

Protected Member Functions

- Singleton ()
- virtual ~Singleton ()

Static Protected Attributes

- static Type * instance = NULL

6.49.1 Detailed Description

template<class Type> class JGTL::Singleton< Type >

This class handles Singletons (Global Single-Instance Classes).

Author:

Jason Gauci 2008

6.49.2 Constructor & Destructor Documentation

6.49.2.1 template<class Type> JGTL::Singleton< Type >::Singleton ()

`[inline, protected]`

6.49.2.2 template<class Type> virtual JGTL::Singleton< Type >::~Singleton ()

`[inline, protected, virtual]`

6.49.3 Member Function Documentation

6.49.3.1 template<class Type> static Type* JGTL::Singleton< Type >::getInstance (void)

`[inline, static]`

6.49.3.2 template<class Type> static void JGTL::Singleton< Type >::destroyInstance ()

`[inline, static]`

Reimplemented in JGTL::Profiler.

6.49.4 Member Data Documentation

6.49.4.1 template<class Type> Type * JGTL::Singleton< Type >::instance = NULL

`[inline, static, protected]`

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

- JGTL_Singleton.h

6.50 JGTL::SortedList< Data > Class Template Reference

`#include <JGTL_SortedList_delete.h>`

Public Member Functions

- SortedList (size_t _maxElements)
- virtual ~SortedList ()
- void addData (const Data &data)
- const size_t & getDataSize ()
- const bool hasData (const Data &other) const
- const Data & getData (size_t index) const
- const Data * getDataPtr (size_t index) const

Protected Attributes

- size_t numElements
- size_t maxElements
- Data * dataList

template<class Data> class JGTL::SortedList< Data >

6.50.1 Constructor & Destructor Documentation

6.50.1.1 template<class Data> JGTL::SortedList< Data >::SortedList (size_t *_maxElements*) [inline]

6.50.1.2 template<class Data> virtual JGTL::SortedList< Data >::~SortedList () [inline, virtual]

6.50.2 Member Function Documentation

6.50.2.1 template<class Data> void JGTL::SortedList< Data >::addData (const Data & *data*) [inline]

6.50.2.2 template<class Data> const size_t& JGTL::SortedList< Data >::getDataSize () [inline]

6.50.2.3 template<class Data> const bool JGTL::SortedList< Data >::hasData (const Data & *other*) const [inline]

6.50.2.4 template<class Data> const Data& JGTL::SortedList< Data >::getData (size_t *index*) const [inline]

6.50.2.5 template<class Data> const Data* JGTL::SortedList< Data >::getDataPtr (size_t *index*) const [inline]

6.50.3 Member Data Documentation

6.50.3.1 template<class Data> size_t JGTL::SortedList< Data >::numElements [protected]

6.50.3.2 template<class Data> size_t JGTL::SortedList< Data >::maxElements [protected]

6.50.3.3 template<class Data> Data* JGTL::SortedList< Data >::dataList [protected]

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

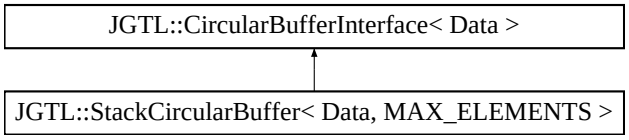
- JGTL_SortedList_delete.h

6.51 JGTL::StackCircularBuffer< Data, MAX_ELEMENTS > Class Template Reference

The StackCircularBuffer Class handles a Circular Buffer.

`#include <JGTL_StackCircularBuffer.h>`

Inheritance diagram for JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::



Public Member Functions

- StackCircularBuffer ()
- StackCircularBuffer (const StackCircularBuffer &other)
- const StackCircularBuffer & operator= (const StackCircularBuffer &other)
- virtual ~StackCircularBuffer ()
- virtual bool resize (int newSize)

Protected Member Functions

- void copyFrom (const StackCircularBuffer &other)

Protected Attributes

- unsigned char data [MAX_ELEMENTS ∗sizeof(Data)]

6.51.1 Detailed Description

template<class Data, int MAX_ELEMENTS = 32> class JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >

The StackCircularBuffer Class handles a Circular Buffer.

Author:

Jason Gauci 2008

6.51.2 Constructor & Destructor Documentation

6.51.2.1 template<class Data, int MAX_ELEMENTS = 32> JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::StackCircularBuffer () [inline]

6.51.2.2 template<class Data, int MAX_ELEMENTS = 32> JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::StackCircularBuffer (const StackCircularBuffer< Data, MAX_ELEMENTS > & *other*) [inline]

6.51.2.3 template<class Data, int MAX_ELEMENTS = 32> virtual JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::~StackCircularBuffer () [inline, virtual]

6.51.3 Member Function Documentation

6.51.3.1 template<class Data, int MAX_ELEMENTS = 32> const StackCircularBuffer& JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::operator= (const StackCircularBuffer< Data, MAX_ELEMENTS > & *other*) [inline]

6.51.3.2 template<class Data, int MAX_ELEMENTS = 32> virtual bool JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::resize (int *newSize*) [inline, virtual]

Implements JGTL::CircularBufferInterface< Data >.

6.51.3.3 template<class Data, int MAX_ELEMENTS = 32> void JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::copyFrom (const StackCircularBuffer< Data, MAX_ELEMENTS > & *other*) [inline, protected]

6.51.4 Member Data Documentation

6.51.4.1 template<class Data, int MAX_ELEMENTS = 32> unsigned char JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >::data[MAX_ELEMENTS *sizeof(Data)] [protected]

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

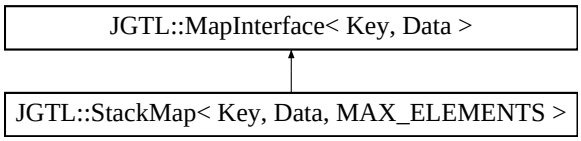
- JGTL_StackCircularBuffer.h

6.52 JGTL::StackMap< Key, Data, MAX_ELEMENTS > Class Template Reference

The StackMap Class is a fixed, array-based, sorted key structure.

`#include <JGTL_StackMap.h>`

Inheritance diagram for JGTL::StackMap< Key, Data, MAX_ELEMENTS >::



Public Member Functions

- StackMap ()
- StackMap (const StackMap &other)
- const StackMap & operator= (const StackMap &other)
- virtual ~StackMap ()
- virtual bool resize (int newSize)

Protected Member Functions

- virtual void copyFrom (const StackMap &other)

Protected Attributes

- unsigned char data [MAX_ELEMENTS ∗sizeof(std::pair< Key, Data >)]

6.52.1 Detailed Description

template<class Key, class Data, int MAX_ELEMENTS = 8> class JGTL::StackMap< Key, Data, MAX_ELEMENTS >

The StackMap Class is a fixed, array-based, sorted key structure.

Author:

Jason Gauci 2008

6.52.2 Constructor & Destructor Documentation

6.52.2.1 template<class Key, class Data, int MAX_ELEMENTS = 8> JGTL::StackMap< Key, Data, MAX_ELEMENTS >::StackMap ()

`[inline]`

6.52.2.2 template<class Key, class Data, int MAX_ELEMENTS = 8> JGTL::StackMap< Key, Data, MAX_ELEMENTS >::StackMap (const StackMap< Key, Data, MAX_ELEMENTS > & *other*)

`[inline]`

6.52.2.3 template<class Key, class Data, int MAX_ELEMENTS = 8> virtual JGTL::StackMap< Key, Data, MAX_ELEMENTS >::~StackMap ()

`[inline, virtual]`

6.52.3 Member Function Documentation

6.52.3.1 template<class Key, class Data, int MAX_ELEMENTS = 8> const StackMap& JGTL::StackMap< Key, Data, MAX_ELEMENTS >::operator= (const StackMap< Key, Data, MAX_ELEMENTS > & *other*)

`[inline]`

6.52.3.2 template<class Key, class Data, int MAX_ELEMENTS = 8> virtual bool JGTL::StackMap< Key, Data, MAX_ELEMENTS >::resize (int *newSize*)

`[inline, virtual]`

Implements JGTL::MapInterface< Key, Data >.

