Reference Manual

Generated by Doxygen 1.6.3

Sun Nov 7 18:34:39 2010

Contents

Chapter 1

TinkerCell Core Library

The TinkerCell Core library is a set of C++ classes that utilize Nokia's Qt Toolkit. The classes provide functions for drawing networks as well as storing information associated with each node and connection in the network.Being built using Qt Toolkit, the Core library makes extensive use of Qt's Signal/Slot framework. When signals are emitted, e.g. mousePressed(...), the signals are received by one or more slots. Slots are functions that respond to the signals. In the Core library, the MainWindow class acts like a "signal hub". Numerous Tools classes (aka "plug-ins") implement the slots for processing the MainWindow's signals. The Core library does not do anything by itself, except display the main window. Tools, or plug-ins, perform all the work. The set of plug-ins in the "BasicTools" folder perform numerous tasks such as inserting, highlighting selected items, renaming an item when the text is changed, etc. Other folders such as "ModelingTools" consist of plug-ins that are used to generate dynamic models of biological system. These plug-ins are not part of TinkerCellCore, but they are very important for the TinkerCell application.

The MainWindow class provides the top-level window. It is also a "hub" for numerous signals. Any programmer writing a plug-in must be familiar with all of these signals in order to utilize the Core library well. The MainWindow holds multiple NetworkHandle class instances. The NetworkHandle class is basically what defines a "network". The NetworkHandle stores a collection of ItemHandle instances. The ItemHandle class represents individual nodes (NodeHandle) or connections (ConnectionHandle). It is important to understand that each network can be displayed in multiple windows and each node or connection can be displayed using multiple graphical items on the screen. The NetworkWindow class is a single window that represents either the entire network or just part of a network. A NetworkHandle contains one or more NetworkWindow instances. Each NetworkWindow hold either a GraphicsView or a TextEditor, but never both. Therefore, a "network" (i.e. NetworkHandle) can displayed to the user using one or more graphical diagrams (GraphicsView) or text (TextEditor).

To understand the design of the Core library, it is imperative to understand ItemHandle. To build well-behaved plug-ins, it is imperative to understand how the Core library uses Undo Commands and Signals. It is also important to review the functions available in the MainWindow, GraphicsScene, and NetworkHandle classes.

DataTable<T>

This is a template class that stores a 2 dimensional table, including the row and column headers. The contents of the table can belong to any type. Typically, TinkerCell only uses double and QString types because those are the two allowed data types in the ItemHandle class. The DataTable class is composed of three vectors: the data, the column headers, and the row headers. The class provides functions for obtaining the data values using header names or index values, removing or adding rows and columns, swapping rows and columns, and resizing the table. NumericalDataTable is an alias for DataTable<double> and TextDataTable is for DataTable<QString>.

```
NumericalDataTable * dat = new NumericalDataTable;
```

```
    dat->resize(10,4);
    dat->colName(0) = "column 1";
    dat->seRowNames( QStringList() << "row A" << "row B" << "row C" );
    dat->value("row A", "column 1") = 10.0;
    dat->removeCol(2);
    dat->addCol(3,"column 3");  //insert new column at position 3
    dat->value("X", "Y") = 5.0;   //automatically creates a new row called X
  and new column called Y
    int r = dat->rows();
    int c = dat->cols();
    NumericalDataTable dat2 = dat->transpose();
```

Undo Commands

Numerous classes are defined in the UndoCommands.h file that inherit from QUndoCommand. These classes contain an undo() and a redo() method. These functions undo and redo a single action without any other side effects. All changes made to a network are generally done using one of these QUndoCommand classes. Examples of undo command classes include MoveCommand, InsertGraphicsCommand and RemoveGraphicsCommand, InsertTextCommand and RemoveTextCommand, ChangeDataCommand, and RenameCommand. There are several more, one for each "atomic" operation. CompositeCommand can be used to construct a more complex command from atomic commands. For example, the "paste" operation is a composite command made from InsertCommand, MoveCommand, and RenameCommand (for renaming newly inserted items). Other plug-ins also use the composite command.

The common procedure for using an undo command is as follows:

```
    QList<QGraphicsItem*> graphicsItems;
    //add some items into graphicsItems
    QUndoCommand * cmd = new InsertGraphicsCommand("some informative message
  here",graphicsItems,handles);

    if (mainWindow && mainWindow->historyStack())
            mainWindow->historyStack()->push(cmd);
```

Alternatively, the NetworkHandle class and GraphicsScene class provide functions that automatically do the same operations:

```
    QList<QGraphicsItem*> graphicsItems;
    //add some items into graphicsItems
    GraphicsScene * scene = currentScene();
    scene->insert("informative message here", graphicsItems);
```

ItemHandle class

This class is arguable the most integral aspect in the TinkerCell Core library. The ItemHandle can be thought of as a "package" with four important components: the graphics items for drawing a node or connection, the data table associated with that node or connection, the tools associated with the node of connection, and the family that the node or connection is identified with. The ItemHandle is the complete package that is required to obtain all the information about any item in the network. Since TinkerCell networks can be constructed using text of graphics interface, the ItemHandle is not required to have graphical items. For networks constructed using the text editor, the data inside each ItemHandle is what is most important.

NodeHandle and ConnectionHandle inherit from ItemHandle. For text based models, it is possible to store connections between nodes and connections using ConnectionHandle::addNode() method, which takes a NodeHandle and an integer describing the "role" of that node in the connection. The interpretation of the "role" is open to the plug-in using it.

Here is a code example, where two graphics items are placed inside a handle, and a new table is added to the handle:

```
    NodeHandle * nodeHandle = new NodeHandle;

    //make a node item from an XML file
    NodeGraphicsItem * node = new NodeGraphicsItem;
    NodeGraphicsReader reader;
    reader.readXML(node,"mynode.xml");

    //make a text graphics item
    TextGraphicsItem * text = new TextGraphicsItem("hello world");

    //add graphics items to the handle
    nodeHandle->graphicsItems << node << text;

    nodeHandle->textData("magic word") = "please";
    nodeHandle->numericalData("magic numbers","pi","value") = 3.141593;
    nodeHandle->numericalData("magic numbers","e","value") = 2.718282;

    //get the entire table
    DataTable<qreal> magicNumbers = nodeHandle->numericalDataTable("magic num
  bers");
    //set the entire table
    nodeHandle->numericalDataTable("magic numbers") = magicNumbers;

    //get list of all tables
    nodeHandle->getNumericalDataNames();
    nodeHandle->getTextDataNames();
```

ItemHandle contains several functions for conveniently retrieving information or the list of child items. Please see the ItemHandle documentation . Each ItemHandle instance contains a list of pointers to tools, or classes that inherit from class Tool. These tools are associated with this item. When items are selected by a user, the list of contextMenuActions from each of these tools is placed in context menu and the list of graphics items are displayed to the side.

ItemFamily class

The ItemFamily class is used to describe a family that a node or connection belongs in. Nodes and connections are not required to belong in a family. Each family can have multiple parent families. The two main child classes are NodeFamily and ConnectionFamily. NodeFamily stores the default graphics item(s) that is used to draw an item of that family, and ConnectionFamily stores the default arrow head that is used when drawing connections of a given family. The family information is useful for tools in order to distinguish items and insert data tables according to the family of the item.

```
    NodeFamily * f1 = new NodeFamily("family A");
    NodeFamily * f2 = new NodeFamily("family B");
    f2->setParent(f1);  //family B is a sub-family of family A

    NodeHandle * node = new NodeHandle("x",f2);

    if (node->isA("family A")) // will return true
    {
    }
```

ItemData

The "Data" inside an ItemHandle is an instance of class ItemData. This class is just composed of two hash tables, numericalData and textData. Each hash table maps a string to a DataTable. These hash tables store all the information needed to describe a node or connection. For example, numericalData["parameters"] might contain all the parameters belonging to this item. The data tables inside each item are added by tools, which often use the family information to decide what data tables to insert in a given item. For example, connections might contain textData["rates"] to describe the flux equations whereas nodes of a particular family might contains some other information, such as textData["DNA sequence"]. It is important to note that each entry is a 2D table of strings or numbers; of course, they can be a 1x1 table as well.

MainWindow class

The MainWindow is always the top-most widget that is created in the main() function. The central widget inside the MainWindow is a Tab Widget with windows that can be popped out. Each widget inside the tab widget is a NetworkWindow. Each NetworkWindow can contain a TextEditor or a GraphicsScene. The MainWindow constructor has two arguments for specifying whether the documents should only contain TextEditors or only GraphicsScene or both. Each GraphicsScene is displayed using a GraphicsView.

The MainWindow class inherits from Qt's QMainWindow. The MainWindow has two main functions:

1. Provide the main window for the docking windows, menus, text editors, and drawing canvas

2. Serve as a Signal hub that routes the signals from each scene or text editor to the plug-ins listening to those signals. Thus, the plug-ins do not need to connect to every single scene and text editor; they only need to connect to the MainWindow's signals. These connections are made in a plug-in's setMainWindow() method.

The MainWindow also provides several Slots that are connected to C function pointers via the C_API_-Slots class. These functions include find, rename, move, remove, and other functions for changing the data tables within an item in the network.

Nearly all the members in the MainWindow class are public. This includes the three toolbars: 1. toolBarBasic, which stores buttons for basic functions such as new, open, and save; 2. toolBarEdits, which stores buttons such as copy and pase; 3. toolBarForTools, which is intended for other tools. Tools may also add new toolbars using the addToolBar method in QMainWindow. The context menu (mouse right button) for TextEditor and GraphicsScene are also defined in MainWindow. The menus named contextItemsMenu and contextScreenMenu are used by GraphicsScene when items are selected and when no item is selected, resp.. The menus named contextSelectionMenu and contextEditorMenu are used by TextEditor when text is highlighted and when no text is highlighted, resp. Menu items such as file menu, edit menu, settings menu, and view menu are also public, allowing tools to add new actions to them.

When items are inserted or removed from a GraphicsScene or TextEditor, each class emits a signal indicating that graphics item(s) have been removed and text item(s) have been removed, resp. These signals are connected to signals in the MainWindow with the same names. In addition, MainWindow also emits two signals called itemsInserted and itemsRemoved that only contain the ItemHandles instead of the graphics items or text items. Signals that contain only ItemHandles are useful for tools that do not need to know whether the network was constructed using text or graphical interface.

itemsAboutToBeInserted and itemsAboutToBeRemoved: these signal are emitted just before items are inserted or removed from a network, respectively. It can be used to automatically add or remove items from the list. The signal contains a list of QUndoCommands; new commands can be added to this list to perform additional actions along with the insertion event. **itemsInserted and itemsRemoved**: these signals are emitted after items are inserted or removed from a network, respectively. It can used to modify the items that have been inserted based on the placement of the items or other conditions. It is also used to add tools to the handle::tools list of the new items. **dataChanged**: this signal is emitted whenever any handle's data entry is changed. It is also emitted when items are inserted or removed. This signal can be used to check when a model need to be updated. Note that undo events are not captured by this signal, which is only captured by historyChanged signal. **historyChanged**: this signal is emitted whenever any recorded change occurs. This signal can be used to check when a model need to be updated. **networkOpened, networkClosed, and networkChanged**: these signals are emitted whenever a new network is opened, a network has been closed, or a the user has clicked on a different network window (respectively). These signals are usually used to reset contents of widgets that display information about a network. **networkOpening and networkClosing**: these signals are sent before opening or closing networks (respectively). They can be used to check if the network has been saved. **mousePressed, mouseReleased, mouseDragged, mouseDoubleClicked, sceneRightClicked**: These signals are emitted due to mouse events. These signals are emitted even if the useDefaultBehavior switch is off in GraphicsScene. **keyPressed, keyReleased**: These signals are emitted due to keyboard events. These signals are emitted even if the useDefaultBehavior switch is off in GraphicsScene.

NetworkHandle

The NetworkHandle is used to store all the information inside a network. The three main components of a NetworkHandle are: historyStack, symbolsTable, and networkWindows. The history stack is used to store the QUndoCommands that provide the undo/redo capabilities. The symbolsTable stores all the nodes and connections in the network. The list networkWindows stores all the windows that are used to display the network to the user. The NetworkHandle provides convenience functions such as changeData(...) or rename(...). These functions create a QUndoCommand, add it to the history stack. Each NetworkHandle can be represented using one or more windows. All of these windows are connected to the same symbols table and the same history stack. NetworkHandle also contains functions such as find() for finding any string in the network and parseMath for validating a mathematical expression (uses muparser).

NetworkWindow

The NetworkWindow is a window (QMainWindow) inside the MainWindow's tab widget. This window can contain either a TextEditor or a GraphicsScene, but not both. Each NetworkWindow can contain its own toolbar or dock widgets. Each NetworkWindow has functions for replacing its current scene or text editor (warning: this operation cannot be undone). Each NetworkWindow can contain an ItemHandle pointer. This handle can be used for multiple purposes. It is designed for particular scenarios in which each individual window is associated with a handle. By default, this pointer is zero.

SymbolsTable

The SymbolsTable class is used to store all the string found in a network model. These strings include the node and connection names and the row names and column names of all the data contained within each node and connection. The purpose of the symbols table is to easily look-up a symbol and find the network objects associated with that symbol. The symbols table keeps a hash table of names and pointers to the node or connection with that name.

The SymbolsTable is also used to get all the ItemHandles in a network, except for "hidden" ItemHandles. ItemHandles represent objects in a network, whether the model is represented as text or graphics.

Full names are always unique, e.g. Cell1.p1. Just the first name, e.g p1, need not be unique. The symbols table keeps a one-to-one hash table that maps full names to object pointers and a one-to-many that maps the first names to object pointers. The uniqueData hash table stores prefixed strings, e.g. p1.param1, as well as non-prefixed strings, e.g. param1. For each string, the hash table stores all the objects that contain that string and the name of the data table which contains that string.

Each NetworkWindow contains one SymbolsTable instance. This instance is updated during any change (history update) to the network.

GraphicsScene

The GraphicsScene class is used to construct a network visually. It is one of the largest classes in Tinker-Cell. The GraphicsScene inherits from Qt's QGraphicsScene. The primary duty of the GraphicsScene class is to receive mouse and keyboard events and emit necessary signals such as itemsSelected, itemsMoved, or mouseOverItem.

The GraphicsScene also handles selection of objects on the scene and moving objects on the scene. The selected objects are placed in the selected() list, and the moving objects are placed in the moving() list. These lists can be modified by plug-ins in order to modify which objects are selected or moved. Moving items are always grouped together when moving; this makes the movement smoother. For example, if a node has other nodes attached to it, a plug-in can ensure that all the nodes move together by adding each node to the moving() list when any one of them is selected. The GraphicsScene's selection and moving operations can be disabled by setting useDefaultBehavior = false.

In addition to emitting signals and handling selection and moving, the GraphicsScene houses numerous functions for conveniently making changes to a network. The functions include insert, remove, move, re-name, and changeData. Each of these functions do three things: make a QUndoCommand object, push the undo command to the history stack, and emit the necessary signal(s) such as itemsInserted or itemsRemoved.

The GraphicsScene is always contained inside a NetworkWindow. Therefore it uses the parent Network-Window's history stack and symbols table. Many functions such as changeData, rename, or allHandles simple call the parent NetworkWindow's function.

Configuring GraphicsScene

Various visual features, such as the color of the selection rectangle in a scene and default grid size can be set using global variables: GraphicsScene::SelectionRectangleBrush, GraphicsScene::SelectionRectanglePen, GraphicsScene::BackgroundBrush, GraphicsScene::ForegroundBrush, GraphicsScene::GRID, GraphicsS-cene::GridPen. GraphicsScene::MIN_DRAG_DISTANCE can be used to set the minimum distance that is considered a valid drag, i.e. moving the mouse less than this distance will be considered an accidental movement of the mouse and ignored.

GraphicsView

The GraphicsView is a class for viewing a GraphicsScene. It inherits from QGraphicsView, and provides a few extra features such as drag-and-drop and zooming.

Graphics items

Qt's QGraphicsItem class is used to draw all the items in the GraphicsScene. The two main graphics item classes are NodeGraphicsItem and ConnectionGraphicsItem. Supporting graphics items are TextGraphic-sItem and ControlPoint.

The qgraphicsitem_cast<class> function can used to cast a generic QGraphicsItem to one of these four classes. In addition, NodeGraphicsItem::cast and ConnectionGraphicsItem::cast can also be used to get the top-most node or connection item from a generic QGraphicsItem instance. Each NodeGraphicsItem and ConnectionGraphicsItem also contains a string named ClassType, which is used to statically cast sub-classes of Node or Connection. For example, ArrowHeadItem is a NodeGraphicsItem with classType = "Arrow Head Item". example usage: if (node->className == ArrowHeadItem::CLASSNAME) static_-cast<ArrowHeadItem∗>(node)

ControlPoint

The ControlPoint class is used to identify key locations of a NodeGraphicsItem or ConnectionGraphicsItem that can be used to change the appearance of that item. For example, NodeGraphicsItem uses control points around its bounding box, allowing a user to drag the control points in order to resize the item. Connection-GraphicsItem uses control points to define the line or beziers used to draw the connection. See image to the right: the small squares and circles are control points. Control points are generally not child items of the item that they belong with. The two main sub-classes of ControlPoint are NodeGraphicsItem::ControlPoint and ConnectionGraphicsItem::ControlPoint.

NodeGraphicsItem

This class is used to draw nodes on the GraphicsScene. NodeGraphicsItem inherits from QGraphicsItem-Group, which is used to group several graphics items together. Each NodeGraphicsItem is a set of points and a set of shapes that are defined using those points. The points belong to the ControlPoint class and the shapes belong to the Shape class. The entire NodeGraphicsItem can be saved as an XML file using NodeGraphicsItemWriter (and NodeGraphicsItemReader for reading the XML). The XML file uses the SBML render extension format, which is similar to SVG.

The NodeGraphicsItem has convenient functions such as connections(). The set of connections connected to a given node is retrieved by looking at the control points that are child items of that node. Each connec-tion must have a control point that is the child item of the node that is it connected to.

Shape This class is a polygon constructed using lines, beziers, or arcs. The Shape class inherits from QGraphicsPolygonItem. The polygon must be closed. The refresh() method is used whenever the shape's control points are changed. This updates the shape's polygon.

ConnectionGraphicsItem

This class is used to draw connections between nodes. ConnectionGraphicsItem is composed of a list of CurveSegment instances. Each CurveSegment is a collection of control points that define a single path, usually with the same central control point. Each curve segment also has two arrow head items -- one at either ends (they can be null). If there is a node at the end of any of the paths, then the control points at the end will be child items (see QGraphicsItem) of that node; so, looking at the parent items of each of the control points at the ends is the correct way to find all the nodes that are connected by a connection.

The ConnectionGraphicsItem also contains an optional centerRegionItem, which is a node that sits at the center of the connection. This node is used when one connection item needs to connect to another connection item. Since connections can only be connected to nodes, the center region item is used when connecting a connection to another.

The control points that constitute a connection are generally parent-free, except for the end control points. As mentioned earlier, if a control point is at the end of a connection and is connected to a node, then the control point will be set as the child of the node item. This allows the control point to move along with the node. The ConnectionGraphicsItem class retrieves all the nodes that it is connected to by looking at the parent items of each of its end control points. ConnectionGraphicsItem provides convenient functions such as nodes(), nodesWithArrows(), nodesWithoutArrows(), where "WithArrows" means that there is an arrow head at the arc leading to the node. It is important to understand that these functions do not imply that the curve segments represent a reaction or some other specific process. They indicate the visual representation, which is then translated to more specific meanings by the plug-ins.

refresh() is used whenever the connection is changed. This function updates the arcs and the shape() of the connection using the control point positions.

The ConnectionGraphicsReader and Writer can be used to read and write a connection item to an XML file.

The default arrow head can be set using ConnectionGraphicsItem::DefaultArrowHeadFile. Similarly, the default middle item (the box at the center) can be set using ConnectionGraphicsItem::DefaultMiddleItemFile. For example:

ConnectionGraphicsItem::DefaultArrowHeadFile = appDir + QString("/ArrowItems/Reaction.xml"); ConnectionGraphicsItem::DefaultMiddleItemFile = appDir + QString("/OtherItems/simplecircle.xml"); TextEditor class

TextEditor

The TextEditor class is used to construct a network using a text-based language. The syntax is not defined by TextEditor and must be provided by a supporting plug-in. The supporting plug-in is expected to make use of the lineChanged(...) and textChanged(...) signals emitted by TextEditor to identify changes by a user and call the insertItem(...), removeItem(...), or setItem(...) methods in order to modify the network.

Tool (plug-in)

Tool is the parent class for all TinkerCell "plug-ins". The most important method in the Tool class is setMainWindow(), which is used by a new tool to connect with the MainWindow's signals and slots.

Each Tool can choose to create instances of Tool::GraphicsItem and place them on the scene. When these graphics items are selected by the user, TinkerCell Core will call the select(int) method of the Tool that is associated with the graphics item.

Console Window

The ConsoleWindow class provides a generic framework for Tools to receive command-line input as well as display messages or execute commands. Each tool can access the ConsoleWindow using console() or mainWindow->console(). For example:

Tools can also interact with the user by connecting to the ConsoleWindow's commandExecuted signal.

This signal is emitted whenever the user pressed <return> after entering a text at the command prompt. The Tools can process the string and carry out necessary operations.

```
  if (console())
  {
          console()->message("hello world");    //print a message on the co
nsole window
          console()->error("incorrect response");  //print an error message
 on the console window
          console()->eval("print 1+2");  //evaluate this expression (only r
uns if a plugin such as python plugin is available)
  }

  DataTable<double> data;
  console()->printTable(data); //print a table (tab-delimited)

  ConsoleWindow * console = console();
  if (console)
  {
          connect(editor, SIGNAL( commandExecuted(const QString&) ),
                  this, SLOT( commandExecuted(const QString&) ));
  }
```

Tools may also disable and re-enable the ConsoleWindow while they are processing the command by using:

```
  console()->freeze();    //lock the console window
  console()->unfreeze();  //unlock the console window

  Alternatively, Tools may also connect with the freeze() and unfreeze() sl
ots:

  CommandTextEdit * editor = console()->editor();
  if (editor)
  {
          connect(this, SIGNAL(freeze()), editor,SLOT(freeze()));
          connect(this, SIGNAL(unfreeze()), editor,SLOT(unfreeze()));
          connect(this, SIGNAL(setFreeze(bool)), editor,SLOT(setFreeze(bool
)));
          connect(editor, SIGNAL( commandExecuted(const QString&) ),
                          this, SLOT( commandExecuted(const QString&) ));
  }
```

CThread

This class is used to run C plugins as separate threads.

InterpreterThread

This class inherits from CThread. It is used to run interpreters such as Python and Octave interpreter.

PythonInterpreterThread

This class inherits from InterpreterThread. It is used to embed Python interpreter. This class uses the C program python/runpy.c.in

OctaveInterpreterThread

This class inherits from CThreads. It is used to embed Octave interpreter. This class uses the C++ program octave/runOctave.cpp (for embedding Octave) and assumes that SWIG has been used to generate tinkercell.oct library (which extends Octave).

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Module Documentation

5.1 TinkerCell Core classes

The main classes in TinkerCell Core. These form the base for all the plug-ins.

Classes

- class Tinkercell::ArrowHeadItem

  A node graphics item that is used to draw arrow heads on connection items.

- class Tinkercell::ConnectionGraphicsItem

  A graphics nodes item that draws connection between two or more nodes and the arrow heads at the ends.

- class Tinkercell::ConnectionGraphicsItem::ControlPoint

  A control point with a pointer to a ConnectionGraphicsItem.

- class Tinkercell::ConnectionGraphicsItem::CurveSegment

  A set of control points and two arrow heads.

- class Tinkercell::ControlPoint

  A simple circle or square that is used for changing specific locations.

- class Tinkercell::ProcessThread

  This class is used to run a process (command + args) as a separate thread as a separate thread.

- class Tinkercell::DataTable< T >

  DataTable is a 2D vector with row names and column names.

- class Tinkercell::GraphicsScene

  The primary task of the graphics scene is to draws items. All interactions with the GraphicsScene is done through MainWindow or NetworkHandle. NetworkHandle provides functions such as move, insert, and remove. MainWindow relays all the signals, such as mouse and key events, from the GraphicsScene. So, there is rarely a need to directly interact with the GraphicsScene.

- class Tinkercell::GraphicsView

GraphicsView class that is used to view the contents of a GraphicsScene. The class inherits from QGraphicsView.

- class Tinkercell::Unit

  A unit of measurement.

- class Tinkercell::ItemFamily

  This class defines the family of a node or connection. The class contains the icon for the family, family name, and minimal data that defines the family. Each family has a name, which is internally converted to an integer (ID) The ID is used to perform isA checks, thus avoiding repeated string matches.

- class Tinkercell::NodeFamily

  This class defines the family of a node. Inherits from ItemFamily. It contains a list of NodeGraphicsItems that is the default for this family of nodes.

- class Tinkercell::ConnectionFamily

  This class defines the family of a connection. Inherits from ItemFamily It contains a list ofConnectioGraphicsItems that is the default for this family of connections.

- class Tinkercell::ItemHandle

  The ItemHandle represents a complete object in the network, whether it is a node or a connection. The ItemHandle contains the name of the object and pointers to all the QGraphicsItems that are used to represent the object. Tools associated with the object can be stored within the ItemHandle as well. The ItemHandle can also optionally contain an ItemFamily, which can be used to distinguish different types of nodes or connections, if needed. Each ItemHandle can contain one parent. Several functions are available for convinently getting the parents and children of an ItemHandle.

- class Tinkercell::NodeHandle

  The handles are used to bring together data and graphics items. Node Handle contains pointers to all the graphics items that belong to it, the tools that apply to this item, the data for this item, and the family that it belongs with.

- class Tinkercell::ConnectionHandle

  The handles are used to bring together data and graphics items. Connection Handle contains pointers to all the graphics items that belong to it, the tools that apply to this item, the data for this item, the family that it belongs with, and pointers to nodes connected (in and out).

- class Tinkercell::MainWindow

  MainWindow is the parent container for all the other widgets in TinkerCell The central widget in MainWindow is a tab widget. Each tab widget can hold a GraphicsView or a TextEditor. One of the main roles of MainWindow is to serve as a signal/slot hub for Tools.

- class Tinkercell::NetworkHandle

  A class that is used to store a network. The network is a collection of Item Handles. The history stack is also a key component of a network. The network can either be represented as text using TextEditor or visualized with graphical items in the GraphicsScene. Each node and connection are contained in a handle, and each handle can either be represented as text or as graphics. The two main components of NetworkWindow are the SymbolsTable and HistoryStack This class provides functions for inserting items, removing items, and changing information inside the model.

- class Tinkercell::NodeGraphicsItem

  A simple figure made from one or more polygons. The class can be represented in an XML file.

- class Tinkercell::NodeGraphicsItem::ControlPoint

  a control point with a pointer to a NodeGraphicsItem

- class Tinkercell::NodeGraphicsItem::Shape

  A closed polygon path made from arcs, lines, and beziers.

- class Tinkercell::NodeGraphicsReader

  An xml reader that reads a NodeGraphicsItem file.

- class Tinkercell::SymbolsTable

  The symbols table is updated every time the scene or text editor changes. The symbols table contains the list of item names and ItemHandle pointers as well as names and pointers to each data entry in each item.

- class Tinkercell::TextEditor

  This is the window that allows used to construct networks using text, as opposed to graphics, which is done by GraphicsScene. The TextEditor requires a supporting tool that parses the text and calls the itemsInserted or itemsRemoved methods. Without a supporting parser tool, the TextEditor will not do anything.

- class Tinkercell::TextGraphicsItem

  editable text item

- class Tinkercell::TextParser

  TextParser is the parent class for all parsers. Parsers are classes that interpret the string in a TextEditor and insert items or modify items as needed. TinkerCell can support multiple parsers through the use of the TextParser interface.

- class Tinkercell::Tool

  everything other than the main window is a tool

- class Tinkercell::ToolGraphicsItem

  tools that are drawn on the scene instead of displayed as a window

Typedefs

- typedef DataTable< QString > Tinkercell::TextDataTable

  a numerical data table

- typedef DataTable< qreal > Tinkercell::NumericalDataTable

  a numerical data table

Functions

- QGraphicsItem ∗ Tinkercell::getGraphicsItem (QGraphicsItem ∗item)

  gets the parent of this item that is a node, text, connection, or control point

- QGraphicsItem ∗ Tinkercell::cloneGraphicsItem (QGraphicsItem ∗item)

  Clone a graphics item. The item handle will NOT be duplicated.

- QList< QGraphicsItem ∗ > Tinkercell::cloneGraphicsItems (QList< QGraphicsItem ∗ > &items, QList< ItemHandle ∗ > &newHandles, bool deep=true)

  Clone a list of graphics items.

- ItemHandle ∗ Tinkercell::getHandle (QGraphicsItem ∗)

  get the handle from a graphics item

- QList< ItemHandle ∗ > Tinkercell::getHandle (const QList< QGraphicsItem ∗ > &)

  get the handles from graphics items

- void Tinkercell::setHandle (QGraphicsItem ∗, ItemHandle ∗)

  set the handle of a graphics item (use 0 to remove handle)

5.1.1 Detailed Description

The main classes in TinkerCell Core. These form the base for all the plug-ins.

5.1.2 Function Documentation

5.1.2.1 TINKERCELLEXPORT QGraphicsItem ∗ Tinkercell::cloneGraphicsItem (QGraphicsItem ∗ *item*)

Clone a graphics item. The item handle will NOT be duplicated.

Parameters

QGraphicsItem ∗ a pointer to a QGraphicsItem

Returns

QGraphicsItem ∗ a QGraphicsItem that is clone of the argument

5.1.2.2 TINKERCELLEXPORT QList< QGraphicsItem ∗ > Tinkercell::cloneGraphicsItems (QList< QGraphicsItem ∗ > & *items*, QList< ItemHandle ∗ > & *newHandles*, bool *deep* = `true`)

Clone a list of graphics items.

Parameters

QList<QGraphicsItem∗> a list of pointers to a QGraphicsItems

QList<ItemHandle∗> return value: returns all the new handles here

bool duplicate the handles as well (default = true).

Returns

QList<QGraphicsItem∗> a new list of QGraphicsItems that are clones of the corresponding argument

5.1.2.3 TINKERCELLEXPORT QGraphicsItem $*$ Tinkercell::getGraphicsItem (QGraphicsItem $*$ *item*)

gets the parent of this item that is a node, text, connection, or control point

Parameters

QGraphicsItem $*$ Qt graphics item

Returns

QGraphicsItem $*$ node, connection, text, or control point

5.1.2.4 TINKERCELLEXPORT QList< ItemHandle $*$ > Tinkercell::getHandle (const QList< QGraphicsItem $*$ > &)

get the handles from graphics items

Parameters

QList<QGraphicsItem$*$> graphics item

Returns

QList<ItemHandle$*$> item handles

5.1.2.5 TINKERCELLEXPORT ItemHandle $*$ Tinkercell::getHandle (QGraphicsItem $*$)

get the handle from a graphics item

Parameters

QGraphicsItem$*$ graphics item

Returns

ItemHandle$*$ item handle (0 if none)

5.1.2.6 TINKERCELLEXPORT void Tinkercell::setHandle (QGraphicsItem $*$, ItemHandle $*$)

set the handle of a graphics item (use 0 to remove handle)

Parameters

QGraphicsItem$*$ graphics item

ItemHandle$*$ handle (use 0 to remove handle)

5.2 Helper functions and classes

Helper classes and functions that are used by the core classes.

Classes

- class Tinkercell::HistoryWindow

  This is a small class extending QUndoView that manages a stack of undo commands.

- class Tinkercell::ItemData

  This class is used to store information about nodes or connections. It contains a hashtable of data tables, which is used by different tools to store specific data. The versions queue can be used to keep previous versions of the data.

Functions

- QPointF Tinkercell::pointOnEdge (const QRectF &rect0, const QPointF &p1, qreal dist, bool straight)

  gets the point on the edge of the rect such that it is in the same line as the center of the rect and the point (arg)

- QPointF Tinkercell::pointOnEdge (const NodeGraphicsItem &node, const QPointF &pt, qreal dist, bool straight)

  gets the point on the edge of the shape such that it is in the same line as the center of the rect and the point (arg)

- tc_matrix Tinkercell::emptyMatrix ()

  construct a tc_matrix with 0 rows and columns

- ItemHandle ∗ Tinkercell::ConvertValue (long)

  convert void∗ to ItemHandle pointer

- long Tinkercell::ConvertValue (ItemHandle ∗)

  convert ItemHandle pointer to void ∗

- QList< ItemHandle ∗ > ∗ Tinkercell::ConvertValue (tc_items)

  convert tc_items to QList of ItemHandle pointers

- tc_items Tinkercell::ConvertValue (const QList< ItemHandle ∗ > &)

  convert QList of ItemHandle pointers to tc_items

- QString Tinkercell::ConvertValue (const char ∗)

  convert char∗ to QString

- const char ∗ Tinkercell::ConvertValue (const QString &)

  convert QString to null-terminated char∗

- DataTable< QString > ∗ Tinkercell::ConvertValue (tc_table) *convert tc_table to DataTable of QString*

- tc_table Tinkercell::ConvertValue (const DataTable< QString > &)

  convert DataTable of QStrings to tc_table

- DataTable< qreal > ∗ Tinkercell::ConvertValue (tc_matrix)

  convert matrix to datatable<double> (see DataTable.h and TC_structs.h)

- tc_matrix Tinkercell::ConvertValue (const DataTable< qreal > &)

  convert Datatable<double> to tc_matrix (see DataTable.h and TC_structs.h)

- QStringList Tinkercell::ConvertValue (tc_strings)

  convert tc_strings to QStringList

- tc_strings Tinkercell::ConvertValue (const QStringList &)

  convert QStringList to tc_strings

- QString Tinkercell::RemoveDisallowedCharactersFromName (const QString &)

  This function replaces disallowed characters in a name string.

5.2.1 Detailed Description

Helper classes and functions that are used by the core classes.

5.2.2 Function Documentation

5.2.2.1 TINKERCELLEXPORT tc_strings Tinkercell::ConvertValue (const QStringList &)

convert QStringList to tc_strings

Returns

tc_strings

5.2.2.2 TINKERCELLEXPORT QStringList Tinkercell::ConvertValue (tc_strings)

convert tc_strings to QStringList

Returns

QStringList

5.2.2.3 TINKERCELLEXPORT tc_matrix Tinkercell::ConvertValue (const DataTable< qreal > &)

convert Datatable<double> to tc_matrix (see DataTable.h and TC_structs.h)

Returns

tc_matrix

5.2.2.4 TINKERCELLEXPORT DataTable< qreal > ∗ Tinkercell::ConvertValue (tc_matrix)

convert matrix to datatable<double> (see DataTable.h and TC_structs.h)

Returns

DataTable of qreals

5.2.2.5 TINKERCELLEXPORT tc_table Tinkercell::ConvertValue (const DataTable< QString > &)

convert DataTable of QStrings to tc_table

Returns

tc_table

5.2.2.6 TINKERCELLEXPORT DataTable< QString > ∗ Tinkercell::ConvertValue (tc_table)

convert tc_table to DataTable of QString

Returns

QStringList

5.2.2.7 TINKERCELLEXPORT const char ∗ Tinkercell::ConvertValue (const QString &)

convert QString to null-terminated char∗

Returns

null-terminated char∗

5.2.2.8 TINKERCELLEXPORT QString Tinkercell::ConvertValue (const char ∗)

convert char∗ to QString

Returns

QString

5.2.2.9 TINKERCELLEXPORT tc_items Tinkercell::ConvertValue (const QList< ItemHandle ∗ > &)

convert QList of ItemHandle pointers to tc_items

Returns

tc_items

5.2.2.10 TINKERCELLEXPORT QList< ItemHandle ∗ > ∗ Tinkercell::ConvertValue (tc_items)

convert tc_items to QList of ItemHandle pointers

Returns

QList<ItemHandle∗>

5.2.2.11 TINKERCELLEXPORT long Tinkercell::ConvertValue (ItemHandle ∗)

convert ItemHandle pointer to void ∗

Returns

void∗

5.2.2.12 TINKERCELLEXPORT ItemHandle ∗ Tinkercell::ConvertValue (long)

convert void∗ to ItemHandle pointer

Returns

ItemHandle∗

5.2.2.13 TINKERCELLEXPORT tc_matrix Tinkercell::emptyMatrix ()

construct a tc_matrix with 0 rows and columns

Returns

tc_matrix

5.2.2.14 TINKERCELLEXPORT QPointF Tinkercell::pointOnEdge (const NodeGraphicsItem & *node*, const QPointF & *pt*, qreal *dist*, bool *straight*)

gets the point on the edge of the shape such that it is in the same line as the center of the rect and the point (arg)

gets the point on the edge of the shape such that it is in the same line as the center of the shape's bounding rect and the point (arg)

Parameters

shape	
point	outside rectangle

Returns

the point on the edge of the shape

Parameters

QPainterPath the shape

QPointF point outside shape

Returns

QPointF the point on the edge of the shape

5.2.2.15 TINKERCELLEXPORT QPointF Tinkercell::pointOnEdge (const QRectF & *rect0*, const QPointF & *p1*, qreal *dist*, bool *straight*)

gets the point on the edge of the rect such that it is in the same line as the center of the rect and the point (arg)

Parameters

rectangle

point outside rectangle

Returns

the point on the edge of the rectangle

Parameters

QRectF rectangle

QPointF point outside rectangle

Returns

QPointF the point on the edge of the rectangle

5.2.2.16 TINKERCELLEXPORT QString Tinkercell::RemoveDisallowedCharactersFromName (const QString &)

This function replaces disallowed characters in a name string.

Parameters

QString original string

5.3 Input and output

Classes that read/write graphics information and data information from/to files as well as serve as input/output devices for C functions.

Classes

- class Tinkercell::AbstractInputWindow

  Classes that inherit from this class can be used as GUI windows that provide interface to C programs (library files).

- class Tinkercell::SimpleInputWindow

  Used to create an input window that can receive user inputs for C plugins.

- class Tinkercell::ConnectionGraphicsReader

  An xml reader that reads a NodeGraphicsItem file.

- class Tinkercell::ConnectionGraphicsWriter

  This class is an xml writer that specifically writes a ConnectionGraphicsItem.

- class Tinkercell::CommandTextEdit

  A command-line type text box that other tools can use for scripting interface.

- class Tinkercell::ConsoleWindow

  Used to create an output window that can display outputs.

- class Tinkercell::ModelReader

  reads an xml file with handle names and data table information and generates a list of item handles

- class Tinkercell::ModelWriter

  writes to an xml file handle names and data table information from a list of item handles

- class Tinkercell::MultithreadedSliderWidget

  This class is used to run specific functions inside a C dynamic library as a separate thread. Uses CThread to call the C functions.

- class Tinkercell::NodeGraphicsWriter

  An xml reader that reads a NodeGraphicsItem file.

5.3.1 Detailed Description

Classes that read/write graphics information and data information from/to files as well as serve as input/output devices for C functions.

5.4 Undo commands

A set of classes that allow undo/redo (using Qt Undo framework).

Classes

- class Tinkercell::ChangeDataCommand< T >

  This template class allows undo and redo of a change made to a data table.
- class Tinkercell::Change2DataCommand< T1, T2 >

  Changes two different data tables.
- class Tinkercell::TextUndoCommand

  this command performs a text change
- class Tinkercell::InsertHandlesCommand

  this command inserts new handles to a NetworkHandle
- class Tinkercell::RemoveHandlesCommand

  this command inserts new handles to a NetworkHandle
- class Tinkercell::MoveCommand

  this command performs a move and allows redo/undo of that move
- class Tinkercell::InsertGraphicsCommand

  this command performs an insert and allows redo/undo of that insert
- class Tinkercell::RemoveGraphicsCommand

  this command performs an removal and allows redo/undo of that removal
- class Tinkercell::ChangeBrushCommand

  this command changes the brush of an item
- class Tinkercell::ChangePenCommand

  this command changes the pen of an item
- class Tinkercell::ChangeBrushAndPenCommand

  this command changes the pen and/or brush of an item
- class Tinkercell::ChangeZCommand

  this command changes the pen of an item
- class Tinkercell::TransformCommand

  this command changes the size, angle, and orientation of an item
- class Tinkercell::ChangeParentCommand

  this command changes the parent of a graphics item (not handles)
- class Tinkercell::RenameCommand

this command changes the name of the handle of an item. important: use full name of the items!

- class Tinkercell::CompositeCommand

  this command can be used to combine multiple commands into one command

- class Tinkercell::ReverseUndoCommand

  this command can be used to invert another undo command (i.e. flip the redo/undo)

- class Tinkercell::ReplaceNodeGraphicsCommand

  this command can be used to replace the graphical representation of a node from an xml file

- class Tinkercell::AssignHandleCommand

  this command assigns handles to items

- class Tinkercell::MergeHandlesCommand

  this command places all the graphics items inside one handle into the other

- class Tinkercell::SetParentHandleCommand

  this command assigns parent(s) to one or more handles

- class Tinkercell::SetGraphicsSceneVisibilityCommand

  this command is used to hide graphics items. Hidden graphics items will be part (unless their handles are also hidden) of the network but not visible on the screen.

- class Tinkercell::SetHandleFamilyCommand

  this command is used to hide graphics items. Hidden graphics items will be part (unless their handles are also hidden) of the network but not visible on the screen.

- class Tinkercell::AddControlPointCommand

  An command that adds a new control point to a connection item; it has undo and redo functionality.

- class Tinkercell::RemoveControlPointCommand

  A command that removed control points. Allows undo and redo.

- class Tinkercell::AddCurveSegmentCommand

  An command that adds a new control point to a connection item; it has undo and redo functionality.

- class Tinkercell::RemoveCurveSegmentCommand

  A command that removed control points. Allows undo and redo.

- class Tinkercell::ReplaceConnectedNodeCommand

  this command replaces one node item in a connection item with another

Typedefs

- typedef ChangeDataCommand< QString > Tinkercell::ChangeTextDataCommand

  this command is used to replace text data inside a handle

- typedef ChangeDataCommand< qreal > Tinkercell::ChangeNumericalDataCommand

  this command is used to replace numerical data inside a handle

5.4.1 Detailed Description

A set of classes that allow undo/redo (using Qt Undo framework).

5.5 C API

C functions that are provided by the TinkerCell Core library and Plug-ins (tools).

Classes

- class Tinkercell::C_API_Slots

  A set of slots that are called by C libraries.

- class Tinkercell::CThread

  This class is used to run specific functions inside a C dynamic library as a separate thread. The class can be used to load a library or just run a specific function inside an already loaded library. If the library is loaded by this class, the library will be unloaded upon completion on the function. To prevent the automatic unloading, use the setAutoUnload option. Only four types of functions are supported.

- class Tinkercell::InterpreterThread

  This class is used to run interpreters such as python, perl, octave, R, etc. This is the parent class that provides the basic structure for loading the library that will embed one of these languages.

