computeMST

Generated by Doxygen 1.8.11

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical 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

Namespace Documentation

4.1 cmst Namespace Reference

Classes

- class Edge2D
- class Graph2D
- class IndexEdge2D

  Edge with start and end point indices in an array.
- class Point2D

  Points in a 2D plane.
- class Stat
- class Timer
- class Window

Enumerations

- enum Menu {
  LOAD, NEW, NEW_4_10, NEW_11_100,
  NEW_101_1000, NEW_1001_5000, NEW_5001_10000, SHOW,
  SHOW_VORONOI, SHOW_DELAUNAY, SHOW_POINT, SHOW_MST,
  SHOW_ST, TEST, TEST_5, TEST_20,
  VALIDATOR, PRINT, QUIT }

  Return values for GLUT menus.

Functions

- int randomInt (int a, int b)
- double randomDouble (double a, double b)
- std::vector< Point2D > TestcaseGenerator (int num_lower_bound=100, int num_upper_bound=500, double x_upper_bound=MAX_X, double y_upper_bound=MAX_Y)

4.1.1 Enumeration Type Documentation

4.1.1.1 enum cmst::Menu

Return values for GLUT menus.

Enumerator

- ***LOAD***
- ***NEW***
- ***NEW_4_10***
- ***NEW_11_100***
- ***NEW_101_1000***
- ***NEW_1001_5000***
- ***NEW_5001_10000***
- ***SHOW***
- ***SHOW_VORONOI***
- ***SHOW_DELAUNAY***
- ***SHOW_POINT***
- ***SHOW_MST***
- ***SHOW_ST***
- ***TEST***
- ***TEST_5***
- ***TEST_20***
- ***VALIDATOR***
- ***PRINT***
- ***QUIT***

4.1.2 Function Documentation

4.1.2.1 double cmst::randomDouble (double *a*, double *b*)

Generate a floating-point number in the range [a, b]

Needs to be improved using other random classes

Here is the caller graph for this function:

4.1.2.2 int cmst::randomInt (int *a*, int *b*)

Generate an integer in the range [a, b]

Needs to be improved using other random classes

Here is the caller graph for this function:

4.1.2.3 std::vector< cmst::Point2D > cmst::TestcaseGenerator (int *num_lower_bound =* `100`, int *num_upper_bound =* `500`, double *x_upper_bound =* `MAX_X`, double *y_upper_bound =* `MAX_Y`)

Generate some random points.

The number of points is generated randomly in the range [num_lower_bound, num_upper_bound], and the x, y coordinates of the points are respectively in the range [0, x_upper_bound] and [0, y_upper_bound].

Here is the call graph for this function:

Here is the caller graph for this function:

Chapter 5

Class Documentation

5.1 cmst::Edge2D Class Reference

Inheritance diagram for cmst::Edge2D:

Collaboration diagram for cmst::Edge2D:

Public Member Functions

- Edge2D ()
- Edge2D (const Point2D &start, const Point2D &end)
- double length () const
- Point2D start () const
- Point2D end () const
- bool operator< (const Edge2D &right) const

    Compares edges by length.
- bool operator== (const Edge2D &right) const

Protected Member Functions

- void swap_points ()

    Swaps the start and end point.

Private Attributes

- Point2D m_start

    Start point.
- Point2D m_end

    End point.
- double m_length

    Length.

Friends

- std::ostream & operator<< (std::ostream &out, const Edge2D &e)

5.1.1 Detailed Description

Stores edges in 2D plane.

The start and end points are stored in the edge.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 cmst::Edge2D::Edge2D ( ) `[inline]`

5.1.2.2 cmst::Edge2D::Edge2D (const **Point2D** & *start,* const **Point2D** & *end*) `[inline]`

Constructor

Calculates the length.

Here is the call graph for this function:

5.1.3 Member Function Documentation

5.1.3.1 **Point2D** cmst::Edge2D::end ( ) const `[inline]`

Returns the end point.