6.52.3.3 template<class Key, class Data, int MAX_ELEMENTS = 8> virtual void JGTL::StackMap< Key, Data, MAX_ELEMENTS >::copyFrom (const StackMap< Key, Data, MAX_ELEMENTS > & *other*)

`[inline, protected, virtual]`

6.52.4 Member Data Documentation

6.52.4.1 template<class Key, class Data, int MAX_ELEMENTS = 8> unsigned char JGTL::StackMap< Key, Data, MAX_ELEMENTS >::data[MAX_ELEMENTS *sizeof(std::pair< Key, Data >)]

`[protected]`

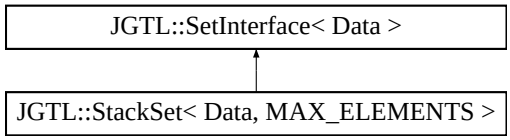
The documentation for this class was generated from the following file:

- JGTL_StackMap.h

6.53 JGTL::StackSet< Data, MAX_ELEMENTS > Class Template Reference

`#include <JGTL_StackSet.h>`

Inheritance diagram for JGTL::StackSet< Data, MAX_ELEMENTS >::



Public Member Functions

- StackSet ()
- StackSet (const StackSet &other)
- const StackSet & operator= (const StackSet &other)
- virtual void copyFrom (const StackSet &other)
- virtual ~StackSet ()

Protected Attributes

- unsigned char data [MAX_ELEMENTS ∗sizeof(Data)]

template<class Data, int MAX_ELEMENTS = 8> class JGTL::StackSet< Data, MAX_ELEMENTS >

6.53.1 Constructor & Destructor Documentation

6.53.1.1 template<class Data, int MAX_ELEMENTS = 8> JGTL::StackSet< Data, MAX_ELEMENTS >::StackSet () [inline]

6.53.1.2 template<class Data, int MAX_ELEMENTS = 8> JGTL::StackSet< Data, MAX_ELEMENTS >::StackSet (const StackSet< Data, MAX_ELEMENTS > & *other*) [inline]

6.53.1.3 template<class Data, int MAX_ELEMENTS = 8> virtual JGTL::StackSet< Data, MAX_ELEMENTS >::~StackSet () [inline, virtual]

6.53.2 Member Function Documentation

6.53.2.1 template<class Data, int MAX_ELEMENTS = 8> const StackSet& JGTL::StackSet< Data, MAX_ELEMENTS >::operator= (const StackSet< Data, MAX_ELEMENTS > & *other*) [inline]

6.53.2.2 template<class Data, int MAX_ELEMENTS = 8> virtual void JGTL::StackSet< Data, MAX_ELEMENTS >::copyFrom (const StackSet< Data, MAX_ELEMENTS > & *other*) [inline, virtual]

6.53.3 Member Data Documentation

6.53.3.1 template<class Data, int MAX_ELEMENTS = 8> unsigned char JGTL::StackSet< Data, MAX_ELEMENTS >::data[MAX_ELEMENTS *sizeof(Data)] [protected]

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

- JGTL_StackSet.h

6.54 JGTL::STATIC_MAX_SIZE< One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two = One, class Three = One, class Four = One, class Five = One, class Six = One, class Seven = One, class Eight = One, class Nine = One, class Ten = One> struct JGTL::STATIC_MAX_SIZE< One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten >

6.54.1 Member Data Documentation

6.54.1.1 template<class One, class Two = One, class Three = One, class Four = One, class Five = One, class Six = One, class Seven = One, class Eight = One, class Nine = One, class Ten = One> const int JGTL::STATIC_MAX_SIZE< One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,Two,Four,Five,Six,Seven,Eight,Nine,Ten>::RESULT,
STATIC_MAX_SIZE<One,One,Three,Four,Five,Six,Seven,Eight,Nine,Ten>::RESULT
>::RESULT
```

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

- JGTL_Variant.h

6.55 JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, Two > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, One, Two >

6.55.1 Member Data Documentation

6.55.1.1 template<class One, class Two> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, One, Two >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(One) > sizeof(Two)),
sizeof(One),
sizeof(Two)
>::RESULT
```

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

- JGTL_Variant.h

6.56 JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, Two, Three > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two, class Three> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, Two, Three >

6.56.1 Member Data Documentation

6.56.1.1 template<class One, class Two, class Three> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, Two, Three >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,One,One,One,One,One,One,Two>::RESULT,
STATIC_MAX_SIZE<One,One,One,One,One,One,One,One,Three>::RESULT
>::RESULT
```

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

- JGTL_Variant.h

6.57 JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, Two, Three, Four > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two, class Three, class Four> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, Two, Three, Four >

6.57.1 Member Data Documentation

6.57.1.1 template<class One, class Two, class Three, class Four> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, Two, Three, Four >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,One,One,One,One,One,Two,Four>::RESULT,
STATIC_MAX_SIZE<One,One,One,One,One,One,One,Three,Four>::RESULT
>::RESULT
```

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

- JGTL_Variant.h

6.58 JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, Two, Three, Four, Five > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two, class Three, class Four, class Five> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, Two, Three, Four, Five >

6.58.1 Member Data Documentation

6.58.1.1 template<class One, class Two, class Three, class Four, class Five> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, Two, Three, Four, Five >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,One,One,One,One,Two,Four,Five>::RESULT,
STATIC_MAX_SIZE<One,One,One,One,One,One,Three,Four,Five>::RESULT
>::RESULT
```

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

- JGTL_Variant.h

6.59 JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, Two, Three, Four, Five, Six > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two, class Three, class Four, class Five, class Six> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, Two, Three, Four, Five, Six >

6.59.1 Member Data Documentation

6.59.1.1 template<class One, class Two, class Three, class Four, class Five, class Six> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, Two, Three, Four, Five, Six >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,One,One,One,One,Two,Four,Five,Six>::RESULT,
STATIC_MAX_SIZE<One,One,One,One,One,One,Three,Four,Five,Six>::RESULT
>::RESULT
```

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

- JGTL_Variant.h

6.60 JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, Two, Three, Four, Five, Six, Seven > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two, class Three, class Four, class Five, class Six, class Seven> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, Two, Three, Four, Five, Six, Seven >

6.60.1 Member Data Documentation

6.60.1.1 template<class One, class Two, class Three, class Four, class Five, class Six, class Seven> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, Two, Three, Four, Five, Six, Seven >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,One,One,One,Two,Four,Five,Six,Seven>::RESULT,
STATIC_MAX_SIZE<One,One,One,One,One,Three,Four,Five,Six,Seven>::RESULT
>::RESULT
```

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

- JGTL_Variant.h

6.61 JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, Two, Three, Four, Five, Six, Seven, Eight > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two, class Three, class Four, class Five, class Six, class Seven, class Eight> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, Two, Three, Four, Five, Six, Seven, Eight >

6.61.1 Member Data Documentation

6.61.1.1 template<class One, class Two, class Three, class Four, class Five, class Six, class Seven, class Eight> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, Two, Three, Four, Five, Six, Seven, Eight >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,One,One,Two,Four,Five,Six,Seven,Eight>::RESULT,
STATIC_MAX_SIZE<One,One,One,One,Three,Four,Five,Six,Seven,Eight>::RESUL
>::RESULT
```

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

- JGTL_Variant.h

6.62 JGTL::JGTL::STATIC_MAX_SIZE< One, One, Two, Three, Four, Five, Six, Seven, Eight, Nine > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const int RESULT

template<class One, class Two, class Three, class Four, class Five, class Six, class Seven, class Eight, class Nine> struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, Two, Three, Four, Five, Six, Seven, Eight, Nine >

6.62.1 Member Data Documentation

6.62.1.1 template<class One, class Two, class Three, class Four, class Five, class Six, class Seven, class Eight, class Nine> const int JGTL::JGTL::STATIC_MAX_SIZE< One, One, Two, Three, Four, Five, Six, Seven, Eight, Nine >::RESULT [static]