- class Tinkercell::OctaveInterpreterThread

  This class is used to embed an octave interpreter inside a TinkerCell application. The C library responsible for embedding octave is called runOctave.cpp and is located inside the octave folder. The octave interpreter uses two libraries -- one for embedding octave in TinkerCell and another for extending Octave with the TinkerCell C API.

- class Tinkercell::PythonInterpreterThread

  This class is used to embed an python interpreter inside a TinkerCell application. The C library responsible for embedding python is called runpy.c and is located inside the python/ folder.

5.5.1 Detailed Description

C functions that are provided by the TinkerCell Core library and Plug-ins (tools).

5.6 TinkerCell plug-ins

Plug-ins, which are classes that inheir from Tool class, provide the large majority of the important features in TinkerCell.

Classes

- class Tinkercell::PlotTool

  A docking widget that can contains one or more PlotWidget instances. Each PlotWidget can either be a text output, 2D graph, or 3D graph. Alternatively, the PlotTool can use an separate Gnuplot window to generate plots.

5.6.1 Detailed Description

Plug-ins, which are classes that inheir from Tool class, provide the large majority of the important features in TinkerCell.

Chapter 6

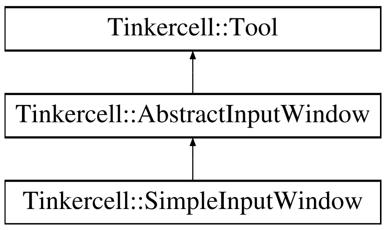
Class Documentation

6.1 Tinkercell::AbstractInputWindow Class Reference

Classes that inherit from this class can be used as GUI windows that provide interface to C programs (library files).

`#include <AbstractInputWindow.h>`

Inheritance diagram for Tinkercell::AbstractInputWindow:


Tinkercell::Tool
Tinkercell::AbstractInputWindow
Tinkercell::SimpleInputWindow

Public Slots

- virtual void escapeSignal (const QWidget ∗)

  Escape signal is a request to stop the current process. This class will hide itself as a response.

- virtual void exec ()

  Executes the CThread.

- virtual void loadAPI (Tool ∗)

  Uses MainWindow's setupNewThread function to setup this window's thread.

Signals

- void updateThread ()

  update the thread

Protected Member Functions

- AbstractInputWindow (const QString &name=tr(""), CThread ∗thread=0)

  constructor

- virtual bool setMainWindow (MainWindow ∗main)

  Sets the main window. This function will set this tool as a docked widget by default and registed the escapeSignal event. Overwrite this function to prevent that default behavior.

- virtual void setInput (const DataTable< qreal > &)

  set the input for this input window

- virtual void setThread (CThread ∗)

  set the thread that will be started by this input window

- virtual CThread ∗ thread () const

  the thread that will be started by this input window

- virtual void enterEvent (QEvent ∗event)

  when mouse enters this widget, the cthread is updated

Protected Attributes

- CThread ∗ cthread

  the target thread

- QDockWidget ∗ dockWidget

  the docked window for this widget (0 if not a docked widget)

- void(∗ targetFunction)(tc_matrix)

  target function for this input window

6.1.1 Detailed Description

Classes that inherit from this class can be used as GUI windows that provide interface to C programs (library files).

See also

LPSolveInput

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Tinkercell::AbstractInputWindow::AbstractInputWindow (const QString & *name* = `tr("")`, CThread ∗ *thread* = 0) `[protected]`

constructor

Parameters

QString name of this tool

CThread the target thread to run from this input window

6.1.3 Member Function Documentation

6.1.3.1 void Tinkercell::AbstractInputWindow::exec () `[virtual, slot]`

Executes the CThread.

See also

CThread

Reimplemented in Tinkercell::SimpleInputWindow.

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

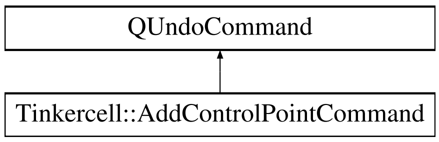
- AbstractInputWindow.h
- AbstractInputWindow.cpp

6.2 Tinkercell::AddControlPointCommand Class Reference

An command that adds a new control point to a connection item; it has undo and redo functionality.

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::AddControlPointCommand:



Public Member Functions

- AddControlPointCommand (const QString &name, GraphicsScene ∗scene, ConnectionGraphicsItem::ControlPoint ∗item)

  constructor that makes the command. If added to history stack, also does redo

- AddControlPointCommand (const QString &name, GraphicsScene ∗scene, QList< ConnectionGraphicsItem::ControlPoint ∗ > items)

  constructor that makes the command. If added to history stack, also does redo

- virtual ∼AddControlPointCommand ()

  destructor. deletes all control points that do not belong a scene

- void redo ()

  Adds a new control point. Control points were set in the constructor.

- void undo ()

  Remove new control points. Control points were set in the constructor.

Public Attributes

- GraphicsScene ∗ graphicsScene

  graphics scene to which control points were added

- QList< ConnectionGraphicsItem::ControlPoint ∗ > graphicsItems

  control points that were added

- QList< int > listK1

  the poisition(s) at which the control points were added

- QList< int > **listK2**

6.2.1 Detailed Description

An command that adds a new control point to a connection item; it has undo and redo functionality.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Tinkercell::AddControlPointCommand::AddControlPointCommand (const QString & *name*, GraphicsScene ∗ *scene*, ConnectionGraphicsItem::ControlPoint ∗ *item*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.2.2.2 Tinkercell::AddControlPointCommand::AddControlPointCommand (const QString & *name*, GraphicsScene ∗ *scene*, QList< ConnectionGraphicsItem::ControlPoint ∗ > *items*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.2.3 Member Function Documentation

6.2.3.1 void Tinkercell::AddControlPointCommand::redo ()

Adds a new control point. Control points were set in the constructor.

Parameters

void

Returns

void

6.2.3.2 void Tinkercell::AddControlPointCommand::undo ()

Remove new control points. Control points were set in the constructor.

Parameters

void

Returns

void

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

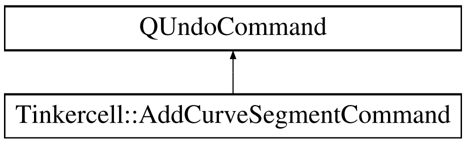
- UndoCommands.h
- UndoCommands.cpp

6.3 Tinkercell::AddCurveSegmentCommand Class Reference

An command that adds a new control point to a connection item; it has undo and redo functionality.

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::AddCurveSegmentCommand:



Public Member Functions

- AddCurveSegmentCommand (const QString &name, GraphicsScene ∗scene, ConnectionGraphicsItem ∗connection, ConnectionGraphicsItem::CurveSegment &item)

  constructor that makes the command. If added to history stack, also does redo

- AddCurveSegmentCommand (const QString &name, GraphicsScene ∗scene, ConnectionGraphicsItem ∗connection, QList< ConnectionGraphicsItem::CurveSegment > items)

  constructor that makes the command. If added to history stack, also does redo

- virtual ∼AddCurveSegmentCommand ()

  destructor. deletes all control points that do not belong a scene

- void redo ()

  Adds a new control point. Control points were set in the constructor.

- void undo ()

  Remove new control points. Control points were set in the constructor.

Public Attributes

- GraphicsScene ∗ graphicsScene

  graphics scene to which control points were added

- ConnectionGraphicsItem ∗ connectionItem

  graphics item to which control points were added

- QList< ConnectionGraphicsItem::CurveSegment > curveSegments

  vector of control points that were added

- QList< int > listK1

  the poisition(s) at which the control point vectors were added

6.3.1 Detailed Description

An command that adds a new control point to a connection item; it has undo and redo functionality.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Tinkercell::AddCurveSegmentCommand::AddCurveSegmentCommand (const QString & *name*, GraphicsScene ∗ *scene*, ConnectionGraphicsItem ∗ *connection*, ConnectionGraphicsItem::CurveSegment & *item*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.3.2.2 Tinkercell::AddCurveSegmentCommand::AddCurveSegmentCommand (const QString & *name*, GraphicsScene ∗ *scene*, ConnectionGraphicsItem ∗ *connection*, QList< ConnectionGraphicsItem::CurveSegment > *items*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.3.3 Member Function Documentation

6.3.3.1 void Tinkercell::AddCurveSegmentCommand::redo ()

Adds a new control point. Control points were set in the constructor.

Parameters

void

Returns

void

6.3.3.2 void Tinkercell::AddCurveSegmentCommand::undo ()

Remove new control points. Control points were set in the constructor.

Parameters

void

Returns

void

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

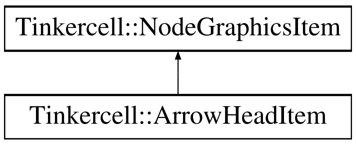
- UndoCommands.h
- UndoCommands.cpp

6.4 Tinkercell::ArrowHeadItem Class Reference

A node graphics item that is used to draw arrow heads on connection items.

```
#include <ConnectionGraphicsItem.h>
```

Inheritance diagram for Tinkercell::ArrowHeadItem:



Public Types

- enum { **Type** = UserType + 6 }

  for enabling dynamic_cast

Public Member Functions

- ArrowHeadItem (ConnectionGraphicsItem ∗c=0)

  constructor -- initializes the angle and connection item

- ArrowHeadItem (const QString &, ConnectionGraphicsItem ∗c=0)

  construct from file

- ArrowHeadItem (const ArrowHeadItem &)

  copy constructor

- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option=new QStyleOption-GraphicsItem(), QWidget ∗widget=0)

  paint this arrow item. performs rotation using the angle member.

- virtual NodeGraphicsItem ∗ clone () const

  returns a duplicate of this arrow head

- virtual int type () const

  for enabling dynamic_cast

Static Public Member Functions

- static ArrowHeadItem ∗ cast (QGraphicsItem ∗)

  cast a graphics item to a node graphics item using qgraphicsitem_cast

Public Attributes

- ConnectionGraphicsItem ∗ connectionItem

  The connection item that this arrow head belongs with.

- qreal angle

  the direction (angle) that the arrow is pointing

Static Public Attributes

- static const QString CLASSNAME = QString("ArrowHeadItem")

  for safe static casting

6.4.1 Detailed Description

A node graphics item that is used to draw arrow heads on connection items.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Tinkercell::ArrowHeadItem::ArrowHeadItem (ConnectionGraphicsItem ∗ *connection* = 0)

constructor -- initializes the angle and connection item

Constructor: init everything

6.4.2.2 Tinkercell::ArrowHeadItem::ArrowHeadItem (const QString & *filename*, ConnectionGraphicsItem ∗ *connection* = 0)

construct from file

Constructor: init everything

6.4.2.3 Tinkercell::ArrowHeadItem::ArrowHeadItem (const ArrowHeadItem & *copy*)

copy constructor

Constructor: init everything

6.4.3 Member Function Documentation

6.4.3.1 ArrowHeadItem ∗ Tinkercell::ArrowHeadItem::cast (QGraphicsItem ∗ *q*) `[static]`

cast a graphics item to a node graphics item using qgraphicsitem_cast

Parameters

QGraphicsItem∗	graphics item

Returns

ArrowHeadItem∗ can be 0 if the cast is invalid

Reimplemented from Tinkercell::NodeGraphicsItem.

6.4.3.2 NodeGraphicsItem ∗ Tinkercell::ArrowHeadItem::clone () const `[virtual]`

returns a duplicate of this arrow head

make a copy of this item

Returns

duplicate arrow head item

Reimplemented from Tinkercell::NodeGraphicsItem.

6.4.3.3 void Tinkercell::ArrowHeadItem::paint (QPainter ∗ *painter*, const QStyleOptionGraphicsItem ∗ *option* = `new QStyleOptionGraphicsItem()`, QWidget ∗ *widget* = `0`) `[virtual]`

paint this arrow item. performs rotation using the angle member.

Returns

void

Reimplemented from Tinkercell::NodeGraphicsItem.

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

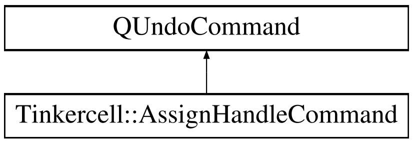
- ConnectionGraphicsItem.h
- ConnectionGraphicsItem.cpp

6.5 Tinkercell::AssignHandleCommand Class Reference

this command assigns handles to items

```
#include <UndoCommands.h>
```

Inheritance diagram for Tinkercell::AssignHandleCommand:



Public Member Functions

- **AssignHandleCommand** (const QString &text, QGraphicsItem ∗item, ItemHandle ∗handle)
- **AssignHandleCommand** (const QString &text, const QList< QGraphicsItem ∗ > &items, ItemHandle ∗handle)
- **AssignHandleCommand** (const QString &text, const QList< QGraphicsItem ∗ > &items, QList< ItemHandle ∗ > &handles)
- void **redo** ()
- void **undo** ()

Public Attributes

- QList< QGraphicsItem ∗ > **graphicsItems**
- QList< ItemHandle ∗ > **oldHandles**
- QList< ItemHandle ∗ > **newHandles**

6.5.1 Detailed Description

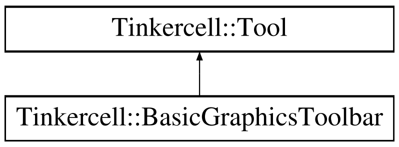
this command assigns handles to items

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

- UndoCommands.h
- UndoCommands.cpp

6.6 Tinkercell::BasicGraphicsToolbar Class Reference

Inheritance diagram for Tinkercell::BasicGraphicsToolbar:



Public Slots

- void **setBackgroundImage** ()
- void **unsetBackgroundImage** ()
- void **bringToFront** ()
- void **sendToBack** ()
- void **zoomIn** ()
- void **find** ()
- void **closeFind** ()
- void **rename** ()
- void **zoomOut** ()
- void **fitAll** ()
- void **changeBrush** ()
- void **changePen** ()
- void **selectBrushColor1** ()
- void **selectBrushAlpha1** ()
- void **selectBrushColor2** ()
- void **selectBrushAlpha2** ()
- void **selectPenWidth** ()
- void **noGradient** ()
- void **linearGradient** ()
- void **radialGradient** ()
- void **alignLeft** ()
- void **alignRight** ()
- void **alignTop** ()
- void **alignBottom** ()
- void **alignCompactVertical** ()
- void **alignCompactHorizontal** ()
- void **alignEvenSpacedVertical** ()
- void **alignEvenSpacedHorizontal** ()
- void **alignSelected** ()
- void **mousePressed** (GraphicsScene ∗scene, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers)
- void **mouseDragged** (GraphicsScene ∗scene, QPointF from, QPointF to, Qt::MouseButton, Qt::KeyboardModifiers modifiers)
- void **mouseMoved** (GraphicsScene ∗scene, QGraphicsItem ∗item, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers, QList< QGraphicsItem ∗ > &)
- void **mouseReleased** (GraphicsScene ∗scene, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers)
- void **escapeSlot** (const QWidget ∗)

Public Member Functions

- bool setMainWindow (MainWindow ∗main)

  set the main window for this tool

Protected Types

- enum **Mode** {
  none, **gradient**, **brush**, **pen**,
  zoom, **unzoom** }
- enum **AlignMode** {
  left, **right**, **bottom**, **top**,
  centervertical, **centerhorizontal**, **evenspacedvertical**, **evenspacedhorizontal**,
  compactvertical, **compacthorizontal** }

Protected Member Functions

- QList< QGraphicsItem ∗ > **itemsToAlign** (QList< QGraphicsItem ∗ > &)
- void **moveTextGraphicsItems** (QList< QGraphicsItem ∗ > &, QList< QPointF > &)
- void **moveChildItems** (QList< QGraphicsItem ∗ > &, QList< QPointF > &)
- void **init** ()

Protected Attributes

- QList< QGraphicsItem ∗ > **targetItems**
- QGradient::Type **gradientType**
- QPointF **gradientPos1**
- QPointF **gradientPos2**
- QToolBar ∗ **findToolBar**
- QColor **brushColor1**
- QColor **brushColor2**
- QColor **penColor**
- qreal **penWidth**
- QAction ∗ **changeBrushColor1**
- QAction ∗ **changeBrushColor2**
- QAction ∗ **changePenWidth**
- QAction ∗ **changeBrushAlpha1**
- QAction ∗ **changeBrushAlpha2**
- QAction ∗ **findAction**
- QSpinBox ∗ **brushAlpha1**
- QSpinBox ∗ **brushAlpha2**
- QSpinBox ∗ **penAlpha**
- QLineEdit ∗ **findText**
- QLineEdit ∗ **replaceText**
- QMenu ∗ **gradientMenu**
- QIcon **linearGradientIcon**
- QIcon **radialGradientIcon**

- Mode **mode**
- QGraphicsRectItem **zoomRect**
- QAction * **alignButton**
- AlignMode **alignMode**

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

- BasicGraphicsToolbar.h
- BasicGraphicsToolbar.cpp

6.7 Tinkercell::C_API_Slots Class Reference

A set of slots that are called by C libraries.

```
#include <C_API_Slots.h>
```

Signals

- void **saveNetwork** (const QString &)

Public Member Functions

- **C_API_Slots** (MainWindow ∗)

6.7.1 Detailed Description

A set of slots that are called by C libraries.

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

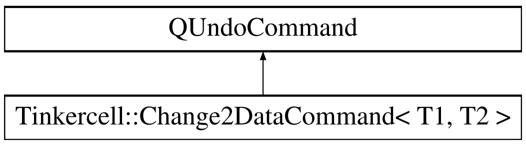
- C_API_Slots.h
- C_API_Slots.cpp

6.8 Tinkercell::Change2DataCommand< T1, T2 > Class Template Reference

Changes two different data tables.

`#include <DataTable.h>`

Inheritance diagram for Tinkercell::Change2DataCommand< T1, T2 >:



Public Member Functions

- Change2DataCommand (const QString &name, DataTable< T1 > *oldDataTable1, const DataTable< T1 > *newDataTable1, DataTable< T2 > *oldDataTable2, const DataTable< T2 > *newDataTable2)

  constructor

- Change2DataCommand (const QString &name, const QList< DataTable< T1 > * > &oldDataTable1, const QList< DataTable< T1 > * > &newDataTable1, const QList< DataTable< T2 > * > &oldDataTable2, const QList< DataTable< T2 > * > &newDataTable2)

  constructor

- void redo ()

  redo the changes

- void undo ()

  undo the changes

Public Attributes

- QList< DataTable< T1 > * > targetDataTable1

  target tables of type T1

- QList< DataTable< T1 > > newDataTable1

  new tables of type T1

- QList< DataTable< T1 > > oldDataTable1

  old tables of type T1

- QList< DataTable< T2 > * > targetDataTable2

  target tables of type T2

- QList< DataTable< T2 > > newDataTable2

  new tables of type T2

- QList< DataTable< T2 > > oldDataTable2

    old tables of type T2

6.8.1 Detailed Description

template<typename T1, typename T2> class Tinkercell::Change2DataCommand< T1, T2 >

Changes two different data tables.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 template<typename T1, typename T2> Tinkercell::Change2DataCommand< T1, T2 >::Change2DataCommand (const QString & *name*, DataTable< T1 > ∗ *oldDataTable1*, const DataTable< T1 > ∗ *newDataTable1*, DataTable< T2 > ∗ *oldDataTable2*, const DataTable< T2 > ∗ *newDataTable2*) `[inline]`

constructor

Parameters

name	of the command
old	table of type T1
new	table of type T1
old	table of type T2
new	table of type T2

6.8.2.2 template<typename T1, typename T2> Tinkercell::Change2DataCommand< T1, T2 >::Change2DataCommand (const QString & *name*, const QList< DataTable< T1 > ∗ > & *oldDataTable1*, const QList< DataTable< T1 > ∗ > & *newDataTable1*, const QList< DataTable< T2 > ∗ > & *oldDataTable2*, const QList< DataTable< T2 > ∗ > & *newDataTable2*) `[inline]`

constructor

Parameters

name	of the command
old	tables of type T1
new	tables of type T1
old	tables of type T2
new	tables of type T2

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

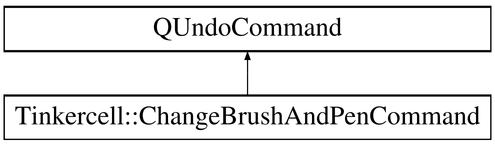
- DataTable.h

6.9 Tinkercell::ChangeBrushAndPenCommand Class Reference

this command changes the pen and/or brush of an item

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ChangeBrushAndPenCommand:



Public Member Functions

- ChangeBrushAndPenCommand (const QString &name, QGraphicsItem ∗item, const QBrush &brush, const QPen &pen)

  constructor

- ChangeBrushAndPenCommand (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QBrush > &brushes, const QList< QPen > &pens)

  constructor

- void **redo** ()
- void **undo** ()

6.9.1 Detailed Description

this command changes the pen and/or brush of an item

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Tinkercell::ChangeBrushAndPenCommand::ChangeBrushAndPenCommand (const QString & *name*, QGraphicsItem ∗ *item*, const QBrush & *brush*, const QPen & *pen*)

constructor

Parameters

QString	name of command
GraphicsScene∗	scene where change happened
QGraphicsItem∗	item that is affected
QBrush	new brushes (one for each item)
QPen	new pens (one for each item)

6.9.2.2 Tinkercell::ChangeBrushAndPenCommand::ChangeBrushAndPenCommand (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QBrush > & *brushes*, const QList< QPen > & *pens*)

constructor

Parameters

QString name of command

GraphicsScene∗ scene where change happened

QList<QGraphicsItem∗>& items that are affected

QList<QBrush>& new brushes (one for each item)

QList<QPen>& new pens (one for each item)

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

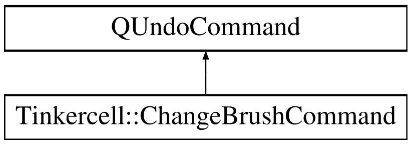
- UndoCommands.h
- UndoCommands.cpp

6.10 Tinkercell::ChangeBrushCommand Class Reference

this command changes the brush of an item

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ChangeBrushCommand:



Public Member Functions

- ChangeBrushCommand (const QString &name, QGraphicsItem ∗item, const QBrush &to)

  constructor

- ChangeBrushCommand (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QBrush > &to)

  constructor

- void **redo** ()
- void **undo** ()

6.10.1 Detailed Description

this command changes the brush of an item

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Tinkercell::ChangeBrushCommand::ChangeBrushCommand (const QString & *name*, QGraphicsItem ∗ *item*, const QBrush & *to*)

constructor

Parameters

QString name of command

***GraphicsScene*∗** scene where change happened

***QGraphicsItem*∗** item that is affected

QBrush new brush

6.10.2.2 Tinkercell::ChangeBrushCommand::ChangeBrushCommand (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QBrush > & *to*)

constructor

Parameters

QString name of command

***GraphicsScene*∗** scene where change happened

QList<QGraphicsItem∗>& items that are affected

QList<QBrush>& new brushes (one for each item)

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

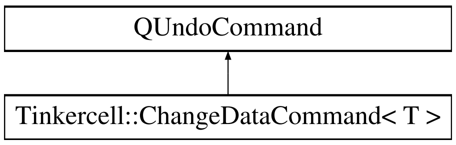
- UndoCommands.h
- UndoCommands.cpp

6.11 Tinkercell::ChangeDataCommand< T > Class Template Reference

This template class allows undo and redo of a change made to a data table.

`#include <DataTable.h>`

Inheritance diagram for Tinkercell::ChangeDataCommand< T >:



Public Member Functions

- ChangeDataCommand (const QString &name, DataTable< T > ∗oldDataTable, const DataTable< T > ∗newDataTable)

  constructor

- ChangeDataCommand (const QString &name, const QList< DataTable< T > ∗ > &oldDataTable, const QList< DataTable< T > ∗ > &newDataTable)

  constructor

- void redo ()

  redo the changes

- void undo ()

  undo the changes

Public Attributes

- QList< DataTable< T > ∗ > targetDataTable

  pointers to target tables

- QList< DataTable< T > > newDataTable

  new tables

- QList< DataTable< T > > oldDataTable

  old tables

6.11.1 Detailed Description

template<typename T> class Tinkercell::ChangeDataCommand< T >

This template class allows undo and redo of a change made to a data table.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 template<typename T > Tinkercell::ChangeDataCommand< T >::ChangeDataCommand (const QString & *name*, DataTable< T > ∗ *oldDataTable*, const DataTable< T > ∗ *newDataTable*) `[inline]`

constructor

Parameters

- ***name*** of the change
- ***old*** tables
- ***new*** tables

6.11.2.2 template<typename T > Tinkercell::ChangeDataCommand< T >::ChangeDataCommand (const QString & *name*, const QList< DataTable< T > ∗ > & *oldDataTable*, const QList< DataTable< T > ∗ > & *newDataTable*) `[inline]`

constructor

Parameters

- ***name*** of the change
- ***old*** table
- ***new*** table

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

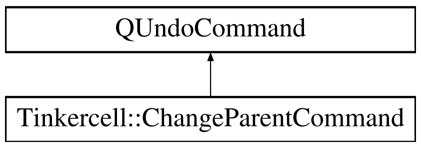
- DataTable.h

6.12 Tinkercell::ChangeParentCommand Class Reference

this command changes the parent of a graphics item (not handles)

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ChangeParentCommand:



Public Member Functions

- ChangeParentCommand (const QString &name, QGraphicsScene ∗scene, QGraphicsItem ∗item, QGraphicsItem ∗newParent)

  constructor

- ChangeParentCommand (const QString &name, QGraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, const QList< QGraphicsItem ∗ > &newParents)

  constructor

- void **redo** ()
- void **undo** ()

6.12.1 Detailed Description

this command changes the parent of a graphics item (not handles)

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Tinkercell::ChangeParentCommand::ChangeParentCommand (const QString & *name*, QGraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, QGraphicsItem ∗ *newParent*)

constructor

Parameters

QString	name of command
GraphicsScene∗	scene where change happened
QGraphicsItem∗	item that is affected
QGraphicsItem∗	new parent item

6.12.2.2 Tinkercell::ChangeParentCommand::ChangeParentCommand (const QString & *name*, QGraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QList< QGraphicsItem ∗ > & *newParents*)

constructor

Parameters

- ***QString*** name of command
- ***GraphicsScene**** scene where change happened
- ***QList<QGraphicsItem*** *>& items that are affected
- ***QList<QGraphicsItem*** *>& new parent items

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

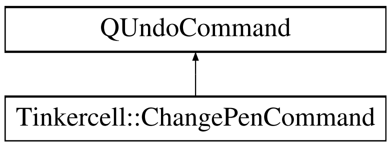
- UndoCommands.h
- UndoCommands.cpp

6.13 Tinkercell::ChangePenCommand Class Reference

this command changes the pen of an item

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ChangePenCommand:



Public Member Functions

- ChangePenCommand (const QString &name, QGraphicsItem ∗item, const QPen &to)

  constructor

- ChangePenCommand (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QPen > &to)

  constructor

- void **redo** ()
- void **undo** ()

6.13.1 Detailed Description

this command changes the pen of an item

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Tinkercell::ChangePenCommand::ChangePenCommand (const QString & *name*, QGraphicsItem ∗ *item*, const QPen & *to*)

constructor

Parameters

QString name of command

***GraphicsScene*∗** scene where change happened

***QGraphicsItem*∗** item that is affected

QBrush new pen

6.13.2.2 Tinkercell::ChangePenCommand::ChangePenCommand (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QPen > & *to*)

constructor

Parameters

QString name of command

***GraphicsScene*∗** scene where change happened

QList<QGraphicsItem∗>& items that are affected

QList<QPen>& new pens (one for each item)

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

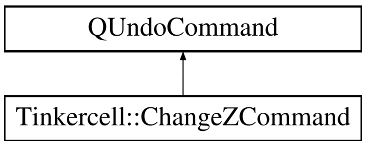
- UndoCommands.h
- UndoCommands.cpp

6.14 Tinkercell::ChangeZCommand Class Reference

this command changes the pen of an item

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ChangeZCommand:



Public Member Functions

- ChangeZCommand (const QString &name, QGraphicsScene ∗scene, QGraphicsItem ∗item, qreal to)

  constructor
- ChangeZCommand (const QString &name, QGraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, const QList< qreal > &to)

  constructor
- void **redo** ()
- void **undo** ()

6.14.1 Detailed Description

this command changes the pen of an item

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Tinkercell::ChangeZCommand::ChangeZCommand (const QString & *name*, QGraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, qreal *to*)

constructor

Parameters

QString	name of command
GraphicsScene∗	scene where change happened
QGraphicsItem∗	item that is affected
double	new Z value

6.14.2.2 Tinkercell::ChangeZCommand::ChangeZCommand (const QString & *name*, QGraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QList< qreal > & *to*)

constructor

Parameters

QString name of command

GraphicsScene* scene where change happened

QList<QGraphicsItem*>& item that is affected

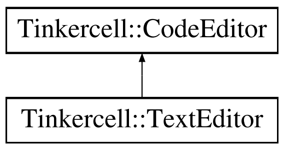
QList<qreal>& new Z (one for each item)

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

- UndoCommands.h
- UndoCommands.cpp

6.15 Tinkercell::CodeEditor Class Reference

Inheritance diagram for Tinkercell::CodeEditor:



Public Slots

- void **setText** (const QString &)

Public Member Functions

- **CodeEditor** (QWidget ∗parent=0)
- void **lineNumberAreaPaintEvent** (QPaintEvent ∗event)
- int **lineNumberAreaWidth** ()
- void **setCompleter** (QCompleter ∗c)
- QCompleter ∗ **completer** () const
- void **zoomIn** (int r=1)
- void **zoomOut** (int r=1)
- QString **text** () const

Public Attributes

- QWidget ∗ **lineNumberArea**
- QColor **lineHighlightColor**
- QColor **lineNumberBackground**
- QColor **lineNumberText**

Protected Member Functions

- void **resizeEvent** (QResizeEvent ∗event)
- virtual void **wheelEvent** (QWheelEvent ∗wheelEvent)
- void **keyPressEvent** (QKeyEvent ∗e)
- void **focusInEvent** (QFocusEvent ∗e)

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

- CodeEditor.h
- CodeEditor.cpp

6.16 Tinkercell::CommandTextEdit Class Reference

A command-line type text box that other tools can use for scripting interface.

`#include <ConsoleWindow.h>`

Public Slots

- virtual void eval (const QString &)

  evaluate a command (just emits a commandExecuted signal)

- virtual void error (const QString &)

  post an error message to this console text box

- virtual void message (const QString &)

  post a message to this console text box

- virtual void clearText ()

  clear all text

- virtual void freeze ()

  equivalent to setFreeze(true)

- virtual void unfreeze ()

  equivalent to setFreeze(false)

- virtual void setFreeze (bool frozen=true)

  Set frozen state. The text box will not respond to user inputs while it is frozen.

- virtual void setBackgroundColor (const QColor &)

  set background color

- virtual void setPlainTextColor (const QColor &)

  set plain text color

- virtual void setOutputTextColor (const QColor &)

  set output message color

- virtual void setErrorTextColor (const QColor &)

  set error message color

- virtual void setTableTextColor (const QColor &)

  set table headers color

Signals

- void commandExecuted (const QString &command)

  the user requested to execute the given command

- void commandInterrupted ()

  the user requested to interrupt the current process

Public Member Functions

- CommandTextEdit (MainWindow ∗parent=0)

  default constructor

- virtual bool isFrozen ()

  Whether or not this console in the frozen state. The text box will not add or remove text while it is frozen.

- void setCompleter (QCompleter ∗c)

  set code completion

- QCompleter ∗ completer () const

  code completion

Protected Member Functions

- virtual void keyPressEvent (QKeyEvent ∗event)

  manages the console-type interface, where the user is not allowed to type outside the >>

- virtual void wheelEvent (QWheelEvent ∗wheelEvent)

  zoom in or out using mouse wheel

- virtual void focusInEvent (QFocusEvent ∗e)

  focus returned from code completer

Protected Attributes

- QStringList historyStack

  list of previously executed commands

- QStringList messagesStack

  list of messages pending

- QStringList errorsStack

  list of errors pending

- int currentHistoryIndex

  current position in the history of commands

- int currentPosition

  current position of the cursor in the text box

- bool frozen

  frozen state = 0 or 1

- QTextCharFormat errorFormat

  font format for error messages

- QTextCharFormat messageFormat

  font format for regular messages

- QTextCharFormat tableHeaderFormat

  font format for table headers

- QTextCharFormat normalFormat

  font format for user inputs

6.16.1 Detailed Description

A command-line type text box that other tools can use for scripting interface.

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

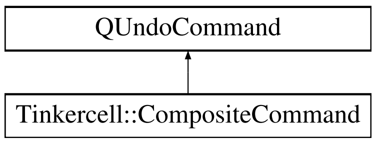
- ConsoleWindow.h
- ConsoleWindow.cpp

6.17 Tinkercell::CompositeCommand Class Reference

this command can be used to combine multiple commands into one command

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::CompositeCommand:



Public Member Functions

- CompositeCommand (const QString &, const QList< QUndoCommand ∗ > &, const QList< QUndoCommand ∗ > &noClear=QList< QUndoCommand ∗ >())

  Constructor. Composite command takes ownership of these commands unless specified otherwise.

- CompositeCommand (const QString &, QUndoCommand ∗, QUndoCommand ∗, bool deleteCommands=true)

  constructor for grouping two commands. Composite command takes ownership of these commands unless specified otherwise.

- ∼CompositeCommand ()

  destructor automatically deletes any command not in the doNotDelete list

- void redo ()

  undo

- void undo ()

  undo

Public Attributes

- QList< QUndoCommand ∗ > commands

  commands grouped inside this composite command

- QList< QUndoCommand ∗ > doNotDelete

  commands that should not be deleted along with the composite command

6.17.1 Detailed Description

this command can be used to combine multiple commands into one command

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Tinkercell::CompositeCommand::CompositeCommand (const QString & *name*, const QList< QUndoCommand ∗ > & *list*, const QList< QUndoCommand ∗ > & *noClear* = `QList`<QUndoCommand∗>`()`)

Constructor. Composite command takes ownership of these commands unless specified otherwise.

Parameters

QString name of command

QList<QUndoCommand∗>& the commands that make up this composite command

QList<QUndoCommand∗>& the commands that should not be deleted by composite command's destructor (default = none)

6.17.2.2 Tinkercell::CompositeCommand::CompositeCommand (const QString & *name*, QUndoCommand ∗ *cmd1*, QUndoCommand ∗ *cmd2*, bool *deleteCommands* = `true`)

constructor for grouping two commands. Composite command takes ownership of these commands unless specified otherwise.

Parameters

QString name of command

QUndoCommand∗ a command to be gouped

QUndoCommand∗ another command to be gouped

bool delete both commands automatically (default = true)

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

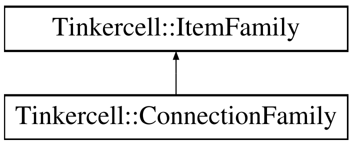
- UndoCommands.h
- UndoCommands.cpp

6.18 Tinkercell::ConnectionFamily Class Reference

This class defines the family of a connection. Inherits from ItemFamily It contains a list ofConnectio-GraphicsItems that is the default for this family of connections.

```
#include <ItemFamily.h>
```

Inheritance diagram for Tinkercell::ConnectionFamily:



Public Member Functions

- virtual ItemFamily ∗ parent () const

  get the parent for this family. If there are more than one parents, returns the first

- virtual QList< ItemFamily ∗ > parents () const

  get all the parents for this family.

- virtual QList< ItemFamily ∗ > children () const

  get all the families that make up this family.

- virtual void setParent (ConnectionFamily ∗)

  set parent family

- virtual ∼ConnectionFamily ()

  destructor.

- ConnectionFamily (const QString &name=QString())

  constructor.

- virtual bool isA (const QString &) const

  indicates whether or not the given string is the name of this family or any of its parent families

- virtual bool isA (const ItemFamily ∗) const

  indicates whether or not the given family is the name of this family or any of its parent families

  - virtual bool addParticipant (const QString &role, const QString &family)

    add a participant

  - virtual QString participantFamily (const QString &role) const

    get participant family

  - virtual QStringList participantRoles () const

    get all participant roles

- virtual QStringList participantTypes () const

  get all participant family names

- virtual bool isValidSet (const QList< NodeHandle ∗ > &nodes, bool checkFull=true)

  checks if this family is compatible with a connection composed of the given set of nodes

- virtual QList< ItemFamily ∗ > findValidChildFamilies (const QList< NodeHandle ∗ > &, bool checkFull=true)

  find child-families of this family that the given set of nodes can potentially belong with

- virtual int numberOfIdenticalNodesFamilies (ConnectionFamily ∗) const

  finds the number of node families that are common between the two connections (exactly the same, not using isA)

Static Public Member Functions

- static ConnectionFamily ∗ cast (ItemFamily ∗)

  cast to connection family

Protected Member Functions

- virtual bool isA (int) const

  indicates whether or not the given ID is this family or any of its parent families

Protected Attributes

- QList< ConnectionFamily ∗ > parentFamilies

  all the parents

- QList< ConnectionFamily ∗ > childFamilies

  all the families that are under this family

- QList< QPair< int, int > > nodeRoles

  the role ID and type ID of each node that is involved in this connection

Static Protected Attributes

- static QHash< QString, int > ROLEID

  stored a list of all possible node roles as IDs

- static QStringList ALLROLENAMES

  all role names. used to assign role IDs

6.18.1 Detailed Description

This class defines the family of a connection. Inherits from ItemFamily It contains a list ofConnectio-GraphicsItems that is the default for this family of connections.

6.18.2 Member Function Documentation

6.18.2.1 bool Tinkercell::ConnectionFamily::addParticipant (const QString & *role*, const QString & *family*) `[virtual]`

add a participant

in a connection and related functions

Parameters

QString role of participant

QString type of participant, must be a family name of a node

Returns

bool false if the participant family does not exist (i.e role not added)

6.18.2.2 QList< ItemFamily ∗ > Tinkercell::ConnectionFamily::findValidChildFamilies (const QList< NodeHandle ∗ > & *nodes*, bool *checkFull* = `true`) `[virtual]`

find child-families of this family that the given set of nodes can potentially belong with

Parameters

bool QList<NodeHandle∗> node handles

bool use false here if the list of nodes is a partial list

Returns

QList<ItemFamily∗> valid connection families

6.18.2.3 bool Tinkercell::ConnectionFamily::isA (int *id*) const `[protected, virtual]`

indicates whether or not the given ID is this family or any of its parent families

indicates whether or not the given string is the name of this family or any of its parent families

Reimplemented from Tinkercell::ItemFamily.

6.18.2.4 bool Tinkercell::ConnectionFamily::isValidSet (const QList< NodeHandle ∗ > & *nodes*, bool *checkFull* = `true`) `[virtual]`

checks if this family is compatible with a connection composed of the given set of nodes

Parameters

bool QList<NodeHandle∗> node handles

bool use false here if the list of nodes is a partial list

Returns

Boolean

6.18.2.5 int Tinkercell::ConnectionFamily::numberOfIdenticalNodesFamilies (ConnectionFamily ∗ *other*) const `[virtual]`

finds the number of node families that are common between the two connections (exactly the same, not using isA)

Parameters

ConnectionFamily ∗

Returns

bool

6.18.2.6 QString Tinkercell::ConnectionFamily::participantFamily (const QString & *role*) const `[virtual]`

get participant family

Parameters

QString role of participant

Returns

QString family name (empty if none)

6.18.2.7 QStringList Tinkercell::ConnectionFamily::participantRoles () const `[virtual]`

get all participant roles

Returns

QStringList role names (may not be unique)

6.18.2.8 QStringList Tinkercell::ConnectionFamily::participantTypes () const `[virtual]`

get all participant family names

Returns

QStringList family names (may not be unique)

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

- ItemFamily.h
- ItemFamily.cpp

6.19 Tinkercell::ConnectionGraphicsItem Class Reference

A graphics nodes item that draws connection between two or more nodes and the arrow heads at the ends.

`#include <ConnectionGraphicsItem.h>`

Classes

- class ControlPoint

  A control point with a pointer to a ConnectionGraphicsItem.

- class CurveSegment

  A set of control points and two arrow heads.

Public Types

- enum LineType { **line**, **bezier** }

  line or beizier

- enum { **Type** = UserType + 5 }

  for enabling dynamic_cast

Public Member Functions

- ConnectionGraphicsItem (QGraphicsItem ∗parent=0)
- ConnectionGraphicsItem (const QList< NodeGraphicsItem ∗ > &, const QList< NodeGraphicsItem ∗ > &, QGraphicsItem ∗parent=0)
- ConnectionGraphicsItem (const ConnectionGraphicsItem &copy)
- virtual ConnectionGraphicsItem & operator= (const ConnectionGraphicsItem &copy)
- virtual ConnectionGraphicsItem & copyPoints (const ConnectionGraphicsItem &copy)
- virtual ConnectionGraphicsItem ∗ clone () const

  make a copy of this connection item

- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option=new QStyleOptionGraphicsItem(), QWidget ∗widget=0)

  returns the bounding rectangle for this reaction figure

- virtual bool isValid ()

  checks that this is a valid drawable

- virtual ItemHandle ∗ handle () const

  get the handle of this connection

- virtual void setHandle (ItemHandle ∗)

  set the handle of this connection

- virtual QList< ControlPoint ∗ > controlPoints (bool includeEnds=false) const *list of pointers to all the control points*

- virtual QList< QGraphicsItem ∗ > controlPointsAsGraphicsItems (bool includeEnds=false) const

  list of pointers to all the control points

- virtual QPainterPath shape () const

  gets a path that represents this reactionimage

- virtual void clear (bool all=false)

  Clear all shapes and control points.