Returns

end point

Here is the caller graph for this function:

5.1.3.2 double cmst::Edge2D::length ( ) const `[inline]`

Returns the length of the edge.

Returns

length of the edge

Here is the caller graph for this function:

5.1.3.3 bool cmst::Edge2D::operator< (const **Edge2D** & *right*) const [inline]

Compares edges by length.

5.1.3.4 bool cmst::Edge2D::operator== (const **Edge2D** & *right*) const [inline]

Compares cmst::Edge2D by start point and end point.

Take the cmst::Edge2D as undirected.

5.1.3.5 **Point2D** cmst::Edge2D::start ( ) const [inline]

Returns the start point.

Returns

start point

Here is the caller graph for this function:

5.1.3.6 void cmst::Edge2D::swap_points ( ) [inline], [protected]

Swaps the start and end point.

Here is the caller graph for this function:

5.1.4 Friends And Related Function Documentation

5.1.4.1 std::ostream& operator<< (std::ostream & *out*, const **Edge2D** & *e*) [friend]

Prints information about the edge.

Prints the length, start point and end point.

5.1.5 Member Data Documentation

5.1.5.1 **Point2D** cmst::Edge2D::m_end [private]

End point.

5.1.5.2 double cmst::Edge2D::m_length [private]

Length.

5.1.5.3 Point2D cmst::Edge2D::m_start [private]

Start point.

5.2 cmst::Graph2D Class Reference

Collaboration diagram for cmst::Graph2D:

Classes

- struct ST

  Store a spanning tree of the graph.

Public Member Functions

- Graph2D (std::vector< Point2D > &points)
- double Kruskal ()
- double naiveKruskal ()
- void drawPoint ()

  Use GLUT to draw the points in the graph.
- void drawDelaunay ()

  Use GLUT to draw the Delaunay Diagram of the graph.
- void drawMST ()

  Use GLUT to draw the MST computed by Kruskal().
- void drawST ()

  Use GLUT to draw the ST computed by naiveKruskal().
- bool print (std::string file="graph.txt")

  Print the graph information to file.
- void changeSTDisplay (int direc)
- void printSTInfo ()

  Print the information of the current spanning tree displayed.
- double mstLength ()
- int delaunayTime () const

  Return the time used for computing Delaunay diagram.
- int mstTime ()
- int graphConstructTime () const
- int pointNum () const

  Return the number of points in this graph.
- int edgeNum () const

  Return the number of edges in the Delaunay diagram.
- bool validateDone () const

  Return if the MST has been validated.

Protected Member Functions

- int findFather (int x)

  Find the father of x in the Union-find Sets structure.
- void initFather ()

  Initializes the father array for Union-find Sets structure.

Protected Attributes

- std::vector< int > father

  Father array for Union-find Sets structure.
- std::vector< Point2D > m_points

  Points in the graph.
- std::vector< IndexEdge2D > m_delaunayEdge

  Delaunay edges of the graph.
- std::vector< IndexEdge2D > m_MSTEdge

  MST edges of the graph.
- std::vector< IndexEdge2D > m_edges

  All possible edges in the graph.
- std::vector< std::vector< int > > m_graph

  Adjacency list of the Delaunay diagram of the graph.
- Delaunay m_delaunay

  CGAL data structure for storing and computing a Delaunay diagram.
- std::vector< ST > m_ST

  Spanning trees of the graph.

Private Attributes

- bool m_mstDone

  If Kruskal() has been called.
- bool m_validateDone

  If naiveKruskal() has been called.
- double m_mstLength

  Length of the MST.
- int m_delaunayTime

  Time used for computing the Delaunay diagram.
- int m_mstTime

  Time used for computing the MST.
- int m_graphConstructTime

  Time used for reconstructing the graph.
- int m_displaySTNum

5.2.1 Constructor & Destructor Documentation

5.2.1.1 cmst::Graph2D::Graph2D (std::vector< Point2D > & *points*)

Constructor which does everything.