Initial value:

```
TYPEIF<
int,
(sizeof(Two) > sizeof(Three)),
STATIC_MAX_SIZE<One,One,One,Two,Four,Five,Six,Seven,Eight,Nine>::RESULT,
STATIC_MAX_SIZE<One,One,One,Three,Four,Five,Six,Seven,Eight,Nine>::RESULT
>::RESULT
```

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

- JGTL_Variant.h

6.63 JGTL::STATIC_MOD< Type, a, b > Struct Template Reference

`#include <JGTL_IntegralUnits.h>`

Static Public Attributes

- static const Type VALUE = (a%b)

template<class Type, Type a, Type b> struct JGTL::STATIC_MOD< Type, a, b >

6.63.1 Member Data Documentation

6.63.1.1 template<class Type, Type a, Type b> const Type JGTL::STATIC_MOD< Type, a, b >::VALUE = (a%b) [static]

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

- JGTL_IntegralUnits.h

6.64 JGTL::TreeList< Data > Class Template Reference

`#include <JGTL_TreeList.h>`

Public Member Functions

- TreeList ()
- ~TreeList ()

Protected Attributes

- shared_ptr< TreeListNode > root

template<class Data> class JGTL::TreeList< Data >

6.64.1 Constructor & Destructor Documentation

6.64.1.1 template<class Data> JGTL::TreeList< Data >::TreeList () [inline]

6.64.1.2 template<class Data> JGTL::TreeList< Data >::~TreeList () [inline]

6.64.2 Member Data Documentation

6.64.2.1 template<class Data> shared_ptr<TreeListNode> JGTL::TreeList< Data >::root [protected]

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

- JGTL_TreeList.h

6.65 JGTL::TreeListNode< Data > Class Template Reference

`#include <JGTL_TreeList.h>`

Public Attributes

- Data sibling
- Data child

template<class Data> class JGTL::TreeListNode< Data >

6.65.1 Member Data Documentation

6.65.1.1 template<class Data> Data JGTL::TreeListNode< Data >::sibling

6.65.1.2 template<class Data> Data JGTL::TreeListNode< Data >::child

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

- JGTL_TreeList.h

6.66 JGTL::TYPEIF< Type, condition, Then, Else > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const Type RESULT = Then

template<class Type, bool condition, Type Then, Type Else> struct JGTL::TYPEIF< Type, condition, Then, Else >

6.66.1 Member Data Documentation

6.66.1.1 template<class Type, bool condition, Type Then, Type Else> static const Type JGTL::TYPEIF< Type, condition, Then, Else >::RESULT = Then [static]

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

- JGTL_IntegralUnits.h
- JGTL_Variant.h

6.67 JGTL::JGTL::TYPEIF< Type, false, Then, Else > Struct Template Reference

`#include <JGTL_Variant.h>`

Static Public Attributes

- static const Type RESULT = Else

template<class Type, Type Then, Type Else> struct JGTL::JGTL::TYPEIF< Type, false, Then, Else >

6.67.1 Member Data Documentation

6.67.1.1 template<class Type, Type Then, Type Else> static const Type JGTL::JGTL::TYPEIF< Type, false, Then, Else >::RESULT = Else [static]

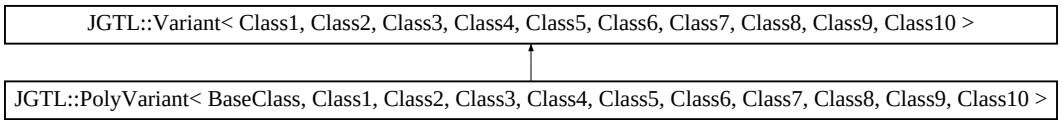
The documentation for this struct was generated from the following files:

- JGTL_IntegralUnits.h
- JGTL_Variant.h

6.68 JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 > Class Template Reference

`#include <JGTL_Variant.h>`

Inheritance diagram for JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::



Public Member Functions

- Variant ()
- Variant (const Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 > &other)
- template<class T>
  bool isOfType ()
- template<class ReturnValueClass>
  ReturnValueClass getValue () const
- template<class ReturnValueClass>
  ReturnValueClass & getValueRef ()
- template<class ReturnValueClass>
  const ReturnValueClass & getValueRef () const
- template<class ReturnValueClass>
  ReturnValueClass * getValuePtr ()
- template<class ReturnValueClass>
  const ReturnValueClass * getValuePtr () const
- void clearValue ()
- template<class ValueToSet>
  void setValue (const ValueToSet &newValue)

Protected Attributes

- char data [STATIC_MAX_SIZE< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::RESULT]
- unsigned char typeOfData

template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> class JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >

6.68.1 Constructor & Destructor Documentation

6.68.1.1 template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::Variant () `[inline]`

6.68.1.2 template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::Variant (const Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 > & *other*) `[inline]`

6.68.2 Member Function Documentation

6.68.2.1 template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> template<class T> bool JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::isOfType () `[inline]`

6.68.2.2 template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> template<class ReturnValueClass> ReturnValueClass JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::getValue () const `[inline]`

6.68.2.3 template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> template<class ReturnValueClass> ReturnValueClass& JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::getValueRef () `[inline]`

6.68.2.4 template<class Class1, class Class2 = NullVariantClass, class Class3 =

6.68.3 Member Data Documentation

6.68.3.1 template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> char JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::data[STATIC_MAX_SIZE< Class1,Class2,Class3,Class4,Class5,Class6,Class7,Class8,Class9,Class10 >::RESULT] [protected]

6.68.3.2 template<class Class1, class Class2 = NullVariantClass, class Class3 = NullVariantClass, class Class4 = NullVariantClass, class Class5 = NullVariantClass, class Class6 = NullVariantClass, class Class7 = NullVariantClass, class Class8 = NullVariantClass, class Class9 = NullVariantClass, class Class10 = NullVariantClass> unsigned char JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >::typeOfData [protected]

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

- JGTL_Variant.h

6.69 JGTL::Vector2< T > Class Template Reference

This class handles 2D Vectors.

`#include <JGTL_Vector2.h>`

Public Member Functions

- Vector2 ()
- Vector2 (T _x, T _y)
- template<class TT>
  Vector2 (const Vector2< TT > &other)
- template<class TT>
  const Vector2 & operator= (const Vector2< TT > &other)
- template<class TT>
  bool operator== (const Vector2< TT > &other) const
- template<class TT>
  bool operator!= (const Vector2< TT > &other) const
- template<class TT>
  bool operator< (const Vector2< TT > &other) const
- Vector2< T > operator/ (T divisor) const
- void operator/= (T divisor)
- Vector2< T > operator- () const
- template<class TT>
  Vector2< T > operator- (const Vector2< TT > &other) const
- template<class TT>
  Vector2< T > operator+ (const Vector2< TT > &other) const
- Vector2< T > operator* (const T &coeff) const
- template<class TT>
  void operator+= (const Vector2< TT > &other)
- template<class TT>
  void operator-= (const Vector2< TT > &other)
- void operator*= (T val)
- template<class TT>
  T dot (const Vector2< TT > &other) const
- template<class TT>
  T cross (const Vector2< TT > &other) const
- Vector2< T > rightHandNormal () const
- T magnitudeSquared () const
- T magnitude () const
- void normalize ()
- Vector2< T > normalizeCopy () const
- template<class TT>
  Vector2< T > projectOn (const Vector2< TT > &other) const

- template<class TT>
  T distance (const Vector2< TT > &other) const
- template<class TT>
  T distance (TT _x, TT _y) const
- template<class TT>
  T distanceSquared (const Vector2< TT > &other) const
- template<class TT>
  T chessDistance (const Vector2< TT > &other) const
- template<class TT>
  T angleTo (const Vector2< TT > &other) const
- template<class TT>
  void rotate (TT angleTheta)
- template<class TT>
  Vector2< T > rotateCopy (TT angleTheta) const
- T getAngle () const
- bool isContainedIn (T left, T top, T right, T bottom) const
- template<class TT>
  T manhatDistance (const Vector2< TT > &other) const

Static Public Member Functions

- template<class TT, class TTT>
  static Vector2< T > fromMagnitudeAngle (TT magnitude, TTT angle)

Public Attributes

- T x
- T y

6.69.1 Detailed Description

template<class T> class JGTL::Vector2< T >

This class handles 2D Vectors.