- virtual void refresh (bool arrows=true)

  refresh the path if any controlpoints have moved

- virtual void setControlPointsVisible (bool visible=true)

  set visibility of control points

- void showControlPoints ()

  show control points. same as setControlPointsVisible(true)

- void hideControlPoints ()

  hide control points. same as setControlPointsVisible(false)

- virtual bool isModifier () const

  check is this connection represents a modifier, i.e. points to the centerRegion of another connection

- virtual QList< NodeGraphicsItem ∗ > nodes () const

  get all nodes that are connected

- virtual QList< NodeGraphicsItem ∗ > nodesWithArrows () const

  get all nodes that have an arrow pointing to them

- virtual QList< NodeGraphicsItem ∗ > nodesWithoutArrows () const

  get all nodes that do NOT have an arrow pointing to them

- virtual QList< NodeGraphicsItem ∗ > nodesDisconnected () const

  get all nodes that are not directle connected to the main connection, such as modifier nodes

- virtual QList< QGraphicsItem ∗ > nodesAsGraphicsItems () const

  get all nodes that are connected

- virtual QList< ArrowHeadItem ∗ > arrowHeads () const

  get all the arrowHeads associated with the nodes. The order is the same order as nodes(), so values can be 0

- virtual QList< QGraphicsItem ∗ > arrowHeadsAsGraphicsItems () const

  get all the arrowHeads associated with the nodes The order is the same order as nodes(), so values can be 0

- virtual QList< ArrowHeadItem ∗ > modifierArrowHeads () const

  get all the arrowHeads NOT associated with the nodes

- virtual NodeGraphicsItem ∗ nodeAt (int index) const
  get the node that connected to the particular path
- virtual int indexOf (QGraphicsItem ∗node) const
  get the index of the node
- virtual void replaceNodeAt (int, NodeGraphicsItem ∗)
  replace the node at the particular position with a new node
- virtual void replaceNode (NodeGraphicsItem ∗, NodeGraphicsItem ∗)
  replace one node in the reaction with another
- virtual ArrowHeadItem ∗ arrowAt (int index) const
  get the arrow head at the particular index
- virtual ArrowHeadItem ∗ modifierArrowAt (int index) const
  get the modifier arrow head at the particular index
- virtual ∼ConnectionGraphicsItem ()
- virtual qreal slopeAtPoint (const QPointF &point)
  get slope at the given point (or closest point)
- virtual ControlPoint ∗ centerPoint () const
  the center point (if one exists)
- virtual QPointF centerLocation () const
  the center point (if one exists)
- virtual QRectF boundingRect () const
  bounding rect
- virtual QRectF sceneBoundingRect () const
  scene bounding rect
- virtual int type () const
  for enabling dynamic_cast

Static Public Member Functions

- static ConnectionGraphicsItem ∗ cast (QGraphicsItem ∗)
  cast a graphics item to a connection graphics item using qgraphicsitem_cast
- static QList< ConnectionGraphicsItem ∗ > cast (const QList< QGraphicsItem ∗ > &)
  cast a list of graphics item to a list of connection graphics items using qgraphicsitem_cast
- static ConnectionGraphicsItem ∗ topLevelConnectionItem (QGraphicsItem ∗item, bool includeControlPoints=false)
  gets the connection graphics item from its child item

Public Attributes

- QString name

  just a name used identifying the connection

- QString className

  used for checking type before static casts

- QBrush defaultBrush

  permanent brush for this control point

- QPen defaultPen

  permanent pen for this control point

- QString groupID

  for identifying which scene this item belongs in

- LineType lineType

  type of line for this reaction - line or beizier

- QList< CurveSegment > curveSegments

  vector of vector of control point

- qreal arrowHeadDistance

  distance from arrow head to the item that it is connected to

- bool controlPointsVisible

  indicates whether to show lines around the curves

- QSizeF centerRegion

  a rectangle that sits at the center of the connector

- ArrowHeadItem ∗ centerRegionItem

  the image on the rectangle that sits at the center of the connector

Static Public Attributes

- static const QString CLASSNAME = QString("ConnectionGraphicsItem")

  used for checking type before static casts

- static QString DefaultMiddleItemFile

  used to initialize the middle item for a connection

- static QString DefaultArrowHeadFile

  used to initialize the arrow heads for a connection

- static const int numLineTypes = 2

  number of different type of shapes available

Protected Member Functions

- virtual void refreshBoundaryPath ()

  update the boundary path

- virtual void adjustEndPoints (bool arrows=true)

  adjust the end control points so that they point straight

Protected Attributes

- ItemHandle ∗ itemHandle

  Tinkercell object that this drawable belongs in.

- QGraphicsPathItem ∗ boundaryPathItem

  path of the boundary region of the entire connection

- QPainterPath pathShape

  path of the selection region of the entire connection

- QRectF pathBoundingRect

  the boundary rectangle for this path. It is recomputed during each refresh.

6.19.1 Detailed Description

A graphics nodes item that draws connection between two or more nodes and the arrow heads at the ends.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Tinkercell::ConnectionGraphicsItem::ConnectionGraphicsItem (QGraphicsItem ∗ *parent* = 0)

Constructor: does nothing

Constructor: initialize everything

6.19.2.2 Tinkercell::ConnectionGraphicsItem::ConnectionGraphicsItem (const QList< NodeGraphicsItem ∗ > & *from*, const QList< NodeGraphicsItem ∗ > & *to*, QGraphicsItem ∗ *parent* = 0)

Constructor: constructs linear curve segments with arrow heads on the second set of nodes

Parameters

QList<NodeGraphicsItem∗>	list of nodes to connect from (no arrow heads)
QList<NodeGraphicsItem∗>	list of nodes to connect to (have arrow heads)

6.19.2.3 Tinkercell::ConnectionGraphicsItem::ConnectionGraphicsItem (const ConnectionGraphicsItem & *copy*)

Copy Constructor: copies handle but not control points

Copy Constructor: deep copy of all pointers

6.19.2.4 Tinkercell::ConnectionGraphicsItem::∼ConnectionGraphicsItem () `[virtual]`

Destructor: deletes all control points

Destructor: deletes all shapes and control points

6.19.3 Member Function Documentation

6.19.3.1 void Tinkercell::ConnectionGraphicsItem::adjustEndPoints (bool *arrowTransform* = `true`) `[protected, virtual]`

adjust the end control points so that they point straight

Parameters

bool adjust arrow transformations
void

Returns

void

6.19.3.2 ArrowHeadItem ∗ Tinkercell::ConnectionGraphicsItem::arrowAt (int *index*) const `[virtual]`

get the arrow head at the particular index

find the arrow head at the particular index

Parameters

index less than size of curveSegments

Returns

node item or 0

6.19.3.3 QList< ArrowHeadItem ∗ > Tinkercell::ConnectionGraphicsItem::arrowHeads () const `[virtual]`

get all the arrowHeads associated with the nodes. The order is the same order as nodes(), so values can be 0

get all the arrow heads in the same order as nodes

Returns

node item list

6.19.3.4 QList< QGraphicsItem ∗ > Tinkercell::ConnectionGraphicsItem::arrowHeadsAsGraphicsItems () const `[virtual]`

get all the arrowHeads associated with the nodes The order is the same order as nodes(), so values can be 0

get all the arrow heads in the same order as nodes

Returns

arrow item list
node item list

6.19.3.5 QList< ConnectionGraphicsItem ∗ > Tinkercell::ConnectionGraphicsItem::cast (const QList< QGraphicsItem ∗ > & *list*) `[static]`

cast a list of graphics item to a list of connection graphics items using qgraphicsitem_cast

Parameters

QList<QGraphicsItem∗> graphics items

Returns

QList<ConnectionGraphicsItem∗> can be empty if no cast is invalid

6.19.3.6 ConnectionGraphicsItem ∗ Tinkercell::ConnectionGraphicsItem::cast (QGraphicsItem ∗ *q*) `[static]`

cast a graphics item to a connection graphics item using qgraphicsitem_cast

Parameters

QGraphicsItem∗ graphics item

Returns

ConnectionGraphicsItem∗ can be 0 if the cast is invalid

6.19.3.7 QPointF Tinkercell::ConnectionGraphicsItem::centerLocation () const `[virtual]`

the center point (if one exists)

the center location

6.19.3.8 void Tinkercell::ConnectionGraphicsItem::clear (bool *all* = `false`) `[virtual]`

Clear all shapes and control points.

Parameters

void

Returns

void

6.19.3.9 ConnectionGraphicsItem ∗ Tinkercell::ConnectionGraphicsItem::clone () const `[virtual]`

make a copy of this connection item

make a copy of this item

6.19.3.10 ConnectionGraphicsItem & Tinkercell::ConnectionGraphicsItem::copyPoints (const ConnectionGraphicsItem & *copy*) `[virtual]`

operator =: copy just the control point positions and pen

6.19.3.11 void Tinkercell::ConnectionGraphicsItem::hideControlPoints ()

hide control points. same as setControlPointsVisible(false)

Returns

void

6.19.3.12 int Tinkercell::ConnectionGraphicsItem::indexOf (QGraphicsItem ∗ *target*) const `[virtual]`

get the index of the node

find the index of the node

Parameters

node in this connection

Returns

index, -1 if node not found

6.19.3.13 bool Tinkercell::ConnectionGraphicsItem::isModifier () const `[virtual]`

check is this connection represents a modifier, i.e. points to the centerRegion of another connection

Returns

boolean

6.19.3.14 ArrowHeadItem ∗ Tinkercell::ConnectionGraphicsItem::modifierArrowAt (int *index*) const `[virtual]`

get the modifier arrow head at the particular index

find the modifier arrow head at the particular index

Parameters

index less than size of curveSegments

Returns

node item or 0

6.19.3.15 QList< ArrowHeadItem ∗ > Tinkercell::ConnectionGraphicsItem::modifierArrowHeads () const `[virtual]`

get all the arrowHeads NOT associated with the nodes

find all the modifier arrow heads in the same order as nodes

Returns

graphics item list
node item list

6.19.3.16 NodeGraphicsItem ∗ Tinkercell::ConnectionGraphicsItem::nodeAt (int *index*) const `[virtual]`

get the node that connected to the particular path

find the node that connected to the particular path

Parameters

index less than size of curveSegments

Returns

node item or 0

6.19.3.17 QList< NodeGraphicsItem ∗ > Tinkercell::ConnectionGraphicsItem::nodes () const `[virtual]`

get all nodes that are connected

find all the nodes that are connected

Returns

node item list
node item list or 0

6.19.3.18 QList< QGraphicsItem ∗ > Tinkercell::ConnectionGraphicsItem::nodesAsGraphicsItems () const `[virtual]`

get all nodes that are connected

find all the nodes that are connected

Returns

graphics item list
node item list or 0

6.19.3.19 QList< NodeGraphicsItem ∗ > Tinkercell::ConnectionGraphicsItem::nodesDisconnected () const `[virtual]`

get all nodes that are not directle connected to the main connection, such as modifier nodes

find all the nodes that are connected

Returns

node item list
node item list or 0

6.19.3.20 QList< NodeGraphicsItem ∗ > Tinkercell::ConnectionGraphicsItem::nodesWithArrows () const `[virtual]`

get all nodes that have an arrow pointing to them

find all the nodes that are connected

Returns

node item list
node item list or 0

6.19.3.21 QList< NodeGraphicsItem ∗ > Tinkercell::ConnectionGraphicsItem::nodesWithoutArrows () const `[virtual]`

get all nodes that do NOT have an arrow pointing to them

find all the nodes that are connected

Returns

node item list
node item list or 0

6.19.3.22 ConnectionGraphicsItem & Tinkercell::ConnectionGraphicsItem::operator= (const ConnectionGraphicsItem & *copy*) `[virtual]`

operator =: remove everything from original connection and copy everything from the given connection

operator =: copy just the control point positions and pen

6.19.3.23 void Tinkercell::ConnectionGraphicsItem::paint (QPainter ∗ *painter*, const QStyleOptionGraphicsItem ∗ *option* = `new QStyleOptionGraphicsItem()`, QWidget ∗ *widget* = `0`) `[virtual]`

returns the bounding rectangle for this reaction figure

paint method. Call's parent's paint after setting antialiasing to true

paint method. Call's parent's after drawing boundary true

6.19.3.24 void Tinkercell::ConnectionGraphicsItem::refresh (bool *arrowTransform* = `true`) `[virtual]`

refresh the path if any controlpoints have moved

Parameters

bool tranform arrow heads

Returns

void

Parameters

void

Returns

void

6.19.3.25 void Tinkercell::ConnectionGraphicsItem::replaceNode (NodeGraphicsItem ∗ *oldNode*, NodeGraphicsItem ∗ *newNode*) `[virtual]`

replace one node in the reaction with another

Parameters

target node to replace

new node

Returns

void

6.19.3.26 void Tinkercell::ConnectionGraphicsItem::replaceNodeAt (int *index*, NodeGraphicsItem ∗ *nodeItem*) `[virtual]`

replace the node at the particular position with a new node

Parameters

index where to insert the new node

new node

Returns

void

6.19.3.27 void Tinkercell::ConnectionGraphicsItem::setControlPointsVisible (bool *visible* = `true`) `[virtual]`

set visibility of control points

Parameters

visible = true, invisible = false

Returns

void

6.19.3.28 QPainterPath Tinkercell::ConnectionGraphicsItem::shape () const `[virtual]`

gets a path that represents this reactionimage

gets a path that is constructed by uniting all the shape paths

6.19.3.29 void Tinkercell::ConnectionGraphicsItem::showControlPoints ()

show control points. same as setControlPointsVisible(true)

Returns

void

6.19.3.30 qreal Tinkercell::ConnectionGraphicsItem::slopeAtPoint (const QPointF & *point*) `[virtual]`

get slope at the given point (or closest point)

find slope at the given point (or closest point)

6.19.3.31 ConnectionGraphicsItem ∗ Tinkercell::ConnectionGraphicsItem::topLevelConnectionItem (QGraphicsItem ∗ *item*, bool *includeControlPoints* = `false`) `[static]`

gets the connection graphics item from its child item

Parameters

QGraphicsItem∗ the target item

bool using true here will return the connection item for a control point, otherwise control points are ignored

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

- ConnectionGraphicsItem.h
- ConnectionGraphicsItem.cpp

6.20 Tinkercell::ConnectionGraphicsReader Class Reference

An xml reader that reads a NodeGraphicsItem file.

`#include <ConnectionGraphicsReader.h>`

Public Member Functions

- QXmlStreamReader::TokenType readNext ()

  Reads up to the next start node.

Static Public Member Functions

- static ConnectionGraphicsItem ∗ readConnectionGraphics (const QList< NodeGraphicsItem ∗ > &nodes, const QList< ConnectionGraphicsItem ∗ > &connections, NodeGraphicsReader ∗reader)

  Reads a ConnectionGraphicsItem from XML, given all the nodes for the connection are already in the scene.

- static QList< ConnectionGraphicsItem::ControlPoint ∗ > readControlPoints (QXmlStreamReader ∗)

  Reads all control points from an XML file.

- static ConnectionGraphicsItem::CurveSegment readCurveSegment (QHash< QString, ItemHandle ∗ > &nodes, QHash< QString, ItemHandle ∗ > &connections, QList< ConnectionGraphicsItem::ControlPoint ∗ > &controlPoints, NodeGraphicsReader ∗, const QString &groupID=QString())

  Reads a shape into an NodeGraphicsItem from an XML file.

- static ConnectionGraphicsItem::ControlPoint ∗ readControlPoint (QXmlStreamReader ∗)

  Reads a control point from an XML file.

- static ArrowHeadItem ∗ readArrow (NodeGraphicsReader &reader, QString name)

  Reads an arrow item from xml file. The procedure is very similar to reading a node.

- static void readCenterRegion (ConnectionGraphicsItem ∗connection, NodeGraphicsReader ∗reader)

  Reads the center region of a connection from xml file.

6.20.1 Detailed Description

An xml reader that reads a NodeGraphicsItem file.

6.20.2 Member Function Documentation

6.20.2.1 ArrowHeadItem ∗ Tinkercell::ConnectionGraphicsReader::readArrow (NodeGraphicsReader & *reader*, QString *name*) `[static]`

Reads an arrow item from xml file. The procedure is very similar to reading a node.

Parameters

- ***node*** reader
- ***name*** of the entry, i.e. ArrowAtStart or ArrowAtEnd

Returns

arrow item

6.20.2.2 void Tinkercell::ConnectionGraphicsReader::readCenterRegion (ConnectionGraphicsItem ∗ *connection*, NodeGraphicsReader ∗ *reader*) `[static]`

Reads the center region of a connection from xml file.

Parameters

- ***target*** connection
- ***name*** of the entry

Returns

arrow item

6.20.2.3 ConnectionGraphicsItem ∗ Tinkercell::ConnectionGraphicsReader::readConnectionGraphics (const QList< NodeGraphicsItem ∗ > & *nodes*, const QList< ConnectionGraphicsItem ∗ > & *connections*, NodeGraphicsReader ∗ *reader*) `[static]`

Reads a ConnectionGraphicsItem from XML, given all the nodes for the connection are already in the scene.

Parameters

- ***list*** of nodes
- ***list*** of other connections
- ***xml*** reader in use

Returns

list of control points

Parameters

- ***list*** of nodes
- ***xml*** reader in use

Returns

list of control points

6.20.2.4 ConnectionGraphicsItem::ControlPoint ∗ Tinker-cell::ConnectionGraphicsReader::readControlPoint (QXmlStreamReader ∗ *reader*) `[static]`

Reads a control point from an XML file.

Parameters

XML reader in use

Returns

control point

Parameters

XML reader in use

Returns

void

6.20.2.5 QList< ConnectionGraphicsItem::ControlPoint ∗ > Tinker-cell::ConnectionGraphicsReader::readControlPoints (QXmlStreamReader ∗ *reader*) `[static]`

Reads all control points from an XML file.

Parameters

xml reader in use

Returns

list of control points

6.20.2.6 ConnectionGraphicsItem::CurveSegment Tinker-cell::ConnectionGraphicsReader::readCurveSegment (QHash< QString, ItemHandle ∗ > & *nodes*, QHash< QString, ItemHandle ∗ > & *connections*, QList< ConnectionGraphicsItem::ControlPoint ∗ > & *controlPoints*, NodeGraphicsReader ∗ *reader*, const QString & *groupID* = `QString()`) `[static]`

Reads a shape into an NodeGraphicsItem from an XML file.

Parameters

hash table of fullname -> node handle

list of control points to use

the xml reader in use

Returns

path vector with all the control points and nodes and arrows

6.20.2.7 QXmlStreamReader::TokenType Tinkercell::ConnectionGraphicsReader::readNext ()

Reads up to the next start node.

Returns

Token Typer

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

- ConnectionGraphicsReader.h
- ConnectionGraphicsReader.cpp

6.21 Tinkercell::ConnectionGraphicsWriter Class Reference

This class is an xml writer that specifically writes a ConnectionGraphicsItem.

`#include <ConnectionGraphicsWriter.h>`

Public Member Functions

- ConnectionGraphicsWriter ()

  default constructor

- bool writeXml (ConnectionGraphicsItem ∗connection, const QString &fileName)

  Writes an Connection item XML file with the document headers.

- bool writeXml (ConnectionGraphicsItem ∗connection, QIODevice ∗device)

  Writes an Connection item XML file with the document headers.

- bool writeConnectionGraphics (ConnectionGraphicsItem ∗connection, QIODevice ∗device)

  Writes an Connection as an XML file using the IO device provided.

Static Public Member Functions

- static bool writeConnectionGraphics (ConnectionGraphicsItem ∗connection, QXmlStreamWriter ∗)

  Writes an NodeImage as an XML file using the xml writer provided.

6.21.1 Detailed Description

This class is an xml writer that specifically writes a ConnectionGraphicsItem.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Tinkercell::ConnectionGraphicsWriter::ConnectionGraphicsWriter ()

default constructor

constructor. Sets autoformatting to true

6.21.3 Member Function Documentation

6.21.3.1 bool Tinkercell::ConnectionGraphicsWriter::writeConnectionGraphics (ConnectionGraphicsItem ∗ *connection*, QXmlStreamWriter ∗ *writer*) `[static]`

Writes an NodeImage as an XML file using the xml writer provided.

Parameters

connection item pointer to write as XML

xml writer in use

Returns

void

6.21.3.2 bool Tinkercell::ConnectionGraphicsWriter::writeConnectionGraphics (ConnectionGraphicsItem * *connection*, QIODevice * *device*)

Writes an Connection as an XML file using the IO device provided.

Writes an NodeImage as an XML file using the xml writer provided.

Parameters

connection item pointer to write as XML
QIODevice to use

Returns

void

Parameters

connection item pointer to write as XML
xml writer in use

Returns

void

6.21.3.3 bool Tinkercell::ConnectionGraphicsWriter::writeXml (ConnectionGraphicsItem * *connection*, QIODevice * *device*)

Writes an Connection item XML file with the document headers.

Writes an ConnectionGraphicsItem XML file with the document headers.

Parameters

connection item pointer to write as XML
QIODevice to use

Returns

void

Parameters

ConnectionGraphicsItem pointer to write as XML
QIODevice to use

Returns

void

6.21.3.4 bool Tinkercell::ConnectionGraphicsWriter::writeXml (ConnectionGraphicsItem ∗ *connection*, const QString & *fileName*)

Writes an Connection item XML file with the document headers.

Writes an ConnectionGraphicsItem XML file with the document headers.

Parameters

connection	item pointer to write as XML
QIODevice	to use

Returns

void

Parameters

ConnectionGraphicsItem	pointer to write as XML
QIODevice	to use

Returns

void

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

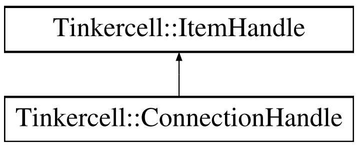
- ConnectionGraphicsWriter.h
- ConnectionGraphicsWriter.cpp

6.22 Tinkercell::ConnectionHandle Class Reference

The handles are used to bring together data and graphics items. Connection Handle contains pointers to all the graphics items that belong to it, the tools that apply to this item, the data for this item, the family that it belongs with, and pointers to nodes connected (in and out).

```
#include <ItemHandle.h>
```

Inheritance diagram for Tinkercell::ConnectionHandle:



Public Member Functions

- virtual QList< NodeHandle ∗ > nodes (int role=0) const

  returns all the nodes connected to all the connectors in this handle

- virtual void addNode (NodeHandle ∗, int role=0)

  add a node to this connection (only applies to connections with NO grpahics items)

- virtual void clearNodes ()

  clear all nodes in connection (only applies to connections with NO graphics items)

- virtual QList< NodeHandle ∗ > nodesIn () const

  returns all the nodes that are on the "input" side of this connection. If this connection is represented by graphics items, then this is determined by looking at which nodes have an arrow-head associated with them in graphics items If there are no graphics items, then this function uses the _nodes list to find the "in" nodes (role = -1).

- virtual QList< NodeHandle ∗ > nodesOut () const

  If this connection is represented by graphics items, then this is determined by looking at which nodes have NO arrow-head associated with them in graphics items If there are no graphics items, then this function uses the _nodes list to find the "out" nodes (role = +1).

- ConnectionHandle (const QString &name=QString(), ConnectionFamily ∗family=0)

  default constructor -- initializes everything

- ConnectionHandle (ConnectionFamily ∗family, const QString &name=QString())

  one parameter constructor -- initializes everything

- ConnectionHandle (const ConnectionHandle &)

  copy constructor -- deep copy of data, but shallow copy of graphics items

- virtual ConnectionHandle & operator= (const ConnectionHandle &)

  operator =

- ConnectionHandle (ConnectionFamily ∗family, ConnectionGraphicsItem ∗item) *two parameter constructor*

- virtual void setFamily (ItemFamily ∗family, bool useCommand=true)

  set the family for this handle

- virtual ItemHandle ∗ clone () const

  clone of this handle

- virtual ItemFamily ∗ family () const

  family for this handle

- virtual QList< ItemFamily ∗ > findValidChildFamilies () const

  find child-families of the current family that this connection can potentially belong with

Static Public Member Functions

- static ConnectionHandle ∗ cast (ItemHandle ∗)

  checks if the item handle is a connection handle and casts it as a connection item. Returns 0 if it is not a node item

- static QList< ConnectionHandle ∗ > cast (const QList< ItemHandle ∗ > &)

  checks if the item handles are connection handles and casts then as connection items. Returns QList<ConnectionHandle∗>

Public Attributes

- ConnectionFamily ∗ connectionFamily

  the family for this connection handle

- QList< QPair< NodeHandle ∗, int > > nodesWithRoles

  the nodes that are connected by this connection and the role of each node. this list is ONLY used for connections with NO graphics items -1 and 1 are reseved roles, indicating in and out nodes

Static Public Attributes

- static const int TYPE = 2

  this number is used to identify when an item handle is a connection handle

6.22.1 Detailed Description

The handles are used to bring together data and graphics items. Connection Handle contains pointers to all the graphics items that belong to it, the tools that apply to this item, the data for this item, the family that it belongs with, and pointers to nodes connected (in and out).

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Tinkercell::ConnectionHandle::ConnectionHandle (ConnectionFamily ∗ *family*, const QString & *name* = `QString()`)

one parameter constructor -- initializes everything

Parameters

ConnectionFamily∗ connection family

QString name

6.22.2.2 Tinkercell::ConnectionHandle::ConnectionHandle (ConnectionFamily ∗ *family*, ConnectionGraphicsItem ∗ *item*)

two parameter constructor

Parameters

ConnectionFamily∗ initial family

ConnectionGraphicsItem∗ connection graphics item

6.22.3 Member Function Documentation

6.22.3.1 void Tinkercell::ConnectionHandle::addNode (NodeHandle ∗ *h*, int *role* = `0`) `[virtual]`

add a node to this connection (only applies to connections with NO grpahics items)

Parameters

NodeHandle∗ node

int role of this node. -1 is for "in" nodes. +1 is for "out" nodes. Use any other values for specific purposes

6.22.3.2 QList< ConnectionHandle ∗ > Tinkercell::ConnectionHandle::cast (const QList< ItemHandle ∗ > & *items*) `[static]`

checks if the item handles are connection handles and casts then as connection items. Returns QList<ConnectionHandle∗>

Parameters

Returns QList<ItemHandle∗> items

6.22.3.3 ConnectionHandle ∗ Tinkercell::ConnectionHandle::cast (ItemHandle ∗ *item*) `[static]`

checks if the item handle is a connection handle and casts it as a connection item. Returns 0 if it is not a node item

Parameters

ItemHandle∗ item

6.22.3.4 ItemHandle ∗ Tinkercell::ConnectionHandle::clone () const `[virtual]`

clone of this handle

Returns

ItemFamily∗ connection handle as item handle

Reimplemented from Tinkercell::ItemHandle.

6.22.3.5 ItemFamily ∗ Tinkercell::ConnectionHandle::family () const `[virtual]`

family for this handle

Returns

ItemFamily∗ connection family as item family

Reimplemented from Tinkercell::ItemHandle.

6.22.3.6 QList< ItemFamily ∗ > Tinkercell::ConnectionHandle::findValidChildFamilies () const `[virtual]`

find child-families of the current family that this connection can potentially belong with

Returns

QList<ItemFamily∗> valid connection families

6.22.3.7 QList< NodeHandle ∗ > Tinkercell::ConnectionHandle::nodes (int *role* = 0) const `[virtual]`

returns all the nodes connected to all the connectors in this handle

Returns

QList<NodeHandle∗> list of node handles

6.22.3.8 QList< NodeHandle ∗ > Tinkercell::ConnectionHandle::nodesIn () const `[virtual]`

returns all the nodes that are on the "input" side of this connection. If this connection is represented by graphics items, then this is determined by looking at which nodes have an arrow-head associated with them in graphics items If there are no graphics items, then this function uses the _nodes list to find the "in" nodes (role = -1).

Returns

QList<NodeHandle∗> list of node handles

6.22.3.9 QList< NodeHandle ∗ > Tinkercell::ConnectionHandle::nodesOut () const `[virtual]`

If this connection is represented by graphics items, then this is determined by looking at which nodes have NO arrow-head associated with them in graphics items If there are no graphics items, then this function uses the _nodes list to find the "out" nodes (role = +1).

Returns

QList<NodeHandle∗> list of node handles

6.22.3.10 void Tinkercell::ConnectionHandle::setFamily (ItemFamily ∗ *family*, bool *useCommand* = `true`) `[virtual]`

set the family for this handle

Parameters

***ConnectionFamily*∗**	connection family

Reimplemented from Tinkercell::ItemHandle.

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

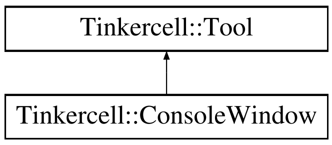
- ItemHandle.h
- ItemHandle.cpp

6.23 Tinkercell::ConsoleWindow Class Reference

Used to create an output window that can display outputs.

`#include <ConsoleWindow.h>`

Inheritance diagram for Tinkercell::ConsoleWindow:



Public Slots

- void eval (const QString &)

  send a command to the console window to be evaluated

- void message (const QString &)

  print a message in the output window

- void error (const QString &)

  print an error message in the output window

- void printTable (const DataTable< qreal > &dataTable)

  print a data table (tab-delimited) in the output window

- void clear ()

  clear the output window

- void freeze ()

  freeze the output window. Frozen window will not be responsive to commands

- void unfreeze ()

  unfreeze the output window. Frozen window will not be responsive to commands

Signals

- void commandExecuted (const QString &command)

  the user requested to execute the given command

- void commandInterrupted ()

  the user requested to interrupt the current process

Public Member Functions

- ConsoleWindow (MainWindow ∗main=0)

    constructor -- initialize main window

- CommandTextEdit ∗ editor ()

    the command window's editor

Static Public Attributes

- static QString Prompt

    the string used at the prompt

- static QColor BackgroundColor = QColor("#000000")

    the background color for console

- static QColor PlainTextColor = QColor("#FEFFEC")

    the font color for plain text

- static QColor ErrorTextColor = QColor("#FF6F0F")

    the font color for error messages

- static QColor OutputTextColor = QColor("#33FF00")

    the font color for outputs

- static QColor TableTextColor = QColor("#FFFF00")

    the font color for table headers

Protected Attributes

- CommandTextEdit commandTextEdit

    the command window

6.23.1 Detailed Description

Used to create an output window that can display outputs.

6.23.2 Member Function Documentation

6.23.2.1 void Tinkercell::ConsoleWindow::message (const QString & *s*) `[slot]`

print a message in the output window

show a message text in the output window

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

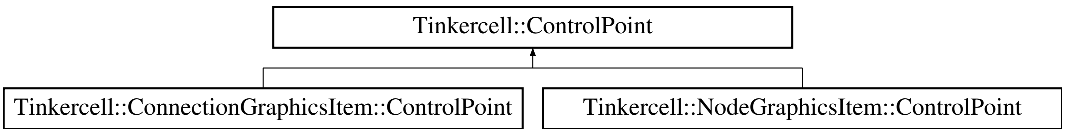
- ConsoleWindow.h
- ConsoleWindow.cpp

6.24 Tinkercell::ControlPoint Class Reference

A simple circle or square that is used for changing specific locations.

`#include <ControlPoint.h>`

Inheritance diagram for Tinkercell::ControlPoint:


Tinkercell::ControlPoint
Tinkercell::ConnectionGraphicsItem::ControlPoint
Tinkercell::NodeGraphicsItem::ControlPoint

Public Types

- enum { **Type** = UserType + 1 }

  paint method. Call's parent's paint after setting antialiasing to true

- enum ShapeType { **circle**, **square**, **triangle** }

  type of shape to paint.

Public Member Functions

- virtual qreal x ()

  x position

- virtual qreal y ()

  y position

- ControlPoint (QGraphicsItem ∗parent=0)

  Constructor: Setup colors and z value.

- ControlPoint (const ControlPoint &copy)

  copy constructor

- virtual int type () const

  for enabling dynamic_cast

- virtual void sideEffect ()

  side effect when moved. always call this after moving

- virtual ControlPoint ∗ clone () const

  make a copy of this control point

- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option=new QStyleOption-GraphicsItem(), QWidget ∗widget=0)

  paint method.

- virtual QRectF boundingRect () const

  bounding rect method.

- virtual void setRect (const QRectF &)

  set size rect.

- virtual QRectF rect () const

  get size rect.

- virtual ItemHandle ∗ handle () const

  get the handle of this control point, usually 0 or the parent's handle

- virtual void setHandle (ItemHandle ∗)

  set the handle of this control point, usually sets parent's handle or does nothing

Static Public Member Functions

- static ControlPoint ∗ cast (QGraphicsItem ∗item)

  Gets the control point item from one of its child items.

Public Attributes

- QBrush defaultBrush

  permanent brush for this control point

- QPen defaultPen

  permanent pen for this control point

- QSizeF defaultSize

  default size for this item

- ShapeType shapeType

  type of shape to paint.

Protected Attributes

- QRectF **bounds**

6.24.1 Detailed Description

A simple circle or square that is used for changing specific locations.

6.24.2 Member Enumeration Documentation

6.24.2.1 anonymous enum

paint method. Call's parent's paint after setting antialiasing to true

for enabling dynamic_cast

6.24.3 Constructor & Destructor Documentation

6.24.3.1 Tinkercell::ControlPoint::ControlPoint (const ControlPoint & *copy*)

copy constructor

Copy Constructor.

6.24.4 Member Function Documentation

6.24.4.1 ControlPoint ∗ Tinkercell::ControlPoint::clone () const `[virtual]`

make a copy of this control point

make a copy of this item

Reimplemented in Tinkercell::ConnectionGraphicsItem::ControlPoint, and Tinkercell::NodeGraphicsItem::ControlPoint.

6.24.4.2 void Tinkercell::ControlPoint::paint (QPainter ∗ *painter*, const QStyleOptionGraphicsItem ∗ *option* = `new QStyleOptionGraphicsItem()`, QWidget ∗ *widget* = `0`) `[virtual]`

paint method.

paint method. draw one of the shapes

Reimplemented in Tinkercell::NodeGraphicsItem::ControlPoint.

6.24.4.3 QRectF Tinkercell::ControlPoint::rect () const `[virtual]`

get size rect.

bounding rect method.

6.24.4.4 void Tinkercell::ControlPoint::setRect (const QRectF & *rect*) `[virtual]`

set size rect.

set size.

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

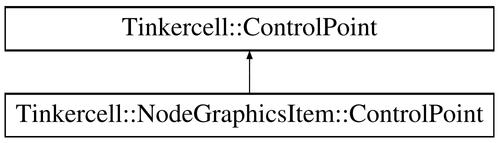
- ControlPoint.h
- ConnectionGraphicsItem.cpp
- ControlPoint.cpp

6.25 Tinkercell::NodeGraphicsItem::ControlPoint Class Reference

a control point with a pointer to a NodeGraphicsItem

```
#include <NodeGraphicsItem.h>
```

Inheritance diagram for Tinkercell::NodeGraphicsItem::ControlPoint:



Public Types

- enum { **Type** = UserType + 2 }

  for enabling dynamic_cast

Public Member Functions

- ControlPoint (NodeGraphicsItem ∗idrawable_ptr=0, QGraphicsItem ∗parent=0)

  Constructor: Setup colors and z value.

- ControlPoint (const ControlPoint &copy)

  Copy Constructor.

- virtual ControlPoint & operator= (const ControlPoint &copy)

  operator =

- virtual Tinkercell::ControlPoint ∗ clone () const

  make a copy of this control point

- virtual int type () const

  for enabling dynamic_cast

- virtual void sideEffect ()

  side effect when moved. always call this after moving

- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option=new QStyleOption-GraphicsItem(), QWidget ∗widget=0)

  paint method.

- virtual ItemHandle ∗ handle () const

  same as nodeItem->handle()

- virtual void setHandle (ItemHandle ∗)

  set the nodeItem->setHandle(..)

- ∼ControlPoint ()

  destructor

Public Attributes

- NodeGraphicsItem ∗ nodeItem

  idrawables that this control point belong in

6.25.1 Detailed Description

a control point with a pointer to a NodeGraphicsItem

6.25.2 Member Function Documentation

6.25.2.1 Tinkercell::ControlPoint ∗ Tinkercell::NodeGraphicsItem::ControlPoint::clone () const `[virtual]`

make a copy of this control point

make a copy of this item

Reimplemented from Tinkercell::ControlPoint.

6.25.2.2 NodeGraphicsItem::ControlPoint & Tinkercell::NodeGraphicsItem::ControlPoint::operator= (const ControlPoint & *copy*) `[virtual]`

operator =

Copy operator

6.25.2.3 void Tinkercell::NodeGraphicsItem::ControlPoint::paint (QPainter ∗ *painter*, const QStyleOptionGraphicsItem ∗ *option* = `new QStyleOptionGraphicsItem()`, QWidget ∗ *widget* = `0`) `[virtual]`

paint method.

paint method. Call's parent's

Reimplemented from Tinkercell::ControlPoint.

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

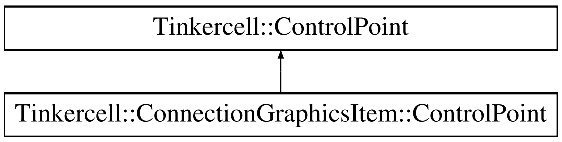
- NodeGraphicsItem.h
- NodeGraphicsItem.cpp

6.26 Tinkercell::ConnectionGraphicsItem::ControlPoint Class Reference

A control point with a pointer to a ConnectionGraphicsItem.

`#include <ConnectionGraphicsItem.h>`

Inheritance diagram for Tinkercell::ConnectionGraphicsItem::ControlPoint:



Public Types

- enum { **Type** = UserType + 7 }

    for enabling dynamic_cast

Public Member Functions

- ControlPoint (ConnectionGraphicsItem ∗reaction_ptr=0, QGraphicsItem ∗parent=0)

    Constructor: Setup colors and z value.

- ControlPoint (const QPointF &pos, ConnectionGraphicsItem ∗reaction_ptr=0, QGraphicsItem ∗parent=0)

    Constructor: constructor with position.

- ControlPoint (const ControlPoint &copy)

    Copy Constructor.

- virtual ControlPoint & operator= (const ControlPoint &copy)

    operator =

- virtual int type () const

    for enabling dynamic_cast

- ∼ControlPoint ()

    destructor

- virtual Tinkercell::ControlPoint ∗ clone () const

    side effect when moved. always call this after moving

- virtual ItemHandle ∗ handle () const

    same as connectionItem->handle()

- virtual void setHandle (ItemHandle ∗)

    same as connectionItem->setHandle(...)

Public Attributes

- ConnectionGraphicsItem ∗ connectionItem

  idrawables that this control point belong in

6.26.1 Detailed Description

A control point with a pointer to a ConnectionGraphicsItem.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Tinkercell::ConnectionGraphicsItem::ControlPoint::∼ControlPoint ()

destructor

destructor

6.26.3 Member Function Documentation

6.26.3.1 ControlPoint ∗ Tinkercell::ConnectionGraphicsItem::ControlPoint::clone () const `[virtual]`

side effect when moved. always call this after moving

make a copy of this item

make a copy of this control point

Reimplemented from Tinkercell::ControlPoint.

6.26.3.2 ConnectionGraphicsItem::ControlPoint & Tinkercell::ConnectionGraphicsItem::ControlPoint::operator= (const ControlPoint & *copy*) `[virtual]`

operator =

Copy operator

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

- ConnectionGraphicsItem.h
- ConnectionGraphicsItem.cpp

6.27 Tinkercell::Core_FtoS Class Reference

Function to Signal converter for MainWindow.

```
#include <C_API_Slots.h>
```

Signals

- void **allItems** (QSemaphore ∗, QList< ItemHandle ∗ > ∗)
- void **selectedItems** (QSemaphore ∗, QList< ItemHandle ∗ > ∗)
- void **itemsOfFamily** (QSemaphore ∗, QList< ItemHandle ∗ > ∗, const QString &)
- void **itemsOfFamily** (QSemaphore ∗, QList< ItemHandle ∗ > ∗, const QList< ItemHandle ∗ > &, const QString &)
- void **find** (QSemaphore ∗, ItemHandle ∗∗, const QString &)
- void **findItems** (QSemaphore ∗, QList< ItemHandle ∗ > ∗, const QStringList &)
- void **select** (QSemaphore ∗, ItemHandle ∗)
- void **deselect** (QSemaphore ∗)
- void **removeItem** (QSemaphore ∗, ItemHandle ∗)
- void **setPos** (QSemaphore ∗, ItemHandle ∗, qreal, qreal)
- void **setPos** (QSemaphore ∗, const QList< ItemHandle ∗ > &, DataTable< qreal > &)
- void **getPos** (QSemaphore ∗, const QList< ItemHandle ∗ > &, DataTable< qreal > ∗)
- void **getY** (QSemaphore ∗, qreal ∗, ItemHandle ∗)
- void **getX** (QSemaphore ∗, qreal ∗, ItemHandle ∗)
- void **moveSelected** (QSemaphore ∗, qreal, qreal)
- void **getFamily** (QSemaphore ∗, QString ∗, ItemHandle ∗)
- void **getName** (QSemaphore ∗, QString ∗, ItemHandle ∗)
- void **getUniqueName** (QSemaphore ∗, QString ∗, ItemHandle ∗)
- void **setName** (QSemaphore ∗, ItemHandle ∗, const QString &)
- void **getNames** (QSemaphore ∗, QStringList ∗, const QList< ItemHandle ∗ > &)
- void **getUniqueNames** (QSemaphore ∗, QStringList ∗, const QList< ItemHandle ∗ > &)
- void **isA** (QSemaphore ∗, int ∗, ItemHandle ∗, const QString &)
- void **outputText** (QSemaphore ∗, const QString &)
- void **errorReport** (QSemaphore ∗, const QString &)
- void **printFile** (QSemaphore ∗, const QString &)
- void **clearText** (QSemaphore ∗)
- void **outputTable** (QSemaphore ∗, const DataTable< qreal > &)
- void **createInputWindow** (QSemaphore ∗, const DataTable< qreal > &, const QString &, const QString &, const QString &)
- void **createInputWindow** (QSemaphore ∗, long, const DataTable< qreal > &, const QString &, MatrixInputFunction)
- void **createSliders** (QSemaphore ∗, CThread ∗, const DataTable< qreal > &, MatrixInputFunction)

- void **addInputWindowOptions** (QSemaphore ∗, const QString &, int i, int j, const QStringList &)
- void **addInputWindowCheckbox** (QSemaphore ∗, const QString &, int i, int j)
- void **openNewWindow** (QSemaphore ∗, const QString &)
- void **isWindows** (QSemaphore ∗, int ∗)
- void **isMac** (QSemaphore ∗, int ∗)
- void **isLinux** (QSemaphore ∗, int ∗)
- void **appDir** (QSemaphore ∗, QString ∗)
- void **homeDir** (QSemaphore ∗, QString ∗)

- void **zoom** (QSemaphore ∗, qreal)
- void **getNumericalDataNames** (QSemaphore ∗, QStringList ∗, ItemHandle ∗)
- void **getTextDataNames** (QSemaphore ∗, QStringList ∗, ItemHandle ∗)
- void **getNumericalData** (QSemaphore ∗, DataTable< qreal > ∗, ItemHandle ∗, const QString &)
- void **setNumericalData** (QSemaphore ∗, ItemHandle ∗, const QString &, const DataTable< qreal > &)
- void **getTextData** (QSemaphore ∗, DataTable< QString > ∗, ItemHandle ∗, const QString &)
- void **setTextData** (QSemaphore ∗, ItemHandle ∗, const QString &, const DataTable< QString > &)

- void **getChildren** (QSemaphore ∗, QList< ItemHandle ∗ > ∗, ItemHandle ∗)
- void **getParent** (QSemaphore ∗, ItemHandle ∗∗, ItemHandle ∗)
- void **getString** (QSemaphore ∗, QString ∗, const QString &)
- void **getFilename** (QSemaphore ∗, QString ∗)
- void **getSelectedString** (QSemaphore ∗, int ∗, const QString &, const QStringList &, const QString &)
- void **getNumber** (QSemaphore ∗, qreal ∗, const QString &)
- void **getNumbers** (QSemaphore ∗, const QStringList &, qreal ∗)
- void **askQuestion** (QSemaphore ∗, const QString &, int ∗)
- void **messageDialog** (QSemaphore ∗, const QString &)
- void **openFile** (QSemaphore ∗, const QString &)
- void **saveToFile** (QSemaphore ∗, const QString &)
- void **setSize** (QSemaphore ∗, ItemHandle ∗, double, double, int)
- void **getWidth** (QSemaphore ∗, ItemHandle ∗, double ∗)
- void **getHeight** (QSemaphore ∗, ItemHandle ∗, double ∗)
- void **setAngle** (QSemaphore ∗, ItemHandle ∗, double, int)
- void **getAngle** (QSemaphore ∗, ItemHandle ∗, double ∗)
- void **getColor** (QSemaphore ∗, QString ∗, ItemHandle ∗)
- void **setColor** (QSemaphore ∗, ItemHandle ∗, const QString &, int)
- void **changeGraphics** (QSemaphore ∗, ItemHandle ∗, const QString &)
- void **changeArrowHead** (QSemaphore ∗, ItemHandle ∗, const QString &)
- void **screenshot** (QSemaphore ∗, const QString &, int, int)
- void **screenHeight** (QSemaphore ∗, int ∗)
- void **screenWidth** (QSemaphore ∗, int ∗)
- void **screenX** (QSemaphore ∗, int ∗)
- void **screenY** (QSemaphore ∗, int ∗)

Public Member Functions

- void **zoom** (double)
- tc_items **allItems** ()
- tc_items **itemsOfFamily** (const char ∗)
- tc_items **itemsOfFamily** (const char ∗, tc_items)
- tc_items **selectedItems** ()
- long **find** (const char ∗)
- tc_items **findItems** (tc_strings)
- void **select** (long)
- void **deselect** ()
- const char ∗ **getName** (long)
- const char ∗ **getUniqueName** (long)
- void **setName** (long, const char ∗)

- tc_strings **getNames** (tc_items)
- tc_strings **getUniqueNames** (tc_items)
- const char ∗ **getFamily** (long)
- int **isA** (long, const char ∗)
- void **removeItem** (long)
- void **setPos** (long, double, double)
- void **setPos** (tc_items, tc_matrix)
- tc_matrix **getPos** (tc_items)
- double **getY** (long)
- double **getX** (long)
- void **moveSelected** (double, double)
- void **outputTable** (tc_matrix m)
- void **outputText** (const char ∗)
- void **errorReport** (const char ∗)
- void **clearText** ()
- void **printFile** (const char ∗)
- void **createInputWindow** (tc_matrix, const char ∗, const char ∗, const char ∗)
- void **createInputWindow** (long, tc_matrix, const char ∗, MatrixInputFunction)
- void **createSliders** (long, tc_matrix, MatrixInputFunction)
- void **addInputWindowOptions** (const char ∗, int i, int j, tc_strings)
- void **addInputWindowCheckbox** (const char ∗, int i, int j)
- void **openNewWindow** (const char ∗)
- int **isWindows** ()
- int **isMac** ()
- int **isLinux** ()
- const char ∗ **appDir** ()
- const char ∗ **homeDir** ()
- tc_strings **getNumericalDataNames** (long)
- tc_strings **getTextDataNames** (long)
- tc_matrix **getNumericalData** (long, const char ∗)
- void **setNumericalData** (long, const char ∗, tc_matrix)
- tc_table **getTextData** (long, const char ∗)
- void **setTextData** (long, const char ∗, tc_table)
- tc_items **getChildren** (long)
- long **getParent** (long)
- const char ∗ **getString** (const char ∗)
- const char ∗ **getFilename** ()
- int **getSelectedString** (const char ∗, tc_strings, const char ∗)
- double **getNumber** (const char ∗)
- void **getNumbers** (tc_strings, double ∗)
- int **askQuestion** (const char ∗)
- void **messageDialog** (const char ∗)
- void **openFile** (const char ∗)
- void **saveToFile** (const char ∗)
- void **setSize** (long, double, double, int)
- double **getWidth** (long)
- double **getHeight** (long)
- void **setAngle** (long, double, int)
- double **getAngle** (long)
- const char ∗ **getColor** (long)

- void **setColor** (long, const char ∗, int)
- void **changeGraphics** (long, const char ∗)
- void **changeArrowHead** (long, const char ∗)
- void **screenshot** (const char ∗, int, int)
- int **screenHeight** ()
- int **screenWidth** ()
- int **screenX** ()
- int **screenY** ()

6.27.1 Detailed Description

Function to Signal converter for MainWindow.