- Compute Delaunay graph
- Reconstruct the graph

Here is the call graph for this function:

5.2.2 Member Function Documentation

5.2.2.1 void cmst::Graph2D::changeSTDisplay (int *direc*) `[inline]`

Change the displaying spanning tree

To-do: calculate the top k spanning trees

Here is the caller graph for this function:

5.2.2.2 int cmst::Graph2D::delaunayTime ( ) const `[inline]`

Return the time used for computing Delaunay diagram.

Here is the caller graph for this function:

5.2.2.3 void cmst::Graph2D::drawDelaunay ( )

Use GLUT to draw the Delaunay Diagram of the graph.

Here is the caller graph for this function:

5.2.2.4 void cmst::Graph2D::drawMST ( )

Use GLUT to draw the MST computed by Kruskal().

Here is the caller graph for this function:

5.2.2.5 void cmst::Graph2D::drawPoint ( )

Use GLUT to draw the points in the graph.

Here is the caller graph for this function:

5.2.2.6 void cmst::Graph2D::drawST ( )

Use GLUT to draw the ST computed by naiveKruskal().

Here is the call graph for this function:

Here is the caller graph for this function:

5.2.2.7 int cmst::Graph2D::edgeNum ( ) const `[inline]`

Return the number of edges in the Delaunay diagram.

Here is the caller graph for this function:

5.2.2.8 int cmst::Graph2D::findFather (int *x*) [protected]

Find the father of x in the Union-find Sets structure.

Here is the caller graph for this function:

5.2.2.9 int cmst::Graph2D::graphConstructTime ( ) const [inline]

Return the time used for reconstructing the graph.

When using CGAL library, the internal data structure is different from the one used in this program. So you need some conversion.

Here is the caller graph for this function:

5.2.2.10 void cmst::Graph2D::initFather ( ) [protected]

Initializes the father array for Union-find Sets structure.

Here is the caller graph for this function:

5.2.2.11 double cmst::Graph2D::Kruskal ( )

The Kruskal algorithm for finding the minimal spanning tree.

Use the CGAL computed Delaunay Diagram.

Returns

The length of the MST.

Here is the call graph for this function:

Here is the caller graph for this function:

5.2.2.12 double cmst::Graph2D::mstLength ( ) [inline]

Return the length of the MST

Returns

Length of MST

Here is the call graph for this function:

Here is the caller graph for this function:

5.2.2.13 int cmst::Graph2D::mstTime ( ) `[inline]`

Return the time used for computing MST, using Kruskal's algorithm

Here is the call graph for this function:

Here is the caller graph for this function:

5.2.2.14 double cmst::Graph2D::naiveKruskal ( )

The naive Kruskal algorithm.

Construct all the edges in the graph, then run Kruskal.

Returns

The length of the MST.

Here is the call graph for this function:

Here is the caller graph for this function:

5.2.2.15 int cmst::Graph2D::pointNum ( ) const `[inline]`

Return the number of points in this graph.

Here is the caller graph for this function:

5.2.2.16 bool cmst::Graph2D::print (std::string *file* = `"graph.txt"`)

Print the graph information to file.

Here is the caller graph for this function:

5.2.2.17 void cmst::Graph2D::printSTInfo ( ) `[inline]`

Print the information of the current spanning tree displayed.

Here is the caller graph for this function:

5.2.2.18 bool cmst::Graph2D::validateDone ( ) const `[inline]`

Return if the MST has been validated.

Here is the caller graph for this function:

5.2.3 Member Data Documentation

5.2.3.1 std::vector<int> cmst::Graph2D::father [protected]

Father array for Union-find Sets structure.

5.2.3.2 Delaunay cmst::Graph2D::m_delaunay [protected]

CGAL data structure for storing and computing a Delaunay diagram.

5.2.3.3 std::vector<IndexEdge2D> cmst::Graph2D::m_delaunayEdge [protected]

Delaunay edges of the graph.

5.2.3.4 int cmst::Graph2D::m_delaunayTime [private]

Time used for computing the Delaunay diagram.