Author:

Jason Gauci 2008

6.69.2 Constructor & Destructor Documentation

6.69.2.1 template<class T> JGTL::Vector2< T >::Vector2 () `[inline]`

6.69.2.2 template<class T> JGTL::Vector2< T >::Vector2 (T *_x*, T *_y*) `[inline]`

6.69.2.3 template<class T> template<class TT> JGTL::Vector2< T >::Vector2 (const Vector2< TT > & *other*) `[inline]`

6.69.3 Member Function Documentation

6.69.3.1 template<class T> template<class TT, class TTT> static Vector2<T> JGTL::Vector2< T >::fromMagnitudeAngle (TT *magnitude*, TTT *angle*) `[inline, static]`

6.69.3.2 template<class T> template<class TT> const Vector2& JGTL::Vector2< T >::operator= (const Vector2< TT > & *other*) `[inline]`

6.69.3.3 template<class T> template<class TT> bool JGTL::Vector2< T >::operator== (const Vector2< TT > & *other*) const `[inline]`

6.69.3.4 template<class T> template<class TT> bool JGTL::Vector2< T >::operator!= (const Vector2< TT > & *other*) const `[inline]`

6.69.3.5 template<class T> template<class TT> bool JGTL::Vector2< T >::operator< (const Vector2< TT > & *other*) const `[inline]`

6.69.3.6 template<class T> Vector2<T> JGTL::Vector2< T >::operator/ (T *divisor*) const `[inline]`

6.69.3.7 template<class T> void JGTL::Vector2< T >::operator/= (T *divisor*) `[inline]`

6.69.3.8 template<class T> Vector2<T> JGTL::Vector2< T >::operator- () const `[inline]`

6.69.3.9 template<class T> template<class TT> Vector2<T> JGTL::Vector2< T >::operator- (const Vector2< TT > & *other*) const `[inline]`

6.69.3.10 template<class T> template<class TT> Vector2<T> JGTL::Vector2< T >::operator+ (const Vector2< TT > & *other*) const `[inline]`

6.69.3.11 template<class T> Vector2<T> JGTL::Vector2< T >::operator* (const T & *coeff*) const `[inline]`

6.69.3.12 template<class T> template<class TT> void JGTL::Vector2< T >::operator+= (const Vector2< TT > & *other*) `[inline]`

6.69.3.13 template<class T> template<class TT> void JGTL::Vector2< T >::operator-= (const Vector2< TT > & *other*) `[inline]`

- JGTL_Vector2.h

6.70 JGTL::Vector3< T > Class Template Reference

`#include <JGTL_Vector3.h>`

Public Member Functions

- Vector3 ()
- Vector3 (T _x, T _y, T _z)
- template<class TT>
  Vector3 (const Vector3< TT > &other)
- template<class TT>
  const Vector3 & operator= (const Vector3< TT > &other)
- template<class TT>
  bool operator== (const Vector3< TT > &other) const
- template<class TT>
  bool operator!= (const Vector3< TT > &other) const
- template<class TT>
  bool operator< (const Vector3< TT > &other) const
- Vector3< T > operator/ (T divisor) const
- void operator/= (T divisor)
- Vector3< T > operator- () const
- template<class TT>
  Vector3< T > operator- (const Vector3< TT > &other) const
- template<class TT>
  Vector3< T > operator+ (const Vector3< TT > &other) const
- Vector3< T > operator* (const T &coeff) const
- template<class TT>
  void operator+= (const Vector3< TT > &other)
- template<class TT>
  void operator-= (const Vector3< TT > &other)
- void operator*= (T val)
- template<class TT>
  T dot (const Vector3< TT > &other) const
- template<class TT>
  Vector3< T > cross (const Vector3< TT > &other) const
- T magnitude () const
- T magnitudeSquared () const
- void normalize ()
- Vector3< T > normalizeCopy () const
- template<class TT>
  Vector3< T > projectOn (const Vector3< TT > &other) const
- template<class TT>
  T distance (const Vector3< TT > &other) const

- template<class TT>
  T distance (TT _x, TT _y, TT _z) const

- template<class TT>
  T distanceSquared (const Vector3< TT > &other) const

- template<class TT>
  T distanceSquared (TT _x, TT _y, TT _z) const

- template<class TT>
  T chessDistance (const Vector3< TT > &other) const

- template<class TT>
  T manhatDistance (const Vector3< TT > &other) const

Public Attributes

- T x

- T y

- T z

template<class T> class JGTL::Vector3< T >

6.70.1 Constructor & Destructor Documentation

6.70.1.1 template<class T> JGTL::Vector3< T >::Vector3 () `[inline]`

6.70.1.2 template<class T> JGTL::Vector3< T >::Vector3 (T *_x*, T *_y*, T *_z*) `[inline]`

6.70.1.3 template<class T> template<class TT> JGTL::Vector3< T >::Vector3 (const Vector3< TT > & *other*) `[inline]`

6.70.2 Member Function Documentation

6.70.2.1 template<class T> template<class TT> const Vector3& JGTL::Vector3< T >::operator= (const Vector3< TT > & *other*) `[inline]`

6.70.2.2 template<class T> template<class TT> bool JGTL::Vector3< T >::operator== (const Vector3< TT > & *other*) const `[inline]`

6.70.2.3 template<class T> template<class TT> bool JGTL::Vector3< T >::operator!= (const Vector3< TT > & *other*) const `[inline]`

6.70.2.4 template<class T> template<class TT> bool JGTL::Vector3< T >::operator< (const Vector3< TT > & *other*) const `[inline]`

6.70.2.5 template<class T> Vector3<T> JGTL::Vector3< T >::operator/ (T *divisor*) const `[inline]`

6.70.2.6 template<class T> void JGTL::Vector3< T >::operator/= (T *divisor*) `[inline]`

6.70.2.7 template<class T> Vector3<T> JGTL::Vector3< T >::operator- () const `[inline]`

6.70.2.8 template<class T> template<class TT> Vector3<T> JGTL::Vector3< T >::operator- (const Vector3< TT > & *other*) const `[inline]`

6.70.2.9 template<class T> template<class TT> Vector3<T> JGTL::Vector3< T >::operator+ (const Vector3< TT > & *other*) const `[inline]`

6.70.2.10 template<class T> Vector3<T> JGTL::Vector3< T >::operator* (const T & *coeff*) const `[inline]`

6.70.2.11 template<class T> template<class TT> void JGTL::Vector3< T >::operator+= (const Vector3< TT > & *other*) `[inline]`

6.70.2.12 template<class T> template<class TT> void JGTL::Vector3< T >::operator-= (const Vector3< TT > & *other*) `[inline]`

6.70.2.13 template<class T> void JGTL::Vector3< T >::operator*= (T *val*) `[inline]`

- JGTL_Vector3.h

6.71 JGTL::Vector4< T > Class Template Reference

`#include <JGTL_Vector4.h>`

Public Member Functions

- Vector4 ()
- Vector4 (T _x, T _y, T _z, T _w)
- template<class TT>
  Vector4 (const Vector4< TT > &other)
- template<class TT>
  const Vector4 & operator= (const Vector4< TT > &other)
- template<class TT>
  bool operator== (const Vector4< TT > &other) const
- template<class TT>
  bool operator!= (const Vector4< TT > &other) const
- template<class TT>
  bool operator< (const Vector4< TT > &other) const
- Vector4< T > operator/ (T divisor) const
- void operator/= (T divisor)
- Vector4< T > operator- () const
- template<class TT>
  Vector4< T > operator- (const Vector4< TT > &other) const
- template<class TT>
  Vector4< T > operator+ (const Vector4< TT > &other) const
- Vector4< T > operator* (const T &coeff) const
- template<class TT>
  void operator+= (const Vector4< TT > &other)
- template<class TT>
  void operator-= (const Vector4< TT > &other)
- void operator*= (T val)
- template<class TT>
  T dot (const Vector4< TT > &other) const
- T magnitude () const
- T magnitudeSquared () const
- void normalize ()
- Vector4< T > normalizeCopy () const
- template<class TT>
  Vector4< T > projectOn (const Vector4< TT > &other) const
- template<class TT>
  T distance (const Vector4< TT > &other) const
- template<class TT>
  T distance (TT _x, TT _y, TT _z) const