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

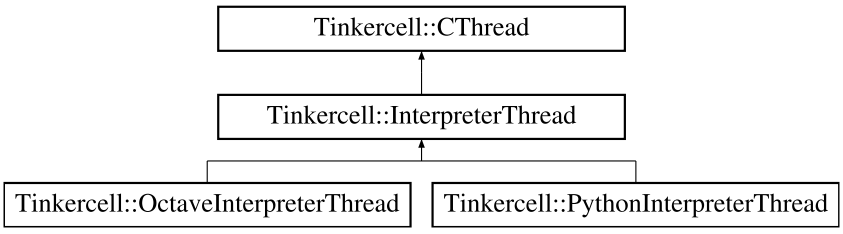
- C_API_Slots.h
- C_API_Slots.cpp

6.28 Tinkercell::CThread Class Reference

This class is used to run specific functions inside a C dynamic library as a separate thread. The class can be used to load a library or just run a specific function inside an already loaded library. If the library is loaded by this class, the library will be unloaded upon completion on the function. To prevent the automatic unloading, use the setAutoUnload option. Only four types of functions are supported.

```
#include <CThread.h>
```

Inheritance diagram for Tinkercell::CThread:


Tinkercell::CThread
Tinkercell::InterpreterThread
Tinkercell::OctaveInterpreterThread
Tinkercell::PythonInterpreterThread

Public Slots

- virtual void unload ()

  uload the C library

- virtual void update ()

  call the callback function, if one exists

Signals

- void progress (int)

  display progress of this thread (0-100). This signal is usually connected to a slot in ProgressBarSignalItem

Public Member Functions

- virtual void emitSignal (int i)

  emits the progress signal

- CThread (MainWindow ∗main, QLibrary ∗lib=0, bool autoUnload=false)

  constructor

- CThread (MainWindow ∗main, const QString &lib=tr(""), bool autoUnload=false)

  constructor

- virtual ∼CThread ()

  destructor. unload and deletes the library

- virtual void setFunction (void(∗f)(void)) *set the function to run inside this threads*

- virtual void setFunction (void(∗f)(double))

  set the function to run inside this threads

- virtual void setFunction (void(∗f)(const char ∗))

  set the function to run inside this threads

- virtual void setFunction (void(∗f)(tc_matrix))

  set the function to run inside this threads

- virtual void setVoidFunction (const char ∗)

  set the function to run inside this threads

- virtual void setDoubleFunction (const char ∗)

  set the function to run inside this threads

- virtual void setCharFunction (const char ∗)

  set the function to run inside this threads

- virtual void setMatrixFunction (const char ∗)

  set the function to run inside this threads

- virtual void setLibrary (QLibrary ∗)

  set the dynamic library for this threads. The library will be loaded if it has not already been loaded

- virtual void setLibrary (const QString &)

  set the dynamic library for this threads.

- virtual QLibrary ∗ library ()

  the library used inside this thread

- virtual void setAutoUnload (bool)

  set whether or not to automatically unload the library when the thread is done running

- virtual bool autoUnload ()

  whether or not to automatically unload the library when the thread is done running

- virtual void setArg (double)

  set the argument for the target function

- virtual void setArg (const QString &)

  set the argument for the target function

- virtual void setArg (const DataTable< qreal > &)

  set the argument for the target function

Static Public Member Functions

- static QLibrary ∗ loadLibrary (const QString &name, QObject ∗parent=0)

  search the default tinkercell folders for the library and load it

- static QWidget ∗ dialog (CThread ∗, const QString &title, const QIcon &icon=QIcon(), bool progressBar=true)

  Creates a dialog with a progress bar for running a new thread. The dialog allows the user to terminate the thread.

Public Attributes

- MainWindow ∗ mainWindow

  main window

Static Public Attributes

- static QString style = QString("background-color: qlineargradient(x1: 0, y1: 1, x2: 0, y2: 0, stop: 1.0 #585858, stop: 0.5 #0E0E0E, stop: 0.5 #9A9A9A, stop: 1.0 #E2E2E2);")

  style sheet for the dialog

- static QList< CThread ∗ > cthreads

  hash stores the name and progress bar pointers for updating progress on different threads

Protected Slots

- virtual void cleanupAfterTerminated ()

  cleanup (such as unload libraries) upon termination

Protected Member Functions

- virtual void setupCFunctionPointers ()

  setup the C pointers in TC_Main.h

- virtual void call_tc_main ()

  call tc_main

- virtual void run ()

  the main function that runs one of the specified functions

Protected Attributes

- bool autoUnloadLibrary

    whether or not to automatically unload the library when the thread is done running

- void(∗ f1)(void)

    one of the functions that can be run inside this thread

- void(∗ f2)(double)

    one of the functions that can be run inside this thread

- void(∗ f3)(const char ∗)

    one of the functions that can be run inside this thread

- void(∗ f4)(tc_matrix)

    one of the functions that can be run inside this thread

- void(∗ callbackPtr)(void)

    callback function

- void(∗ callWhenExitPtr)(void)

    call when exit function

- QLibrary ∗ lib

    the library where the functions are located that can be run inside this thread

- double argDouble

    the argument for one of the the run function

- QString argString

    the argument for one of the the run function

- DataTable< qreal > argMatrix

    the argument for one of the the run function

6.28.1 Detailed Description

This class is used to run specific functions inside a C dynamic library as a separate thread. The class can be used to load a library or just run a specific function inside an already loaded library. If the library is loaded by this class, the library will be unloaded upon completion on the function. To prevent the automatic unloading, use the setAutoUnload option. Only four types of functions are supported.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Tinkercell::CThread::CThread (MainWindow ∗ *main*, QLibrary ∗ *lib* = `0`, bool *autoUnload* = `false`)

constructor

Parameters

MainWindow the Tinkercell main window

QLibrary the dynamic library to load (optional)

bool whether or not to automatically unload the library

6.28.2.2 Tinkercell::CThread::CThread (MainWindow * *main*, const QString & *lib* = `tr("")`, bool *autoUnload* = `false`)

constructor

Parameters

MainWindow the Tinkercell main window

QString the name of the dynamic library to load (optional)

bool whether or not to automatically unload the library

6.28.3 Member Function Documentation

6.28.3.1 bool Tinkercell::CThread::autoUnload () `[virtual]`

whether or not to automatically unload the library when the thread is done running

Returns

bool

6.28.3.2 QWidget * Tinkercell::CThread::dialog (CThread * *newThread*, const QString & *title*, const QIcon & *icon* = `QIcon()`, bool *progressBar* = `true`) `[static]`

Creates a dialog with a progress bar for running a new thread. The dialog allows the user to terminate the thread.

Parameters

CThread * target thread

QString display text for the dialog

QIcon display icon for the dialog

bool whether or not to show a progress bar

6.28.3.3 QLibrary * Tinkercell::CThread::library () `[virtual]`

the library used inside this thread

Returns

QLibrary*

6.28.3.4 QLibrary ∗ Tinkercell::CThread::loadLibrary (const QString & *name*, QObject ∗ *parent* = 0) `[static]`

search the default tinkercell folders for the library and load it

Parameters

QString	name of library (with or without full path)
QObject	parent

Returns

QLibrary∗ the loaded library. 0 if cannot be loaded.

6.28.3.5 void Tinkercell::CThread::setArg (const DataTable< qreal > & *dat*) `[virtual]`

set the argument for the target function

Parameters

DataTable

6.28.3.6 void Tinkercell::CThread::setArg (const QString & *s*) `[virtual]`

set the argument for the target function

Parameters

QString

6.28.3.7 void Tinkercell::CThread::setArg (double *d*) `[virtual]`

set the argument for the target function

Parameters

double

6.28.3.8 void Tinkercell::CThread::setAutoUnload (bool *b*) `[virtual]`

set whether or not to automatically unload the library when the thread is done running

Parameters

bool

6.28.3.9 void Tinkercell::CThread::setCharFunction (const char ∗ *f*) `[virtual]`

set the function to run inside this threads

Parameters

void name of the function inside the library that has been loaded in this thread.

6.28.3.10 void Tinkercell::CThread::setDoubleFunction (const char ∗ *f*) `[virtual]`

set the function to run inside this threads

Parameters

void name of the function inside the library that has been loaded in this thread.

6.28.3.11 void Tinkercell::CThread::setFunction (void(∗)(tc_matrix) *f*) `[virtual]`

set the function to run inside this threads

Parameters

void function pointer

6.28.3.12 void Tinkercell::CThread::setFunction (void(∗)(const char ∗) *f*) `[virtual]`

set the function to run inside this threads

Parameters

void function pointer

6.28.3.13 void Tinkercell::CThread::setFunction (void(∗)(double) *f*) `[virtual]`

set the function to run inside this threads

Parameters

void function pointer

6.28.3.14 void Tinkercell::CThread::setFunction (void(∗)(void) *f*) `[virtual]`

set the function to run inside this threads

Parameters

void function pointer

6.28.3.15 void Tinkercell::CThread::setLibrary (const QString & *libname*) `[virtual]`

set the dynamic library for this threads.

Parameters

QLibrary*∗ library

6.28.3.16 void Tinkercell::CThread::setLibrary (QLibrary ∗ *lib*) `[virtual]`

set the dynamic library for this threads. The library will be loaded if it has not already been loaded

Parameters

QLibrary*∗ library

6.28.3.17 void Tinkercell::CThread::setMatrixFunction (const char ∗ *f*) `[virtual]`

set the function to run inside this threads

Parameters

void name of the function inside the library that has been loaded in this thread.

6.28.3.18 void Tinkercell::CThread::setVoidFunction (const char ∗ *f*) `[virtual]`

set the function to run inside this threads

Parameters

void name of the function inside the library that has been loaded in this thread.

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

- CThread.h
- CThread.cpp

6.29 Tinkercell::ConnectionGraphicsItem::CurveSegment Class Reference

A set of control points and two arrow heads.

`#include <ConnectionGraphicsItem.h>`

Public Member Functions

- **CurveSegment** (int)
- **CurveSegment** (int, ConnectionGraphicsItem::ControlPoint ∗)
- **CurveSegment** (const CurveSegment &)

Public Attributes

- ArrowHeadItem ∗ **arrowStart**
- ArrowHeadItem ∗ **arrowEnd**

6.29.1 Detailed Description

A set of control points and two arrow heads.

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

- ConnectionGraphicsItem.h
- ConnectionGraphicsItem.cpp

6.30 Tinkercell::DataColumn Class Reference

Public Member Functions

- **DataColumn** (DataTable< qreal > *data, int, int, int dt=1)
- virtual QwtData * **copy** () const
- virtual size_t **size** () const
- virtual double **x** (size_t index) const
- virtual double **y** (size_t index) const

Friends

- class **DataPlot**
- class **Plot2DWidget**

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

- Plot2DWidget.h
- Plot2DWidget.cpp

6.31 Tinkercell::Plot3DWidget::DataFunction Class Reference

Public Member Functions

- **DataFunction** (SurfacePlot &)
- double **operator()** (double x, double y)

Public Attributes

- DataTable< qreal > ∗ **dataTable**
- double **minX**
- double **minY**
- double **maxX**
- double **maxY**

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

- Plot3DWidget.h
- Plot3DWidget.cpp

6.32 Tinkercell::DataPlot Class Reference

Public Member Functions

- **DataPlot** (QWidget ∗parent=0)
- void **plot** (const DataTable< qreal > &, int x, const QString &title, int dt=1)
- virtual QSize **minimumSizeHint** () const
- virtual QSize **sizeHint** () const
- virtual void **setLogX** (bool)
- virtual void **setLogY** (bool)

Protected Slots

- void **itemChecked** (QwtPlotItem ∗, bool)
- void **setXAxis** (int)

Protected Member Functions

- void **processData** ()
- void **replotUsingHideList** ()

Protected Attributes

- DataTable< qreal > **dataTable**
- QwtPlotZoomer ∗ **zoomer**
- QStringList **hideList**
- int **xcolumn**
- int **delta**
- PlotTool::PlotType **type**

Static Protected Attributes

- static QList< QPen > **penList** = QList<QPen>()

Friends

- class **Plot2DWidget**
- class **GetPenInfoDialog**
- class **ShowHideLegendItemsWidget**

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

- Plot2DWidget.h
- Plot2DWidget.cpp

6.33 Tinkercell::DataTable< T > Class Template Reference

DataTable is a 2D vector with row names and column names.

`#include <DataTable.h>`

Public Member Functions

- virtual QString description () const

  get description of this table
- virtual QString & description ()

  get or set description of this table
- virtual QStringList columnNames () const

  get the column names
- virtual bool hasRow (const QString &) const

  check is this table has a row with the given name
- virtual bool hasColumn (const QString &) const

  check is this table has a column with the given name
- virtual QStringList rowNames () const

  get the row names
- virtual QString rowName (int i) const

  get the ith row name reference. can be used to change the row name
- virtual QString columnName (int i) const

  get the ith column name. cannot be used to change the column name
- virtual void setRowName (int i, const QString &name)

  get the ith row name. cannot be used to change the row name
- virtual void setColumnName (int i, const QString &name)

  get the ith column name reference. can be used to change the column name
- virtual void setColumnNames (const QStringList &names)

  set all the column names.
- virtual void setRowNames (const QStringList &names)

  set all the row names.
- virtual int rows () const

  get the number of rows
- virtual int columns () const

  get the number of columns

- virtual T & value (int i, int j=0)

  get the value at the ith row and jth column. can also be used to set the value

- virtual T & value (const QString &r, const QString &c)

  get the value using row and column names. can also be used to set the value. Slower than using value(int,int)

- virtual T & value (const QString &r, int j=0)

  get the value using row name. can also be used to set the value. Slower than using value(int,int)

- virtual T & value (int i, const QString &c)

  get the value using column name. can also be used to set the value. Slower than using value(int,int)

- virtual bool operator== (const DataTable< T > &D)

  checks if the two data table's headers and contents are the same

- virtual bool operator!= (const DataTable< T > &D)

  exactly opposite of operator ==

- virtual T at (int i, int j=0) const

  get the value using row and column number. cannot also be used to set the value.

- virtual T at (const QString &r, const QString &c) const

  get the value using row and column name. cannot also be used to set the value.

- virtual T at (const QString &r, int j=0) const

  get the value using row name. cannot also be used to set the value.

- virtual T at (int i, const QString &c) const

  get the value using column name. cannot also be used to set the value.

- virtual void resize (int m, int n=1)

  set the size of the data table

- virtual bool insertRow (int k, const QString &row)

  insert a new row at the given location with the given name. Insertion will fail if there is already a row with the same name

- virtual bool insertColumn (int k, const QString &col)

  insert a new column at the given location with the given name. Insertion will fail if there is already a column with the same name

- virtual bool removeRow (int k)

  remove an existing row at the given index.

- virtual bool removeRow (const QString &name)

  remove an existing row with the given name.

- virtual bool removeColumn (int k)

  remove an existing column at the given index.

- virtual bool removeColumn (const QString &name)

  remove an existing col with the given name.

- virtual void swapRows (int i1, int i2)

  swap two rows. Nothing will happen if the given numbers are outside the table

- virtual void swapColumns (int j1, int j2)

  swap two columns. Nothing will happen if the given numbers are outside the table

- virtual void swapRows (const QString &s1, const QString &s2)

  swap two rows using their name. Nothing will happen if the given numbers are outside the table

- virtual void swapColumns (const QString &s1, const QString &s2)

  swap two columns using their name. Nothing will happen if the given numbers are outside the table

- virtual DataTable< T > transpose () const

  get transpose of the table. complexity = n∗m (use sparingly)

- void appendColumns (DataTable< T > ∗)

  append another data table's columns to this data table

- void appendRows (DataTable< T > ∗)

  append another data table's rows to this data table

Static Public Member Functions

- static DataTable< T > appendColumns (const QList< DataTable< T > ∗ > &)

  append multiple data tables column-wise

- static DataTable< T > appendRows (const QList< DataTable< T > ∗ > &)

  append multiple data tables row-wise

Protected Attributes

- QVector< T > dataMatrix

  the values in the table

- QVector< QString > colHeaders

  the column and row names

- QVector< QString > **rowHeaders**
- QHash< QString, int > colHash

  hash for quick lookup of row and columns by name

- QHash< QString, int > **rowHash**
- QString desc

  a description of this table (optional)

6.33.1 Detailed Description

template<typename T> class Tinkercell::DataTable< T >

DataTable is a 2D vector with row names and column names.

6.33.2 Member Function Documentation

6.33.2.1 template<typename T> DataTable< T > Tinkercell::DataTable< T >::appendColumns (const QList< DataTable< T > * > & *list*) `[inline, static]`

append multiple data tables column-wise

append multiple data tables' columns

Parameters

QList< DataTable<T>* > list of tables

Returns

DataTable<T> new data table

6.33.2.2 template<typename T> void Tinkercell::DataTable< T >::appendColumns (DataTable< T > * *other*) `[inline]`

append another data table's columns to this data table

append another data table's columns

Parameters

DataTable<T>* table to append

Returns

void

6.33.2.3 template<typename T> DataTable< T > Tinkercell::DataTable< T >::appendRows (const QList< DataTable< T > * > & *list*) `[inline, static]`

append multiple data tables row-wise

append multiple data tables' rows

Parameters

QList< DataTable<T>* > list of tables to append

Returns

DataTable<T> new data table

6.33.2.4 template<typename T> void Tinkercell::DataTable< T >::appendRows (DataTable< T > * *other*) `[inline]`

append another data table's rows to this data table

append another data table's rows

Parameters

DataTable<T>* table to append

Returns

void

6.33.2.5 template<typename T > T Tinkercell::DataTable< T >::at (int *i*, const QString & *c*) const `[inline, virtual]`

get the value using column name. cannot also be used to set the value.

Parameters

int row number

int column name

Returns

T copy of value at given row and column. returns value at 0 if row and column are not in the table

6.33.2.6 template<typename T > T Tinkercell::DataTable< T >::at (const QString & *r*, int *j* = `0`) const `[inline, virtual]`

get the value using row name. cannot also be used to set the value.

Parameters

QString row name

int column number (defaults to 0)

Returns

T copy of value at given row and column. returns value at 0 if row and column are not in the table

Parameters

QString row name

int column number

Returns

T copy of value at given row and column. returns value at 0 if row and column are not in the table

6.33.2.7 template<typename T > T Tinkercell::DataTable< T >::at (const QString & *r*, const QString & *c*) const `[inline, virtual]`

get the value using row and column name. cannot also be used to set the value.

Parameters

QString row name

QString column name

Returns

T copy of value at given row and column. returns value at 0 if row and column are not in the table

6.33.2.8 template<typename T > T Tinkercell::DataTable< T >::at (int *i*, int *j* = `0`) const `[inline, virtual]`

get the value using row and column number. cannot also be used to set the value.

Parameters

int row number

int column number (defaults to 0)

Returns

T copy of value at given row and column. returns value at 0 if row and column are not in the table

Parameters

int row number

int column number

Returns

T copy of value at given row and column. returns value at 0 if row and column are not in the table

6.33.2.9 template<typename T > QString Tinkercell::DataTable< T >::columnName (int *i*) const `[inline, virtual]`

get the ith column name. cannot be used to change the column name

Parameters

int col number

Returns

QString copy of the ith column name

6.33.2.10 template<typename T > QStringList Tinkercell::DataTable< T >::columnNames () const `[inline, virtual]`

get the column names

Returns

QStringList column names (copy)
QVector reference to the actural column names

6.33.2.11 template<typename T > int Tinkercell::DataTable< T >::columns () const `[inline, virtual]`

get the number of columns

Returns

int number of columns

6.33.2.12 template<typename T > bool Tinkercell::DataTable< T >::hasColumn (const QString & *s*) const `[inline, virtual]`

check is this table has a column with the given name

Parameters

QString column name

Returns

bool true if the column with the name exists

6.33.2.13 template<typename T > bool Tinkercell::DataTable< T >::hasRow (const QString & *s*) const `[inline, virtual]`

check is this table has a row with the given name

Parameters

QString row name

Returns

bool true if the row with the name exists

6.33.2.14 template<typename T > bool Tinkercell::DataTable< T >::insertColumn (int *k*, const QString & *col*) `[inline, virtual]`

insert a new column at the given location with the given name. Insertion will fail if there is already a column with the same name

Parameters

int column number

QString column name

Returns

Boolean false if failed, true if successful

6.33.2.15 template<typename T > bool Tinkercell::DataTable< T >::insertRow (int *k*, const QString & *row*) `[inline, virtual]`

insert a new row at the given location with the given name. Insertion will fail if there is already a row with the same name

Parameters

int row number

QString row name

Returns

Boolean false if failed, true if successful

6.33.2.16 template<typename T> bool Tinkercell::DataTable< T >::operator!= (const DataTable< T > & *D*) `[inline, virtual]`

exactly opposite of operator ==

Parameters

DataTable<T>

Returns

bool

6.33.2.17 template<typename T> bool Tinkercell::DataTable< T >::operator== (const DataTable< T > & *D*) `[inline, virtual]`

checks if the two data table's headers and contents are the same

Parameters

DataTable<T>

Returns

bool

6.33.2.18 template<typename T > bool Tinkercell::DataTable< T >::removeColumn (const QString & *name*) `[inline, virtual]`

remove an existing col with the given name.

Parameters

QString row name

Returns

Boolean false if failed, true if successful

6.33.2.19 template<typename T > bool Tinkercell::DataTable< T >::removeColumn (int *k*) `[inline, virtual]`

remove an existing column at the given index.

Parameters

int column number

Returns

Boolean false if failed, true if successful

6.33.2.20 template<typename T > bool Tinkercell::DataTable< T >::removeRow (const QString & *name*) `[inline, virtual]`

remove an existing row with the given name.

Parameters

QString row name

Returns

Boolean false if failed, true if successful

6.33.2.21 template<typename T > bool Tinkercell::DataTable< T >::removeRow (int *k*) `[inline, virtual]`

remove an existing row at the given index.

Parameters

int row number

Returns

Boolean false if failed, true if successful

6.33.2.22 template<typename T > void Tinkercell::DataTable< T >::resize (int *m*, int *n* = `1`) `[inline, virtual]`

set the size of the data table

Parameters

int row count

int column count (defaults to 1)

Returns

void

Parameters

int row count

int column count

Returns

void

6.33.2.23 template<typename T > QString Tinkercell::DataTable< T >::rowName (int *i*) const `[inline, virtual]`

get the ith row name reference. can be used to change the row name

Parameters

int col number

Returns

QString copy to the ith row name

6.33.2.24 template<typename T > QStringList Tinkercell::DataTable< T >::rowNames () const `[inline, virtual]`

get the row names

Returns

QStringList row names (copy)
QVector reference to the actural row names

6.33.2.25 template<typename T > int Tinkercell::DataTable< T >::rows () const `[inline, virtual]`

get the number of rows

Returns

int number of rows

6.33.2.26 template<typename T > void Tinkercell::DataTable< T >::setColumnName (int *i*, const QString & *name*) `[inline, virtual]`

get the ith column name reference. can be used to change the column name

Parameters

int col number

QString name

Returns

QString reference to the ith column name

6.33.2.27 template<typename T > void Tinkercell::DataTable< T >::setColumnNames (const QStringList & *lst*) `[inline, virtual]`

set all the column names.

Parameters

QStringList vector of strings

Returns

void

6.33.2.28 template<typename T > void Tinkercell::DataTable< T >::setRowName (int *i*, const QString & *name*) `[inline, virtual]`

get the ith row name. cannot be used to change the row name

Parameters

int row number

QString name

Returns

QString reference of the ith row name

Parameters

int row number

Returns

QString reference of the ith row name

6.33.2.29 template<typename T > void Tinkercell::DataTable< T >::setRowNames (const QStringList & *lst*) `[inline, virtual]`

set all the row names.

Parameters

QStringList vector of strings

Returns

void

6.33.2.30 template<typename T > void Tinkercell::DataTable< T >::swapColumns (const QString & *s1*, const QString & *s2*) `[inline, virtual]`

swap two columns using their name. Nothing will happen if the given numbers are outside the table

Parameters

int first column name

int second column name

Returns

void

6.33.2.31 template<typename T > void Tinkercell::DataTable< T >::swapColumns (int *j1*, int *j2*) `[inline, virtual]`

swap two columns. Nothing will happen if the given numbers are outside the table

Parameters

int first column number

int second column number

Returns

void

6.33.2.32 template<typename T > void Tinkercell::DataTable< T >::swapRows (const QString & *s1*, const QString & *s2*) `[inline, virtual]`

swap two rows using their name. Nothing will happen if the given numbers are outside the table

Parameters

int first row name

int second row name

Returns

void

6.33.2.33 template<typename T > void Tinkercell::DataTable< T >::swapRows (int *i1*, int *i2*) `[inline, virtual]`

swap two rows. Nothing will happen if the given numbers are outside the table

Parameters

int first row number

int second row number

Returns

void

6.33.2.34 template<typename T > DataTable< T > Tinkercell::DataTable< T >::transpose () const `[inline, virtual]`

get transpose of the table. complexity = n∗m (use sparingly)

Returns

DataTable<T> new data table
new data table

6.33.2.35 template<typename T > T & Tinkercell::DataTable< T >::value (int *i*, const QString & *c*) `[inline, virtual]`

get the value using column name. can also be used to set the value. Slower than using value(int,int)

Parameters

int row number

QString column name

Returns

T reference to value at given row and column. returns value at 0 if row and column are not in the table

6.33.2.36 template<typename T > T & Tinkercell::DataTable< T >::value (const QString & *r*, int *j* = 0) `[inline, virtual]`

get the value using row name. can also be used to set the value. Slower than using value(int,int)

Parameters

QString row name

int column number (defaults to 0)

Returns

T reference to value at given row and column. returns value at 0 if row and column are not in the table

Parameters

QString row name

int column number

Returns

T reference to value at given row and column. returns value at 0 if row and column are not in the table

6.33.2.37 template<typename T > T & Tinkercell::DataTable< T >::value (const QString & *r*, const QString & *c*) `[inline, virtual]`

get the value using row and column names. can also be used to set the value. Slower than using value(int,int)

Parameters

QString row name

QString column name

Returns

T reference to value at given row and column. returns value at 0 if row and column are not in the table

6.33.2.38 template<typename T > T & Tinkercell::DataTable< T >::value (int *i*, int *j* = `0`) `[inline, virtual]`

get the value at the ith row and jth column. can also be used to set the value

Parameters

int row number

int column number (defaults to 0)

Returns

T reference to value at ith row and jth column. returns value at 0 if i or j are not inside the table

Parameters

int row number (i)

int column number (j)

Returns

T reference to value at ith row and jth column. returns value at 0 if i or j are not inside the table

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

- DataTable.h

6.34 Tinkercell::GetPenInfoDialog Class Reference

Public Member Functions

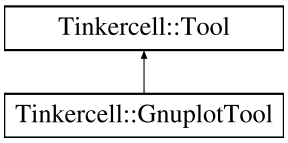
- **GetPenInfoDialog** (QWidget ∗parent)
- void **setPen** (const QPen &, int)
- QPen **getPen** () const
- int **currentIndex** () const

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

- Plot2DWidget.h
- Plot2DWidget.cpp

6.35 Tinkercell::GnuplotTool Class Reference

Inheritance diagram for Tinkercell::GnuplotTool:



Public Slots

- void **runScriptFile** (const QString &)
- void **makeScript** (const QString &)
- void **runScript** (const QString &)

Public Member Functions

- GnuplotTool (QWidget ∗parent=0)

  default constructor

- bool setMainWindow (MainWindow ∗main)

  set main window

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

- GnuplotTool.h
- GnuplotTool.cpp

6.36 Tinkercell::GraphicsScene Class Reference

The primary task of the graphics scene is to draws items. All interactions with the GraphicsScene is done through MainWindow or NetworkHandle. NetworkHandle provides functions such as move, insert, and remove. MainWindow relays all the signals, such as mouse and key events, from the GraphicsScene. So, there is rarely a need to directly interact with the GraphicsScene.

```
#include <GraphicsScene.h>
```

Signals

- void copyItems (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &)

    signals just before items are copied

- void itemsAboutToBeRemoved (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &, QList< QUndoCommand ∗ > &)

    signals just before items are deleted

- void itemsRemoved (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &, const QList< ItemHandle ∗ > &)

    signals whenever items are deleted

- void itemsAboutToBeInserted (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &, QList< QUndoCommand ∗ > &)

    signals whenever items are going to be added

- void itemsInserted (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &, const QList< ItemHandle ∗ > &)

    signals whenever items are added

- void itemsSelected (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, QPointF point, Qt::KeyboardModifiers modifiers)

    signals whenever items are selected (item can be sub-item, not top-level)

- void mousePressed (GraphicsScene ∗scene, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

    signals whenever an empty node of the screen is clicked

- void mouseReleased (GraphicsScene ∗scene, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

    signals whenever an empty node of the screen is clicked

- void mouseDoubleClicked (GraphicsScene ∗scene, QPointF point, QGraphicsItem ∗, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

    emits event when mouse is double clicked

- void mouseDragged (GraphicsScene ∗scene, QPointF from, QPointF to, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

    signals whenever mouse is dragged from one point to another

- void itemsAboutToBeMoved (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &item, QList< QPointF > &distance, QList< QUndoCommand ∗ > &)

  signals whenever items are going to be moved (each item is the top-most item)

- void itemsMoved (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &item, const QList< QPointF > &distance)

  signals whenever items are being moved (each item is the top-most item)

- void mouseMoved (GraphicsScene ∗scene, QGraphicsItem ∗item, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers, QList< QGraphicsItem ∗ > &)

  signals whenever mouse moves, and indicates whether it is on top of an item

- void mouseOnTopOf (GraphicsScene ∗scene, QGraphicsItem ∗item, QPointF point, Qt::KeyboardModifiers modifiers, QList< QGraphicsItem ∗ > &)

  signals whenever mouse is on top of an item

- void sceneRightClick (GraphicsScene ∗scene, QGraphicsItem ∗item, QPointF point, Qt::KeyboardModifiers modifiers)

  signals whenever right click is made on an item or sceen

- void keyPressed (GraphicsScene ∗scene, QKeyEvent ∗)

  signals whenever a key is pressed

- void keyReleased (GraphicsScene ∗scene, QKeyEvent ∗)

  signals whenever a key is released

- void escapeSignal (const QWidget ∗sender)

  signals whenever the current activities need to be stopped

- void filesDropped (const QList< QFileInfo > &files)

  signals whenever file(s) are dropped on the canvas

- void colorChanged (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items)

  signals whenever color of items are changed

- void parentItemChanged (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, const QList< QGraphicsItem ∗ > &parents)

  signals whenever item parents are changed

Public Member Functions

- MainWindow ∗ mainWindow () const

  the main window for this network

- ConsoleWindow ∗ console () const

  same as network->mainWindow->console()

- ItemHandle ∗ localHandle () const

  same as networkWindow->handle

- ItemHandle ∗ globalHandle () const

  same as network->globalHandle()

- virtual QRectF viewport () const

  Returns the currently visible window from the current graphics view.

- virtual void setBackground (const QPixmap &) const

  set the background image for the scene

- virtual void setForeground (const QPixmap &) const

  set the foreground image for the scene

- virtual QPointF & lastPoint ()

  Returns the point where mouse was clicked last on the scene coordinates.

- virtual QPoint & lastScreenPoint ()

  Returns the point where mouse was clicked last on the screen coordinates.

- virtual QList< QGraphicsItem ∗ > & selected ()

  Returns the list of pointers to items that are currently selected.

- virtual QRectF selectedRect ()

  Returns a rectangle that includes all the selected items.

- virtual QList< QGraphicsItem ∗ > & moving ()

  Returns the list of pointers to items that are currently being moved.

- virtual qreal ZValue ()

  top Z value

- GraphicsScene (NetworkHandle ∗network)

  Constructor: sets 10000x10000 scene.

- virtual ∼GraphicsScene ()

  destructor

- virtual void enableGrid (int sz=100)

  set the grid mode ON with the given grid size

- virtual void disableGrid ()

  set the grid mode OFF, which is same as setting grid size to 0

- virtual void setGridSize (int sz=100)

  set the grid size. If > 0, grid will be enabled. If 0, grid will be disabled

- virtual int gridSize () const

  get the grid size being used (0 = no grid)

- virtual void addItem (QGraphicsItem ∗item)

Add a new item to the scene (different from insert).

- virtual void centerOn (const QPointF &point) const

  place center at the point

- virtual void fitAll () const

  adjusts view to include all items

- virtual void fitInView (const QRectF &) const

  adjusts view to include the given rect

- virtual void popOut ()

  calls main window's popOut

- virtual void popIn ()

  calls main window's popIn

- virtual void clearSelection ()

  Clear all selection and moving items list.

- virtual void print (QPaintDevice ∗printer, const QRectF &rect=QRectF()) const

  send everything on the screen to a printer

- virtual void select (QGraphicsItem ∗item)

  select one item (does not deselect other items)

- virtual void select (const QList< QGraphicsItem ∗ > &item)

  select items (does not deselect previously selected items)

- virtual void selectAll ()

  select all items

- virtual void find (const QString &)

  select items with the given text

- virtual void deselect (QGraphicsItem ∗item)

  deselect one item

- virtual void deselect ()

  deselect all selected items

- virtual void copy ()

  copy selected items

- virtual void cut ()

  cut selected items

- virtual void paste ()

  paste copied items

- virtual void showToolTip (QPointF, const QString &)

  show a tooltip a the given position

- virtual void move (QGraphicsItem ∗item, const QPointF &distance)

  a simple move operation that also adds undo command to history window and emits associated signal(s)

- virtual void move (const QList< QGraphicsItem ∗ > &items, const QPointF &distance)

  a simple move operation that also adds undo command to history window and emits associated signal(s)

- virtual void move (const QList< QGraphicsItem ∗ > &items, const QList< QPointF > &distance)

  a simple move operation that also adds undo command to history window and emits associated signal(s)

- virtual void insert (const QString &name, QGraphicsItem ∗item)

  this command performs an insert and also adds undo command to history window and emits associated signal(s)

- virtual void insert (const QString &name, const QList< QGraphicsItem ∗ > &items)

  this command performs an insert and also adds undo command to history window and emits associated signal(s)

- virtual void remove (const QString &name, QGraphicsItem ∗item)

  this command performs an removal and also adds undo command to history window and emits associated signal(s)

- virtual void remove (const QString &name, const QList< QGraphicsItem ∗ > &items)

  this command performs an removal and also adds undo command to history window and emits associated signal(s)

- virtual void removeSelected ()

  remove selected items

- virtual void setBrush (const QString &name, QGraphicsItem ∗item, const QBrush &to)

  this command changes the brush of an item

- virtual void setBrush (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QBrush > &to)

  this command changes the brush of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setZValue (const QString &name, QGraphicsItem ∗item, qreal to)

  this command changes the z value of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setZValue (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< qreal > &to)

  this command changes the z value of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setPen (const QString &name, QGraphicsItem ∗item, const QPen &to)

  this command changes the pen of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setPen (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QPen > &to)

  this command changes the pen of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setBrushAndPen (const QString &name, QGraphicsItem ∗item, const QBrush &brush, const QPen &pen)

  this command changes the pen and/or brush of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setBrushAndPen (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QBrush > &brushes, const QList< QPen > &pens)

  this command changes the pen and/or brush of an item and also adds undo command to history window and emits associated signal(s)

- virtual void transform (const QString &name, QGraphicsItem ∗item, const QPointF &sizechange, qreal anglechange=0.0, bool VFlip=false, bool HFlip=false)

  this command changes the size, angle, and orientation of an item and also adds undo command to history window and emits associated signal(s)

- virtual void transform (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QPointF > &sizechange, const QList< qreal > &anglechange=QList< qreal >(), bool VFlip=false, bool HFlip=false)

  this command changes the size, angle, and orientation of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setParentItem (const QString &name, QGraphicsItem ∗item, QGraphicsItem ∗newParent)

  this command changes the parent of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setParentItem (const QString &name, const QList< QGraphicsItem ∗ > &items, QGraphicsItem ∗newParent)

  this command changes the parent of an item and also adds undo command to history window and emits associated signal(s)

- virtual void setParentItem (const QString &name, const QList< QGraphicsItem ∗ > &items, const QList< QGraphicsItem ∗ > &newParents)

  this command changes the parent of an item and also adds undo command to history window and emits associated signal(s)

- virtual void snapToGrid (QGraphicsItem ∗)

  snap the node item to the grid

- virtual void scaleView (qreal scaleFactor)

  zoom Precondition: None Postcondition: None

Public Attributes

- NetworkHandle ∗ network

  the network represented by this scene

- NetworkWindow ∗ networkWindow

  the network window widget inside of which this scene is located

- bool useDefaultBehavior

  indicates whether this scene is free to perform actions

- QMenu ∗ contextItemsMenu

  the context menu that is shown during right-click event on selected graphical items. Plugins can add new actions to this menu.

- QMenu ∗ contextScreenMenu

  the context menu that is shown during right-click event on the scene. Plugins can add new actions to this menu.

Static Public Attributes

- static bool USE_DEFAULT_BEHAVIOR = true

  each graphics scene has a default behavior, i.e. moving, selecing, deleting. Whether or not to use the default behavior is set using scene->useDefaultBehavior. This static variable is the default value for each scene's useDefaultBehavior variable, i.e. setting this to true will cause a newly constructed graphics scene to NOT use default behaviors.

- static int GRID = 0

  setting grid to a non-zero value forces node items to "fit" on the grid, where the gap between the grid lines is determined by this variable. The default is 0, i.e. no grid

- static QPen SelectionRectanglePen = Qt::NoPen

  pen that is used to draw the selection rectangle

- static QBrush SelectionRectangleBrush = QBrush(QColor(0,132,255,50))

  brush that is used to color the selection rectangle

- static QBrush BackgroundBrush = Qt::NoBrush

  brush used to draw the background for the scene

- static QPen GridPen = QPen(Qt::lightGray,2)

  pen used to draw the grid for the scene

- static QBrush ForegroundBrush = Qt::NoBrush

  brush used to draw the foreground for the scene

- static QBrush ToolTipBackgroundBrush = QBrush(QColor(36,28,28,125))

  brush used to draw the background of tool tips

- static QBrush ToolTipTextBrush = QBrush(QColor(255,255,255,255)) *brush used to draw the text for tool tips*

- static qreal MIN_DRAG_DISTANCE = 2.0

  the minimum distance that gets classified as a "drag". Anything less will be considered just a click.