5.2.3.5 int cmst::Graph2D::m_displaySTNum [private]

5.2.3.6 std::vector<IndexEdge2D> cmst::Graph2D::m_edges [protected]

All possible edges in the graph.

5.2.3.7 std::vector<std::vector<int> > cmst::Graph2D::m_graph [protected]

Adjacency list of the Delaunay diagram of the graph.

5.2.3.8 int cmst::Graph2D::m_graphConstructTime [private]

Time used for reconstructing the graph.

5.2.3.9 bool cmst::Graph2D::m_mstDone [private]

If Kruskal() has been called.

5.2.3.10 std::vector<IndexEdge2D> cmst::Graph2D::m_MSTEdge [protected]

MST edges of the graph.

5.2.3.11 double cmst::Graph2D::m_mstLength [private]

Length of the MST.

5.2.3.12 int cmst::Graph2D::m_mstTime [private]

Time used for computing the MST.

5.2.3.13 std::vector<Point2D> cmst::Graph2D::m_points [protected]

Points in the graph.

5.2.3.14 std::vector<ST> cmst::Graph2D::m_ST [protected]

Spanning trees of the graph.

5.2.3.15 bool cmst::Graph2D::m_validateDone [private]

If naiveKruskal() has been called.

5.3 cmst::IndexEdge2D Class Reference

Edge with start and end point indices in an array.

Inheritance diagram for cmst::IndexEdge2D:

Collaboration diagram for cmst::IndexEdge2D:

Public Member Functions

- IndexEdge2D ()
- IndexEdge2D (Point2D p1, Point2D p2, int index1, int index2)
- int startIndex () const

  The index of the starting point.
- int endIndex () const

  The index of the end point.
- bool operator< (const IndexEdge2D &right) const

  Compares edges by length.
- bool operator> (const IndexEdge2D &right) const

  Compares edges by length.

Private Attributes

- int m_index [2]

  Indices of the end points.

Friends

- std::ostream & operator<< (std::ostream &str, const IndexEdge2D &e)

Additional Inherited Members

5.3.1 Detailed Description

Edge with start and end point indices in an array.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 cmst::IndexEdge2D::IndexEdge2D ( ) `[inline]`

5.3.2.2 cmst::IndexEdge2D::IndexEdge2D (**Point2D** *p1*, **Point2D** *p2*, int *index1*, int *index2*) `[inline]`

Store the edge as an undirected one. The two end points will be sorted according to indices.

Here is the call graph for this function:

5.3.3 Member Function Documentation

5.3.3.1 int cmst::IndexEdge2D::endIndex ( ) const `[inline]`

The index of the end point.

5.3.3.2 bool cmst::IndexEdge2D::operator< (const **IndexEdge2D** & *right*) const `[inline]`

Compares edges by length.

Here is the call graph for this function:

5.3.3.3 bool cmst::IndexEdge2D::operator> (const **IndexEdge2D** & *right*) const `[inline]`

Compares edges by length.

Here is the call graph for this function:

5.3.3.4 int cmst::IndexEdge2D::startIndex ( ) const `[inline]`

The index of the starting point.

5.3.4 Friends And Related Function Documentation

5.3.4.1 std::ostream& operator<< (std::ostream & *str*, const IndexEdge2D & *e*) [friend]

5.3.5 Member Data Documentation

5.3.5.1 int cmst::IndexEdge2D::m_index[2] [private]

Indices of the end points.

5.4 cmst::Point2D Class Reference

Points in a 2D plane.

Collaboration diagram for cmst::Point2D:

Public Member Functions

- Point2D (double x=0.0, double y=0.0)

  Constructor.
- Point2D (const Point2D &other)

  Copy-constructor.
- double x () const
- double y () const
- bool operator< (const Point2D &right) const

  Compare points by x coordinates and y coordinates.
- bool operator== (const Point2D &right) const

Private Attributes

- double m_x

  x coordinate
- double m_y

  y coordinate

Friends

- std::ostream & operator<< (std::ostream &out, const Point2D &p)

5.4.1 Detailed Description

Points in a 2D plane.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 cmst::Point2D::Point2D (double *x* = `0.0`, double *y* = `0.0`) `[inline]`

Constructor.