- template<class TT>
  T distanceSquared (const Vector4< TT > &other) const

- template<class TT>
  T distanceSquared (TT _x, TT _y, TT _z) const

- template<class TT>
  T manhatDistance (const Vector4< TT > &other) const

Public Attributes

- T x

- T y

- T z

- T w

template<class T> class JGTL::Vector4< T >

6.71.1 Constructor & Destructor Documentation

6.71.1.1 template<class T> JGTL::Vector4< T >::Vector4 () [inline]

6.71.1.2 template<class T> JGTL::Vector4< T >::Vector4 (T *_x*, T *_y*, T *_z*, T *_w*) [inline]

6.71.1.3 template<class T> template<class TT> JGTL::Vector4< T >::Vector4 (const Vector4< TT > & *other*) [inline]

6.71.2 Member Function Documentation

6.71.2.1 template<class T> template<class TT> const Vector4& JGTL::Vector4< T >::operator= (const Vector4< TT > & *other*) [inline]

6.71.2.2 template<class T> template<class TT> bool JGTL::Vector4< T >::operator== (const Vector4< TT > & *other*) const [inline]

6.71.2.3 template<class T> template<class TT> bool JGTL::Vector4< T >::operator!= (const Vector4< TT > & *other*) const [inline]

6.71.2.4 template<class T> template<class TT> bool JGTL::Vector4< T >::operator< (const Vector4< TT > & *other*) const [inline]

6.71.2.5 template<class T> Vector4<T> JGTL::Vector4< T >::operator/ (T *divisor*) const [inline]

6.71.2.6 template<class T> void JGTL::Vector4< T >::operator/= (T *divisor*) [inline]

6.71.2.7 template<class T> Vector4<T> JGTL::Vector4< T >::operator- () const [inline]

6.71.2.8 template<class T> template<class TT> Vector4<T> JGTL::Vector4< T >::operator- (const Vector4< TT > & *other*) const [inline]

6.71.2.9 template<class T> template<class TT> Vector4<T> JGTL::Vector4< T >::operator+ (const Vector4< TT > & *other*) const [inline]

6.71.2.10 template<class T> Vector4<T> JGTL::Vector4< T >::operator* (const T & *coeff*) const [inline]

6.71.2.11 template<class T> template<class TT> void JGTL::Vector4< T >::operator+= (const Vector4< TT > & *other*) [inline]

6.71.2.12 template<class T> template<class TT> void JGTL::Vector4< T >::operator-= (const Vector4< TT > & *other*) [inline]

6.71.2.13 template<class T> void JGTL::Vector4< T >::operator*= (T *val*) [inline]

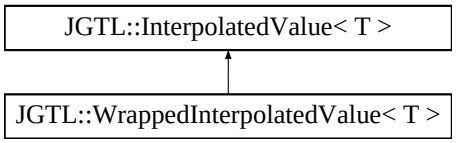
- JGTL_Vector4.h

6.72 JGTL::WrappedInterpolatedValue< T > Class Template Reference

The WrappedInterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff; This special instance of an InterpolatedValue is for values which wrap (angles, for example, which wrap around 2∗PI).

`#include <JGTL_WrappedInterpolatedValue.h>`

Inheritance diagram for JGTL::WrappedInterpolatedValue< T >::



Public Member Functions

- WrappedInterpolatedValue (const T &_minValue, const T &_maxValue)
- WrappedInterpolatedValue (float _interpolationCoeff, const T &_minValue, const T &_maxValue)
- WrappedInterpolatedValue (const T &_value, float _interpolationCoeff, const T &_minValue, const T &_maxValue)
- const WrappedInterpolatedValue< T > & operator= (T _potentialValue)
- virtual void setValue (const T &_value)
- virtual void update (int times=1)

Protected Member Functions

- void clampValues ()

Protected Attributes

- T minValue
- T maxValue
- T spread

6.72.1 Detailed Description

template<class T> class JGTL::WrappedInterpolatedValue< T >

The WrappedInterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff; This special instance of an InterpolatedValue is for values which wrap (angles, for example, which wrap around 2∗PI).

Author:

Jason Gauci 2008

6.72.2 Constructor & Destructor Documentation

6.72.2.1 template<class T> JGTL::WrappedInterpolatedValue< T >::WrappedInterpolatedValue (const T & *_minValue*, const T & *_maxValue*) [inline]

6.72.2.2 template<class T> JGTL::WrappedInterpolatedValue< T >::WrappedInterpolatedValue (float *_interpolationCoeff*, const T & *_minValue*, const T & *_maxValue*) [inline]

6.72.2.3 template<class T> JGTL::WrappedInterpolatedValue< T >::WrappedInterpolatedValue (const T & *_value*, float *_interpolationCoeff*, const T & *_minValue*, const T & *_maxValue*) [inline]

6.72.3 Member Function Documentation

6.72.3.1 template<class T> const WrappedInterpolatedValue<T>& JGTL::WrappedInterpolatedValue< T >::operator= (T *_potentialValue*) [inline]

Reimplemented from JGTL::InterpolatedValue< T >.

6.72.3.2 template<class T> virtual void JGTL::WrappedInterpolatedValue< T >::setValue (const T & *_value*) [inline, virtual]

Reimplemented from JGTL::InterpolatedValue< T >.

6.72.3.3 template<class T> virtual void JGTL::WrappedInterpolatedValue< T >::update (int *times* = 1) [inline, virtual]

Reimplemented from JGTL::InterpolatedValue< T >.

6.72.3.4 template<class T> void JGTL::WrappedInterpolatedValue< T >::clampValues () [inline, protected]

6.72.4 Member Data Documentation

6.72.4.1 template<class T> T JGTL::WrappedInterpolatedValue< T >::minValue [protected]

6.72.4.2 template<class T> T JGTL::WrappedInterpolatedValue< T >::maxValue [protected]

6.72.4.3 template<class T> T JGTL::WrappedInterpolatedValue< T >::spread [protected]

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

- JGTL_WrappedInterpolatedValue.h

6.73 JGTL::XorSpace< Rectangle, Point > Class Template Reference

This class handles Xor spaces. Think of this as a way to handle things like rectangular doughnuts. A positive space followed by a smaller concentric negative space would represent a doughnut.

`#include <JGTL_XorSpace.h>`

Public Types

- typedef vector< XorSpaceRect< Rectangle, Point > >::iterator SpaceIterator
- typedef vector< XorSpaceRect< Rectangle, Point > >::const_iterator ConstSpaceIterator

Public Member Functions

- XorSpace ()
- XorSpace (const Rectangle &start)
- void addSpace (const Rectangle &space, bool positive)
- void removeSpace (const Rectangle &space, bool positive)
- bool contains (const Point &point) const
- const Point & getTopLeft () const
- const Point & getBottomRight () const
- Point getSize () const
- Point getFirstPoint () const
- bool getNextPoint (Point &curPoint) const
- void pack ()

Protected Attributes

- vector< XorSpaceRect< Rectangle, Point > > spaces
- Point topLeft
- Point bottomRight

6.73.1 Detailed Description

template<class Rectangle, class Point> class JGTL::XorSpace< Rectangle, Point >

This class handles Xor spaces. Think of this as a way to handle things like rectangular doughnuts. A positive space followed by a smaller concentric negative space would represent a doughnut.