Protected Member Functions

- virtual void hideToolTips ()

  hide the all tool tips

- virtual void hideGraphicalTools ()

  hide the all graphical tools

- virtual void showGraphicalTools ()

  show graphical tools for selected items

- virtual void scaleGraphicalTools ()

  scale the visible graphical tools according to viewport size

- virtual void mousePressEvent (QGraphicsSceneMouseEvent ∗mouseEvent)

  when mouse is pressed, the item at the position is added to selected list and moving list

- virtual void mouseDoubleClickEvent (QGraphicsSceneMouseEvent ∗mouseEvent)

  when mouse is double clicked, the item at the position is added to selected list and moving list

- virtual void mouseMoveEvent (QGraphicsSceneMouseEvent ∗mouseEvent)

  when mouse is moving, all items in moving list are moved

- virtual void mouseReleaseEvent (QGraphicsSceneMouseEvent ∗mouseEvent)

  when mouse is released, moving list is cleared

- virtual void keyPressEvent (QKeyEvent ∗event)

  when key is pressed

- virtual void keyReleaseEvent (QKeyEvent ∗event)

  when key is released

- virtual void contextMenuEvent (QGraphicsSceneContextMenuEvent ∗contextMenuEvent)

  context menu for the scene

- virtual void populateContextMenu ()

  populate the context menu using selected items' tools actions

- virtual void drawBackground (QPainter ∗painter, const QRectF &rect)

  draw background grid if in grid mode

- virtual void selectConnections (const QPointF &)

  used to select the entire connection during mouse click

Static Protected Member Functions

- static void clearStaticItems ()

  clears copied items

Protected Attributes

- int gridSz

  grid size. If zero, then disabled

- qreal lastZ

  topmost Z value

- bool contextMenuJustActivated

  a hack to prevent strange mouse movements after context menu event

- QGraphicsRectItem selectionRect

  rectanglular selection area

- QList< QGraphicsItem ∗ > toolTips

  list of temporary tool tips

- QPointF clickedPoint

  point where mouse is clicked

- QPoint clickedScreenPoint

  point where mouse is clicked on the screen

- Qt::MouseButton clickedButton

  button that was used when mouse was clicked

- bool mouseDown

  mouse is being pressed

- QList< QGraphicsItem ∗ > selectedItems

  list of pointers to selected items

- QList< ToolGraphicsItem ∗ > visibleTools

  list of pointers to tool items

- QList< QGraphicsItem ∗ > movingItems

  list of pointers to moving items

- QGraphicsItemGroup ∗ movingItemsGroup

  group of moving items

Static Protected Attributes

- static QList< QGraphicsItem ∗ > duplicateItems

  used to store copied items

- static GraphicsScene ∗ copiedFromScene

  used to store copied items

Friends

- class **MainWindow**
- class **NetworkWindow**
- class **NetworkHandle**
- class **GraphicsView**
- class **SymbolsTable**

6.36.1 Detailed Description

The primary task of the graphics scene is to draws items. All interactions with the GraphicsScene is done through MainWindow or NetworkHandle. NetworkHandle provides functions such as move, insert, and remove. MainWindow relays all the signals, such as mouse and key events, from the GraphicsScene. So, there is rarely a need to directly interact with the GraphicsScene.

6.36.2 Member Function Documentation

6.36.2.1 void Tinkercell::GraphicsScene::addItem (QGraphicsItem ∗ *item*) `[virtual]`

Add a new item to the scene (different from insert).

Add a new item to the scene Precondition: None Postcondition: None.

See also

insert

Parameters

QGraphicsItem∗ Tinkercell object

Returns

void

Parameters

Tinkercell object

Returns

void

6.36.2.2 void Tinkercell::GraphicsScene::centerOn (const QPointF & *point*) const `[virtual]`

place center at the point

place center at the point Precondition: None Postcondition: None

Parameters

QPointF point

Returns

void

Parameters

point

Returns

void

6.36.2.3 void Tinkercell::GraphicsScene::clearSelection () `[virtual]`

Clear all selection and moving items list.

Clear all selection and moving items list Precondition: None Postcondition: None.

Returns

void

6.36.2.4 void Tinkercell::GraphicsScene::colorChanged (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*) `[signal]`

signals whenever color of items are changed

Parameters

GraphicsScene ∗ scene where the event took place

QList<QGraphicsItem∗>& items that changed color

Returns

void

6.36.2.5 void Tinkercell::GraphicsScene::contextMenuEvent (QGraphicsSceneContextMenuEvent ∗ *mouseEvent*) `[protected, virtual]`

context menu for the scene

context menu for the scene Precondition: None Postcondition: None

Parameters

QGraphicsSceneContextMenuEvent ∗ context menu event

Returns

void

Parameters

context menu event

Returns

void

6.36.2.6 void Tinkercell::GraphicsScene::copyItems (GraphicsScene ∗ *scene*, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &) `[signal]`

signals just before items are copied

Parameters

GraphicsScene ∗ scene where the items are going to be copied

QList<QGraphicsItem∗>& list of graphics items going to be copied

QList<ItemHandle∗>& list of handles going to be copied (does NOT have to be the same number as items removed)

Returns

void

6.36.2.7 void Tinkercell::GraphicsScene::deselect () `[virtual]`

deselect all selected items

deselect items

Returns

void

6.36.2.8 void Tinkercell::GraphicsScene::deselect (QGraphicsItem ∗ *item*) `[virtual]`

deselect one item

deselect items

Parameters

QGraphicsItem∗ item to deselect

Returns

void

6.36.2.9 void Tinkercell::GraphicsScene::disableGrid () `[virtual]`

set the grid mode OFF, which is same as setting grid size to 0

Returns

void

6.36.2.10 void Tinkercell::GraphicsScene::enableGrid (int *sz* = `100`) `[virtual]`

set the grid mode ON with the given grid size

Parameters

double grid size (0 will disable grid)

Returns

void

6.36.2.11 void Tinkercell::GraphicsScene::escapeSignal (const QWidget ∗ *sender*) `[signal]`

signals whenever the current activities need to be stopped

Parameters

QWidget ∗ the widget that send the signal

Returns

void

6.36.2.12 void Tinkercell::GraphicsScene::filesDropped (const QList< QFileInfo > & *files*) `[signal]`

signals whenever file(s) are dropped on the canvas

Parameters

QList<QFileInfo>& the name(s) of the file(s)

Returns

void

6.36.2.13 void Tinkercell::GraphicsScene::fitAll () const `[virtual]`

adjusts view to include all items

Returns

void

6.36.2.14 void Tinkercell::GraphicsScene::fitInView (const QRectF & *rect*) const `[virtual]`

adjusts view to include the given rect

adjusts view to include rect

Parameters

QRectF

Returns

void

6.36.2.15 int Tinkercell::GraphicsScene::gridSize () const `[virtual]`

get the grid size being used (0 = no grid)

Returns

int

6.36.2.16 void Tinkercell::GraphicsScene::insert (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*) `[virtual]`

this command performs an insert and also adds undo command to history window and emits associated signal(s)

this command performs an insert and allows redo/undo of that insert

6.36.2.17 void Tinkercell::GraphicsScene::insert (const QString & *name*, QGraphicsItem ∗ *item*) `[virtual]`

this command performs an insert and also adds undo command to history window and emits associated signal(s)

Parameters

QString name of new item

QList<QPointF>& distance to move the items specified for each item

Returns

void

6.36.2.18 void Tinkercell::GraphicsScene::itemsAboutToBeInserted (GraphicsScene ∗ *scene*, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &, QList< QUndoCommand ∗ > &) `[signal]`

signals whenever items are going to be added

Parameters

GraphicsScene∗ scene where the items are added

QList<QGraphicsItem∗>& list of new graphics items

QList<ItemHandle∗>& list of new handles (does NOT have to be the same number as items)

QList<QUndoCommand∗>& list of commands that will be executed right before items are inserted

Returns

void

6.36.2.19 void Tinkercell::GraphicsScene::itemsAboutToBeMoved (GraphicsScene ∗ *scene*, QList< QGraphicsItem ∗ > & *item*, QList< QPointF > & *distance*, QList< QUndoCommand ∗ > &) `[signal]`

signals whenever items are going to be moved (each item is the top-most item)

Parameters

GraphicsScene∗ scene where the items were moved

QList<QGraphicsItem∗>& list of pointers to all moving items

QPointF distance by which items moved

Qt::KeyboardModifiers modifier keys being used when mouse clicked

QList<QUndoCommand∗>& list of commands that will be executed right before items are inserted

Returns

void

6.36.2.20 void Tinkercell::GraphicsScene::itemsAboutToBeRemoved (GraphicsScene ∗ *scene*, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &, QList< QUndoCommand ∗ > &) `[signal]`

signals just before items are deleted

Parameters

GraphicsScene ∗ scene where the items are going to be removed

QList<QGraphicsItem∗>& list of graphics items going to be removed

QList<ItemHandle∗>& list of handles going to be removed (does NOT have to be the same number as items removed)

QList<QUndoCommand∗>& list of commands that will be executed right before items are removed

Returns

void

6.36.2.21 void Tinkercell::GraphicsScene::itemsInserted (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > &, const QList< ItemHandle ∗ > &) `[signal]`

signals whenever items are added

Parameters

GraphicsScene∗ scene where the items were added

QList<QGraphicsItem∗>& list of new graphics items

QList<ItemHandle∗>& list of new handles (does NOT have to be the same number as items)

Returns

void

6.36.2.22 void Tinkercell::GraphicsScene::itemsMoved (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *item*, const QList< QPointF > & *distance*) `[signal]`

signals whenever items are being moved (each item is the top-most item)

Parameters

GraphicsScene∗ scene where the items were moved

QList<QGraphicsItem∗>& list of pointers to all moving items

QPointF distance by which items moved

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.36.2.23 void Tinkercell::GraphicsScene::itemsRemoved (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > &, const QList< ItemHandle ∗ > &) `[signal]`

signals whenever items are deleted

Parameters

GraphicsScene∗ scene where the items were removed

QList<QGraphicsItem∗>& list of items removed

QList<ItemHandle∗>& list of handles removed (does NOT have to be the same number as items removed)

Returns

void

6.36.2.24 void Tinkercell::GraphicsScene::itemsSelected (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, QPointF *point*, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever items are selected (item can be sub-item, not top-level)

Parameters

GraphicsScene∗ scene where items are selected

QList<QGraphicsItem∗>& list of all selected item pointers

QPointF point where mouse is clicked

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.36.2.25 void Tinkercell::GraphicsScene::keyPressed (GraphicsScene ∗ *scene*, QKeyEvent ∗) `[signal]`

signals whenever a key is pressed

Parameters

GraphicsScene∗ scene where the event took place

QKeyEvent ∗ key that is pressed

Returns

void

6.36.2.26 void Tinkercell::GraphicsScene::keyPressEvent (QKeyEvent ∗ *keyEvent*) `[protected, virtual]`

when key is pressed

when key is pressed Precondition: None Postcondition: None

Parameters

QKeyEvent ∗ key event

Returns

void

Parameters

key event

Returns

void

6.36.2.27 void Tinkercell::GraphicsScene::keyReleased (GraphicsScene ∗ *scene*, QKeyEvent ∗) `[signal]`

signals whenever a key is released

Parameters

GraphicsScene∗ scene where the event took place
QKeyEvent ∗ key that is released

Returns

void

6.36.2.28 void Tinkercell::GraphicsScene::keyReleaseEvent (QKeyEvent ∗ *keyEvent*) `[protected, virtual]`

when key is released

when key is released Precondition: None Postcondition: None

Parameters

QKeyEvent ∗ key event

Returns

void

Parameters

key event

Returns

void

6.36.2.29 QPointF & Tinkercell::GraphicsScene::lastPoint () `[virtual]`

Returns the point where mouse was clicked last on the scene coordinates.

Returns the point where mouse was clicked last Precondition: None Postcondition: None.

Parameters

void

Returns

QPointF& ref to last clicked point on the scene

Parameters

void

Returns

ref to last clicked point

6.36.2.30 QPoint & Tinkercell::GraphicsScene::lastScreenPoint () `[virtual]`

Returns the point where mouse was clicked last on the screen coordinates.

Returns the point where mouse was clicked last Precondition: None Postcondition: None.

Parameters

void

Returns

QPointF& ref to last clicked point on the screen

Parameters

void

Returns

ref to last clicked point

6.36.2.31 void Tinkercell::GraphicsScene::mouseDoubleClicked (GraphicsScene ∗ *scene*, QPointF *point*, QGraphicsItem ∗, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

emits event when mouse is double clicked

Parameters

GraphicsScene∗ scene where the event took place
point where mouse is clicked
modifier keys being used when mouse clicked

Returns

void

6.36.2.32 void Tinkercell::GraphicsScene::mouseDoubleClickEvent (QGraphicsSceneMouseEvent ∗ *mouseEvent*) `[protected, virtual]`

when mouse is double clicked, the item at the position is added to selected list and moving list

emits signal when mouse is double clicked Precondition: None Postcondition: None

Parameters

QGraphicsSceneMouseEvent ∗ mouse event

Returns

void

Parameters

mouse event

Returns

void

6.36.2.33 void Tinkercell::GraphicsScene::mouseDragged (GraphicsScene ∗ *scene*, QPointF *from*, QPointF *to*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever mouse is dragged from one point to another

Parameters

GraphicsScene∗ scene where the event took place

QPointF point where mouse is clicked first

QPointF point where mouse is released

Qt::MouseButton button being pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.36.2.34 void Tinkercell::GraphicsScene::mouseMoved (GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, QPointF *point*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*, QList< QGraphicsItem ∗ > &) `[signal]`

signals whenever mouse moves, and indicates whether it is on top of an item

Parameters

GraphicsScene∗ scene where the event took place

QGraphicsItem∗ pointer to item that mouse is on top of

QPointF point where mouse is clicked

Qt::MouseButton button being pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

QList<QGraphicsItem∗>& list of items that are being moved with the mouse

Returns

void

6.36.2.35 void Tinkercell::GraphicsScene::mouseMoveEvent (QGraphicsSceneMouseEvent ∗ *mouseEvent*) `[protected, virtual]`

when mouse is moving, all items in moving list are moved

when mouse is moving, all items in moving list are moved Precondition: None Postcondition: None

Parameters

QGraphicsSceneMouseEvent ∗ mouse event

Returns

void

Parameters

mouse event

Returns

void

6.36.2.36 void Tinkercell::GraphicsScene::mouseOnTopOf (GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, QPointF *point*, Qt::KeyboardModifiers *modifiers*, QList< QGraphicsItem ∗ > &) `[signal]`

signals whenever mouse is on top of an item

Parameters

GraphicsScene∗ scene where the event took place

QGraphicsItem∗ pointer to item that mouse is on top of

QPointF point where mouse is clicked

Qt::KeyboardModifiers modifier keys being used when mouse clicked

QList<QGraphicsItem∗>& list of items that are being moved with the mouse

Returns

void

6.36.2.37 void Tinkercell::GraphicsScene::mousePressed (GraphicsScene ∗ *scene*, QPointF *point*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever an empty node of the screen is clicked

Parameters

GraphicsScene∗ scene where the event took place

QPointF point where mouse is clicked

Qt::MouseButton which button was pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.36.2.38 void Tinkercell::GraphicsScene::mousePressEvent (QGraphicsSceneMouseEvent ∗ *mouseEvent*) `[protected, virtual]`

when mouse is pressed, the item at the position is added to selected list and moving list

when mouse is pressed, the item at the position is added to selected list and moving list Precondition: None Postcondition: None

Parameters

QGraphicsSceneMouseEvent ∗ mouse event

Returns

void

Parameters

mouse event

Returns

void

6.36.2.39 void Tinkercell::GraphicsScene::mouseReleased (GraphicsScene ∗ *scene*, QPointF *point*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever an empty node of the screen is clicked

Parameters

GraphicsScene∗ scene where the event took place

QPointF point where mouse is clicked

Qt::MouseButton which button was pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.36.2.40 void Tinkercell::GraphicsScene::mouseReleaseEvent (QGraphicsSceneMouseEvent ∗ *mouseEvent*) `[protected, virtual]`

when mouse is released, moving list is cleared

when mouse is released, moving list is cleared Precondition: None Postcondition: None

Parameters

QGraphicsSceneMouseEvent ∗ mouse event

Returns

void

Parameters

mouse event

Returns

void

6.36.2.41 void Tinkercell::GraphicsScene::move (const QList< QGraphicsItem * > & *items*, const QList< QPointF > & *distance*) `[virtual]`

a simple move operation that also adds undo command to history window and emits associated signal(s)

a simple move operation with undo

Parameters

QList<QGraphicsItem*>& items to move

QList<QPointF>& distance to move the items specified for each item

Returns

void

6.36.2.42 void Tinkercell::GraphicsScene::move (const QList< QGraphicsItem * > & *items*, const QPointF & *distance*) `[virtual]`

a simple move operation that also adds undo command to history window and emits associated signal(s)

a simple move operation with undo

Parameters

QList<QGraphicsItem*>& items to move

QPointF distance to move the items (same for all items)

Returns

void

6.36.2.43 void Tinkercell::GraphicsScene::move (QGraphicsItem * *item*, const QPointF & *distance*) `[virtual]`

a simple move operation that also adds undo command to history window and emits associated signal(s)

a simple move operation with undo

Parameters

QGraphicsItem * item to move

QPointF distance to move the item

Returns

void

6.36.2.44 QList< QGraphicsItem * > & Tinkercell::GraphicsScene::moving () `[virtual]`

Returns the list of pointers to items that are currently being moved.

Returns the list of pointers to items that are currently being moved Precondition: None Postcondition: None.

Parameters

void

Returns

QList<QGraphicsItem∗>& list of pointers to moving items

Parameters

void

Returns

list of pointers to moving items

6.36.2.45 void Tinkercell::GraphicsScene::parentItemChanged (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QList< QGraphicsItem ∗ > & *parents*) `[signal]`

signals whenever item parents are changed

Parameters

GraphicsScene	∗ scene where the event took place
QList<QGraphicsItem∗>&	items
QList<QGraphicsItem∗>&	new parents

Returns

void

6.36.2.46 void Tinkercell::GraphicsScene::popIn () `[virtual]`

calls main window's popIn

Returns

void

6.36.2.47 void Tinkercell::GraphicsScene::popOut () `[virtual]`

calls main window's popOut

Returns

void

6.36.2.48 void Tinkercell::GraphicsScene::populateContextMenu () `[protected, virtual]`

populate the context menu using selected items' tools actions

Returns

void

6.36.2.49 void Tinkercell::GraphicsScene::print (QPaintDevice ∗ *printer*, const QRectF & *rect* = `QRectF()`) const `[virtual]`

send everything on the screen to a printer

prints the current scene

Parameters

QPaintDevice ∗ printer

QRectF region to print

Returns

void

6.36.2.50 void Tinkercell::GraphicsScene::remove (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*) `[virtual]`

this command performs an removal and also adds undo command to history window and emits associated signal(s)

this command performs an removal and allows redo/undo of that removal

6.36.2.51 void Tinkercell::GraphicsScene::remove (const QString & *name*, QGraphicsItem ∗ *item*) `[virtual]`

this command performs an removal and also adds undo command to history window and emits associated signal(s)

this command performs an removal and allows redo/undo of that removal

6.36.2.52 void Tinkercell::GraphicsScene::scaleView (qreal *scaleFactor*) `[virtual]`

zoom Precondition: None Postcondition: None

Parameters

scale factor

Returns

void

6.36.2.53 void Tinkercell::GraphicsScene::sceneRightClick (GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, QPointF *point*, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever right click is made on an item or sceen

Parameters

GraphicsScene∗ scene where the event took place

QGraphicsItem∗ pointer to item that mouse is clicked on

QPointF point where mouse is clicked

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.36.2.54 void Tinkercell::GraphicsScene::select (const QList< QGraphicsItem ∗ > & *item*) `[virtual]`

select items (does not deselect previously selected items)

select items

Parameters

QList<QGraphicsItem∗>& items to select

Returns

void

6.36.2.55 void Tinkercell::GraphicsScene::select (QGraphicsItem ∗ *item*) `[virtual]`

select one item (does not deselect other items)

select items

Parameters

QGraphicsItem∗ item to select

Returns

void

6.36.2.56 QList< QGraphicsItem ∗ > & Tinkercell::GraphicsScene::selected () `[virtual]`

Returns the list of pointers to items that are currently selected.

Returns the list of pointers to items that are currently selected Precondition: None Postcondition: None.

Parameters

void

Returns

QList<QGraphicsItem∗>& list of pointers to selected items

Parameters

void

Returns

list of pointers to selected items

6.36.2.57 QRectF Tinkercell::GraphicsScene::selectedRect () `[virtual]`

Returns a rectangle that includes all the selected items.

Returns a rectangle that includes all the selected items Precondition: None Postcondition: None.

Parameters

void

Returns

QRectF bounding rect for selected items

Parameters

void

Returns

bounding rect for selected items

6.36.2.58 void Tinkercell::GraphicsScene::setBrush (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QBrush > & *to*) `[virtual]`

this command changes the brush of an item and also adds undo command to history window and emits associated signal(s)

this command changes the brush of an item

6.36.2.59 void Tinkercell::GraphicsScene::setBrushAndPen (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QBrush > & *brushes*, const QList< QPen > & *pens*) `[virtual]`

this command changes the pen and/or brush of an item and also adds undo command to history window and emits associated signal(s)

this command changes the pen of an item

6.36.2.60 void Tinkercell::GraphicsScene::setBrushAndPen (const QString & *name*, QGraphicsItem ∗ *item*, const QBrush & *brush*, const QPen & *pen*) `[virtual]`

this command changes the pen and/or brush of an item and also adds undo command to history window and emits associated signal(s)

this command changes the pen of an item

6.36.2.61 void Tinkercell::GraphicsScene::setGridSize (int *sz* = `100`) `[virtual]`

set the grid size. If > 0, grid will be enabled. If 0, grid will be disabled

Parameters

double grid size (0 will disable grid)

Returns

void

6.36.2.62 void Tinkercell::GraphicsScene::setParentItem (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QGraphicsItem ∗ > & *newParents*) `[virtual]`

this command changes the parent of an item and also adds undo command to history window and emits associated signal(s)

this command changes the parent of an item

6.36.2.63 void Tinkercell::GraphicsScene::setParentItem (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, QGraphicsItem ∗ *newParent*) `[virtual]`

this command changes the parent of an item and also adds undo command to history window and emits associated signal(s)

this command changes the parent of an item

6.36.2.64 void Tinkercell::GraphicsScene::setParentItem (const QString & *name*, QGraphicsItem ∗ *item*, QGraphicsItem ∗ *newParent*) `[virtual]`

this command changes the parent of an item and also adds undo command to history window and emits associated signal(s)

this command changes the parent of an item

6.36.2.65 void Tinkercell::GraphicsScene::setPen (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QPen > & *to*) `[virtual]`

this command changes the pen of an item and also adds undo command to history window and emits associated signal(s)

this command changes the pen of an item

6.36.2.66 void Tinkercell::GraphicsScene::setPen (const QString & *name*, QGraphicsItem ∗ *item*, const QPen & *to*) `[virtual]`

this command changes the pen of an item and also adds undo command to history window and emits associated signal(s)

this command changes the pen of an item

6.36.2.67 void Tinkercell::GraphicsScene::snapToGrid (QGraphicsItem ∗ *item*) `[virtual]`

snap the node item to the grid

Parameters

NodeGraphicsItem∗

Returns

void

6.36.2.68 void Tinkercell::GraphicsScene::transform (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, const QList< QPointF > & *sizechange*, const QList< qreal > & *anglechange* = `QList<qreal>()`, bool *VFlip* = `false`, bool *HFlip* = `false`) `[virtual]`

this command changes the size, angle, and orientation of an item and also adds undo command to history window and emits associated signal(s)

this command changes the size, angle, and orientation of an item

6.36.2.69 void Tinkercell::GraphicsScene::transform (const QString & *name*, QGraphicsItem ∗ *item*, const QPointF & *sizechange*, qreal *anglechange* = `0.0`, bool *VFlip* = `false`, bool *HFlip* = `false`) `[virtual]`

this command changes the size, angle, and orientation of an item and also adds undo command to history window and emits associated signal(s)

this command changes the size, angle, and orientation of an item

6.36.2.70 QRectF Tinkercell::GraphicsScene::viewport () const `[virtual]`

Returns the currently visible window from the current graphics view.

Returns the currently visible window.

Parameters

void

Returns

QRectF rectangle

Parameters

void

Returns

rectangle

6.36.2.71 qreal Tinkercell::GraphicsScene::ZValue () `[virtual]`

top Z value

top Z value Precondition: None Postcondition: None

Returns

double

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

- GraphicsScene.h
- GraphicsScene.cpp

6.37 Tinkercell::GraphicsView Class Reference

GraphicsView class that is used to view the contents of a GraphicsScene. The class inherits from QGraphicsView.

`#include <GraphicsView.h>`

Signals

- void itemsDropped (GraphicsScene ∗, const QString &, const QPointF &)

  signal is emitted when some object OTHER than files are dropped on the canvas

Protected Member Functions

- virtual void drawBackground (QPainter ∗painter, const QRectF &rect)

  draw background
- virtual void drawForeground (QPainter ∗painter, const QRectF &rect)

  draw foreground
- virtual void dropEvent (QDropEvent ∗)

  drag and drop
- virtual void dragEnterEvent (QDragEnterEvent ∗event)

  drag and drop
- virtual void dragMoveEvent (QDragMoveEvent ∗event)

  drag and drop
- virtual void wheelEvent (QWheelEvent ∗event)

  mouse wheel event
- virtual void scrollContentsBy (int dx, int dy)

  scroll event
- virtual void mousePressEvent (QMouseEvent ∗event)

  mouse event. sets the currentGraphicsView for NetworkWindow
- virtual void keyPressEvent (QKeyEvent ∗event)

  mouse event. sets the currentGraphicsView for NetworkWindow

Friends

- class **GraphicsScene**
- class **NetworkWindow**
- class **NetworkHandle**
- class **MainWindow**

6.37.1 Detailed Description

GraphicsView class that is used to view the contents of a GraphicsScene. The class inherits from QGraphicsView.

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

- GraphicsView.h
- GraphicsView.cpp

6.38 Tinkercell::HistoryWindow Class Reference

This is a small class extending QUndoView that manages a stack of undo commands.

`#include <HistoryWindow.h>`

Public Slots

- void **undo** ()
- void **redo** ()
- void **push** (QUndoCommand ∗command)

6.38.1 Detailed Description

This is a small class extending QUndoView that manages a stack of undo commands.

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

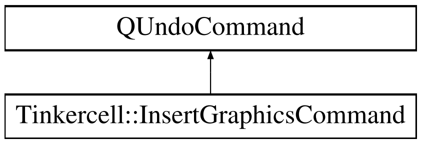
- HistoryWindow.h
- HistoryWindow.cpp

6.39 Tinkercell::InsertGraphicsCommand Class Reference

this command performs an insert and allows redo/undo of that insert

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::InsertGraphicsCommand:



Public Member Functions

- InsertGraphicsCommand (const QString &name, GraphicsScene ∗scene, QGraphicsItem ∗item, bool checkNames=true)

  constructor

- InsertGraphicsCommand (const QString &name, GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, bool checkNames=true)

  constructor

- void redo ()

  redo the change

- void undo ()

  undo the change

- virtual ∼InsertGraphicsCommand ()

  destructor

6.39.1 Detailed Description

this command performs an insert and allows redo/undo of that insert

6.39.2 Constructor & Destructor Documentation

6.39.2.1 Tinkercell::InsertGraphicsCommand::InsertGraphicsCommand (const QString & *name*, GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, bool *checkNames* = `true`)

constructor

Parameters

QString	name of command
GraphicsScene∗	where change happened
QGraphicsItem∗	item that is inserted
bool	check for uniqueness of names before inserting (default = true)

6.39.2.2 Tinkercell::InsertGraphicsCommand::InsertGraphicsCommand (const QString & *name*, GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, bool *checkNames* = `true`)

constructor

Parameters

QString name of command

GraphicsScene∗ where change happened

QList<QGraphicsItem∗>& items that are inserted

bool check for uniqueness of names before inserting (default = true)

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

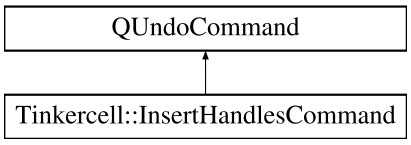
- UndoCommands.h
- UndoCommands.cpp

6.40 Tinkercell::InsertHandlesCommand Class Reference

this command inserts new handles to a NetworkHandle

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::InsertHandlesCommand:



Public Member Functions

- InsertHandlesCommand (TextEditor ∗, const QList< ItemHandle ∗ > &, bool checkNames=true)

  constructor
- InsertHandlesCommand (TextEditor ∗, ItemHandle ∗, bool checkNames=true)

  constructor
- ∼InsertHandlesCommand ()

  destructor. deletes all text items and their handles (if not containing any graphics items)
- void redo ()

  redo the change
- void undo ()

  undo the change

6.40.1 Detailed Description

this command inserts new handles to a NetworkHandle

6.40.2 Constructor & Destructor Documentation

6.40.2.1 Tinkercell::InsertHandlesCommand::InsertHandlesCommand (TextEditor ∗ *textEditor*, const QList< ItemHandle ∗ > & *list*, bool *checkNames* = `true`)

constructor

Parameters

NetworkHandle∗	window where items are inserted
QList<ItemHandle∗>	new items
bool	check for uniqueness of names before inserting

6.40.2.2 Tinkercell::InsertHandlesCommand::InsertHandlesCommand (TextEditor ∗ *textEditor*, ItemHandle ∗ *h*, bool *checkNames* = `true`)

constructor

Parameters

***NetworkHandle*∗** window where items are inserted

***ItemHandle*∗** new item

bool check for uniqueness of names before inserting

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

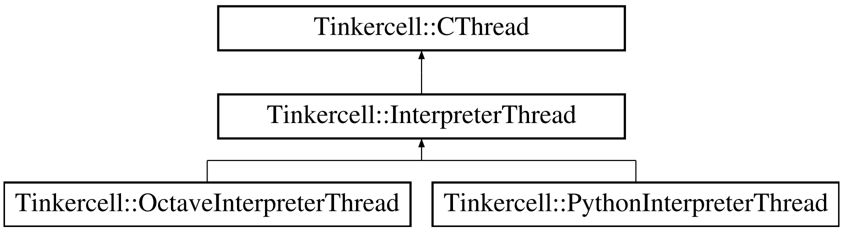
- UndoCommands.h
- UndoCommands.cpp

6.41 Tinkercell::InterpreterThread Class Reference

This class is used to run interpreters such as python, perl, octave, R, etc. This is the parent class that provides the basic structure for loading the library that will embed one of these languages.

`#include <InterpreterThread.h>`

Inheritance diagram for Tinkercell::InterpreterThread:


Tinkercell::CThread
Tinkercell::InterpreterThread
Tinkercell::OctaveInterpreterThread
Tinkercell::PythonInterpreterThread

Public Slots

- virtual void **initialize** ()
- virtual void **exec** (const QString &)
- virtual void **finalize** ()
- virtual void **toolLoaded** (Tool ∗)

Public Member Functions

- InterpreterThread (const QString &, MainWindow ∗main)

  load an embedded interpreter (e.g. python)

- virtual ∼InterpreterThread ()

  unloads the library

- virtual void setCPointers ()

  requests main window to load all the C pointers for the C API inside the embedded library

Protected Member Functions

- virtual void run ()

  the main function that runs one of the specified functions

Protected Attributes

- QString **code**
- QQueue< QString > **codeQueue**

6.41.1 Detailed Description

This class is used to run interpreters such as python, perl, octave, R, etc. This is the parent class that provides the basic structure for loading the library that will embed one of these languages.

See also

PythonInterpreterThread
OctaveInterpreterThread

6.41.2 Constructor & Destructor Documentation

6.41.2.1 Tinkercell::InterpreterThread::InterpreterThread (const QString & *dllname*, MainWindow ∗ *main*)

load an embedded interpreter (e.g. python)

Parameters

QString	name of the embed library
MainWindow	∗ TinkerCell main window

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

- InterpreterThread.h
- InterpreterThread.cpp

6.42 Tinkercell::ItemData Class Reference

This class is used to store information about nodes or connections. It contains a hashtable of data tables, which is used by different tools to store specific data. The versions queue can be used to keep previous versions of the data.

```
#include <ItemHandle.h>
```

Friends

- class **ItemHandle**

6.42.1 Detailed Description

This class is used to store information about nodes or connections. It contains a hashtable of data tables, which is used by different tools to store specific data. The versions queue can be used to keep previous versions of the data.

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

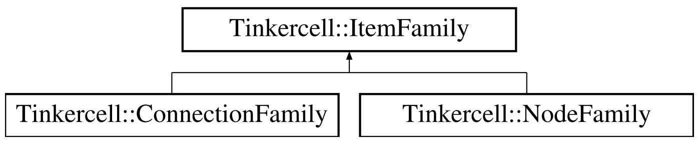
- ItemHandle.h
- ItemHandle.cpp

6.43 Tinkercell::ItemFamily Class Reference

This class defines the family of a node or connection. The class contains the icon for the family, family name, and minimal data that defines the family. Each family has a name, which is internally converted to an integer (ID) The ID is used to perform isA checks, thus avoiding repeated string matches.

```
#include <ItemFamily.h>
```

Inheritance diagram for Tinkercell::ItemFamily:



Public Member Functions

- virtual QString name () const

  name of this family

- virtual void setName (const QString &)

  set name of this family

- virtual bool isA (const QString &) const

  indicates whether or not the given string is the name of this family or any of its parent families

- virtual bool isA (const ItemFamily ∗) const

  indicates whether or not the given family is the name of this family or any of its parent families

- virtual ItemFamily ∗ root () const

  get the top-most family

- virtual bool isRelatedTo (const ItemFamily ∗) const

  checks if the given family shares its root family with this family

- virtual ItemFamily ∗ parent () const

  get the parent for this family. If there are more than one parents, returns the first

- virtual QList< ItemFamily ∗ > parents () const

  get all the parents for this family.

- virtual QList< ItemFamily ∗ > children () const

  get all the families that inherit directly from this family

- virtual QList< ItemFamily ∗ > allChildren () const

  get all the families that inherit from this family. the list will be ordered in a breadth-first ordering

- ItemFamily (const QString &name=QString())

  constructor.

- virtual ~ItemFamily ()

  destructor.

Public Attributes

- QString description

  description of this family

- Unit measurementUnit

  the measurement name and unit for items in this family

- QHash< QString, qreal > numericalAttributes

  the list of numerical attributes that are common to all members of this family

- QHash< QString, QString > textAttributes

  the list of string attributes that are common to all members of this family

- QList< QGraphicsItem ∗ > graphicsItems

  the default set of graphics items used to represent items of this family

- QPixmap pixmap

  the icon representing this family

Protected Member Functions

- virtual bool isA (int ID) const

  indicates whether or not the given family ID is the name of this family or any of its parent families

Protected Attributes

- int type

  used for casting between different sub-classes

- QString _name

  name of this family

- int ID

  the ID for this family. It is used for quick equality checks (instead of using strings)

Static Protected Attributes

- static QStringList ALLNAMES

  all family names. This list's lenth is used to assign the next ID

- static QHash< QString, int > NAMETOID

  the hash stores names for each ID

Friends

- class **NodeFamily**
- class **ConnectionFamily**

6.43.1 Detailed Description

This class defines the family of a node or connection. The class contains the icon for the family, family name, and minimal data that defines the family. Each family has a name, which is internally converted to an integer (ID) The ID is used to perform isA checks, thus avoiding repeated string matches.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 Tinkercell::ItemFamily::ItemFamily (const QString & *name* = `QString()`)

constructor.

Parameters

QString name

6.43.3 Member Function Documentation

6.43.3.1 QList< ItemFamily ∗ > Tinkercell::ItemFamily::allChildren () const `[virtual]`

get all the families that inherit from this family. the list will be ordered in a breadth-first ordering

Returns

QList<ItemFamily∗>

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

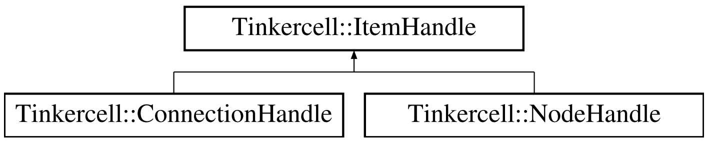
- ItemFamily.h
- ItemFamily.cpp

6.44 Tinkercell::ItemHandle Class Reference

The ItemHandle represents a complete object in the network, whether it is a node or a connection. The ItemHandle contains the name of the object and pointers to all the QGraphicsItems that are used to represent the object. Tools associated with the object can be stored within the ItemHandle as well. The ItemHandle can also optionally contain an ItemFamily, which can be used to distinguish different types of nodes or connections, if needed. Each ItemHandle can contain one parent. Several functions are available for convinently getting the parents and children of an ItemHandle.

`#include <ItemHandle.h>`

Inheritance diagram for Tinkercell::ItemHandle:



Public Member Functions

- ItemHandle (const QString &name=QString())

  default constructor

- ItemHandle (const ItemHandle &)

  copy constructor

- virtual ItemHandle & operator= (const ItemHandle &)

  operator =

- virtual ∼ItemHandle ()

  destructor -- does nothing

- virtual ItemHandle ∗ clone () const

  clone the data and lists

- virtual ItemFamily ∗ family () const

  family that this items belongs in. Used for characterizing the nodes and connections.

- virtual void setFamily (ItemFamily ∗, bool useCommand=true)

  set the family that this items belongs in.

- virtual bool isA (const ItemFamily ∗family) const

  determines whether this handle belongs to the speicific family.

- virtual bool isA (const QString &family) const

  determines whether this handle belongs to the speicific family.

- virtual QString fullName (const QString &sep=QString(".")) const

  The full name includes all the parent names appended using a dot.

- virtual void setParent (ItemHandle ∗parent, bool useCommand=true)

  Set the parent for this handle.

- virtual void rename (const QString &)

  set name of this handle and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &hashstring, const NumericalDataTable ∗newdata)

  change numerical data table and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &hashstring, const TextDataTable ∗newdata)

  change text data table and also adds undo command to history window and emits associated signal(s)

- virtual ItemHandle ∗ root (const QString &family=QString("")) const

  get the top-level handle such that it is of the specified family. If no family is specified, then gets the top-level handle

- virtual ItemHandle ∗ parentOfFamily (const QString &family) const

  get the bottom-most parent handle such that it is of the specified family. If no family is specified, then gets the top-level handle

- virtual bool isChildOf (ItemHandle ∗handle) const

  checks if an item is the parent or parent's parent, or parent's parent's parent, etc. Note: self->isChildOf(self) is false

- virtual int depth () const

  counts the number of parents that have to be traversed in order to reach the root handle. If this handle has no parents, the values returned is 0. If its parent has no parent, then the value is 1, and so on.

- virtual QList< QGraphicsItem ∗ > allGraphicsItems () const

  gets the graphics items belonging to this handle and all child handes

- virtual QList< ItemHandle ∗ > allChildren () const

  gets the all child handles and their child handles

- QStringList numericalDataNames () const

  all the numerical data table names

- QStringList textDataNames () const

  all the numerical text table names

- bool hasNumericalData (const QString &name) const

  does this handle have a numerical data table with this name?

- bool hasTextData (const QString &name) const

  does this handle have a text data table with this name?

- qreal numericalData (const QString &name, int row=0, int column=0) const

  gets a numerical attribute with the given name, row, column

- qreal numericalData (const QString &name, const QString &row, const QString &column=QString()) const *gets a numerical attribute with the given name, row, column*

- QString textData (const QString &name, int row=0, int column=0) const

  gets a text attribute with the given name, row, column

- QString textData (const QString &name, const QString &row, const QString &column=QString()) const

  gets a text attribute with the given name, row, column

- qreal & numericalData (const QString &name, int row=0, int column=0)

  gets a reference to the numerical attribute with the given name, row, column

- qreal & numericalData (const QString &name, const QString &row, const QString &column=QString())

  gets a reference to the numerical attribute with the given name, row, column

- QString & textData (const QString &name, int row=0, int column=0)

  gets a reference to the text attribute with the given name, row, column

- QString & textData (const QString &name, const QString &row, const QString &column=QString())

  gets a reference to the text attribute with the given name, row, column

- NumericalDataTable & numericalDataTable (const QString &name)

  gets reference to a numerical table with the given name. Makes the table if needed

- TextDataTable & textDataTable (const QString &name)

  gets reference to a text table with the given name. Makes the table if needed

Public Attributes

- QString name

  name of this item

- QList< QGraphicsItem ∗ > graphicsItems

  list of graphical items used to draw this handle

- QList< Tool ∗ > tools

  list of tools associated with this handle

- NetworkHandle ∗ network

  the network that this item belongs in

- ItemHandle ∗ parent

  this handles immediate parent (main parent if there are more than one)

- QList< ItemHandle ∗ > children

  child handles that have this handle as a parent

- int type

  type of this handle (sub-classes can specify type)

6.44.1 Detailed Description

The ItemHandle represents a complete object in the network, whether it is a node or a connection. The ItemHandle contains the name of the object and pointers to all the QGraphicsItems that are used to represent the object. Tools associated with the object can be stored within the ItemHandle as well. The ItemHandle can also optionally contain an ItemFamily, which can be used to distinguish different types of nodes or connections, if needed. Each ItemHandle can contain one parent. Several functions are available for convinently getting the parents and children of an ItemHandle. Use setHandle and getHandle functions to get and set the handles for QGraphicsItems. Use h->data->numericalData[string] or h->data->textData[string] to get the DataTable with the particular name. Alternatively, h->numericalData(string) or h->textData(string) can be used to access the data conviniently.

The SymbolsTable is used to store all the handles in a network.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 Tinkercell::ItemHandle::ItemHandle (const QString & *name* = `QString()`)

default constructor

Parameters

QString name

6.44.3 Member Function Documentation

6.44.3.1 QList< ItemHandle ∗ > Tinkercell::ItemHandle::allChildren () const `[virtual]`

gets the all child handles and their child handles

Returns

QList<ItemHandle∗> list of handles

6.44.3.2 QList< QGraphicsItem ∗ > Tinkercell::ItemHandle::allGraphicsItems () const `[virtual]`

gets the graphics items belonging to this handle and all child handes

Returns

QList<QGraphicsItem∗> list of graphics items

6.44.3.3 int Tinkercell::ItemHandle::depth () const `[virtual]`

counts the number of parents that have to be traversed in order to reach the root handle. If this handle has no parents, the values returned is 0. If its parent has no parent, then the value is 1, and so on.

Returns

int

6.44.3.4 QString Tinkercell::ItemHandle::fullName (const QString & *sep* = `QString(".")`) const `[virtual]`

The full name includes all the parent names appended using a dot.

Parameters

QString replace the dot with some other separator

6.44.3.5 bool Tinkercell::ItemHandle::hasNumericalData (const QString & *name*) const

does this handle have a numerical data table with this name?