5.4.2.2 cmst::Point2D::Point2D (const Point2D & *other*) `[inline]`

Copy-constructor.

5.4.3 Member Function Documentation

5.4.3.1 bool cmst::Point2D::operator< (const Point2D & *right*) const `[inline]`

Compare points by x coordinates and y coordinates.

5.4.3.2 bool cmst::Point2D::operator== (const Point2D & *right*) const `[inline]`

Compare if two points are the same.

Some epsilon loss is allowed.

5.4.3.3 double cmst::Point2D::x ( ) const `[inline]`

Returns

x

Here is the caller graph for this function:

5.4.3.4 double cmst::Point2D::y ( ) const `[inline]`

Returns

y

Here is the caller graph for this function:

5.4.4 Friends And Related Function Documentation

5.4.4.1 std::ostream& operator<< (std::ostream & *out*, const Point2D & *p*) `[friend]`

5.4.5 Member Data Documentation

5.4.5.1 double cmst::Point2D::m_x `[private]`

x coordinate

5.4.5.2 double cmst::Point2D::m_y [private]

y coordinate

5.5 cmst::Graph2D::ST Struct Reference

Store a spanning tree of the graph.

Collaboration diagram for cmst::Graph2D::ST:

Public Member Functions

- ST (std::vector< IndexEdge2D > edges=std::vector< IndexEdge2D >(), int stTime=0, double length=0.0)

  Constructor.

Public Attributes

- std::vector< IndexEdge2D > m_edges

  Edges of the spanning tree.
- int m_stTime

  Time used to compute the spanning tree.
- double m_length

  Length of the spanning tree.

5.5.1 Detailed Description

Store a spanning tree of the graph.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 cmst::Graph2D::ST::ST (std::vector< IndexEdge2D > *edges* = std::vector<IndexEdge2D>(), int *stTime* = 0, double *length* = 0.0) [inline]

Constructor.

5.5.3 Member Data Documentation

5.5.3.1 std::vector<IndexEdge2D> cmst::Graph2D::ST::m_edges

Edges of the spanning tree.

5.5.3.2 double cmst::Graph2D::ST::m_length

Length of the spanning tree.

5.5.3.3 int cmst::Graph2D::ST::m_stTime

Time used to compute the spanning tree.

5.6 cmst::Stat Class Reference

Collaboration diagram for cmst::Stat:

Public Member Functions

- Stat ()
- void record (double data)

  Record a datum and update m_min, m_max.
- double min () const
- double max () const
- int count () const
- double mean ()
- double standardDeviation ()
- std::string print ()

Private Attributes

- double m_min

  Minimum of the data.
- double m_max

  Maximum of the data.
- double m_mean

  Average of the data.
- double m_standardDeviation

  Standard deviation of the data.
- std::vector$<$ double $>$ m_data

  Data.

5.6.1 Detailed Description

Simple statistics.

Including:

- Minimum
- Maximum
- Mean
- Standard Deviation

5.6.2 Constructor & Destructor Documentation

5.6.2.1 cmst::Stat::Stat ( ) [inline]

Constructor

Set m_max to DOUBLE_MIN and m_min to DOUBLE_MAX

5.6.3 Member Function Documentation

5.6.3.1 int cmst::Stat::count ( ) const [inline]

Return the number of recorded data.

Returns

The number of recorded data

Return values

0	If no data has been recorded.

5.6.3.2 double cmst::Stat::max ( ) const [inline]

Return the maximum of recorded data.

Returns

Maximum of recorded data

Return values

0.↩ 0	If no data has been recorded

5.6.3.3 double cmst::Stat::mean ( ) [inline]

Return the mean of all data.

Returns

Mean of all data

Return values

0.↩ 0	If no data has been recorded.

Here is the caller graph for this function:

5.6.3.4 double cmst::Stat::min ( ) const [inline]

Return the minimum of recorded data.