Author:

Jason Gauci 2008

6.73.2 Member Typedef Documentation

6.73.2.1 template<class Rectangle, class Point> typedef vector< XorSpaceRect<Rectangle,Point> >::iterator JGTL::XorSpace< Rectangle, Point >::SpaceIterator

6.73.2.2 template<class Rectangle, class Point> typedef vector< XorSpaceRect<Rectangle,Point> >::const_iterator JGTL::XorSpace< Rectangle, Point >::ConstSpaceIterator

6.73.3 Constructor & Destructor Documentation

6.73.3.1 template<class Rectangle, class Point> JGTL::XorSpace< Rectangle, Point >::XorSpace () [inline]

6.73.3.2 template<class Rectangle, class Point> JGTL::XorSpace< Rectangle, Point >::XorSpace (const Rectangle & *start*) [inline]

6.73.4 Member Function Documentation

6.73.4.1 template<class Rectangle, class Point> void JGTL::XorSpace< Rectangle, Point >::addSpace (const Rectangle & *space*, bool *positive*) [inline]

Adds a space with the highest priority to the list of spaces.

Parameters:

space The rectangle to add

positive Whether the rectangle is positive or negative (a negative cancels all positives before it)

6.73.4.2 template<class Rectangle, class Point> void JGTL::XorSpace< Rectangle, Point >::removeSpace (const Rectangle & *space*, bool *positive*) [inline]

Removes a space with the lowest priority and a certain size and polarity

Parameters:

space The rectangle to add

positive Whether the rectangle is positive or negative (a negative cancels all positives before it)

6.73.4.3 template<class Rectangle, class Point> bool JGTL::XorSpace< Rectangle, Point >::contains (const Point & *point*) const [inline]

Checks if the XorSpace contains a certain point

6.73.4.4 template<class Rectangle, class Point> const Point& JGTL::XorSpace< Rectangle, Point >::getTopLeft () const [inline]

6.73.4.5 template<class Rectangle, class Point> const Point& JGTL::XorSpace< Rectangle, Point >::getBottomRight () const [inline]

6.73.4.6 template<class Rectangle, class Point> Point JGTL::XorSpace< Rectangle, Point >::getSize () const [inline]

6.73.4.7 template<class Rectangle, class Point> Point JGTL::XorSpace< Rectangle, Point >::getFirstPoint () const [inline]

Returns the top-left point of an Xor-Space

6.73.4.8 template<class Rectangle, class Point> bool JGTL::XorSpace< Rectangle, Point >::getNextPoint (Point & *curPoint*) const [inline]

Gets the next point within an XorSpace. Works along with getFirstPoint() to iterate over all points.

6.73.4.9 template<class Rectangle, class Point> void JGTL::XorSpace< Rectangle, Point >::pack () [inline]

Tries to eliminate redundant spaces in an XorSpace

6.73.5 Member Data Documentation

6.73.5.1 template<class Rectangle, class Point> vector<XorSpaceRect<Rectangle,Point> > JGTL::XorSpace< Rectangle, Point >::spaces [protected]

6.73.5.2 template<class Rectangle, class Point> Point JGTL::XorSpace< Rectangle, Point >::topLeft [protected]

6.73.5.3 template<class Rectangle, class Point> Point JGTL::XorSpace< Rectangle, Point >::bottomRight [protected]

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

- JGTL_XorSpace.h

6.74 JGTL::XorSpaceRect< Rectangle, Point > Class Template Reference

This handles a single Xor Rectangle.

`#include <JGTL_XorSpace.h>`

Public Member Functions

- XorSpaceRect ()
- XorSpaceRect (Rectangle _rect, bool _positive)

Public Attributes

- Rectangle rect
- bool positive

6.74.1 Detailed Description

template<class Rectangle, class Point> class JGTL::XorSpaceRect< Rectangle, Point >

This handles a single Xor Rectangle.

Author:

Jason Gauci 2008

6.74.2 Constructor & Destructor Documentation

6.74.2.1 template<class Rectangle, class Point> JGTL::XorSpaceRect< Rectangle, Point >::XorSpaceRect () [inline]

6.74.2.2 template<class Rectangle, class Point> JGTL::XorSpaceRect< Rectangle, Point >::XorSpaceRect (Rectangle *_rect*, bool *_positive*) [inline]

6.74.3 Member Data Documentation

6.74.3.1 template<class Rectangle, class Point> Rectangle JGTL::XorSpaceRect< Rectangle, Point >::rect

6.74.3.2 template<class Rectangle, class Point> bool JGTL::XorSpaceRect< Rectangle, Point >::positive

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

- JGTL_XorSpace.h

Chapter 7

File Documentation

7.1 JGTL_Bar.h File Reference

```
#include <iostream>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Bar< BarValueType >

  The Bar Class handles a max/current value system (e.g. a Progress Bar).

Functions

- template<class BarValueType>
  std::ostream & JGTL::operator<< (std::ostream &stream, const Bar< BarValueType > &d)
- template<class BarValueType>
  std::istream & JGTL::operator>> (std::istream &stream, Bar< BarValueType > &d)

7.2 JGTL_CircularBuffer.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::CircularBuffer< Data >

  The CircularBuffer Class handles a Circular Buffer.

Defines

- #define DEBUG_CIRCULAR_BUFFER (0)

7.2.1 Define Documentation

7.2.1.1 #define DEBUG_CIRCULAR_BUFFER (0)

7.3 JGTL_CircularBufferInterface.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::CircularBufferInterface< Data >

  The CircularBufferInterface Class handles a Circular Buffer.

Defines

- #define DEBUG_CIRCULAR_BUFFER_INTERFACE (0)

7.3.1 Define Documentation

7.3.1.1 #define DEBUG_CIRCULAR_BUFFER_INTERFACE (0)

7.4 JGTL_CommandLineParser.h File Reference

```
#include <map>
#include <string>
#include <vector>
```

Namespaces

- namespace JGTL

Classes

- struct JGTL::CCmdParam
- class JGTL::CommandLineParser

Defines

- #define StringType std::string

Typedefs

- typedef std::map< StringType, CCmdParam > JGTL::_CommandLineParser

7.4.1 Define Documentation

7.4.1.1 #define StringType std::string

7.5 JGTL_DataManager.h File Reference

```
#include <cstdlib>
#include <cctype>
#include <string>
#include <algorithm>
#include <map>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::DataManager< Data >

Defines

- #define DEBUG_DATA_MANAGER (0)

7.5.1 Define Documentation

7.5.1.1 #define DEBUG_DATA_MANAGER (0)

7.6 JGTL_DataPool_delete.h File Reference

```
#include <memory>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::DataPool< Data >

Defines

- #define DEBUG_DATA_POOL (0)

7.6.1 Define Documentation

7.6.1.1 #define DEBUG_DATA_POOL (0)

7.7 JGTL_DynamicCircularBuffer.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
#include "JGTL_CircularBufferInterface.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::DynamicCircularBuffer< Data >

  The DynamicCircularBuffer Class handles a Circular Buffer.

Defines

- #define DEBUG_DYNAMIC_CIRCULAR_BUFFER (0)

7.7.1 Define Documentation

7.7.1.1 #define DEBUG_DYNAMIC_CIRCULAR_BUFFER (0)

7.8 JGTL_DynamicPoolMap.h File Reference

```
#include "JGTL_MapInterface.h"
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::DynamicPoolMap< Key, Data >

  The DynamicPoolMap Class is a resizable array-based associative map structure.