Parameters

QString name of tool, e.g. "Numerical Attributes"

Returns

bool true = has a numerical table by this name. false = does not have a numerical table by this name

6.44.3.6 bool Tinkercell::ItemHandle::hasTextData (const QString & *name*) const

does this handle have a text data table with this name?

Parameters

QString name of tool, e.g. "Text Attributes"

Returns

bool true = has a text table by this name. false = does not have a text table by this name

6.44.3.7 bool Tinkercell::ItemHandle::isA (const QString & *family*) const `[virtual]`

determines whether this handle belongs to the speicific family.

Parameters

QString the family name

6.44.3.8 bool Tinkercell::ItemHandle::isA (const ItemFamily ∗ *family*) const `[virtual]`

determines whether this handle belongs to the speicific family.

Parameters

QString the family

6.44.3.9 bool Tinkercell::ItemHandle::isChildOf (ItemHandle ∗ *handle*) const `[virtual]`

checks if an item is the parent or parent's parent, or parent's parent's parent, etc. Note: self->isChildOf(self) is false

Parameters

ItemHandle∗ parent handle

Returns

Boolean is child

6.44.3.10 qreal & Tinkercell::ItemHandle::numericalData (const QString & *name*, const QString & *row*, const QString & *column* = `QString()`)

gets a reference to the numerical attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Numerical Attributes"

QString row name in data table

QString column name data table

Returns

double reference value

6.44.3.11 qreal & Tinkercell::ItemHandle::numericalData (const QString & *name*, int *row* = `0`, int *column* = `0`)

gets a reference to the numerical attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Numerical Attributes"

int row in data table

int column in data table

Returns

double reference value

6.44.3.12 qreal Tinkercell::ItemHandle::numericalData (const QString & *name*, const QString & *row*, const QString & *column* = `QString()`) const

gets a numerical attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Numerical Attributes"

QString row name in data table

QString column name data table

Returns

double value

6.44.3.13 qreal Tinkercell::ItemHandle::numericalData (const QString & *name*, int *row* = `0`, int *column* = `0`) const

gets a numerical attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Numerical Attributes"

int row in data table

int column in data table

Returns

double value

6.44.3.14 QStringList Tinkercell::ItemHandle::numericalDataNames () const

all the numerical data table names

Returns

QStringList

6.44.3.15 DataTable< qreal > & Tinkercell::ItemHandle::numericalDataTable (const QString & *name*)

gets reference to a numerical table with the given name. Makes the table if needed

Parameters

QString name of tool, e.g. "Numerical Attributes"

Returns

DataTable<double>& reference of table

6.44.3.16 ItemHandle ∗ Tinkercell::ItemHandle::parentOfFamily (const QString & *family*) const `[virtual]`

get the bottom-most parent handle such that it is of the specified family. If no family is specified, then gets the top-level handle

Parameters

ItemHandle∗ the family name

6.44.3.17 ItemHandle ∗ Tinkercell::ItemHandle::root (const QString & *family* = `QString("")`) const `[virtual]`

get the top-level handle such that it is of the specified family. If no family is specified, then gets the top-level handle

Parameters

ItemHandle∗ the family name

6.44.3.18 void Tinkercell::ItemHandle::setParent (ItemHandle ∗ *parent*, bool *useCommand* = `true`) `[virtual]`

Set the parent for this handle.

Parameters

ItemHandle ∗ parent

bool (optional) whether to call network's set parent command, which will update the history stack

ItemHandle∗ parent handle

6.44.3.19 QString & Tinkercell::ItemHandle::textData (const QString & *name*, const QString & *row*, const QString & *column* = `QString()`)

gets a reference to the text attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Text Attributes"

QString row name in data table

QString column name data table

Returns

QString& reference value

6.44.3.20 QString & Tinkercell::ItemHandle::textData (const QString & *name*, int *row* = `0`, int *column* = `0`)

gets a reference to the text attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Text Attributes"
int row in data table
int column in data table

Returns

QString reference value

6.44.3.21 QString Tinkercell::ItemHandle::textData (const QString & *name*, const QString & *row*, const QString & *column* = `QString()`) const

gets a text attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Text Attributes"
QString row name in data table
QString column name data table

Returns

QString value

6.44.3.22 QString Tinkercell::ItemHandle::textData (const QString & *name*, int *row* = `0`, int *column* = `0`) const

gets a text attribute with the given name, row, column

Parameters

QString name of tool, e.g. "Text Attributes"
int row in data table
int column in data table

Returns

QString value

6.44.3.23 QStringList Tinkercell::ItemHandle::textDataNames () const

all the numerical text table names

Returns

QStringList

6.44.3.24 DataTable< QString > & Tinkercell::ItemHandle::textDataTable (const QString & *name*)

gets reference to a text table with the given name. Makes the table if needed

Parameters

QString name of tool, e.g. "Numerical Attributes"

Returns

TextDataTable& reference of table

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

- ItemHandle.h
- ItemHandle.cpp

6.45 Tinkercell::LineNumberArea Class Reference

Public Member Functions

- **LineNumberArea** (CodeEditor ∗editor)
- QSize **sizeHint** () const

Protected Member Functions

- void **paintEvent** (QPaintEvent ∗event)

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

- CodeEditor.h

6.46 Tinkercell::MainWindow Class Reference

MainWindow is the parent container for all the other widgets in TinkerCell The central widget in MainWindow is a tab widget. Each tab widget can hold a GraphicsView or a TextEditor. One of the main roles of MainWindow is to serve as a signal/slot hub for Tools.

```
#include <MainWindow.h>
```

Public Types

- enum TOOL_WINDOW_OPTION { **DockWidget**, **ToolBoxWidget**, **NewToolBoxWidget** }

  this enum is used to determine how to place a widget when used in addToolWindow. DockWidget = tool window is placed into a dockable widget ToolBoxWidget = tool window is placed in an existing toolbox, if one exists NewToolBoxWidget = tool window is placed inside a new toolbox

- enum VIEW_MODE { **TabView**, **WindowView** }

  the types of views for multiple documents TabView = tabbed documents WindowView = each documents in a separate subwindow

Public Member Functions

- MainWindow (bool enableScene=true, bool enableText=true, bool enableConsoleWindow=true, bool showHistory=true, bool views=true)

  5-arg (optional) constructor allows disabling of text/graphics modes

- virtual void allowMultipleViewModes (bool)

  allow or disallow changing between different views

- virtual ∼MainWindow ()

  Destructor: delete all the graphics scenes.

- QDockWidget ∗ addToolWindow (QWidget ∗tool, TOOL_WINDOW_OPTION option=DockWidget, Qt::DockWidgetArea initArea=Qt::RightDockWidgetArea, Qt::DockWidgetAreas allowedAreas=Qt::AllDockWidgetAreas, bool inMenu=true)

  Add a new docking window to the main window. The name and icon are obtained using the widget's windowTitle and windowIcon, so be sure to set those before calling this function.

- void addToViewMenu (QWidget ∗tool)

  place a show/hide action in the view menu for the given widget

- void setCursor (QCursor cursor)

  set the cursor for all windows

- void addTool (Tool ∗tool)

  add a new tool to the list of tools stored in the main window

- void initializeMenus (bool enableScene=true, bool enableText=true)

  Initialize the basic menu (save, open, close, exit, etc.).

- void setupNewThread (QSemaphore ∗, QLibrary ∗)

  This function is usually called from a new thread. This function allows all the plugins to add their functionalities to the C function pointer of the new thread.

- void loadDynamicLibrary (const QString &)

  Load a new plugin (dll).

- QPair< QList< ItemHandle ∗ >, QList< QGraphicsItem ∗ > > getItemsFromFile (const QString &filename, ItemHandle ∗root=0)

  get the items inside a file. Some tool must implement this function and connect to the getItemsFromFile signal. The Core library does not implement a read file function.

- GraphicsScene ∗ currentScene () const

  gets the current scene that is active

- TextEditor ∗ currentTextEditor () const

  gets the text editor that is active

- NetworkWindow ∗ currentWindow () const

  gets the current window that is active (each window contains either a scene or editor)

- NetworkHandle ∗ currentNetwork () const

  gets the current window that is active

- QList< NetworkHandle ∗ > networks () const

  gets all the windows in the main window

- QUndoStack ∗ historyStack () const

  the history stack of the current network.

- QUndoView ∗ historyWidget ()

  the history stack widget of the current window.

- virtual Tool ∗ tool (const QString &) const

  get a tool

- virtual QList< Tool ∗ > tools () const

  get all tools

Static Public Member Functions

- static void RegisterDataTypes ()

  register all the TinkerCell data structures with Qt

- static QString homeDir ()

  The TinkerCell user directory, which is User's Documents Folder/TinkerCell by default, but users may change this setting.

- static QString tempDir ()

  The TinkerCell user temporary directory, which is <SYSTEM temp="" folder>="">/TinkerCell.

Public Attributes

- QList< QWidget ∗ > toolWindows

  the set of all windows inseted in the main window using addToolWindow

- QMenu contextItemsMenu

  the context menu that is shown during right-click event on selected graphical items. Plugins can add new actions to this menu.

- QMenu contextScreenMenu

  the context menu that is shown during right-click event on the scene. Plugins can add new actions to this menu.

- QMenu contextSelectionMenu

  the context menu that is shown during right-click event on a text editor with text selected. Plugins can add new actions to this menu.

- QMenu contextEditorMenu

  the context menu that is shown during right-click event on a text editor with no text selected. Plugins can add new actions to this menu.

- QMenu ∗ fileMenu

  The file menu. Plugins can add new actions to this menu.

- QMenu ∗ editMenu

  The edit menu. Plugins can add new actions to this menu.

- QMenu ∗ viewMenu

  The view menu. New docking windows are automatically added here.

- QMenu ∗ helpMenu

  The help menu.

- QMenu ∗ optionsMenu

  the menu for settings such as default plugins, Tinkercell home directory, etc.

- QMenu ∗ parsersMenu

  the menu for choosing one of the available parsers (will be 0 if there are no parsers)

- QToolBar ∗ toolBarBasic

  The tool bar that contains new, open, close, etc. actions.

- QToolBar ∗ toolBarEdits

  The tool bar that contains copy, paste, undo, etc.

- QToolBar ∗ toolBarForTools

  One of the initial tool bars which designated for tools that do not want to create a new toolbar.

Static Public Attributes

- static TOOL_WINDOW_OPTION defaultToolWindowOption = MainWindow::ToolBoxWidget

  the default option to use for tools (optional)

- static TOOL_WINDOW_OPTION defaultHistoryWindowOption = MainWindow::ToolBoxWidget

  the default option to use for history window

- static TOOL_WINDOW_OPTION defaultConsoleWindowOption = MainWindow::DockWidget

  the default option to use for console window

- static QString PROJECTWEBSITE = QObject::tr("www.tinkercell.com")

  the project website

- static QString ORGANIZATIONNAME = QObject::tr("TinkerCell")

  the project organization name

- static QString PROJECTNAME = QObject::tr("TinkerCell")

  the project name

- static QString CPP_ENTRY_FUNCTION = QObject::tr("loadTCTool")

  the default function that is loaded in C++ plugins

- static QString C_ENTRY_FUNCTION = QObject::tr("tc_main")

  the default function that is loaded in C plugins

- static QString PROJECT_VERSION = QObject::tr("0.0.0")

  the default project version

- static QStringList OPEN_FILE_EXTENSIONS

  the default file extensions that can be opened

- static QStringList SAVE_FILE_EXTENSIONS

  the default file extensions that can be saved

Friends

- class **NetworkWindow**
- class **NetworkHandle**
- class **GraphicsScene**
- class **TextEditor**
- class **GraphicsView**

signals

- static QString previousFileName

  stores the last opened directory

- static QHash< void ∗, bool > invalidPointers

  stores list of all pointers that have been deleted (to prevent double-deletions)

- bool allowViewModeToChange

  allowed views

- QHash< QString, QLibrary ∗ > dynamicallyLoadedLibraries

  the loaded dynamic libraries indexed by file name

- ConsoleWindow ∗ consoleWindow

  the general window for command, errors, and messages

- QTabWidget ∗ tabWidget

  the central multi-document interface widget

- QList< NetworkHandle ∗ > allNetworks

  the list of all network windows

- QToolBox ∗ toolBox

  the optional tool box that will only appear if one of the plug-ins uses the toolbox argument in the addToolWindow call

- HistoryWindow historyWindow

  history view, not the stack itself. The stack is stored within each NetworkHandle

- NetworkWindow ∗ currentNetworkWindow

  keep pointer to last selected window. Used by windowChanged signal

- QHash< QString, Tool ∗ > toolsHash

  all the tools (plug-ins) are stored here, indexed by their names

- bool isValidHandlePointer (void ∗p)

  checks if the given address belongs to a handle

- void toolAboutToBeLoaded (Tool ∗tool, bool ∗shouldLoad)

  a new tool is about to be added. This signal can be used to prevent the tool from being added

- void historyChanged (int i=0)

  one of more changed have occurred in the history window of the current scene

- void funtionPointersToMainThread (QSemaphore ∗, QLibrary ∗)

  used internally by MainWindow in order to move from a thread to the main thread

- void toolLoaded (Tool ∗tool)

  signals when a new tool (plugin) is loaded

- void setupFunctionPointers (QLibrary ∗)

  signals when a new FuntionToSignal is constructed

- void networkClosing (NetworkHandle ∗, bool ∗)

signals when a network is going to close

- void networkClosed (NetworkHandle ∗)

  signals after a window is closed

- void prepareNetworkForSaving (NetworkHandle ∗, bool ∗)

  signals when a tool is about to save a network

- void networkSaved (NetworkHandle ∗)

  signals when a tool has saved the network in a file

- void saveNetwork (const QString &filename)

  signals when user selects a file to save the current network to

- void loadNetwork (const QString &filename)

  signals when user selects a file to open in the current network

- void getItemsFromFile (QList< ItemHandle ∗ > &, QList< QGraphicsItem ∗ > &, const QString &filename, ItemHandle ∗root)

  signal sent to a tool so that the tool can get the items inside a file

- void networkLoaded (NetworkHandle ∗)

  signals informs that the current network has just loaded a new Network

- void networkOpened (NetworkHandle ∗)

  signals whenever the new network is opened

- void windowChanged (NetworkWindow ∗, NetworkWindow ∗)

  signals whenever the current window changes

- void itemsSelected (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, QPointF point, Qt::KeyboardModifiers modifiers)

  signals whenever a new item is selected (item can be sub-item, not top-level)

- void mousePressed (GraphicsScene ∗scene, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

  signals whenever an empty node of the screen is clicked

- void mouseReleased (GraphicsScene ∗scene, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

  signals whenever an empty node of the screen is clicked

- void mouseDoubleClicked (GraphicsScene ∗scene, QPointF point, QGraphicsItem ∗, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

  emits event when mouse is double clicked

- void mouseDragged (GraphicsScene ∗scene, QPointF from, QPointF to, Qt::MouseButton, Qt::KeyboardModifiers modifiers)

  signals whenever mouse is dragged from one point to another

- void itemsAboutToBeMoved (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &item, QList< QPointF > &distance, QList< QUndoCommand ∗ > &)

  signals whenever items are going to be moved (each item is the top-most item)

- void itemsMoved (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &item, const QList< QPointF > &distance)

  signals whenever items are being moved (each item is the top-most item)

- void itemsAboutToBeRemoved (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &item, QList< ItemHandle ∗ > &handles, QList< QUndoCommand ∗ > &)

  signals just before items are deleted

- void itemsRemoved (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &item, const QList< ItemHandle ∗ > &handles)

  signals whenever items are deleted

- void itemsAboutToBeInserted (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &, QList< QUndoCommand ∗ > &)

  signals whenever items are going to be added

- void itemsInserted (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &item, const QList< ItemHandle ∗ > &handles)

  signals whenever items are added

- void itemsInserted (NetworkHandle ∗win, const QList< ItemHandle ∗ > &)

  A convenient signal that is emitted when items are inserted from a GraphicsScene or TextEditor. Warning: listening to the other itemsInserted signals may cause redundancy.

- void itemsRemoved (NetworkHandle ∗win, const QList< ItemHandle ∗ > &)

  A convenient signal that is emitted when items are removed from a GraphicsScene or TextEditor. Warning: listening to the other itemsRemoved signals may cause redundancy.

- void copyItems (GraphicsScene ∗scene, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &)

  signals just before items are copied

- void textChanged (TextEditor ∗, const QString &, const QString &, const QString &)

  some text inside this editor has been changed

- void lineChanged (TextEditor ∗, int, const QString &)

  the cursor has moved to a different line

- void parse (TextEditor ∗)

  request to parse the text in the current text editor

- void mouseMoved (GraphicsScene ∗scene, QGraphicsItem ∗item, QPointF point, Qt::MouseButton, Qt::KeyboardModifiers modifiers, QList< QGraphicsItem ∗ > &)

  signals whenever mouse moves, and indicates whether it is on top of an item

- void mouseOnTopOf (GraphicsScene ∗scene, QGraphicsItem ∗item, QPointF point, Qt::KeyboardModifiers modifiers, QList< QGraphicsItem ∗ > &)

signals whenever mouse is on top of an item

- void sceneRightClick (GraphicsScene ∗scene, QGraphicsItem ∗item, QPointF point, Qt::KeyboardModifiers modifiers)

  signals whenever right click is made on an item or sceen

- void keyPressed (GraphicsScene ∗scene, QKeyEvent ∗)

  signals whenever a key is pressed

- void keyReleased (GraphicsScene ∗scene, QKeyEvent ∗)

  signals whenever a key is released

- void colorChanged (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items)

  signals whenever color of items are changed

- void parentItemChanged (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, const QList< QGraphicsItem ∗ > &parents)

  signals whenever item parents are changed

- void itemsRenamed (NetworkHandle ∗window, const QList< ItemHandle ∗ > &items, const QList< QString > &oldnames, const QList< QString > &newnames)

  signals whenever an item is renamed

- void handlesChanged (NetworkHandle ∗scene, const QList< QGraphicsItem ∗ > &items, const QList< ItemHandle ∗ > &old)

  signals whenever the handles for graphics items have changed

- void parentHandleChanged (NetworkHandle ∗scene, const QList< ItemHandle ∗ > &, const QList< ItemHandle ∗ > &)

  signals whenever item parent handle is changed

- void handleFamilyChanged (NetworkHandle ∗network, const QList< ItemHandle ∗ > &, const QList< ItemFamily ∗ > &)

  signals whenever item handles' families are changed

- void dataChanged (const QList< ItemHandle ∗ > &items)

  signals whenever some data is changed

- void escapeSignal (const QWidget ∗sender)

  signals whenever the current activities need to be stopped

- void filesLoaded (const QList< QFileInfo > &files)

  signals whenever file(s) are loaded. Each file can be a model or a plugin

- void itemsDropped (GraphicsScene ∗, const QString &, const QPointF &)

  signal is emitted when some object OTHER than files are dropped on the canvas

- void saveSettings ()

  save initial settings to settingsFileName

- void closeEvent (QCloseEvent ∗event)

  close window event -- asks whether to save file

- virtual void dropEvent (QDropEvent ∗)

  drag and drop

- virtual void dragEnterEvent (QDragEnterEvent ∗event)

  drag and drop

slots

- void setUserHome ()

  asks user for a new directory to be used as the user home directory (must be writtable)

- GraphicsScene ∗ newScene ()

  create new scene

- TextEditor ∗ newTextEditor ()

  create new text editor

- void closeWindow ()

  triggered when the close button is clicked. Closes the current window

- void saveWindow ()

  triggered when the save button is clicked. Opens a file dialog and emits the save signal. The main window itself does not implement the save.

- void saveWindowAs ()

  triggered when the save-as button is clicked. Opens a file dialog and emits the save signal. The main window itself does not implement the save.

- void open ()

  triggered when the open button is clicked. Opens a file dialog. Note: the core library just emits a signal, and other tools are responsible for actually opening a file

- void open (const QString &)

  open a file. Note: the core library just emits a signal, and other tools are responsible for actually opening a file The main window does not implement an function for opening a new file

- void undo ()

  calls current scene or text editor's undo

- void redo ()

  calls current scene or text editor's redo

- void copy ()

  calls current scene or text editor's copy

- void cut ()

calls current scene or text editor's cut

- void paste ()

  calls current scene or text editor's paste

- void selectAll ()

  calls current scene or text editor's selectAll

- void remove ()

  calls current scene or text editor's find

- void print ()

  triggered when the print button is clicked. Calls current scene's print

- void printToFile ()

  triggered when the print-to-file button is clicked. Calls current scene's print on a pdf file

- void sendEscapeSignal (const QWidget ∗w=0)

  sends a signal to all plugins telling them to exit their current processes.

- void addParser (TextParser ∗)

  add a new text parser to the list of available parsers. The current text parser can be obtained using TextParser::currentParser();

- void gridOn ()

  change grid mode for current scene to on (>0)

- void gridOff ()

  change grid mode for current scene to off (=0)

- void setGridSize ()

  set grid size for current scene

- void popOut ()

  pop-out the current window

- ConsoleWindow ∗ console () const

  get the console window

- void readSettings ()

  read initial settings from settingsFileName

- static MainWindow ∗ instance ()

  gets the global main window

- void popOut (NetworkWindow ∗)

  pop-out the given window

- void popIn (NetworkWindow ∗)

  pop-in the given window

- void setCurrentWindow (NetworkWindow ∗)

  sets the active window

- void loadFiles (const QList< QFileInfo > &files)

  loads files (library files or Network files)

- void changeConsoleBgColor ()

  change console background color

- void changeConsoleTextColor ()

  change console text color

- void changeConsoleMsgColor ()

  change console message text color

- void changeConsoleErrorMsgColor ()

  change console error text color

- virtual void tabIndexChanged (int)

  tab changed

- void itemsRemovedSlot (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &item, const QList< ItemHandle ∗ > &handles)

  signals whenever items are deleted

- void itemsInsertedSlot (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &item, const QList< ItemHandle ∗ > &handles)

  signals whenever items are added

- void setupFunctionPointersSlot (QSemaphore ∗, QLibrary ∗)

  send signal to other tools so that they can connect functions to signals

6.46.1 Detailed Description

MainWindow is the parent container for all the other widgets in TinkerCell The central widget in MainWindow is a tab widget. Each tab widget can hold a GraphicsView or a TextEditor. One of the main roles of MainWindow is to serve as a signal/slot hub for Tools.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 Tinkercell::MainWindow::MainWindow (bool *enableScene* = `true`, bool *enableText* = `true`, bool *enableConsoleWindow* = `true`, bool *showHistory* = `true`, bool *views* = `true`)

5-arg (optional) constructor allows disabling of text/graphics modes

Parameters

bool enable text-based network construction (default = true)

bool enable graphics-based network construction (default = true)

bool enable command-line (default = true)

bool enable history window (default = true)

bool allow tabbed and windowed view modes (default = true)

6.46.2.2 Tinkercell::MainWindow::∼MainWindow () `[virtual]`

Destructor: delete all the graphics scenes.

destructor

6.46.3 Member Function Documentation

6.46.3.1 void Tinkercell::MainWindow::addTool (Tool ∗ *tool*)

add a new tool to the list of tools stored in the main window

Parameters

the name of the new tool

the new tool

Returns

void

6.46.3.2 QDockWidget ∗ Tinkercell::MainWindow::addToolWindow (QWidget ∗ *tool*, TOOL_WINDOW_OPTION *option* = `DockWidget`, Qt::DockWidgetArea *initArea* = `Qt::RightDockWidgetArea`, Qt::DockWidgetAreas *allowedAreas* = `Qt::AllDockWidgetAreas`, bool *inMenu* = `true`)

Add a new docking window to the main window. The name and icon are obtained using the widget's windowTitle and windowIcon, so be sure to set those before calling this function.

Parameters

Tool∗ the new tool

Qt::DockWidgetArea the initial docking area

Qt::DockWidgetAreas the allowed docking areas

bool whether or not to place the docking window in the view menu

bool use a QToolBox instead of a dock widget. The widget will not be dockable, but the entire toolbox will be dockable.

Returns

QDockWidget∗ the new docking widget. ToolBoxWidget option is used, the docking widget may be an existing docking widget.

6.46.3.3 void Tinkercell::MainWindow::addToViewMenu (QWidget ∗ *tool*)

place a show/hide action in the view menu for the given widget

Parameters

QWidget∗ the new widget

6.46.3.4 void Tinkercell::MainWindow::allowMultipleViewModes (bool *b*) `[virtual]`

allow or disallow changing between different views

Parameters

bool

6.46.3.5 void Tinkercell::MainWindow::changeConsoleBgColor () `[protected, slot]`

change console background color

Returns

void

6.46.3.6 void Tinkercell::MainWindow::changeConsoleErrorMsgColor () `[protected, slot]`

change console error text color

Returns

void

6.46.3.7 void Tinkercell::MainWindow::changeConsoleMsgColor () `[protected, slot]`

change console message text color

Returns

void

6.46.3.8 void Tinkercell::MainWindow::changeConsoleTextColor () `[protected, slot]`

change console text color

Returns

void

6.46.3.9 void Tinkercell::MainWindow::closeEvent (QCloseEvent ∗ *event*) `[protected]`

close window event -- asks whether to save file

Parameters

QCloseEvent ∗ event

Returns

void

6.46.3.10 void Tinkercell::MainWindow::colorChanged (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*) `[signal]`

signals whenever color of items are changed

Parameters

GraphicsScene ∗ scene where the event took place

QList<QGraphicsItem∗>& items that changed color

Returns

void

6.46.3.11 void Tinkercell::MainWindow::copyItems (GraphicsScene ∗ *scene*, QList< QGraphicsItem ∗ > &, QList< ItemHandle ∗ > &) `[signal]`

signals just before items are copied

Parameters

GraphicsScene ∗ scene where the items are going to be copied

QList<QGraphicsItem∗>& list of graphics items going to be copied

QList<ItemHandle∗>& list of handles going to be copied (does NOT have to be the same number as items removed)

Returns

void

6.46.3.12 NetworkHandle ∗ Tinkercell::MainWindow::currentNetwork () const

gets the current window that is active

Returns

NetworkHandle∗ current network

6.46.3.13 GraphicsScene ∗ Tinkercell::MainWindow::currentScene () const

gets the current scene that is active

Returns

GraphicsScene∗ current scene

6.46.3.14 TextEditor ∗ Tinkercell::MainWindow::currentTextEditor () const

gets the text editor that is active

Returns

TextEditor∗ current editor

6.46.3.15 NetworkWindow ∗ Tinkercell::MainWindow::currentWindow () const

gets the current window that is active (each window contains either a scene or editor)

Returns

NetworkWindow∗ current network window

6.46.3.16 void Tinkercell::MainWindow::dataChanged (const QList< ItemHandle ∗ > & *items*) `[signal]`

signals whenever some data is changed

Parameters

QList<ItemHandle∗>& items handles

Returns

void

6.46.3.17 void Tinkercell::MainWindow::escapeSignal (const QWidget ∗ *sender*) `[signal]`

signals whenever the current activities need to be stopped

Parameters

QWidget ∗ the widget that send the signal

Returns

void

6.46.3.18 void Tinkercell::MainWindow::filesLoaded (const QList< QFileInfo > & *files*) `[signal]`

signals whenever file(s) are loaded. Each file can be a model or a plugin

Parameters

QList<QFileInfo>& the name(s) of the file(s)

Returns

void

6.46.3.19 void Tinkercell::MainWindow::funtionPointersToMainThread (QSemaphore ∗, QLibrary ∗) `[signal]`

used internally by MainWindow in order to move from a thread to the main thread

Parameters

QSemaphore∗ Sempahore that lets the thread run once C API is initialized

QLibrary ∗ the new FuntionToSignal instance

Returns

void

6.46.3.20 void Tinkercell::MainWindow::getItemsFromFile (QList< ItemHandle ∗ > &, QList< QGraphicsItem ∗ > &, const QString & *filename*, ItemHandle ∗ *root*) `[signal]`

signal sent to a tool so that the tool can get the items inside a file

Parameters

QList<ItemHandle∗>& list of items inside the file

QList<QGraphicsItem∗>& list of graphics items in the file

QString& file that is selected by user

ItemHandle ∗ optional root parent handle for all the loaded items

Returns

void

6.46.3.21 QPair< QList< ItemHandle ∗ >, QList< QGraphicsItem ∗ > > Tinkercell::MainWindow::getItemsFromFile (const QString & *filename*, ItemHandle ∗ *root* = 0)

get the items inside a file. Some tool must implement this function and connect to the getItemsFromFile signal. The Core library does not implement a read file function.

Parameters

QString& file that is selected by user

ItemHandle∗ optional parent handle to all the items that will be loaded form file

Returns

QList<ItemHandle∗> list of items inside the file
void

6.46.3.22 void Tinkercell::MainWindow::handleFamilyChanged (NetworkHandle ∗ *network*, const QList< ItemHandle ∗ > &, const QList< ItemFamily ∗ > &) `[signal]`

signals whenever item handles' families are changed

Parameters

NetworkHandle∗ network where the event took place

QList<ItemHandle∗>& child items

QList<ItemFamily∗>& old families

Returns

void

6.46.3.23 void Tinkercell::MainWindow::handlesChanged (NetworkHandle ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QList< ItemHandle ∗ > & *old*) `[signal]`

signals whenever the handles for graphics items have changed

Parameters

GraphicsScene∗ scene where the event took place

QList<GraphicsItem∗>& items that are affected

QList<ItemHandle∗>& old handle for each items

Returns

void

6.46.3.24 void Tinkercell::MainWindow::historyChanged (int *i* = 0) `[signal]`

one of more changed have occurred in the history window of the current scene

Parameters

int number of changes (negative = undos, positive = redos)

Returns

void

6.46.3.25 QUndoStack $*$ Tinkercell::MainWindow::historyStack () const

the history stack of the current network.

Returns

QUndoStack$*$ current scene's history stack or null if current network is null

6.46.3.26 QUndoView $*$ Tinkercell::MainWindow::historyWidget ()

the history stack widget of the current window.

Returns

QUndoView$*$ current scene's history stack or null if current network is null

6.46.3.27 void Tinkercell::MainWindow::initializeMenus (bool *enableScene* = `true`, bool *enableText* = `true`)

Initialize the basic menu (save, open, close, exit, etc.).

Returns

void

6.46.3.28 void Tinkercell::MainWindow::itemsAboutToBeInserted (GraphicsScene $*$ *scene*, QList< QGraphicsItem $*$ > &, QList< ItemHandle $*$ > &, QList< QUndoCommand $*$ > &) `[signal]`

signals whenever items are going to be added

Parameters

GraphicsScene$*$ scene where the items are added

QList<QGraphicsItem$*$>& list of new graphics items

QList<ItemHandle$*$>& list of new handles (does NOT have to be the same number as items)

QList<QUndoCommand$*$>& list of commands that will be executed right before items are inserted

Returns

void

6.46.3.29 void Tinkercell::MainWindow::itemsAboutToBeMoved (GraphicsScene $*$ *scene*, QList< QGraphicsItem $*$ > & *item*, QList< QPointF > & *distance*, QList< QUndoCommand $*$ > &) `[signal]`

signals whenever items are going to be moved (each item is the top-most item)

Parameters

GraphicsScene∗ scene where the items were moved

QList<QGraphicsItem∗>& list of pointers to all moving items

QPointF distance by which items moved

Qt::KeyboardModifiers modifier keys being used when mouse clicked

QList<QUndoCommand∗>& list of commands that will be executed right before items are inserted

Returns

void

6.46.3.30 void Tinkercell::MainWindow::itemsAboutToBeRemoved (GraphicsScene ∗ *scene*, QList< QGraphicsItem ∗ > & *item*, QList< ItemHandle ∗ > & *handles*, QList< QUndoCommand ∗ > &) `[signal]`

signals just before items are deleted

Parameters

GraphicsScene∗ scene where the items are going to be removed

QList<QGraphicsItem∗>& list of items going to be removed

QList<ItemHandle∗>& list of handles going to be removed (does NOT have to be the same number as items removed)

QList<QUndoCommand∗>& list of commands that will be executed right before items are inserted

Returns

void

6.46.3.31 void Tinkercell::MainWindow::itemsDropped (GraphicsScene ∗, const QString &, const QPointF &) `[signal]`

signal is emitted when some object OTHER than files are dropped on the canvas

Parameters

GraphicsScene∗ the scene where objects were dropped

QString the string describing the object that was dropped

QPointF the Scene position where it was dropped

Returns

void

6.46.3.32 void Tinkercell::MainWindow::itemsInserted (NetworkHandle ∗ *win*, const QList< ItemHandle ∗ > &) `[signal]`

A convenient signal that is emitted when items are inserted from a GraphicsScene or TextEditor. Warning: listening to the other itemsInserted signals may cause redundancy.

Parameters

NetworkHandle∗ where the editting happened

QList<TextItem∗*>* new items

6.46.3.33 void Tinkercell::MainWindow::itemsInserted (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *item*, const QList< ItemHandle ∗ > & *handles*) `[signal]`

signals whenever items are added

Parameters

GraphicsScene ∗ scene where the items were added

QList<QGraphicsItem∗*>&* list of new items

QList<ItemHandle∗*>&* list of new handles (does NOT have to be the same number as items)

Returns

void

6.46.3.34 void Tinkercell::MainWindow::itemsInsertedSlot (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *item*, const QList< ItemHandle ∗ > & *handles*) `[protected, slot]`

signals whenever items are added

Parameters

GraphicsScene ∗ scene where the items were added

QList<QGraphicsItem∗*>&* list of new items

QList<ItemHandle∗*>&* list of new handles (does NOT have to be the same number as items)

Returns

void

6.46.3.35 void Tinkercell::MainWindow::itemsMoved (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *item*, const QList< QPointF > & *distance*) `[signal]`

signals whenever items are being moved (each item is the top-most item)

Parameters

GraphicsScene ∗ scene where the items were moved

QList<QGraphicsItem>&* list of pointes to all moving items

QPointF distance by which items moved

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.46.3.36 void Tinkercell::MainWindow::itemsRemoved (NetworkHandle ∗ *win*, const QList< ItemHandle ∗ > &) `[signal]`

A convenient signal that is emitted when items are removed from a GraphicsScene or TextEditor. Warning: listening to the other itemsRemoved signals may cause redundancy.

Parameters

NetworkHandle∗ where the editting happened

ItemHandle∗ removed items

6.46.3.37 void Tinkercell::MainWindow::itemsRemoved (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *item*, const QList< ItemHandle ∗ > & *handles*) `[signal]`

signals whenever items are deleted

Parameters

GraphicsScene ∗ scene where the items were removed

QList<QGraphicsItem>&* list of items removed

QList<ItemHandle>&* list of handles removed (does NOT have to be the same number as items removed)

Returns

void

6.46.3.38 void Tinkercell::MainWindow::itemsRemovedSlot (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *item*, const QList< ItemHandle ∗ > & *handles*) `[protected, slot]`

signals whenever items are deleted

Parameters

GraphicsScene ∗ scene where the items were removed

QList<QGraphicsItem>&* list of items removed

QList<ItemHandle>&* list of handles removed (does NOT have to be the same number as items removed)

Returns

void

6.46.3.39 void Tinkercell::MainWindow::itemsRenamed (NetworkHandle $*$ *window*, const QList< ItemHandle $*$ > & *items*, const QList< QString > & *oldnames*, const QList< QString > & *newnames*) `[signal]`

signals whenever an item is renamed

Parameters

NetworkHandle $*$ window where the event took place

QList<ItemHandle$*$*>&* items

QList<QString>& old names

QList<QString>& new names

Returns

void

6.46.3.40 void Tinkercell::MainWindow::itemsSelected (GraphicsScene $*$ *scene*, const QList< QGraphicsItem $*$ > & *items*, QPointF *point*, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever a new item is selected (item can be sub-item, not top-level)

Parameters

GraphicsScene $*$ scene where items are selected

QList<QGraphicsItem$*$*>&* list of all selected item pointers

QPointF point where mouse is clicked

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.46.3.41 void Tinkercell::MainWindow::keyPressed (GraphicsScene $*$ *scene*, QKeyEvent $*$) `[signal]`

signals whenever a key is pressed

Parameters

GraphicsScene $*$ scene where the event took place

QKeyEvent $*$ key that is pressed

Returns

void

6.46.3.42 void Tinkercell::MainWindow::keyReleased (GraphicsScene ∗ *scene*, QKeyEvent ∗) `[signal]`

signals whenever a key is released

Parameters

GraphicsScene ∗ scene where the event took place

QKeyEvent ∗ key that is released

Returns

void

6.46.3.43 void Tinkercell::MainWindow::lineChanged (TextEditor ∗, int, const QString &) `[signal]`

the cursor has moved to a different line

Parameters

TextEditor∗ editor

int index of the current line

QString current line text

6.46.3.44 void Tinkercell::MainWindow::loadDynamicLibrary (const QString & *dllFile*)

Load a new plugin (dll).

Parameters

the complete path of the dll file

Returns

void

6.46.3.45 void Tinkercell::MainWindow::loadFiles (const QList< QFileInfo > & *files*) `[protected, slot]`

loads files (library files or Network files)

Parameters

QList<QFileInfo>& the name(s) of the file(s)

Returns

void

6.46.3.46 void Tinkercell::MainWindow::loadNetwork (const QString & *filename*) `[signal]`

signals when user selects a file to open in the current network

Parameters

QString& file that is selected by user

Returns

void

6.46.3.47 void Tinkercell::MainWindow::mouseDoubleClicked (GraphicsScene ∗ *scene*, QPointF *point*, QGraphicsItem ∗, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

emits event when mouse is double clicked

Parameters

GraphicsScene ∗ scene where the event took place

point where mouse is clicked

modifier keys being used when mouse clicked

Returns

void

6.46.3.48 void Tinkercell::MainWindow::mouseDragged (GraphicsScene ∗ *scene*, QPointF *from*, QPointF *to*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever mouse is dragged from one point to another

Parameters

GraphicsScene ∗ scene where the event took place

QPointF point where mouse is clicked first

QPointF point where mouse is released

Qt::MouseButton button being pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.46.3.49 void Tinkercell::MainWindow::mouseMoved (GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, QPointF *point*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*, QList< QGraphicsItem ∗ > &) `[signal]`

signals whenever mouse moves, and indicates whether it is on top of an item

Parameters

GraphicsScene ∗ scene where the event took place

QGraphicsItem∗ pointer to item that mouse is on top of

QPointF point where mouse is clicked

Qt::MouseButton button being pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

QList<QGraphicsItem∗*>&* list of items that are being moved with the mouse

Returns

void

6.46.3.50 void Tinkercell::MainWindow::mouseOnTopOf (GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, QPointF *point*, Qt::KeyboardModifiers *modifiers*, QList< QGraphicsItem ∗ > &) `[signal]`

signals whenever mouse is on top of an item

Parameters

GraphicsScene ∗ scene where the event took place

QGraphicsItem∗ pointer to item that mouse is on top of

QPointF point where mouse is clicked

Qt::KeyboardModifiers modifier keys being used when mouse clicked

QList<QGraphicsItem∗*>&* list of items that are being moved with the mouse

Returns

void

6.46.3.51 void Tinkercell::MainWindow::mousePressed (GraphicsScene ∗ *scene*, QPointF *point*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever an empty node of the screen is clicked

Parameters

GraphicsScene ∗ scene where the event took place

QPointF point where mouse is clicked

Qt::MouseButton which button was pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.46.3.52 void Tinkercell::MainWindow::mouseReleased (GraphicsScene ∗ *scene*, QPointF *point*, Qt::MouseButton, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever an empty node of the screen is clicked

Parameters

GraphicsScene ∗ scene where the event took place

QPointF point where mouse is clicked

Qt::MouseButton which button was pressed

Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.46.3.53 void Tinkercell::MainWindow::networkClosed (NetworkHandle ∗) `[signal]`

signals after a window is closed

Parameters

NetworkHandle ∗ the window that was closed

Returns

void

6.46.3.54 void Tinkercell::MainWindow::networkClosing (NetworkHandle ∗, bool ∗) `[signal]`

signals when a network is going to close

Parameters

NetworkHandle ∗ the network that is closing

Boolean setting to false will prevent this window from closing

Returns

void

6.46.3.55 void Tinkercell::MainWindow::networkLoaded (NetworkHandle ∗) `[signal]`

signals informs that the current network has just loaded a new Network

Parameters

NetworkHandle ∗ the window where network was loaded (usually current scene)

Returns

void

6.46.3.56 void Tinkercell::MainWindow::networkOpened (NetworkHandle ∗) `[signal]`

signals whenever the new network is opened

Parameters

NetworkHandle∗ the current new window

Returns

void

6.46.3.57 QList< NetworkHandle ∗ > Tinkercell::MainWindow::networks () const

gets all the windows in the main window

Returns

QList<NetworkHandle∗> list of windows

6.46.3.58 void Tinkercell::MainWindow::networkSaved (NetworkHandle ∗) `[signal]`

signals when a tool has saved the network in a file

Parameters

NetworkHandle ∗ the window where network was loaded (usually current scene)

Returns

void

6.46.3.59 void Tinkercell::MainWindow::parentHandleChanged (NetworkHandle ∗ *scene*, const QList< ItemHandle ∗ > &, const QList< ItemHandle ∗ > &) `[signal]`

signals whenever item parent handle is changed

Parameters

NetworkHandle ∗ window where the event took place

QList<ItemHandle∗>& child items

QList<ItemHandle∗>& old parents

Returns

void

6.46.3.60 void Tinkercell::MainWindow::parentItemChanged (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QList< QGraphicsItem ∗ > & *parents*) `[signal]`

signals whenever item parents are changed

Parameters

GraphicsScene ∗ scene where the event took place

QList<QGraphicsItem∗>& items

QList<QGraphicsItem∗>& new parents

Returns

void

6.46.3.61 void Tinkercell::MainWindow::parse (TextEditor ∗) `[signal]`

request to parse the text in the current text editor

Parameters

TextEditor∗ editor

6.46.3.62 void Tinkercell::MainWindow::prepareNetworkForSaving (NetworkHandle ∗, bool ∗) `[signal]`

signals when a tool is about to save a network

Parameters

NetworkHandle ∗ the window where Network was loaded (usually current scene)

Returns

void

6.46.3.63 void Tinkercell::MainWindow::print () `[slot]`

triggered when the print button is clicked. Calls current scene's print

print the current scene

6.46.3.64 void Tinkercell::MainWindow::printToFile () `[slot]`

triggered when the print-to-file button is clicked. Calls current scene's print on a pdf file

print the current scene

6.46.3.65 void Tinkercell::MainWindow::readSettings () `[slot]`

read initial settings from settingsFileName

Returns

void

6.46.3.66 void Tinkercell::MainWindow::saveNetwork (const QString & *filename*) `[signal]`

signals when user selects a file to save the current network to

Parameters

QString& file that is selected by user

Returns

void

6.46.3.67 void Tinkercell::MainWindow::saveSettings () `[protected]`

save initial settings to settingsFileName

Returns

void

6.46.3.68 void Tinkercell::MainWindow::sceneRightClick (GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, QPointF *point*, Qt::KeyboardModifiers *modifiers*) `[signal]`

signals whenever right click is made on an item or sceen

Parameters

GraphicsScene ∗ scene where the event took place
QGraphicsItem∗ pointer to item that mouse is clicked on
QPointF point where mouse is clicked
Qt::KeyboardModifiers modifier keys being used when mouse clicked

Returns

void

6.46.3.69 void Tinkercell::MainWindow::setCursor (QCursor *cursor*)

set the cursor for all windows

Parameters

QCursor cursor

Returns

void

6.46.3.70 void Tinkercell::MainWindow::setupFunctionPointers (QLibrary ∗) `[signal]`

signals when a new FuntionToSignal is constructed

Parameters

QLibrary ∗ the new FuntionToSignal instance

Returns

void

6.46.3.71 void Tinkercell::MainWindow::setupFunctionPointersSlot (QSemaphore ∗ *s*, QLibrary ∗ *library*) `[protected, slot]`

send signal to other tools so that they can connect functions to signals

Parameters

QSemaphore∗ semaphore

QLibrary ∗ the dynamic library instance

Returns

void

6.46.3.72 void Tinkercell::MainWindow::setupNewThread (QSemaphore ∗ *s*, QLibrary ∗ *f*)

This function is usually called from a new thread. This function allows all the plugins to add their functionalities to the C function pointer of the new thread.