Returns

Minimum of recorded data

Return values

0.↩ 0	If no data has been recorded

5.6.3.5 std::string cmst::Stat::print ( ) [inline]

Print the information of the statistic.

- Average
- Maximum
- Minimum
- Standard deviation

Here is the call graph for this function:

Here is the caller graph for this function:

5.6.3.6 void cmst::Stat::record (double *data*) [inline]

Record a datum and update m_min, m_max.

Here is the caller graph for this function:

5.6.3.7 double cmst::Stat::standardDeviation ( ) [inline]

Return the standard deviation of all data.

Returns

Standard deviation of all data

Return values

0.↩ 0	If no data has been recorded.

Here is the call graph for this function:

Here is the caller graph for this function:

5.6.4 Member Data Documentation

5.6.4.1 std::vector<double> cmst::Stat::m_data [private]

Data.

5.6.4.2 double cmst::Stat::m_max [private]

Maximum of the data.

5.6.4.3 double cmst::Stat::m_mean [private]

Average of the data.

5.6.4.4 double cmst::Stat::m_min [private]

Minimum of the data.

5.6.4.5 double cmst::Stat::m_standardDeviation [private]

Standard deviation of the data.

5.7 cmst::Window::Test Struct Reference

Collaboration diagram for cmst::Window::Test:

Public Member Functions

- Test ()

Public Attributes

- bool m_displayTest

  Whether a test has been generated and displayed.
- int m_displayTestNum

  The number of graphs in the test.
- std::vector< Graph2D > m_testGraphs

  The graphs generated in the test.
- Stat m_delaunayTimeStat

  Statistics of Delaunay Diagram computational time.
- Stat m_graphConstructTimeStat

  Statistics of graph re-construction time.
- Stat m_mstTimeStat

  Statistics of MST computational time.

5.7.1 Detailed Description

Stores information of a test.

Including the generated graphs and statistics of times.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 cmst::Window::Test::Test ( ) `[inline]`

Constructor

No test is generated in initialization.

5.7.3 Member Data Documentation

5.7.3.1 Stat cmst::Window::Test::m_delaunayTimeStat

Statistics of Delaunay Diagram computational time.

5.7.3.2 bool cmst::Window::Test::m_displayTest

Whether a test has been generated and displayed.

5.7.3.3 int cmst::Window::Test::m_displayTestNum

The number of graphs in the test.

5.7.3.4 Stat cmst::Window::Test::m_graphConstructTimeStat

Statistics of graph re-construction time.

5.7.3.5 Stat cmst::Window::Test::m_mstTimeStat

Statistics of MST computational time.

5.7.3.6 std::vector< Graph2D > cmst::Window::Test::m_testGraphs

The graphs generated in the test.

5.8 cmst::Timer Class Reference

Collaboration diagram for cmst::Timer:

Public Member Functions

- Timer ()

  Constructor. Begin the timer.
- int time ()
- void reset ()

  Reset the timer.

Private Attributes

- int m_begin

  The time at construction or reset.

5.8.1 Detailed Description

A class for timing.

Uses simple clock() function.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 cmst::Timer::Timer ( ) [inline]

Constructor. Begin the timer.

5.8.3 Member Function Documentation

5.8.3.1 void cmst::Timer::reset ( ) [inline]

Reset the timer.

Here is the caller graph for this function:

5.8.3.2 int cmst::Timer::time ( ) [inline]

Return the time since construction or reset.

The time unit is ms.

Here is the caller graph for this function:

5.8.4 Member Data Documentation

5.8.4.1 int cmst::Timer::m_begin [private]

The time at construction or reset.