Defines

- #define DEBUG_DYNAMIC_POOL_MAP (0)

7.8.1 Define Documentation

7.8.1.1 #define DEBUG_DYNAMIC_POOL_MAP (0)

7.9 JGTL_DynamicPoolSet.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
#include "JGTL_SetInterface.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::DynamicPoolSet< Data >

Defines

- #define DEBUG_DYNAMIC_POOL_SET (0)

7.9.1 Define Documentation

7.9.1.1 #define DEBUG_DYNAMIC_POOL_SET (0)

7.10 JGTL_FloatingUnits.h File Reference

```
#include <iostream>
#include <complex>
#include <sstream>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR >

Typedefs

- typedef unsigned long long JGTL::units_internal_ulong

Functions

- template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR>
  std::ostream & JGTL::operator<< (std::ostream &stream, const FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR > &d)
- template<class ValueType, units_internal_ulong SCALE_NUMERATOR, units_internal_ulong SCALE_DENOMINATOR>
  std::istream & JGTL::operator>> (std::istream &stream, FloatingUnits< ValueType, SCALE_NUMERATOR, SCALE_DENOMINATOR > &d)

7.11 JGTL_HexTree.h File Reference

```
#include <iostream>
#include <string>
#include "Vector4.h"
#include "boost/pool/pool.hpp"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::HexTreeNode< T >
- class JGTL::HexTreeStub< T >
- class JGTL::HexTreeBranch< T >
- class JGTL::HexTree< T >

7.12 JGTL_Index2.h File Reference

Namespaces

- namespace JGTL

Classes

- class JGTL::Index2

7.13 JGTL_Index3.h File Reference

```
#include <algorithm>
#include <iostream>
#include <cmath>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Index3
- class JGTL::RectangleIndex3

Functions

- ostream & JGTL::operator<< (ostream &stream, const Index3 &d)
- istream & JGTL::operator>> (istream &stream, Index3 &d)
- ostream & JGTL::operator<< (ostream &stream, const RectangleIndex3 &d)
- istream & JGTL::operator>> (istream &stream, RectangleIndex3 &d)

7.14 JGTL_IntegralUnits.h File Reference

```
#include <iostream>
#include <complex>
#include <sstream>
```

Namespaces

- namespace JGTL

Classes

- struct JGTL::IF< condition, Then, Else >
- struct JGTL::JGTL::IF< false, Then, Else >
- struct JGTL::STATIC_MOD< Type, a, b >
- struct JGTL::TYPEIF< Type, condition, Then, Else >
- struct JGTL::JGTL::TYPEIF< Type, false, Then, Else >
- struct JGTL::IntegralUnitsGCD< i, j >
- struct JGTL::JGTL::IntegralUnitsGCD< 1, j >
- struct JGTL::JGTL::IntegralUnitsGCD< i, 1 >
- struct JGTL::JGTL::IntegralUnitsGCD< 1, 1 >
- struct JGTL::JGTL::IntegralUnitsGCD< 0, j >
- struct JGTL::JGTL::IntegralUnitsGCD< i, 0 >
- struct JGTL::JGTL::IntegralUnitsGCD< 0, 0 >
- class JGTL::IntegralUnits< ValueType, SCALE, USEGCD >

Functions

- template<class T>
  units_internal_ulong JGTL::GCD (T a, T b)
- template<>
  units_internal_ulong JGTL::GCD (double a, double b)
- template<>
  units_internal_ulong JGTL::GCD (float a, float b)
- template<class ValueType, ValueType SCALE, bool USEGCD>
  std::ostream & JGTL::operator<< (std::ostream &stream, const IntegralUnits< ValueType, SCALE, USEGCD > &d)
- template<class ValueType, ValueType SCALE, bool USEGCD>
  std::istream & JGTL::operator>> (std::istream &stream, IntegralUnits< ValueType, SCALE, USEGCD > &d)

7.15 JGTL_InterpolatedValue.h File Reference

Namespaces

- namespace JGTL

Classes

- class JGTL::InterpolatedValue< T >

  The InterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff;.

Functions

- template<class T>
  std::ostream & JGTL::operator<< (std::ostream &stream, const InterpolatedValue< T > &d)
- template<class T>
  std::istream & JGTL::operator>> (std::istream &stream, InterpolatedValue< T > &d)

7.16 JGTL_LocatedException.h File Reference

```
#include <string>
#include <sstream>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::LocatedException

  This class handles throwing exceptions which include the file and line number.

Defines

- #define CREATE_PAUSE(X) {cout << X << "\nPress enter to continue" << endl;string line;getline(cin,line);}
- #define CREATE_LOCATEDEXCEPTION_INFO(X) JGTL::LocatedException((X) ,__FILE__,__LINE__);

7.16.1 Define Documentation

7.16.1.1 #define CREATE_LOCATEDEXCEPTION_INFO(X) JGTL::LocatedException((X) ,__FILE__,__LINE__);

7.16.1.2 #define CREATE_PAUSE(X) {cout << X << "\nPress enter to continue" << endl;string line;getline(cin,line);}

7.17 JGTL_MapInterface.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::MapInterface< Key, Data >
  This class acts as a base class for the Map construct.

Defines

- #define DEBUG_MAP_INTERFACE (0)

7.17.1 Define Documentation

7.17.1.1 #define DEBUG_MAP_INTERFACE (0)

7.18 JGTL_PolyVariant.h File Reference

```
#include <iostream>
#include "JGTL_LocatedException.h"
#include "JGTL_Variant.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::PolyVariant< BaseClass, Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >

7.19 JGTL_PoolMap_delete.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::PoolMap< Key, Data >

Defines

- #define DEBUG_POOL_MAP (0)

7.19.1 Define Documentation

7.19.1.1 #define DEBUG_POOL_MAP (0)

7.20 JGTL_Quadratic.h File Reference

Namespaces

- namespace JGTL

Classes

- class JGTL::QuadraticSolution< T >

Functions

- template<class T, class TT, class TTT, class TTTT>
  QuadraticSolution< T > JGTL::solveQuadratic (TT a, TTT b, TTTT c)

7.21 JGTL_QuadTree.h File Reference

```
#include <iostream>
#include <string>
#include "JGTL_Vector2.h"
#include "boost/pool/pool.hpp"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::QuadTreeNode< T >
- class JGTL::QuadTreeStub< T >
- class JGTL::QuadTreeBranch< T >
- class JGTL::QuadTree< T >

7.22 JGTL_QuickProf.h File Reference

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <math.h>
#include <algorithm>
#include "JGTL_LocatedException.h"
#include "JGTL_DynamicPoolMap.h"
#include "JGTL_Singleton.h"
#include <sys/time.h>
```

Namespaces

- namespace JGTL

Classes

- struct JGTL::ProfileBlock
- class JGTL::Clock
- class JGTL::Profiler

  A singleton class that manages timing for a set of profiling blocks.

- class JGTL::ProfileBlockHandler

Defines

- #define PROFILER JGTL::Profiler::getInstance()

Enumerations

- enum JGTL::TimeFormat { JGTL::SECONDS, JGTL::MILLISECONDS, JGTL::MICROSECONDS, JGTL::PERCENT }

  A set of ways to represent timing results.

7.22.1 Define Documentation

7.22.1.1 #define PROFILER JGTL::Profiler::getInstance()

Use this macro to access the profiler singleton. For example: PROFILER.init(); ... PROFILER.beginBlock("foo"); foo(); PROFILER.endBlock("foo");

7.23 JGTL_Ray2.h File Reference

```
#include "JGTL_Vector2.h"
#include <utility>
#include <cmath>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Ray2< T >

  This class handles 2D Rays and Line Segments.

Enumerations

- enum JGTL::IntersectionState { JGTL::IS_NONE, JGTL::IS_ONE, JGTL::IS_-INFINITE }

7.24 JGTL_Ray3.h File Reference

```
#include "JGTL_Vector3.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Ray3< T >

  This class handles 3D Rays and Line Segments.

7.25 JGTL_Rectangle3.h File Reference

```
#include <algorithm>
#include <iostream>
#include <cmath>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Rectangle3< T >

7.26 JGTL_Serialization.h File Reference

```
#include "JGTL_LocatedException.h"
#include <iostream>
#include <fstream>
#include <string>
#include <string.h>
```

Namespaces

- namespace JGTL

Typedefs

- typedef unsigned char JGTL::uchar

Functions

- template<class Data>
  void JGTL::packBuffer (uchar *&buffer, int &bufferSize, const Data &data)
- template<class Data>
  void JGTL::unpackBuffer (uchar *&buffer, int &bufferSize, Data &data)
- template<class Data>
  void JGTL::packBufferStack (uchar *&buffer, int &bufferSize, const Data &data)
- template<class Data>
  void JGTL::unpackBufferStack (uchar *&buffer, int &bufferSize, Data &data)
- void JGTL::packBufferString (uchar *&buffer, int &bufferSize, const char *s)
- void JGTL::packBufferString (uchar *&buffer, int &bufferSize, const std::string &s)
- void JGTL::unpackBufferString (uchar *&buffer, int &bufferSize, string &s)

7.27 JGTL_SetInterface.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::SetInterface< Data >

  This class acts as a base class for the Set construct.