Parameters

QSemaphore∗ used to wait for all the plugins to initialize the thread

QLibrary∗ the library to load

Returns

void

6.46.3.73 void Tinkercell::MainWindow::textChanged (TextEditor ∗, const QString &, const QString &, const QString &) `[signal]`

some text inside this editor has been changed

Parameters

TextEditor∗ editor

QString old text (usually a line)

QString new text (usually a line)

6.46.3.74 Tool ∗ Tinkercell::MainWindow::tool (const QString & *s0*) const `[virtual]`

get a tool

Parameters

QString name of the tool

Returns

Tool∗

6.46.3.75 void Tinkercell::MainWindow::toolAboutToBeLoaded (Tool ∗ *tool*, bool ∗ *shouldLoad*) `[signal]`

a new tool is about to be added. This signal can be used to prevent the tool from being added

Parameters

Tool the tool itself

bool& set this bool to false to prevent the tool from loading

Returns

void

6.46.3.76 void Tinkercell::MainWindow::toolLoaded (Tool ∗ *tool*) `[signal]`

signals when a new tool (plugin) is loaded

Parameters

Tool∗ the new tool

Returns

void

6.46.3.77 QList< Tool ∗ > Tinkercell::MainWindow::tools () const `[virtual]`

get all tools

Returns

QList<Tool∗>

6.46.3.78 void Tinkercell::MainWindow::windowChanged (NetworkWindow ∗, NetworkWindow ∗) `[signal]`

signals whenever the current window changes

Parameters

***NetworkWindow*∗**	the previous windpw
***NetworkWindow*∗**	the current new window

Returns

void

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

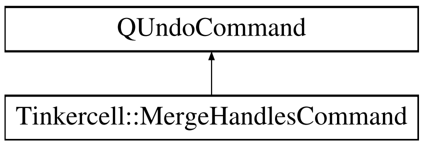
- MainWindow.h
- MainWindow.cpp

6.47 Tinkercell::MergeHandlesCommand Class Reference

this command places all the graphics items inside one handle into the other

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::MergeHandlesCommand:



Public Member Functions

- **MergeHandlesCommand** (const QString &text, NetworkHandle ∗, const QList< ItemHandle ∗ > &handles)
- void **redo** ()
- void **undo** ()

Public Attributes

- QList< ItemHandle ∗ > **oldHandles**
- ItemHandle ∗ **newHandle**

6.47.1 Detailed Description

this command places all the graphics items inside one handle into the other

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

- UndoCommands.h
- UndoCommands.cpp

6.48 Tinkercell::ModelReader Class Reference

reads an xml file with handle names and data table information and generates a list of item handles

```
#include <ModelReader.h>
```

Public Member Functions

- QList< QPair< QString, ItemHandle ∗ > > readHandles (QIODevice ∗device)

  Reads a list of <family,handles> pairs from an XML file using the IO device provided.

- QXmlStreamReader::TokenType readNext ()

  Reads up to the next start node.

6.48.1 Detailed Description

reads an xml file with handle names and data table information and generates a list of item handles

6.48.2 Member Function Documentation

6.48.2.1 QList< QPair< QString, ItemHandle ∗ > > Tinkercell::ModelReader::readHandles (QIODevice ∗ *device*)

Reads a list of <family,handles> pairs from an XML file using the IO device provided.

Parameters

QIODevice to use

Returns

list of item handles

6.48.2.2 QXmlStreamReader::TokenType Tinkercell::ModelReader::readNext ()

Reads up to the next start node.

Returns

Token Typer

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

- ModelReader.h
- ModelReader.cpp

6.49 Tinkercell::ModelWriter Class Reference

writes to an xml file handle names and data table information from a list of item handles

```
#include <ModelWriter.h>
```

Public Member Functions

- ModelWriter ()

  default constructor

- bool writeModel (NetworkHandle ∗, QIODevice ∗device)

  Writes the handles and data for that handle.

- bool writeModel (const QList< ItemHandle ∗ > &, QIODevice ∗device)

  Writes the handles and data for that handle.

Static Public Member Functions

- static bool writeModel (NetworkHandle ∗network, QXmlStreamWriter ∗)

  Writes the handles and data for that handle.

- static bool writeModel (const QList< ItemHandle ∗ > &, QXmlStreamWriter ∗)

  Writes the handles and data for that handle.

- static void writeDataTable (const DataTable< qreal > &, QXmlStreamWriter ∗)

  Writes a data table of doubles into an XML file.

- static void writeDataTable (const DataTable< QString > &, QXmlStreamWriter ∗)

  Writes a data table of strings into an XML file.

- static void writeHandle (ItemHandle ∗, QXmlStreamWriter ∗)

  Writes a handle and all its children.

Static Public Attributes

- static QString sep

  delimiter

6.49.1 Detailed Description

writes to an xml file handle names and data table information from a list of item handles

6.49.2 Constructor & Destructor Documentation

6.49.2.1 Tinkercell::ModelWriter::ModelWriter ()

default constructor

constructor. Sets autoformatting to true

6.49.3 Member Function Documentation

6.49.3.1 void Tinkercell::ModelWriter::writeDataTable (const DataTable< QString > & *table*, QXmlStreamWriter ∗ *writer*) `[static]`

Writes a data table of strings into an XML file.

Parameters

DataTable<QString>	datatable
QXmlStreamWriter∗	xml writer to use

Returns

void

Parameters

NodeImage	pointer to write as XML
index	of shape in NodeImage's shape vector

Returns

void

6.49.3.2 void Tinkercell::ModelWriter::writeDataTable (const DataTable< qreal > & *table*, QXmlStreamWriter ∗ *writer*) `[static]`

Writes a data table of doubles into an XML file.

Parameters

DataTable<qreal>	datatable
QXmlStreamWriter∗	xml writer to use

Returns

void

Parameters

NodeImage	pointer to write as XML
index	of shape in NodeImage's shape vector

Returns

void

6.49.3.3 void Tinkercell::ModelWriter::writeHandle (ItemHandle ∗ *handle*, QXmlStreamWriter ∗ *writer*) `[static]`

Writes a handle and all its children.

Parameters

Item handle pointer to write as XML

Returns

void

6.49.3.4 bool Tinkercell::ModelWriter::writeModel (const QList< ItemHandle ∗ > & *allItems*, QXmlStreamWriter ∗ *writer*) `[static]`

Writes the handles and data for that handle.

Parameters

QList<ItemHandle∗> list of handles (top level)

QXmlStreamWriter∗ xml writer to use

Returns

void

6.49.3.5 bool Tinkercell::ModelWriter::writeModel (NetworkHandle ∗ *network*, QXmlStreamWriter ∗ *writer*) `[static]`

Writes the handles and data for that handle.

Parameters

NetworkHandle∗ network

QXmlStreamWriter∗ xml writer to use

Returns

void

6.49.3.6 bool Tinkercell::ModelWriter::writeModel (const QList< ItemHandle ∗ > & *list*, QIODevice ∗ *device*)

Writes the handles and data for that handle.

Parameters

QList<ItemHandle∗> list of handles (top level)

QIODevice device to use

Returns

void

6.49.3.7 bool Tinkercell::ModelWriter::writeModel (NetworkHandle ∗ *network*, QIODevice ∗ *device*)

Writes the handles and data for that handle.

Parameters

NetworkHandle∗ network

QIODevice device to use

Returns

void

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

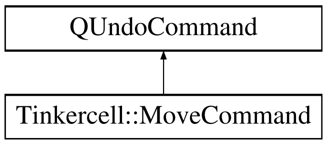
- ModelWriter.h
- ModelWriter.cpp

6.50 Tinkercell::MoveCommand Class Reference

this command performs a move and allows redo/undo of that move

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::MoveCommand:



Public Member Functions

- MoveCommand (GraphicsScene ∗scene, QGraphicsItem ∗item, const QPointF &distance)

  constructor
- MoveCommand (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, const QPointF &distance)

  constructor
- MoveCommand (GraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, const QList< QPointF > &distance)

  constructor
- void redo ()

  redo the change
- void undo ()

  undo the change

Static Public Member Functions

- static void refreshAllConnectionIn (const QList< QGraphicsItem ∗ > &)

  refresh all connectors that are attached to any of the items in the list

6.50.1 Detailed Description

this command performs a move and allows redo/undo of that move

6.50.2 Constructor & Destructor Documentation

6.50.2.1 Tinkercell::MoveCommand::MoveCommand (GraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, const QPointF & *distance*)

constructor

Parameters

***GraphicsScene*∗** scene where change happened

QGraphicsItem ∗ items that are affected

QPointF& amount to move

6.50.2.2 Tinkercell::MoveCommand::MoveCommand (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QPointF & *distance*)

constructor

Parameters

scene where change happened

items that are affected

QPointF& amount to move

6.50.2.3 Tinkercell::MoveCommand::MoveCommand (GraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QList< QPointF > & *distance*)

constructor

Parameters

***GraphicsScene*∗** scene where change happened

QList<QGraphicsItem∗>& items that are affected

QPointF& amount to move

6.50.3 Member Function Documentation

6.50.3.1 void Tinkercell::MoveCommand::refreshAllConnectionIn (const QList< QGraphicsItem ∗ > & *moving*) `[static]`

refresh all connectors that are attached to any of the items in the list

Parameters

items list to check

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

- UndoCommands.h
- UndoCommands.cpp

6.51 Tinkercell::MultithreadedSliderWidget Class Reference

This class is used to run specific functions inside a C dynamic library as a separate thread. Uses CThread to call the C functions.

`#include <MultithreadedSliderWidget.h>`

Public Slots

- virtual void setSliders (const QStringList &options, const QList< double > &minValues, const QList< double > &maxValues)

  setup the sliders options and initial values

- virtual void setVisibleSliders (const QStringList &options)

  set the sliders visible

Signals

- void optionsChanged (const QStringList &)

  the options in the slider have changed

- void valuesChanged (const QList< double > &)

  the values in the slider have changed

Public Member Functions

- MultithreadedSliderWidget (MainWindow ∗parent, CThread ∗thread, Qt::Orientation orientation=Qt::Horizontal)

  constructor

- MultithreadedSliderWidget (MainWindow ∗parent, const QString &lib, const QString &functionName, Qt::Orientation orientation=Qt::Horizontal)

  constructor

- virtual CThread ∗ thread () const

  the cthread that is run every time the sliders change

- virtual DataTable< qreal > data () const

  table containing the variables, current values, min and max

Protected Slots

- virtual void valueChanged ()

  whenver the value text change, the function in the C library is called

- virtual void sliderChanged (int)

whenver the sliders change, the function in the C library is called

- virtual void minmaxChanged ()

  whenver the text change, the function in the C library is called

- virtual void saveValues ()

  copy the values from the slider to the model

Protected Attributes

- CThread ∗ cthread

  whenver the slides change, cthread->start() is called

- Qt::Orientation orientation

  orientation of the sliders

- DataTable< qreal > values

  table storing slider values

- QList< QLabel ∗ > labels

  slider labels in same order as sliders list

- QList< QSlider ∗ > sliders

  all the sliders

- QList< QLineEdit ∗ > minline

  slider min, max, and values in same order as sliders list

- QList< QLineEdit ∗ > **maxline**
- QList< QLineEdit ∗ > **valueline**
- QList< double > min

  slider min and max in same order as sliders list

- QList< double > **max**
- QVBoxLayout ∗ slidersLayout

  slider layout

- QHash< QString, QWidget ∗ > sliderWidgets

  sliders by name

- MainWindow ∗ mainWindow

  main window

6.51.1 Detailed Description

This class is used to run specific functions inside a C dynamic library as a separate thread. Uses CThread to call the C functions.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 Tinkercell::MultithreadedSliderWidget::MultithreadedSliderWidget (MainWindow ∗ *parent*, CThread ∗ *thread*, Qt::Orientation *orientation* = `Qt::Horizontal`)

constructor

Parameters

QWidget ∗ parent
CThread ∗ the thread that is already setup with the correct library and function
Qt::Orientation orientation

6.51.2.2 Tinkercell::MultithreadedSliderWidget::MultithreadedSliderWidget (MainWindow ∗ *parent*, const QString & *lib*, const QString & *functionName*, Qt::Orientation *orientation* = `Qt::Horizontal`)

constructor

Parameters

QWidget ∗ parent
QString the name of the dynamic library to load
QString name of function in the library with signature void f(Matrix)
Qt::Orientation orientation

6.51.3 Member Function Documentation

6.51.3.1 void Tinkercell::MultithreadedSliderWidget::setSliders (const QStringList & *options*, const QList< double > & *minValues*, const QList< double > & *maxValues*) `[virtual, slot]`

setup the sliders options and initial values

Parameters

QStringList names for the sliders
QList<double> minimum value for each of the sliders
QList<double> maximum value for each of the sliders

6.51.3.2 void Tinkercell::MultithreadedSliderWidget::setVisibleSliders (const QStringList & *options*) `[virtual, slot]`

set the sliders visible

Parameters

QStringList names for the sliders

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

- MultithreadedSliderWidget.h
- MultithreadedSliderWidget.cpp

6.52 Tinkercell::NetworkHandle Class Reference

A class that is used to store a network. The network is a collection of Item Handles. The history stack is also a key component of a network. The network can either be represented as text using TextEditor or visualized with graphical items in the GraphicsScene. Each node and connection are contained in a handle, and each handle can either be represented as text or as graphics. The two main components of NetworkWindow are the SymbolsTable and HistoryStack This class provides functions for inserting items, removing items, and changing information inside the model.

```
#include <NetworkHandle.h>
```

Public Slots

slots

update the symbols table that stores all the symbols in the network

- virtual void updateSymbolsTable ()
  updates the symbols table
- virtual void updateSymbolsTable (int)
  updates the symbols table. The int argument is so that this can be connected to the history changed signal
- virtual void close ()
  updates the symbols table. The int argument is so that this can be connected to the history changed signal
- virtual void undo ()
  undo last command
- virtual void redo ()
  redo last command
- virtual void push (QUndoCommand ∗)
  push a new command into the history stack

Public Member Functions

Constructor and destructor

- NetworkHandle (MainWindow ∗)
  constructor
- virtual ∼NetworkHandle ()
  destructor

Get items

get the set of items in the model

- virtual QList< ItemHandle ∗ > handles (bool includeGlobalHandle=true, bool sort=false)
  get all the visible items in this network window
- virtual QList< ItemHandle ∗ > handlesSortedByFamily () const

get list of all items sorted according to family

- virtual ItemHandle ∗ globalHandle ()
  the model global item
- virtual GraphicsScene ∗ currentScene () const
  gets the current scene that is active
- virtual TextEditor ∗ currentTextEditor () const
  gets the text editor that is active
- virtual NetworkWindow ∗ currentWindow () const
  gets the window that is active
- virtual void showScene (GraphicsScene ∗)
  show the window that contains the given scene
- virtual void showTextEditor (TextEditor ∗)
  show the window that contains the given text editor
- ConsoleWindow ∗ console () const
  same as main window's console()

find item handles and data tables

- QList< ItemHandle ∗ > findItem (const QString &) const
  get all the items with the given name. Returns a list for non-unique names
- QList< ItemHandle ∗ > findItem (const QStringList &) const
  get all the items with the given name. returned list may be longer if names are non-unique
- QList< QPair< ItemHandle ∗, QString > > findData (const QString &) const
  get all the items and corresponding data table name that contains the given string. if non-unique, returns a list
- QList< QPair< ItemHandle ∗, QString > > findData (const QStringList &) const
  get all the items and corresponding data table name that contains the given string. if non-unique, returns a list

create scene or editor

- virtual void remove (const QString &name, const QList< QGraphicsItem ∗ > &items)
  this command performs an removal and also adds undo command to history window and emits associated signal(s)
- virtual QList< GraphicsScene ∗ > scenes () const
  get all the graphics scenes used to illustrate this network
- virtual QList< TextEditor ∗ > editors () const
  get all the text editors used to express this network
- virtual GraphicsScene ∗ createScene (const QList< QGraphicsItem ∗ > &insertItems=QList< QGraphicsItem ∗ >())

create a new scene for this network

- virtual GraphicsScene ∗ createScene (ItemHandle ∗, const QRectF &boundingRect=QRectF())

  create a new scene that gets all the items inside the given item handle.

- virtual TextEditor ∗ createTextEditor (const QString &text=QString())

  create a new text editor for this network

- virtual void setWindowTitle (const QString &)

  set all the title for each window representing this network

- virtual QString windowTitle () const

  get the title for current window representing this network

- virtual bool parseMath (QString &, QStringList &)

  checks whether a string is a correct formula.

- virtual QString makeUnique (const QString &, const QStringList &doNotUse-Names=QStringList()) const

  checks whether the given string names a unique item or data entry

- virtual QString makeUnique (ItemHandle ∗handle, const QStringList &doNotUse-Names=QStringList()) const

  checks whether the given handle's name is unique and returns a new name. Note that this can be different from makeUnqiue for strings, because this function will check if an existing name belongs to the given handle, in which case no change is needed.

- virtual QStringList makeUnique (const QStringList &, const QStringList &doNotUse-Names=QStringList()) const

  checks whether the given string names a unique item or data entry

rename items

These functions automatically perform history updates and send appropriate signals, which will inform the other tools that an insertion or deletion has taken place.

- virtual void rename (const QString &oldname, const QString &new_name)

  rename item and also adds undo command to history window and emits associated signal(s)

- virtual void rename (ItemHandle ∗item, const QString &new_name)

  rename an item and also adds undo command to history window and emits associated signal(s)

- virtual void rename (const QList< ItemHandle ∗ > &items, const QList< QString > &new_-names)

  rename items and also adds undo command to history window and emits associated signal(s)

change parents of items

These functions automatically perform history updates and send appropriate signals, which will inform the other tools that an insertion or deletion has taken place.

- virtual void setParentHandle (const QList< ItemHandle ∗ > &handles, const QList< ItemHandle ∗ > &parentHandles)

  change parent handles and also adds undo command to history window and emits associated signal(s)

- virtual void setParentHandle (ItemHandle ∗child, ItemHandle ∗parent)

  change parent handle and also adds undo command to history window and emits associated signal(s)

- virtual void setParentHandle (const QList< ItemHandle ∗ > children, ItemHandle ∗parent)

  change parent for handles and also adds undo command to history window and emits associated signal(s)

- virtual void setHandleFamily (const QList< ItemHandle ∗ > &handles, const QList< ItemFamily ∗ > &newfamilies)

  change handles families and also adds undo command to history window and emits associated signal(s)

- virtual void setHandleFamily (ItemHandle ∗handle, ItemFamily ∗newfamily)

  change handle and also adds undo command to history window and emits associated signal(s)

- virtual void setHandleFamily (const QList< ItemHandle ∗ > handles, ItemFamily ∗newfamily)

  change family for handles and also adds undo command to history window and emits associated signal(s)

change data in one or more items

These functions automatically perform history updates and send appropriate signals, which will inform the other tools that an insertion or deletion has taken place.

- virtual void changeData (const QString &name, ItemHandle ∗handle, const QString &hashstring, const NumericalDataTable ∗newdata)

  change numerical data table and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QList< QString > &hashstring, const QList< NumericalDataTable ∗ > &newdata)

  change a list of numerical data tables and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QString &hashstring, const QList< NumericalDataTable ∗ > &newdata)

  change a list of numerical data tables and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &name, ItemHandle ∗handle, const QString &hashstring, const TextDataTable ∗newdata)

  change text data table and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QList< QString > &hashstring, const QList< TextDataTable ∗ > &newdata)

  change a list of text data tables and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QString &hashstring, const QList< TextDataTable ∗ > &newdata)

  change a list of text data tables and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &name, ItemHandle ∗handle, const QString &hashstring, const NumericalDataTable ∗newdata1, const TextDataTable ∗newdata2)

  change two types of data tables and also adds undo command to history window and emits associated signal(s)

- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QList< QString > &hashstring, const QList< NumericalDataTable ∗ > &newdata1, const QList< TextDataTable ∗ > &newdata2)

  change a list of two types of data tables and also adds undo command to history window and emits associated signal(s)
- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QString &hashstring, const QList< NumericalDataTable ∗ > &newdata1, const QList< TextDataTable ∗ > &newdata2)

  change a list of two types of data tables and also adds undo command to history window and emits associated signal(s)
- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QList< NumericalDataTable ∗ > &olddata1, const QList< NumericalDataTable ∗ > &newdata1)

  change a list of two types of data tables and also adds undo command to history window and emits associated signal(s)
- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QList< TextDataTable ∗ > &olddata2, const QList< TextDataTable ∗ > &newdata2)

  change a list of two types of data tables and also adds undo command to history window and emits associated signal(s)
- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, const QList< NumericalDataTable ∗ > &olddata1, const QList< NumericalDataTable ∗ > &newdata1, const QList< TextDataTable ∗ > &olddata2, const QList< TextDataTable ∗ > &newdata2)

  change a list of two types of data tables and also adds undo command to history window and emits associated signal(s)
- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, NumericalDataTable ∗olddata1, const NumericalDataTable ∗newdata1, TextDataTable ∗olddata2, const TextDataTable ∗newdata2)

  change a two types of data tables and also adds undo command to history window and emits associated signal(s)
- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, NumericalDataTable ∗olddata1, const NumericalDataTable ∗newdata1)

  change a data table and also adds undo command to history window and emits associated signal(s)
- virtual void changeData (const QString &name, const QList< ItemHandle ∗ > &handles, TextDataTable ∗olddata1, const TextDataTable ∗newdata1)

  change a data table and also adds undo command to history window and emits associated signal(s)
- virtual void assignHandles (const QList< QGraphicsItem ∗ > &items, ItemHandle ∗newHandle)

  assign the handle for one or more items
- virtual void mergeHandles (const QList< ItemHandle ∗ > &handles)

  marge the graphics items and children of two or more handles

Public Attributes

- QUndoStack history

the undo stack

- SymbolsTable symbolsTable

  holds a hash of all items and data in this scene.

signals

- class **GraphicsView**
- class **GraphicsScene**
- class **TextEditor**
- class **MainWindow**
- class **NetworkWindow**
- class **SymbolsTable**
- void itemsRenamed (NetworkHandle ∗network, const QList< ItemHandle ∗ > &items, const QList< QString > &oldnames, const QList< QString > &newnames)

  signals whenever an item is renamed

- void parentHandleChanged (NetworkHandle ∗network, const QList< ItemHandle ∗ > &, const QList< ItemHandle ∗ > &)

  signals whenever item parent handle is changed

- void handleFamilyChanged (NetworkHandle ∗network, const QList< ItemHandle ∗ > &, const QList< ItemFamily ∗ > &)

  signals whenever item handles' families are changed

- void dataChanged (const QList< ItemHandle ∗ > &items)

  signals whenever some data is changed

- void handlesChanged (NetworkHandle ∗network, const QList< QGraphicsItem ∗ > &items, const QList< ItemHandle ∗ > &old)

  signals whenever the handles for graphics items have changed

6.52.1 Detailed Description

A class that is used to store a network. The network is a collection of Item Handles. The history stack is also a key component of a network. The network can either be represented as text using TextEditor or visualized with graphical items in the GraphicsScene. Each node and connection are contained in a handle, and each handle can either be represented as text or as graphics. The two main components of NetworkWindow are the SymbolsTable and HistoryStack This class provides functions for inserting items, removing items, and changing information inside the model.

6.52.2 Member Function Documentation

6.52.2.1 void Tinkercell::NetworkHandle::changeData (const QString & *name*, const QList< ItemHandle ∗ > & *handles*, const QString & *hashstring*, const QList< NumericalDataTable ∗ > & *newdata1*, const QList< TextDataTable ∗ > & *newdata2*) `[virtual]`

change a list of two types of data tables and also adds undo command to history window and emits associated signal(s)

change a list of two types of data tables

6.52.2.2 void Tinkercell::NetworkHandle::changeData (const QString & *name*, const QList< ItemHandle ∗ > & *handles*, const QList< QString > & *hashstring*, const QList< NumericalDataTable ∗ > & *newdata1*, const QList< TextDataTable ∗ > & *newdata2*) `[virtual]`

change a list of two types of data tables and also adds undo command to history window and emits associated signal(s)

change a list of two types of data tables

6.52.2.3 void Tinkercell::NetworkHandle::changeData (const QString & *name*, ItemHandle ∗ *handle*, const QString & *hashstring*, const NumericalDataTable ∗ *newdata1*, const TextDataTable ∗ *newdata2*) `[virtual]`

change two types of data tables and also adds undo command to history window and emits associated signal(s)

change two types of data tables

6.52.2.4 void Tinkercell::NetworkHandle::changeData (const QString & *name*, const QList< ItemHandle ∗ > & *handles*, const QString & *hashstring*, const QList< TextDataTable ∗ > & *newdata*) `[virtual]`

change a list of text data tables and also adds undo command to history window and emits associated signal(s)

change a list of text data tables

6.52.2.5 void Tinkercell::NetworkHandle::changeData (const QString & *name*, const QList< ItemHandle ∗ > & *handles*, const QList< QString > & *hashstring*, const QList< TextDataTable ∗ > & *newdata*) `[virtual]`

change a list of text data tables and also adds undo command to history window and emits associated signal(s)

change a list of text data tables

6.52.2.6 void Tinkercell::NetworkHandle::changeData (const QString & *name*, ItemHandle ∗ *handle*, const QString & *hashstring*, const TextDataTable ∗ *newdata*) `[virtual]`

change text data table and also adds undo command to history window and emits associated signal(s) change text data table

6.52.2.7 void Tinkercell::NetworkHandle::changeData (const QString & *name*, const QList< ItemHandle ∗ > & *handles*, const QString & *hashstring*, const QList< NumericalDataTable ∗ > & *newdata*) `[virtual]`

change a list of numerical data tables and also adds undo command to history window and emits associated signal(s)

change a list of numerical data tables

6.52.2.8 void Tinkercell::NetworkHandle::changeData (const QString & *name*, const QList< ItemHandle ∗ > & *handles*, const QList< QString > & *hashstring*, const QList< NumericalDataTable ∗ > & *newdata*) `[virtual]`

change a list of numerical data tables and also adds undo command to history window and emits associated signal(s)

change a list of numerical data tables

6.52.2.9 void Tinkercell::NetworkHandle::changeData (const QString & *name*, ItemHandle ∗ *handle*, const QString & *hashstring*, const NumericalDataTable ∗ *newdata*) `[virtual]`

change numerical data table and also adds undo command to history window and emits associated signal(s)

change numerical data table

6.52.2.10 GraphicsScene ∗ Tinkercell::NetworkHandle::createScene (ItemHandle ∗ *item*, const QRectF & *boundingRect* = `QRectF()`) `[virtual]`

create a new scene that gets all the items inside the given item handle.

Parameters

ItemHandle ∗

QRectF only include the graphicss items

Returns

GraphicsScene∗ the new scene

6.52.2.11 GraphicsScene ∗ Tinkercell::NetworkHandle::createScene (const QList< QGraphicsItem ∗ > & *insertItems* = `QList<QGraphicsItem∗>()`) `[virtual]`

create a new scene for this network

Parameters

QList<QGraphicsItem∗> items to initialize the network with

Returns

GraphicsScene∗ the new scene

6.52.2.12 TextEditor ∗ Tinkercell::NetworkHandle::createTextEditor (const QString & *text* = `QString()`) `[virtual]`

create a new text editor for this network

Parameters

QString (optional) initial script

Returns

TextEditor∗ the new scene

6.52.2.13 GraphicsScene ∗ Tinkercell::NetworkHandle::currentScene () const `[virtual]`

gets the current scene that is active

Returns

GraphicsScene∗ current scene

6.52.2.14 TextEditor ∗ Tinkercell::NetworkHandle::currentTextEditor () const `[virtual]`

gets the text editor that is active

Returns

TextEditor∗ current editor

6.52.2.15 NetworkWindow ∗ Tinkercell::NetworkHandle::currentWindow () const `[virtual]`

gets the window that is active

Returns

NetworkWindow∗ current window

6.52.2.16 void Tinkercell::NetworkHandle::dataChanged (const QList< ItemHandle ∗ > & *items*) `[signal]`

signals whenever some data is changed

Parameters

QList<ItemHandle∗>& items handles

Returns

void

6.52.2.17 QList< TextEditor * > Tinkercell::NetworkHandle::editors () const `[virtual]`

get all the text editors used to express this network

Returns

QList<TextEditor*>

6.52.2.18 QList< QPair< ItemHandle *, QString > > Tinkercell::NetworkHandle::findData (const QStringList & *list*) const

get all the items and corresponding data table name that contains the given string. if non-unique, returns a list

Parameters

QString

Returns

QPair<ItemHandle*,QString>

6.52.2.19 QList< QPair< ItemHandle *, QString > > Tinkercell::NetworkHandle::findData (const QString & *s*) const

get all the items and corresponding data table name that contains the given string. if non-unique, returns a list

Parameters

QString

Returns

QPair<ItemHandle*,QString>

6.52.2.20 QList< ItemHandle * > Tinkercell::NetworkHandle::findItem (const QStringList & *list*) const

get all the items with the given name. returned list may be longer if names are non-unique

Parameters

QStringList

Returns

QList<ItemHandle*>

6.52.2.21 QList< ItemHandle ∗ > Tinkercell::NetworkHandle::findItem (const QString & *s*) const

get all the items with the given name. Returns a list for non-unique names

Parameters

QString

Returns

QList<ItemHandle∗>

6.52.2.22 void Tinkercell::NetworkHandle::handleFamilyChanged (NetworkHandle ∗ *network*, const QList< ItemHandle ∗ > &, const QList< ItemFamily ∗ > &) `[signal]`

signals whenever item handles' families are changed

Parameters

NetworkHandle∗ network where the event took place

QList<ItemHandle∗>& child items

QList<ItemFamily∗>& old families

Returns

void

6.52.2.23 QList< ItemHandle ∗ > Tinkercell::NetworkHandle::handles (bool *includeGlobalHandle* = `true`, bool *sort* = `false`) `[virtual]`

get all the visible items in this network window

Parameters

bool include the global handle (default = true)

bool sort handles by full name (default = false)

6.52.2.24 void Tinkercell::NetworkHandle::handlesChanged (NetworkHandle ∗ *network*, const QList< QGraphicsItem ∗ > & *items*, const QList< ItemHandle ∗ > & *old*) `[signal]`

signals whenever the handles for graphics items have changed

Parameters

NetworkHandle∗ network where the event took place

QList<GraphicsItem∗>& items that are affected

QList<ItemHandle∗>& old handle for each items

Returns

void

6.52.2.25 void Tinkercell::NetworkHandle::itemsRenamed (NetworkHandle ∗ *network*, const QList< ItemHandle ∗ > & *items*, const QList< QString > & *oldnames*, const QList< QString > & *newnames*) `[signal]`

signals whenever an item is renamed

Parameters

NetworkHandle∗ network where the event took place

QList<ItemHandle∗>& items

QList<QString>& old names

QList<QString>& new names

Returns

void

6.52.2.26 QStringList Tinkercell::NetworkHandle::makeUnique (const QStringList & *oldnames*, const QStringList & *doNotUseNames* = `QStringList()`) const `[virtual]`

checks whether the given string names a unique item or data entry

Parameters

QStringList target strings

Returns

QStringList new strings

6.52.2.27 QString Tinkercell::NetworkHandle::makeUnique (ItemHandle ∗ *handle*, const QStringList & *doNotUseNames* = `QStringList()`) const `[virtual]`

checks whether the given handle's name is unique and returns a new name. Note that this can be different from makeUnqiue for strings, because this function will check if an existing name belongs to the given handle, in which case no change is needed.

Parameters

ItemHandle ∗ handle

QStringList any other names that should be disallowed (optional)

Returns

QString new string

6.52.2.28 QString Tinkercell::NetworkHandle::makeUnique (const QString & *str*, const QStringList & *doNotUseNames* = `QStringList()`) const `[virtual]`

checks whether the given string names a unique item or data entry

Parameters

QString target string
QStringList any other names that should be disallowed (optional)

Returns

QString new string

6.52.2.29 void Tinkercell::NetworkHandle::parentHandleChanged (NetworkHandle ∗ *network*, const QList< ItemHandle ∗ > &, const QList< ItemHandle ∗ > &) `[signal]`

signals whenever item parent handle is changed

Parameters

NetworkHandle∗ network where the event took place
QList<ItemHandle∗>& child items
QList<ItemHandle∗>& old parents

Returns

void

6.52.2.30 bool Tinkercell::NetworkHandle::parseMath (QString & *s*, QStringList & *newvars*) `[virtual]`

checks whether a string is a correct formula.

Parameters

QString target string (also the output)
QStringList returns any new variables not found in this network

Returns

Boolean whether or not the string is valid

6.52.2.31 QList< GraphicsScene ∗ > Tinkercell::NetworkHandle::scenes () const `[virtual]`

get all the graphics scenes used to illustrate this network

Returns

QList<GraphicsScene∗>

6.52.2.32 void Tinkercell::NetworkHandle::setWindowTitle (const QString & *title*) `[virtual]`

set all the title for each window representing this network

Parameters

QString

6.52.2.33 void Tinkercell::NetworkHandle::showScene (GraphicsScene ∗ *scene*) `[virtual]`

show the window that contains the given scene

Returns

GraphicsScene ∗ scene

6.52.2.34 void Tinkercell::NetworkHandle::showTextEditor (TextEditor ∗ *editor*) `[virtual]`

show the window that contains the given text editor

Returns

TextEditor ∗ text editor

6.52.2.35 void Tinkercell::NetworkHandle::updateSymbolsTable (int) `[virtual, slot]`

updates the symbols table. The int argument is so that this can be connected to the history changed signal update symbols table

6.52.2.36 void Tinkercell::NetworkHandle::updateSymbolsTable () `[virtual, slot]`

updates the symbols table
update symbols table

6.52.2.37 QString Tinkercell::NetworkHandle::windowTitle () const `[virtual]`

get the title for current window representing this network

Returns

QString

6.52.3 Member Data Documentation

6.52.3.1 SymbolsTable Tinkercell::NetworkHandle::symbolsTable

holds a hash of all items and data in this scene.

See also

SymbolsTable

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

- NetworkHandle.h
- NetworkHandle.cpp

6.53 Tinkercell::NetworkWindow Class Reference

Public Slots

- virtual void popOut ()

  calls main window's popOut

- virtual void popIn ()

  calls main window's popIn

- virtual void setFileName (const QString &)

  set file name and window title

Signals

- void networkClosing (NetworkHandle ∗, bool ∗)

  signals when a window is going to close

- void networkClosed (NetworkHandle ∗)

  signals after a window is closed

Public Member Functions

- virtual GraphicsScene ∗ newScene ()

  replace the current text editor or scene with a new scene

- virtual TextEditor ∗ newTextEditor ()

  replace the current text editor or scene with a new text editor

Public Attributes

- NetworkHandle ∗ network

  the network displayed in this window

- ItemHandle ∗ handle

  this pointer will be non-zero if an ItemHandle is associated with this window

- GraphicsScene ∗ scene

  the scene inside this window. Either the scene or the editor must be 0

- TextEditor ∗ editor

  the editor inside this window. Either the scene or the editor must be 0

Protected Member Functions

- virtual void closeEvent (QCloseEvent ∗event)

  close event sends signal to all tools asking for confirmation becore closing

- virtual void focusInEvent (QFocusEvent ∗)

  focus receved changes the main windows current network pointer

- virtual void resizeEvent (QResizeEvent ∗event)

  resize event checks if the window has been minimized and calls popIn instead of minimizing

- virtual void setAsCurrentWindow ()

  calls main window's setAsCurrentWindow

- virtual void changeEvent (QEvent ∗event)

  calls popIn when minimized

- virtual void connectToMainWindow ()

  make all the main window connections

- NetworkWindow (NetworkHandle ∗network, GraphicsScene ∗scene)

  constructor with scene

- NetworkWindow (NetworkHandle ∗network, TextEditor ∗editor)

  constructor with text editor

- virtual ∼NetworkWindow ()

  destructor

Protected Attributes

- QString filename

  filename associated with this window

Friends

- class **MainWindow**
- class **GraphicsScene**
- class **GraphicsView**
- class **TextEditor**
- class **NetworkHandle**
- class **SymbolsTable**

6.53.1 Member Function Documentation

6.53.1.1 void Tinkercell::NetworkWindow::changeEvent (QEvent ∗ *event*) `[protected, virtual]`

calls popIn when minimized

Returns

void

6.53.1.2 void Tinkercell::NetworkWindow::closeEvent (QCloseEvent ∗ *event*) `[protected, virtual]`

close event sends signal to all tools asking for confirmation becore closing

Parameters

QCloseEvent ∗ event

Returns

void

6.53.1.3 void Tinkercell::NetworkWindow::focusInEvent (QFocusEvent ∗) `[protected, virtual]`

focus receved changes the main windows current network pointer

Parameters

QFocusEvent∗

Returns

void

6.53.1.4 void Tinkercell::NetworkWindow::networkClosed (NetworkHandle ∗) `[signal]`

signals after a window is closed

Parameters

NetworkWindow ∗ the window that was closed

Returns

void

6.53.1.5 void Tinkercell::NetworkWindow::networkClosing (NetworkHandle ∗, bool ∗) `[signal]`

signals when a window is going to close

Parameters

NetworkWindow ∗ the window that is closing

Boolean setting to false will prevent this window from closing

Returns

void

6.53.1.6 GraphicsScene ∗ Tinkercell::NetworkWindow::newScene () `[virtual]`

replace the current text editor or scene with a new scene

Returns

GraphicsScene ∗ scene

6.53.1.7 TextEditor ∗ Tinkercell::NetworkWindow::newTextEditor () `[virtual]`

replace the current text editor or scene with a new text editor

Returns

GraphicsScene ∗ scene

6.53.1.8 void Tinkercell::NetworkWindow::popIn () `[virtual, slot]`

calls main window's popIn

Returns

void

6.53.1.9 void Tinkercell::NetworkWindow::popOut () `[virtual, slot]`

calls main window's popOut

Returns

void

6.53.1.10 void Tinkercell::NetworkWindow::resizeEvent (QResizeEvent ∗ *event*) `[protected, virtual]`

resize event checks if the window has been minimized and calls popIn instead of minimizing

Parameters

QResizeEvent∗

Returns

void

6.53.1.11 void Tinkercell::NetworkWindow::setAsCurrentWindow () `[protected, virtual]`

calls main window's setAsCurrentWindow

Returns

void

6.53.1.12 void Tinkercell::NetworkWindow::setFileName (const QString & *text*) `[virtual, slot]`

set file name and window title

Returns

void

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

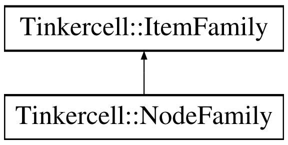
- NetworkWindow.h
- NetworkWindow.cpp

6.54 Tinkercell::NodeFamily Class Reference

This class defines the family of a node. Inherits from ItemFamily. It contains a list of NodeGraphicsItems that is the default for this family of nodes.

`#include <ItemFamily.h>`

Inheritance diagram for Tinkercell::NodeFamily:



Public Member Functions

- virtual ItemFamily $*$ parent () const

  get the parent for this family. If there are more than one parents, returns the first

- virtual QList< ItemFamily $*$ > parents () const

  get all the parents for this family.

- virtual QList< ItemFamily $*$ > children () const

  get all the families that make up this family.

- virtual void setParent (NodeFamily $*$)

  set parent family

- virtual ~NodeFamily ()

  destructor.

- NodeFamily (const QString &name=QString())

  constructor.

- virtual bool isA (const QString &) const

  indicates whether or not the given string is the name of this family or any of its parent families

- virtual bool isA (const ItemFamily $*$) const

  indicates whether or not the given family is the name of this family or any of its parent families

Static Public Member Functions

- static NodeFamily $*$ cast (ItemFamily $*$)

  cast to connection family

Protected Member Functions

- virtual bool isA (int) const

    indicates whether or not the given ID is this family or any of its parent families

Protected Attributes

- QList< NodeFamily ∗ > parentFamilies

    all the parents

- QList< NodeFamily ∗ > childFamilies

    all the families that are under this family

Friends

- class **ConnectionFamily**

6.54.1 Detailed Description

This class defines the family of a node. Inherits from ItemFamily. It contains a list of NodeGraphicsItems that is the default for this family of nodes.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 Tinkercell::NodeFamily::NodeFamily (const QString & *name* = `QString()`)

constructor.

Parameters

QString name

6.54.3 Member Function Documentation

6.54.3.1 bool Tinkercell::NodeFamily::isA (int *id*) const `[protected, virtual]`

indicates whether or not the given ID is this family or any of its parent families

indicates whether or not the given string is the name of this family or any of its parent families

Reimplemented from Tinkercell::ItemFamily.

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

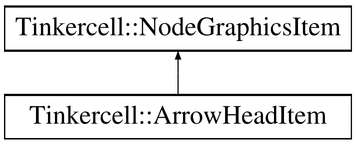
- ItemFamily.h
- ItemFamily.cpp

6.55 Tinkercell::NodeGraphicsItem Class Reference

A simple figure made from one or more polygons. The class can be represented in an XML file.

```
#include <NodeGraphicsItem.h>
```

Inheritance diagram for Tinkercell::NodeGraphicsItem:



Classes

- class ControlPoint

  a control point with a pointer to a NodeGraphicsItem

- class Shape

  A closed polygon path made from arcs, lines, and beziers.