5.9 cmst::Window Class Reference

Collaboration diagram for cmst::Window:

Classes

- struct Test

Public Member Functions

- Graph2D ∗ curGraph ()

  Returns a pointer to the graph in display currently.
- void resetCurGraph (std::vector< Point2D > &points)
- void resetCurGraph ()
- void resetCurGraph (int n)
- void resetCurGraph (int low, int hi)
- bool load ()
- void resetShowDelaunay ()

  Change whether the Delaunay diagram is to be drawn to the GLUT window.
- void resetShowPoint ()

  Change whether the points are to be drawn to the GLUT window.
- void resetShowMST ()

  Change whether the MST is to be drawn to the GLUT window.
- void resetShowST ()

  Change whether the STs are to be drawn to the GLUT window.
- void resetWidth (int width)

  Record the width of current GLUT window.
- void resetHeight (int height)

  Record the height of current GLUT window.
- int width () const
- int height () const
- void draw ()
- void printCurInfo ()

- bool displayTest () const
- void generateTest (int n)
- void printTestInfo ()
- int testDisplayNum () const
- void changeTestDisplay (int direc)
- bool printToFile ()

  Print the information of the current graph to file graph.txt.
- void changeMSTDisplay (int direc)

  Change the MST that is being displayed.
- void printSTInfo ()

  Print information of the current ST to console.
- void runValidate ()

  Run the validator for small graphs.

Static Public Member Functions

- static Window ∗ instance ()

Protected Attributes

- struct cmst::Window::Test m_test

  The test.

Private Member Functions

- Window ()

  Constructor.
- Window (const Window &)

  Private copy-constructor.

Private Attributes

- Graph2D ∗ m_curGraph

  The pointer to the graph that is being displayed.
- bool m_showDelaunay

  Whether the Delaunay iagram is to be drawn.
- bool m_showMST

  Whether the MST is to be drawn.
- bool m_showST

  Whether the MST is to be drawn.
- bool m_showPoint

  Whether the points are to be drawn.
- int m_width

  The width of current GLUT window.
- int m_height

  The height of current GLUT window.

Static Private Attributes

- static Window ∗ m_instance = NULL

  The pointer to an instance of cmst::Window.

5.9.1 Detailed Description

Manipulates the window.

Uses Singleton pattern.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 cmst::Window::Window ( ) `[inline]`, `[private]`

Constructor.

Here is the caller graph for this function:

5.9.2.2 cmst::Window::Window (const Window &) `[private]`

Private copy-constructor.

5.9.3 Member Function Documentation

5.9.3.1 void cmst::Window::changeMSTDisplay (int *direc*) `[inline]`

Change the MST that is being displayed.

Here is the call graph for this function:

5.9.3.2 void cmst::Window::changeTestDisplay (int *direc*) `[inline]`

If a test is being displayed, then changes the graph in the test that is being displayed.

If no test has been generated, does nothing.

Parameters

direc	If negative, display the last graph (if there is one); if positive, display the next graph (if there is one).

5.9.3.3 Graph2D∗ cmst::Window::curGraph ( ) `[inline]`

Returns a pointer to the graph in display currently.

Here is the call graph for this function:

5.9.3.4 bool cmst::Window::displayTest ( ) const [inline]

Returns if a test has been generated

Returns

If a test has been generated

Here is the call graph for this function:

5.9.3.5 void cmst::Window::draw ( )

Draws the current graph

- Points: definitely
- Delaunay Diagram: change whether to draw it by Window::resetShowDelaunay()
- MST: definitely
- Other spanning trees: draws one of them

Here is the call graph for this function:

Here is the caller graph for this function:

5.9.3.6 void cmst::Window::generateTest (int *n*)

Generates a test of n graphs and display the first one.

Parameters

n	The number of graphs in the test to be generated

Here is the call graph for this function:

Here is the caller graph for this function:

5.9.3.7 int cmst::Window::height ( ) const [inline]

Return the height of current GLUT window.

Returns

The height of current GLUT window

Here is the call graph for this function:

Here is the caller graph for this function:

5.9.3.8 static Window* cmst::Window::instance ( ) [inline], [static]

Return the pointer to the instance of cmst::Window class.