Defines

- #define DEBUG_SET_INTERFACE (0)

7.27.1 Define Documentation

7.27.1.1 #define DEBUG_SET_INTERFACE (0)

7.28 JGTL_Singleton.h File Reference

```
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Singleton< Type >

  This class handles Singletons (Global Single-Instance Classes).

7.29 JGTL_SortedList_delete.h File Reference

```
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::SortedList< Data >

Defines

- #define DEBUG_SORTED_LIST (0)

7.29.1 Define Documentation

7.29.1.1 #define DEBUG_SORTED_LIST (0)

7.30 JGTL_StackCircularBuffer.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
#include "JGTL_CircularBufferInterface.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::StackCircularBuffer< Data, MAX_ELEMENTS >
  The StackCircularBuffer Class handles a Circular Buffer.

Defines

- #define DEBUG_STACK_CIRCULAR_BUFFER (0)

7.30.1 Define Documentation

7.30.1.1 #define DEBUG_STACK_CIRCULAR_BUFFER (0)

7.31 JGTL_StackMap.h File Reference

```
#include "JGTL_MapInterface.h"
#include <utility>
#include <stdexcept>
#include <cstdlib>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::StackMap< Key, Data, MAX_ELEMENTS >
  The StackMap Class is a fixed, array-based, sorted key structure.

Defines

- #define DEBUG_STACK_MAP (0)

7.31.1 Define Documentation

7.31.1.1 #define DEBUG_STACK_MAP (0)

7.32 JGTL_StackSet.h File Reference

```
#include "JGTL_SetInterface.h"
#include <utility>
#include <stdexcept>
#include <cstdlib>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::StackSet< Data, MAX_ELEMENTS >

Defines

- #define DEBUG_STACK_SET (0)

7.32.1 Define Documentation

7.32.1.1 #define DEBUG_STACK_SET (0)

7.33 JGTL_StringConverter.h File Reference

```
#include <string>
#include <sstream>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Functions

- template<typename T>
  T JGTL::stringTo (const std::string &s)
- template<typename T>
  void JGTL::stringTo (const std::string &s, T &x)
- template<typename T>
  std::string JGTL::toString (const T &x)
- template<class T>
  T JGTL::getIndexFromName (const char *name, const char **names, T numNames)
- template<class T>
  T JGTL::getIndexFromName (const std::string &name, const char **names, T numNames)

7.34 JGTL_TreeList.h File Reference

Namespaces

- namespace JGTL

Classes

- class JGTL::TreeListNode< Data >
- class JGTL::TreeList< Data >

7.35 JGTL_UnorderedDynamicPoolMap.h File Reference

```
#include "MapInterface.h"
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::DynamicPoolMap< Key, Data >

  The DynamicPoolMap Class is a resizable array-based associative map structure.

Defines

- #define DEBUG_DYNAMIC_POOL_MAP (0)

7.35.1 Define Documentation

7.35.1.1 #define DEBUG_DYNAMIC_POOL_MAP (0)

7.36 JGTL_UnorderedMapInterface.h File Reference

```
#include <utility>
#include <cstdlib>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::BinaryTreeNode
- class JGTL::MapInterface< Key, Data >

  This class acts as a base class for the Map construct.

Defines

- #define DEBUG_MAP_INTERFACE (0)

7.36.1 Define Documentation

7.36.1.1 #define DEBUG_MAP_INTERFACE (0)

7.37 JGTL_Variant.h File Reference

```
#include <iostream>
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- struct JGTL::TYPEIF< Type, condition, Then, Else >
- struct JGTL::JGTL::TYPEIF< Type, false, Then, Else >
- struct JGTL::STATIC_MAX_SIZE< One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, Two, Three, Four, Five, Six, Seven, Eight, Nine >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, Two, Three, Four, Five, Six, Seven, Eight >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, Two, Three, Four, Five, Six, Seven >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, Two, Three, Four, Five, Six >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, Two, Three, Four, Five >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, Two, Three, Four >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, Two, Three >
- struct JGTL::JGTL::STATIC_MAX_SIZE< One, One, One, One, One, One, One, One, One, Two >
- class JGTL::NullVariantClass
- class JGTL::Variant< Class1, Class2, Class3, Class4, Class5, Class6, Class7, Class8, Class9, Class10 >

7.38 JGTL_Vector2.h File Reference

```
#include <string>
#include <iostream>
#include <cmath>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Vector2< T >
  This class handles 2D Vectors.

Functions

- template<class T>
  std::ostream & JGTL::operator<< (std::ostream &stream, const Vector2< T > &d)
- template<class T>
  std::istream & JGTL::operator>> (std::istream &stream, Vector2< T > &d)
- template<class T, class TT>
  T JGTL::convertVector2 (const TT &other)

7.39 JGTL_Vector3.h File Reference

```
#include <algorithm>
#include <iostream>
#include <cmath>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Vector3< T >

Functions

- template<class T, class TT>
  T JGTL::convertVector3 (const TT &other)
- template<class T>
  std::ostream & JGTL::operator<< (std::ostream &stream, const Vector3< T > &d)
- template<class T>
  std::istream & JGTL::operator>> (std::istream &stream, Vector3< T > &d)

7.40 JGTL_Vector4.h File Reference

```
#include <algorithm>
#include <iostream>
#include <cmath>
```

Namespaces

- namespace JGTL

Classes

- class JGTL::Vector4< T >

Functions

- template<class T, class TT>
  T JGTL::convertVector4 (const TT &other)
- template<class T>
  std::ostream & JGTL::operator<< (std::ostream &stream, const Vector4< T > &d)
- template<class T>
  std::istream & JGTL::operator>> (std::istream &stream, Vector4< T > &d)

7.41 JGTL_WrappedInterpolatedValue.h File Reference

```
#include "JGTL_InterpolatedValue.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::WrappedInterpolatedValue< T >

  The WrappedInterpolatedValue Class handles values which approach a limit using the formula NewValue = actualValue + (potentialValue-actualValue)∗interpolationCoeff; This special instance of an InterpolatedValue is for values which wrap (angles, for example, which wrap around 2∗PI).

Functions

- template<class T>
  std::ostream & JGTL::operator<< (std::ostream &stream, const WrappedInterpolatedValue< T > &d)
- template<class T>
  std::istream & JGTL::operator>> (std::istream &stream, WrappedInterpolatedValue< T > &d)

7.42 JGTL_XorSpace.h File Reference

```
#include <algorithm>
#include <iostream>
#include <cmath>
#include "JGTL_StringConverter.h"
#include "JGTL_LocatedException.h"
```

Namespaces

- namespace JGTL

Classes

- class JGTL::XorSpaceRect< Rectangle, Point >

  This handles a single Xor Rectangle.

- class JGTL::XorSpace< Rectangle, Point >

  This class handles Xor spaces. Think of this as a way to handle things like rectangular doughnuts. A positive space followed by a smaller concentric negative space would represent a doughnut.

Functions

- template<class Rectangle, class Point>
  ostream & JGTL::operator<< (ostream &stream, const XorSpace< Rectangle, Point > &d)
- template<class Rectangle, class Point>
  istream & JGTL::operator>> (istream &stream, XorSpace< Rectangle, Point > &d)

Index