Returns

the pointer to the instance

Here is the call graph for this function:

5.9.3.9 bool cmst::Window::load ( )

Here is the call graph for this function:

Here is the caller graph for this function:

5.9.3.10 void cmst::Window::printCurInfo ( )

Prints information about the current displayed graph to console

Information including numbers and computational time

Here is the call graph for this function:

Here is the caller graph for this function:

5.9.3.11 void cmst::Window::printSTInfo ( ) [inline]

Print information of the current ST to console.

Here is the call graph for this function:

5.9.3.12 void cmst::Window::printTestInfo ( )

Prints information about the test that has been generated to console.

If no test has been generated, then nothing is printed.

Here is the call graph for this function:

Here is the caller graph for this function:

5.9.3.13 bool cmst::Window::printToFile ( ) [inline]

Print the information of the current graph to file graph.txt.

Here is the call graph for this function:

5.9.3.14 void cmst::Window::resetCurGraph (std::vector< Point2D > & *points*)

Reset the current graph with a vector of points.

Parameters

points	A vector of points.

5.9.3.15 void cmst::Window::resetCurGraph ( )

Reset the current graph with cmst::TestcaseGenerator

The size of the graph is defaulted.

Here is the call graph for this function:

Here is the caller graph for this function:

5.9.3.16 void cmst::Window::resetCurGraph (int *n*)

Reset the current graph with n random generated points.

Parameters

n	The size of the graph to be generated.

Here is the call graph for this function:

5.9.3.17 void cmst::Window::resetCurGraph (int *low,* int *hi*)

Reset the current graph with random generated points.

The size of the graph to be generated is randomly selected between low and hi.

Parameters

low	The least number of points to be generated.
hi	The most number of points to be generated.

Here is the call graph for this function:

5.9.3.18 void cmst::Window::resetHeight (int *height*) `[inline]`

Record the height of current GLUT window.

Here is the call graph for this function:

5.9.3.19 void cmst::Window::resetShowDelaunay ( ) `[inline]`

Change whether the Delaunay diagram is to be drawn to the GLUT window.

5.9.3.20 void cmst::Window::resetShowMST ( ) [inline]

Change whether the MST is to be drawn to the GLUT window.

5.9.3.21 void cmst::Window::resetShowPoint ( ) [inline]

Change whether the points are to be drawn to the GLUT window.

5.9.3.22 void cmst::Window::resetShowST ( ) [inline]

Change whether the STs are to be drawn to the GLUT window.

5.9.3.23 void cmst::Window::resetWidth (int *width*) [inline]

Record the width of current GLUT window.

Here is the call graph for this function:

5.9.3.24 void cmst::Window::runValidate ( ) [inline]

Run the validator for small graphs.

Here is the call graph for this function:

5.9.3.25 int cmst::Window::testDisplayNum ( ) const [inline]

Returns the number of graphs in the test that has been generated.

Returns

the number of graphs in the test that has been generated.

Return values

0	If no test has been generated.

5.9.3.26 int cmst::Window::width ( ) const [inline]

Return the width of current GLUT window.

Returns

The width of current GLUT window

Here is the caller graph for this function:

5.9.4 Member Data Documentation

5.9.4.1 Graph2D∗ cmst::Window::m_curGraph [private]

The pointer to the graph that is being displayed.

5.9.4.2 int cmst::Window::m_height [private]

The height of current GLUT window.

5.9.4.3 cmst::Window ∗ cmst::Window::m_instance = NULL [static], [private]

The pointer to an instance of cmst::Window.

5.9.4.4 bool cmst::Window::m_showDelaunay [private]

Whether the Delaunay iagram is to be drawn.

5.9.4.5 bool cmst::Window::m_showMST [private]

Whether the MST is to be drawn.

5.9.4.6 bool cmst::Window::m_showPoint [private]

Whether the points are to be drawn.

5.9.4.7 bool cmst::Window::m_showST [private]

Whether the MST is to be drawn.

5.9.4.8 struct cmst::Window::Test cmst::Window::m_test [protected]

The test.

5.9.4.9 int cmst::Window::m_width [private]

The width of current GLUT window.

Index