Public Types

- enum ShapeType { **arc**, **line**, **bezier**, **rectangle** }

  arc, line, or beizier

- enum { **Type** = UserType + 4 }

  for enabling dynamic_cast

Public Member Functions

- virtual ItemHandle ∗ handle () const

  get the handle of this node

- virtual void setHandle (ItemHandle ∗)

  set the handle of this node

- NodeGraphicsItem (QGraphicsItem ∗parent=0)
- NodeGraphicsItem (const QString &filename, QGraphicsItem ∗parent=0)
- NodeGraphicsItem (const NodeGraphicsItem &copy)
- virtual NodeGraphicsItem & operator= (const NodeGraphicsItem &copy)
- virtual NodeGraphicsItem ∗ clone () const

  make a copy of this node item

- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option=new QStyleOption-GraphicsItem(), QWidget ∗widget=0) *paint method. Call's parent's paint after setting antialiasing to true*

- bool isValid () const

  checks that this is a valid drawable

- virtual void addControlPoint (ControlPoint ∗control)

  add a new control point

- virtual void addShape (Shape ∗shape)

  add a shape to the set of shapes

- virtual void removeControlPoint (ControlPoint ∗control)

  remove a control point

- virtual void removeShape (Shape ∗shape)

  add a shape to the set of shapes

- virtual void setBrush (const QBrush &newBrush)

  change fill color of all shapes

- virtual void setAlpha (int value)

  change alpha value for brush and pen of all shapes

- virtual void setPen (const QPen &newPen)

  change outline color of all shapes

- virtual void resetBrush ()

  change fill color of all shapes to the default brush

- virtual void resetPen ()

  change outline color of all shapes to default pen

- virtual QPolygonF polygon () const

  gets a polygon that represents this graphicsItem

- virtual QPainterPath shape () const

  gets a path that represents this graphicsItem

- virtual void refresh ()

  Updates the graphicsItem by re-initializing the vector of shapes Precondition: shapes.size $>$ 1 Postcondition: NA.

- virtual void normalize ()

  normalizes a node graphics item so that its center is 0,0 and width∗height is 10

- virtual void clear ()

  Clear all shapes and control points.

- virtual QRectF boundingRect () const

  bounding rect

- virtual ∼NodeGraphicsItem ()

  Destructor: deletes all shapes and control points.
- virtual QList< Tinkercell::ControlPoint ∗ > allControlPoints () const

  all the control points that are used in this figure
- virtual void adjustBoundaryControlPoints ()

  reset of control points that control the bounding box of this figure
- virtual void adjustToBoundaryControlPoints ()

  set boundary to match control points that control the bounding box of this figure
- virtual void setBoundingRect (const QPointF &, const QPointF &)

  set the top left and bottom right corners of this node item
- virtual void setBoundingBoxVisible (bool visible=true, bool controlPoints=true)

  show or hide the bounding box of this figure
- void showBoundingBox (bool controlPoints=true)

  show the bounding box of this figure. same as setBoundingBoxVisible(true)
- void hideBoundingBox (bool controlPoints=true)

  hide the bounding box of this figure. same as setBoundingBoxVisible(false)
- virtual int type () const

  for enabling dynamic_cast
- virtual QList< ConnectionGraphicsItem ∗ > connections () const

  get all the connection items linked to this node
- virtual QList< NodeGraphicsItem ∗ > connectedNodes () const

  get all the nodes connected to all the connections
- virtual QList< ConnectionGraphicsItem ∗ > connectionsWithArrows () const

  get all the connection items that have an arrow associated with this node
- virtual QList< ConnectionGraphicsItem ∗ > connectionsWithoutArrows () const

  get all the connection items that do NOT have an arrow associated with this node
- virtual QList< ConnectionGraphicsItem ∗ > connectionsDisconnected () const

  get all the connection items where this node is disconnected from the main connection, e.g. modifiers
- virtual QList< QGraphicsItem ∗ > connectionsAsGraphicsItems () const

  get all the connection items linked to this node as a list of qgraphicsitems
- virtual QList< NodeGraphicsItem ∗ > nodesAdjacent () const

  get all the node items that are bordering this node
- virtual QList< NodeGraphicsItem ∗ > nodesUpstream () const

get all the node items that are connected to this node directly or indirectly. only nodes that are coming in are selected (with arrows) Note: if the node contains more than one connections with arrows, this list returns one downstream path from the possible paths

- virtual QList< NodeGraphicsItem ∗ > nodesDownstream () const

  get all the node items that are connected to this node directly or indirectly. only nodes that are going out are selected (without arrows) Note: if the node contains more than one connections without arrows, this list returns one downstream path from the possible paths

- virtual QList< NodeGraphicsItem ∗ > nodesToLeft () const

  nodes to the left of this node in sequence

- virtual QList< NodeGraphicsItem ∗ > nodesToRight () const

  nodes to the right of this node in sequence

- virtual QList< NodeGraphicsItem ∗ > nodesAbove () const

  nodes above of this node in sequence

- virtual QList< NodeGraphicsItem ∗ > nodesBelow () const

  nodes below of this node in sequence

- virtual Shape ∗ tallestShape () const

  get the shape with greatest height inside this group graphics item

- virtual Shape ∗ longestShape () const

  get the shape with greatest width inside this group graphics item

- virtual Shape ∗ leftMostShape () const

  get the shape with lowest x value nside this group graphics item

- virtual Shape ∗ rightMostShape () const

  get the shape with largest x value inside this group graphics item

- virtual Shape ∗ topMostShape () const

  get the shape with lowest y value nside this group graphics item

- virtual Shape ∗ bottomMostShape () const

  get the shape with largest y value inside this group graphics item

Static Public Member Functions

- static NodeGraphicsItem ∗ cast (QGraphicsItem ∗)

  cast a graphics item to a node graphics item using qgraphicsitem_cast

- static QList< NodeGraphicsItem ∗ > cast (const QList< QGraphicsItem ∗ > &)

  cast a list of graphics item to a list of node graphics items using qgraphicsitem_cast

- static NodeGraphicsItem ∗ topLevelNodeItem (QGraphicsItem ∗item, bool ignoreControlPoints=false)

  Gets the node item from one of its child items.

Public Attributes

- QString className

  for safe static casting

- QString name

  file where the graphics item is stored

- QSizeF defaultSize

  default size for this item

- QVector< Shape ∗ > shapes

  shapes that comprise this figure

- QVector< ControlPoint ∗ > controlPoints

  control points that control the shapes in this figure

- QVector< ControlPoint ∗ > boundaryControlPoints

  set of control points that control the bounding box of this figure

- QString groupID

  for identifying which scene this item belongs in

Static Public Attributes

- static const QString CLASSNAME = QString("NodeGraphicsItem")

  for safe static casting

- static const int numShapeTypes = 4

  number of different type of shapes available

Protected Member Functions

- virtual void recomputeBoundingRect ()

  reconstruct bounding rect

- virtual qreal getPenWidthForBoundingRect ()

  get pen width based on bounding rect

Protected Attributes

- QRectF boundingRectangle

  bounding rectangle for the whole group

- ItemHandle ∗ itemHandle

  Tinkercell object that this drawable belongs in.

- QGraphicsRectItem ∗ boundingBoxItem

    the bounding box of this figure

6.55.1 Detailed Description

A simple figure made from one or more polygons. The class can be represented in an XML file.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 Tinkercell::NodeGraphicsItem::NodeGraphicsItem (QGraphicsItem ∗ *parent* = 0)

Constructor: does nothing

6.55.2.2 Tinkercell::NodeGraphicsItem::NodeGraphicsItem (const QString & *filename*, QGraphicsItem ∗ *parent* = 0)

Construct from file using NodeGraphicsReader

6.55.2.3 Tinkercell::NodeGraphicsItem::NodeGraphicsItem (const NodeGraphicsItem & *copy*)

Copy Constructor

Copy Constructor: deep copy of all pointers

copy handle

Copy control points and shapes

6.55.2.4 Tinkercell::NodeGraphicsItem::∼NodeGraphicsItem () `[virtual]`

Destructor: deletes all shapes and control points.

Destructor: deletes all shapes and control points

6.55.3 Member Function Documentation

6.55.3.1 QList< NodeGraphicsItem ∗ > Tinkercell::NodeGraphicsItem::cast (const QList< QGraphicsItem ∗ > & *list*) `[static]`

cast a list of graphics item to a list of node graphics items using qgraphicsitem_cast

Parameters

QList<QGraphicsItem∗>	graphics items

Returns

QList<NodeGraphicsItem∗> can be empty if no cast is invalid

6.55.3.2 NodeGraphicsItem ∗ Tinkercell::NodeGraphicsItem::cast (QGraphicsItem ∗ *q*) `[static]`

cast a graphics item to a node graphics item using qgraphicsitem_cast

Parameters

QGraphicsItem∗ graphics item

Returns

NodeGraphicsItem∗ can be 0 if the cast is invalid

Reimplemented in Tinkercell::ArrowHeadItem.

6.55.3.3 void Tinkercell::NodeGraphicsItem::clear () `[virtual]`

Clear all shapes and control points.

Parameters

void

Returns

void

6.55.3.4 NodeGraphicsItem ∗ Tinkercell::NodeGraphicsItem::clone () const `[virtual]`

make a copy of this node item

make a copy of this item

Reimplemented in Tinkercell::ArrowHeadItem.

6.55.3.5 QList< NodeGraphicsItem ∗ > Tinkercell::NodeGraphicsItem::connectedNodes () const `[virtual]`

get all the nodes connected to all the connections

get all the connected nodes

6.55.3.6 QList< QGraphicsItem ∗ > Tinkercell::NodeGraphicsItem::connectionsAsGraphicsItems () const `[virtual]`

get all the connection items linked to this node as a list of qgraphicsitems

get all the connection items linked to this node

6.55.3.7 QList< ConnectionGraphicsItem ∗ > Tinkercell::NodeGraphicsItem::connectionsDisconnected () const `[virtual]`

get all the connection items where this node is disconnected from the main connection, e.g. modifiers

get all the connection items linked to this node

6.55.3.8 QList< ConnectionGraphicsItem ∗ > Tinkercell::NodeGraphicsItem::connectionsWithArrows () const `[virtual]`

get all the connection items that have an arrow associated with this node

get all the connection items linked to this node

6.55.3.9 QList< ConnectionGraphicsItem ∗ > Tinkercell::NodeGraphicsItem::connectionsWithoutArrows () const `[virtual]`

get all the connection items that do NOT have an arrow associated with this node

get all the connection items linked to this node

6.55.3.10 void Tinkercell::NodeGraphicsItem::normalize () `[virtual]`

normalizes a node graphics item so that its center is 0,0 and width∗height is 10

Parameters

node item pointer to normalize

Returns

void

Parameters

NodeImage pointer to normalize

Returns

void

6.55.3.11 NodeGraphicsItem & Tinkercell::NodeGraphicsItem::operator= (const NodeGraphicsItem & *copy*) `[virtual]`

basically does the same as copy constructor

operator =: deep copy of all pointers

Copy control points and shapes

6.55.3.12 QPolygonF Tinkercell::NodeGraphicsItem::polygon () const `[virtual]`

gets a polygon that represents this graphicsItem

gets a polygon that is constructed by uniting all the shapes

6.55.3.13 void Tinkercell::NodeGraphicsItem::refresh () `[virtual]`

Updates the graphicsItem by re-initializing the vector of shapes Precondition: shapes.size > 1 Postcondition: NA.

Parameters

void

Returns

void

6.55.3.14 void Tinkercell::NodeGraphicsItem::resetBrush () `[virtual]`

change fill color of all shapes to the default brush

change fill color of all shapes to default

6.55.3.15 void Tinkercell::NodeGraphicsItem::resetPen () `[virtual]`

change outline color of all shapes to default pen

change outline color of all shapes to default

6.55.3.16 void Tinkercell::NodeGraphicsItem::setAlpha (int *value*) `[virtual]`

change alpha value for brush and pen of all shapes

change alpha value for brush of all shapes

6.55.3.17 QPainterPath Tinkercell::NodeGraphicsItem::shape () const `[virtual]`

gets a path that represents this graphicsItem

gets a path that is constructed by uniting all the shape paths

6.55.3.18 NodeGraphicsItem ∗ Tinkercell::NodeGraphicsItem::topLevelNodeItem (QGraphicsItem ∗ *item*, bool *ignoreControlPoints* = `false`) `[static]`

Gets the node item from one of its child items.

gets the node graphics item from its child item

Parameters

QGraphicsItem∗ the target item

bool using true here will return the node item for a control point, otherwise control points are ignored

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

- NodeGraphicsItem.h
- NodeGraphicsItem.cpp

6.56 Tinkercell::NodeGraphicsReader Class Reference

An xml reader that reads a NodeGraphicsItem file.

`#include <NodeGraphicsReader.h>`

Classes

- struct **BrushStruct**

Public Member Functions

- bool readXml (NodeGraphicsItem ∗idrawable, const QString &fileName)

  Reads an NodeGraphicsItem from an XML file using the IO device provided.

- void readNodeGraphics (NodeGraphicsItem ∗idrawable, QIODevice ∗device)

  Reads an NodeGraphicsItem from an XML file using the IO device provided.

- QXmlStreamReader::TokenType readNext ()

  Reads up to the next start node.

6.56.1 Detailed Description

An xml reader that reads a NodeGraphicsItem file.

6.56.2 Member Function Documentation

6.56.2.1 QXmlStreamReader::TokenType Tinkercell::NodeGraphicsReader::readNext ()

Reads up to the next start node.

Returns

Token Typer

6.56.2.2 void Tinkercell::NodeGraphicsReader::readNodeGraphics (NodeGraphicsItem ∗ *node*, QIODevice ∗ *device*)

Reads an NodeGraphicsItem from an XML file using the IO device provided.

Reads an NodeGraphicsItem from an XML file using the IO device provided and adds the information to the provided NodeGraphicsItem.

Parameters

NodeGraphicsItem	pointer to write as XML
QIODevice	to use

Returns

NodeGraphicsItem pointer

Parameters

NodeGraphicsItem pointer that will be read into from XML
QIODevice to use

Returns

void

6.56.2.3 bool Tinkercell::NodeGraphicsReader::readXml (NodeGraphicsItem ∗ *node*, const QString & *fileName*)

Reads an NodeGraphicsItem from an XML file using the IO device provided.

Reads an NodeGraphicsItem from an XML file using the IO device provided and adds the information to the provided NodeGraphicsItem.

Parameters

NodeGraphicsItem pointer to write as XML
QIODevice to use

Returns

NodeGraphicsItem pointer

Parameters

NodeGraphicsItem pointer that will be read into from XML
QIODevice to use

Returns

void

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

- NodeGraphicsReader.h
- NodeGraphicsReader.cpp

6.57 Tinkercell::NodeGraphicsWriter Class Reference

An xml reader that reads a NodeGraphicsItem file.

`#include <NodeGraphicsWriter.h>`

Public Member Functions

- NodeGraphicsWriter ()

    default constructor

- bool writeXml (NodeGraphicsItem ∗idrawable, const QString &fileName, bool normalize=true)

    Writes an Node graphics item XML file with the document headers.

- bool writeXml (NodeGraphicsItem ∗idrawable, QIODevice ∗device, bool normalize=true)

    Writes an Node graphics item XML file with the document headers.

- bool writeNodeGraphics (NodeGraphicsItem ∗idrawable, QIODevice ∗device, bool normalize=false)

    Writes an NodeImage as an XML file using the IO device provided.

Static Public Member Functions

- static bool writeNodeGraphics (NodeGraphicsItem ∗idrawable, QXmlStreamWriter ∗, bool normalize=false)

    Writes an NodeImage as an XML file using the xml writer provided.

6.57.1 Detailed Description

An xml reader that reads a NodeGraphicsItem file.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 Tinkercell::NodeGraphicsWriter::NodeGraphicsWriter ()

default constructor

constructor. Sets autoformatting to true

6.57.3 Member Function Documentation

6.57.3.1 bool Tinkercell::NodeGraphicsWriter::writeNodeGraphics (NodeGraphicsItem ∗ *node*, QXmlStreamWriter ∗ *writer*, bool *normalize* = `false`) `[static]`

Writes an NodeImage as an XML file using the xml writer provided.

Writes an NodeImage as an XML file using the IO device provided.

Parameters

NodeImage pointer to write as XML
XML writer to use

Returns

void

MainWindow::invalidPointers.contains(node->shapes[i]) &&

MainWindow::invalidPointers.contains(node->shapes[i]) &&

6.57.3.2 bool Tinkercell::NodeGraphicsWriter::writeNodeGraphics (NodeGraphicsItem ∗ *node*, QIODevice ∗ *device*, bool *normalize* = `false`)

Writes an NodeImage as an XML file using the IO device provided.

Writes an NodeGraphicsItem as an XML file using the IO device provided.

Parameters

NodeImage pointer to write as XML
QIODevice to use

Returns

void

Parameters

NodeGraphicsItem pointer to write as XML
QIODevice to use

Returns

void

6.57.3.3 bool Tinkercell::NodeGraphicsWriter::writeXml (NodeGraphicsItem ∗ *node*, QIODevice ∗ *device*, bool *normalize* = `true`)

Writes an Node graphics item XML file with the document headers.

Writes an NodeGraphicsItem XML file with the document headers.

Parameters

NodeImage pointer to write as XML
QIODevice to use

Returns

void

Parameters

NodeGraphicsItem pointer to write as XML

QIODevice to use

Returns

void

6.57.3.4 bool Tinkercell::NodeGraphicsWriter::writeXml (NodeGraphicsItem ∗ *node*, const QString & *fileName*, bool *normalize* = `true`)

Writes an Node graphics item XML file with the document headers.

Writes an NodeGraphicsItem XML file with the document headers.

Parameters

NodeImage pointer to write as XML

QIODevice to use

Returns

void

Parameters

NodeGraphicsItem pointer to write as XML

QIODevice to use

Returns

void

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

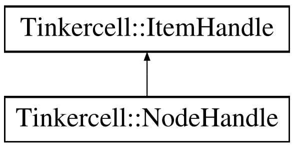
- NodeGraphicsWriter.h
- NodeGraphicsWriter.cpp

6.58 Tinkercell::NodeHandle Class Reference

The handles are used to bring together data and graphics items. Node Handle contains pointers to all the graphics items that belong to it, the tools that apply to this item, the data for this item, and the family that it belongs with.

```
#include <ItemHandle.h>
```

Inheritance diagram for Tinkercell::NodeHandle:



Public Member Functions

- virtual QList< ConnectionHandle ∗ > connections () const

  funcion that returns all the connections from all the nodes in this handle

- NodeHandle (const QString &name=QString(), NodeFamily ∗nodeFamily=0)

  default constructor -- initialize everything

- NodeHandle (const NodeHandle &copy)

  copy constructor -- copies all the data (deep). graphic items are shallow copies

- virtual NodeHandle & operator= (const NodeHandle &)

  operator =

- NodeHandle (NodeFamily ∗nodeFamily, NodeGraphicsItem ∗item)

  constructor using initial family and graphics item

- NodeHandle (NodeFamily ∗nodeFamily, const QString &name=QString())

  constructor using initial family and name

- virtual ItemHandle ∗ clone () const

  return a clone of this handle

- virtual ItemFamily ∗ family () const

  get the node family for this handle

- virtual void setFamily (ItemFamily ∗, bool useCommand=true)

  set the node family for this handle

Static Public Member Functions

- static NodeHandle ∗ cast (ItemHandle ∗)

  checks if the item handle is a node handle and casts it as a node item. Returns 0 if it is not a node item

- static QList< NodeHandle ∗ > cast (const QList< ItemHandle ∗ > &)

  checks if the item handles are node handles and casts then as node items. Returns QList<NodeHandle∗>

Public Attributes

- NodeFamily ∗ nodeFamily

  node family for this node handle

Static Public Attributes

- static const int TYPE = 1

  this number is used to identify when a handle is a node handle

6.58.1 Detailed Description

The handles are used to bring together data and graphics items. Node Handle contains pointers to all the graphics items that belong to it, the tools that apply to this item, the data for this item, and the family that it belongs with.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 Tinkercell::NodeHandle::NodeHandle (NodeFamily ∗ *nodeFamily*, NodeGraphicsItem ∗ *item*)

constructor using initial family and graphics item

Parameters

nodeFamily*∗* node family

NodeGraphicsItem*∗* graphics item

6.58.2.2 Tinkercell::NodeHandle::NodeHandle (NodeFamily ∗ *nodeFamily*, const QString & *name* = `QString()`)

constructor using initial family and name

Parameters

nodeFamily*∗* node family

QString name

6.58.3 Member Function Documentation

6.58.3.1 QList< NodeHandle ∗ > Tinkercell::NodeHandle::cast (const QList< ItemHandle ∗ > & *items*) `[static]`

checks if the item handles are node handles and casts then as node items. Returns QList<NodeHandle∗>

Parameters

Returns QList<ItemHandle∗> items

6.58.3.2 NodeHandle ∗ Tinkercell::NodeHandle::cast (ItemHandle ∗ *item*) `[static]`

checks if the item handle is a node handle and casts it as a node item. Returns 0 if it is not a node item

Parameters

ItemHandle∗ item

6.58.3.3 ItemHandle ∗ Tinkercell::NodeHandle::clone () const `[virtual]`

return a clone of this handle

Returns

ItemFamily∗ node handle as item handle

Reimplemented from Tinkercell::ItemHandle.

6.58.3.4 QList< ConnectionHandle ∗ > Tinkercell::NodeHandle::connections () const `[virtual]`

funcion that returns all the connections from all the nodes in this handle

Returns

QList<ConnectionHandle∗> list of connection handles

6.58.3.5 ItemFamily ∗ Tinkercell::NodeHandle::family () const `[virtual]`

get the node family for this handle

Returns

ItemFamily∗ node family as item family

Reimplemented from Tinkercell::ItemHandle.

6.58.3.6 void Tinkercell::NodeHandle::setFamily (ItemFamily ∗ *p*, bool *useCommand* = `true`) `[virtual]`

set the node family for this handle

Parameters

***NodeFamily*∗**	node family

Reimplemented from Tinkercell::ItemHandle.

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

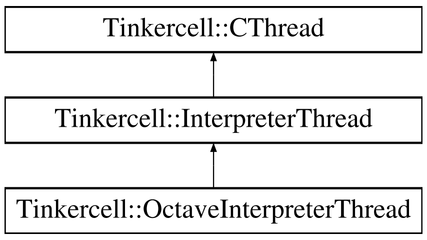
- ItemHandle.h
- ItemHandle.cpp

6.59 Tinkercell::OctaveInterpreterThread Class Reference

This class is used to embed an octave interpreter inside a TinkerCell application. The C library responsible for embedding octave is called runOctave.cpp and is located inside the octave folder. The octave interpreter uses two libraries -- one for embedding octave in TinkerCell and another for extending Octave with the TinkerCell C API.

```
#include <OctaveInterpreterThread.h>
```

Inheritance diagram for Tinkercell::OctaveInterpreterThread:


Tinkercell::CThread
Tinkercell::InterpreterThread
Tinkercell::OctaveInterpreterThread

Public Slots

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **toolLoaded** (Tool ∗)

Public Member Functions

- OctaveInterpreterThread (const QString &, const QString &, MainWindow ∗main)

  initialize the thread that will embed and extend octave. The embed library is ASSUMED to be named tinkercell.oct

- virtual void setCPointers ()

  requests main window to load all the C pointers for the C API inside the embedded library

Static Public Attributes

- static QString OCTAVE_FOLDER

  the folder where tinkercell will look for octave files, defaults to /octave

Protected Member Functions

- virtual void run ()

  the main function that runs one of the specified functions

Protected Attributes

- execFunc **f**
- bool **addpathDone**
- QLibrary ∗ swigLib

    library with all the C API functions

6.59.1 Detailed Description

This class is used to embed an octave interpreter inside a TinkerCell application. The C library responsible for embedding octave is called runOctave.cpp and is located inside the octave folder. The octave interpreter uses two libraries -- one for embedding octave in TinkerCell and another for extending Octave with the TinkerCell C API.

See also

PythonInterpreterThread

6.59.2 Constructor & Destructor Documentation

6.59.2.1 Tinkercell::OctaveInterpreterThread::OctaveInterpreterThread (const QString & *octname*, const QString & *dllname*, MainWindow ∗ *main*)

initialize the thread that will embed and extend octave. The embed library is ASSUMED to be named tinkercell.oct

Parameters

QString	folder where the two octave libraries are located
QString	name of the octave embed library

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

- OctaveInterpreterThread.h
- OctaveInterpreterThread.cpp

6.60 Tinkercell::Plot3DWidget::Plot Class Reference

Public Member Functions

- void **setColor** ()

Public Attributes

- QString **title**
- double **minZ**
- double **maxZ**
- QColor **minColor**
- QColor **maxColor**

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

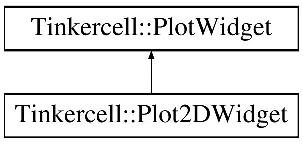
- Plot3DWidget.h
- Plot3DWidget.cpp

6.61 Tinkercell::Plot2DWidget Class Reference

A widget containing a data plot, legend and options.

`#include <Plot2DWidget.h>`

Inheritance diagram for Tinkercell::Plot2DWidget:



Public Slots

- void exportData (const QString &, const QString &)

  export data is some format

- void **logX** (bool)
- void **logY** (bool)
- void **logAxis** (int, bool)
- void **setTitle** ()
- void **setXLabel** ()
- void **setYLabel** ()
- void **setTitle** (const QString &)
- void **setXLabel** (const QString &)
- void **setYLabel** (const QString &)

Public Member Functions

- **Plot2DWidget** (PlotTool ∗parent=0)
- virtual DataTable< qreal > ∗ data ()

  get the data inside this plot

- virtual bool canAppendData () const

  indicates whether or not this plot widget is capable of plotting one graph on top of another

- virtual void appendData (const DataTable< qreal > &)

  append more data to the currently existing plot

- virtual void **plot** (const DataTable< qreal > &matrix, const QString &title, int x=0)
- virtual void updateData (const DataTable< qreal > &)

  update data for the current plot

6.61.1 Detailed Description

A widget containing a data plot, legend and options.

6.61.2 Member Function Documentation

6.61.2.1 void Tinkercell::Plot2DWidget::exportData (const QString & *type*, const QString & *file*) `[virtual, slot]`

export data is some format

Parameters

QString	format

Reimplemented from Tinkercell::PlotWidget.

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

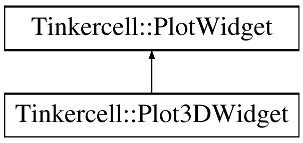
- Plot2DWidget.h
- Plot2DWidget.cpp

6.62 Tinkercell::Plot3DWidget Class Reference

A widget containing a data plot, legend and options.

`#include <Plot3DWidget.h>`

Inheritance diagram for Tinkercell::Plot3DWidget:



Classes

- class DataFunction
- class Plot
- class StandardColor

Public Slots

- void exportData (const QString &, const QString &)

  export data is some format

Public Member Functions

- **Plot3DWidget** (PlotTool ∗parent=0)
- DataTable< qreal > ∗ data ()

  get the data inside this plot

- void updateData (const DataTable< qreal > &)

  update data for the current plot

- void **surface** (const DataTable< qreal > &matrix, const QString &title=QString())

Static Public Attributes

- static QColor **DEFAULT_LOW_COLOR**
- static QColor **DEFAULT_HIGH_COLOR**

Static Protected Member Functions

- static double ∗∗ **tableToArray** (const DataTable< qreal > &)

Protected Attributes

- DataTable< qreal > **dataTable**
- Plot * **surfacePlot**
- DataFunction * **function**

6.62.1 Detailed Description

A widget containing a data plot, legend and options.

6.62.2 Member Function Documentation

6.62.2.1 void Tinkercell::Plot3DWidget::exportData (const QString & *type*, const QString & *file*) `[virtual, slot]`

export data is some format

Parameters

QString format

Reimplemented from Tinkercell::PlotWidget.

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

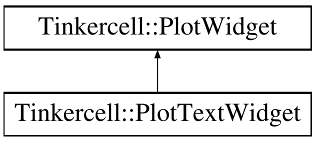
- Plot3DWidget.h
- Plot3DWidget.cpp

6.63 Tinkercell::PlotTextWidget Class Reference

A PlotWidget used to display tab delimited text.

`#include <PlotTextWidget.h>`

Inheritance diagram for Tinkercell::PlotTextWidget:



Public Member Functions

- PlotTextWidget (const DataTable< qreal > &, PlotTool ∗parent=0, const QString &text=QString())
  constructor with data table and plot tool as parent
- virtual DataTable< qreal > ∗ data ()
  get the data

Protected Member Functions

- virtual void keyPressEvent (QKeyEvent ∗event)
  key events

6.63.1 Detailed Description

A PlotWidget used to display tab delimited text.

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

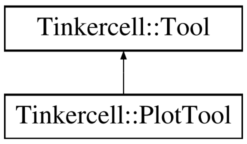
- PlotTextWidget.h
- PlotTextWidget.cpp

6.64 Tinkercell::PlotTool Class Reference

A docking widget that can contains one or more PlotWidget instances. Each PlotWidget can either be a text output, 2D graph, or 3D graph. Alternatively, the PlotTool can use an separate Gnuplot window to generate plots.

```
#include <PlotTool.h>
```

Inheritance diagram for Tinkercell::PlotTool:



Public Types

- enum PlotType {

  Plot2D, **SurfacePlot**, **HistogramPlot**, **ScatterPlot**,

  Text }

  available plot types

Public Slots

- void hold (bool b=true)

  hold current plot (don't close it)

- void overplot (bool b=true)

  plot on top of current plot (if the feature is available for current plot)

- void plot (const DataTable< qreal > &, const QString &title, int xaxis=0, int all=0, PlotType type=Plot2D)

  graph the given data with headers

- void surfacePlot (const DataTable< qreal > &matrix, const QString &title)

  surface plot of the given data

- void addExportOption (const QIcon &, const QString &, const QString &toolTip=QString())

  add export option. This will add a new button to the set of export options. When user selects this option, the exportData method in the current PlotWidget will be invoked

- void exportData (const QString &)

  export data in the given format

Signals

- void plotDataTable (DataTable< qreal > &m, int x, const QString &title, int all)

  plot a 2D graph

- void plotDataTable3D (DataTable< qreal > &m, const QString &title)

  plot a 3D graph

- void plotHist (DataTable< qreal > &m, double bins, const QString &title)

  plot a histogram

- void plotErrorbars (DataTable< qreal > &m, int x, const QString &title)

  plot a 2D graph with error bars, where every alternating column are the errors

- void plotMultiplot (int rows, int columns)

  enable multiple plots (grid)

- void plotScatterplot (DataTable< qreal > &m, const QString &title)

  make a scatterplot

- void gnuplot (const QString &script)

  send a script to gnuplot

Public Member Functions

- PlotTool ()

  default constructor

- virtual QSize sizeHint () const

  default size of this widget

- virtual bool setMainWindow (MainWindow ∗)

  set Tinkercell main window

- virtual void setVisible (bool visible)

  make this widget visible and on top

- virtual void addWidget (PlotWidget ∗)

  add a new plot to the window

- virtual void setStatusBarMessage (const QString &)

  show message at the bottom

- virtual QDockWidget ∗ addDockWidget (const QString &title, QWidget ∗widget, Qt::DockWidgetArea area=Qt::BottomDockWidgetArea)

  add a dock widget to the plot area

Static Public Member Functions

- static void pruneDataTable (DataTable< qreal > &table, int &xaxis, MainWindow ∗main)
  remove all items in the data table that are not visible in any scene

Protected Member Functions

- virtual void **keyPressEvent** (QKeyEvent ∗event)
- virtual void **mouseMoveEvent** (QMouseEvent ∗event)

Friends

- class **PlotWidget**

6.64.1 Detailed Description

A docking widget that can contains one or more PlotWidget instances. Each PlotWidget can either be a text output, 2D graph, or 3D graph. Alternatively, the PlotTool can use an separate Gnuplot window to generate plots.

6.64.2 Member Function Documentation

6.64.2.1 void Tinkercell::PlotTool::addExportOption (const QIcon & *icon*, const QString & *type*, const QString & *toolTip* = `QString()`) `[slot]`

add export option. This will add a new button to the set of export options. When user selects this option, the exportData method in the current PlotWidget will be invoked

Parameters

QIcon	icon for the export opion
QString	name of the export option

6.64.2.2 void Tinkercell::PlotTool::exportData (const QString & *type*) `[slot]`

export data in the given format

Parameters

QString	format

6.64.2.3 void Tinkercell::PlotTool::gnuplot (const QString & *script*) `[signal]`

send a script to gnuplot

Parameters

QString	gnuplot script

6.64.2.4 void Tinkercell::PlotTool::plot (const DataTable< qreal > & *matrix*, const QString & *title*, int *xaxis* = `0`, int *all* = `0`, PlotTool::PlotType *type* = `Plot2D`) `[slot]`

graph the given data with headers

Parameters

DataTable<qreal> table

QString title

QString column in the table that will be used as x-axis

int 0 or 1, indicating whether to plot only those items that are visible on the screen

6.64.2.5 void Tinkercell::PlotTool::plotDataTable (DataTable< qreal > & *m*, int *x*, const QString & *title*, int *all*) `[signal]`

plot a 2D graph

Parameters

NumericalDataTable data

int column for the x-axis

QString title

int(bool) whether or not to graph all the columns or just the handles that exist in the network

6.64.2.6 void Tinkercell::PlotTool::plotDataTable3D (DataTable< qreal > & *m*, const QString & *title*) `[signal]`

plot a 3D graph

Parameters

NumericalDataTable data with 3 columns

QString title

6.64.2.7 void Tinkercell::PlotTool::plotErrorbars (DataTable< qreal > & *m*, int *x*, const QString & *title*) `[signal]`

plot a 2D graph with error bars, where every alternating column are the errors

Parameters

NumericalDataTable data

int index of x-axis

QString title

6.64.2.8 void Tinkercell::PlotTool::plotHist (DataTable< qreal > & *m*, double *bins*, const QString & *title*) `[signal]`

plot a histogram

Parameters

NumericalDataTable data

int number of bins

QString title

6.64.2.9 void Tinkercell::PlotTool::plotMultiplot (int *rows*, int *columns*) `[signal]`

enable multiple plots (grid)

Parameters

int number of rows of plots

int number of columns of plots

6.64.2.10 void Tinkercell::PlotTool::plotScatterplot (DataTable< qreal > & *m*, const QString & *title*) `[signal]`

make a scatterplot

Parameters

NumericalDataTable data

QString title

6.64.2.11 void Tinkercell::PlotTool::surfacePlot (const DataTable< qreal > & *matrix*, const QString & *title*) `[slot]`

surface plot of the given data

Parameters

DataTable<qreal> table where value(x,y) is the z value

QString title

int 0 or 1, indicating whether to plot only those items that are visible on the screen

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

- PlotTool.h
- PlotTool.cpp

6.65 Tinkercell::PlotTool_FtoS Class Reference

Signals

- void **plotDataTable** (QSemaphore ∗, DataTable< qreal > &m, int x, const QString &title, int all)
- void **plotDataTable3D** (QSemaphore ∗, DataTable< qreal > &m, const QString &title)
- void **plotHist** (QSemaphore ∗, DataTable< qreal > &m, double bins, const QString &title)
- void **plotErrorbars** (QSemaphore ∗, DataTable< qreal > &m, int x, const QString &title)
- void **plotMultiplot** (QSemaphore ∗, int x, int y)
- void **getDataTable** (QSemaphore ∗, DataTable< qreal > ∗, int index)
- void **plotScatter** (QSemaphore ∗, DataTable< qreal > &, const QString &title)
- void **gnuplot** (QSemaphore ∗, const QString &script)
- void **savePlotImage** (QSemaphore ∗, const QString &filename)

Friends

- class **PlotTool**

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

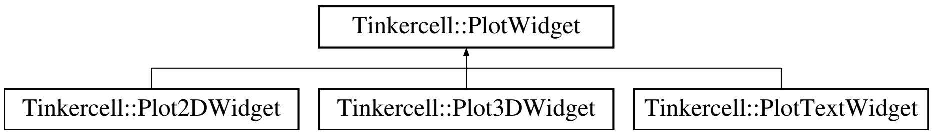
- PlotTool.h
- PlotTool.cpp

6.66 Tinkercell::PlotWidget Class Reference

A widget containing a data plot, legend and options. This class does not perform any plotting. This class serves as a template for other widgets that perform the plotting.

```
#include <PlotWidget.h>
```

Inheritance diagram for Tinkercell::PlotWidget:



Public Slots

- virtual void exportData (const QString &, const QString &file)

  export data is some format

Public Member Functions

- PlotWidget (PlotTool ∗parent=0)

  constructor with plot tool as parent

- PlotWidget (const DataTable< qreal > &, PlotTool ∗parent=0)

  constructor with plot tool as parent

- virtual DataTable< qreal > ∗ data ()

  get the data inside this plot

- virtual bool canAppendData () const

  indicates whether or not this plot widget is capable of plotting one graph on top of another

- virtual void appendData (const DataTable< qreal > &)

  append more data to the currently existing plot

- virtual void updateData (const DataTable< qreal > &)

  update data for the current plot

- virtual QString dataToString (const QString &delim=QString("\t"))

  get the data inside this plot as teb-delimited text

Public Attributes

- PlotTool::PlotType type

  used for identifying the plot type

- QToolBar toolBar

  tool bar containing all the options for this widget

Protected Member Functions

- virtual void keyPressEvent (QKeyEvent ∗event)

  key events

Protected Attributes

- PlotTool ∗ plotTool

  the plot tool that contains this widget

Friends

- class **PlotTool**

6.66.1 Detailed Description

A widget containing a data plot, legend and options. This class does not perform any plotting. This class serves as a template for other widgets that perform the plotting.

6.66.2 Member Function Documentation

6.66.2.1 void Tinkercell::PlotWidget::exportData (const QString & *type*, const QString & *file*) `[virtual, slot]`

export data is some format

Parameters

QString format

Reimplemented in Tinkercell::Plot2DWidget, and Tinkercell::Plot3DWidget.

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

- PlotWidget.h
- PlotWidget.cpp

6.67 Tinkercell::PopupListWidgetDelegate Class Reference

delegate used inside the SimpleInputWindow

```
#include <AbstractInputWindow.h>
```

Public Member Functions

- **PopupListWidgetDelegate** (QObject ∗parent=0)
- QWidget ∗ **createEditor** (QWidget ∗parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
- void **setEditorData** (QWidget ∗editor, const QModelIndex &index) const
- void **setModelData** (QWidget ∗editor, QAbstractItemModel ∗model, const QModelIndex &index) const
- void **updateEditorGeometry** (QWidget ∗editor, const QStyleOptionViewItem &option, const QModelIndex &index) const

Static Public Member Functions

- static QString **displayListWidget** (const QStringList &list, const QString &current=QString())

Public Attributes

- DataTable< QStringList > options

    options for the combo boxes. Uses line edits if empty. Uses check boxes if just one item

6.67.1 Detailed Description

delegate used inside the SimpleInputWindow

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

- AbstractInputWindow.h
- AbstractInputWindow.cpp

6.68 Tinkercell::PopupListWidgetDelegateDialog Class Reference

dialog for list widget

```
#include <AbstractInputWindow.h>
```

Public Slots

- void **acceptListWidget** (QListWidgetItem ∗)

6.68.1 Detailed Description

dialog for list widget

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

- AbstractInputWindow.h

6.69 Tinkercell::ProcessThread Class Reference

This class is used to run a process (command + args) as a separate thread as a separate thread.

`#include <CThread.h>`

Public Member Functions

- ProcessThread (const QString &, const QString &, MainWindow ∗main)

  constructor -- used to initialize the main window, the command name and the args for the command

- virtual QString output () const

  get the results (output stream) from the process

- virtual QString errors () const

  get the errors (error stream) from the process

- virtual ∼ProcessThread ()

  destructor -- free the library that this thread loaded

Static Public Member Functions

- static QWidget ∗ dialog (MainWindow ∗, ProcessThread ∗, const QString &text=QString("Process"), QIcon icon=QIcon())

  creates a dialog that shows the name of the running thread and a button for terminating the thread

Protected Slots

- virtual void stopProcess ()

  unload the library (if loaded) and delete it

Protected Member Functions

- virtual void run ()

  initializes the function pointers through the main window and then runs the target function

Protected Attributes

- QString exe

  the name of the executable

- QString args

  the arguments

- QString outputStream

  the output from the process

- QString errStream

  the error from the process

- MainWindow ∗ mainWindow

  Tinkercell's main window.

- QProcess process

  Tinkercell's main window.

6.69.1 Detailed Description

This class is used to run a process (command + args) as a separate thread as a separate thread.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 Tinkercell::ProcessThread::ProcessThread (const QString & *exe*, const QString & *args*, MainWindow ∗ *main*)

constructor -- used to initialize the main window, the command name and the args for the command

Parameters

QString command

QString arguments

MainWindow main window

6.69.3 Member Function Documentation

6.69.3.1 QWidget ∗ Tinkercell::ProcessThread::dialog (MainWindow ∗ *mainWindow*, ProcessThread ∗ *newThread*, const QString & *text* = `QString("Process")`, QIcon *icon* = `QIcon()`) `[static]`

creates a dialog that shows the name of the running thread and a button for terminating the thread

Parameters

MainWindow main window

ProcessThread

QString text to display

QIcon icon to display

6.69.3.2 QString Tinkercell::ProcessThread::errors () const `[virtual]`

get the errors (error stream) from the process

Returns

QString output

6.69.3.3 QString Tinkercell::ProcessThread::output () const `[virtual]`

get the results (output stream) from the process

Returns

QString output

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

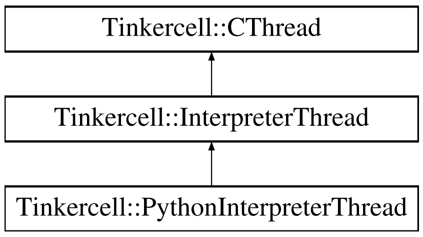
- CThread.h
- CThread.cpp

6.70 Tinkercell::PythonInterpreterThread Class Reference

This class is used to embed an python interpreter inside a TinkerCell application. The C library responsible for embedding python is called runpy.c and is located inside the python/ folder.

```
#include <PythonInterpreterThread.h>
```

Inheritance diagram for Tinkercell::PythonInterpreterThread:


Tinkercell::CThread
Tinkercell::InterpreterThread
Tinkercell::PythonInterpreterThread

Public Slots

- virtual void **initialize** ()
- virtual void **finalize** ()

Public Member Functions

- **PythonInterpreterThread** (const QString &, MainWindow ∗main)

Static Public Attributes

- static QString PYTHON_FOLDER

  the folder where tinkercell will look for python files, defaults to /python

Protected Member Functions

- virtual void run ()

  the main function that runs one of the specified functions

Protected Attributes

- execFunc **f**
- bool **addpathDone**

6.70.1 Detailed Description

This class is used to embed an python interpreter inside a TinkerCell application. The C library responsible for embedding python is called runpy.c and is located inside the python/ folder.

See also

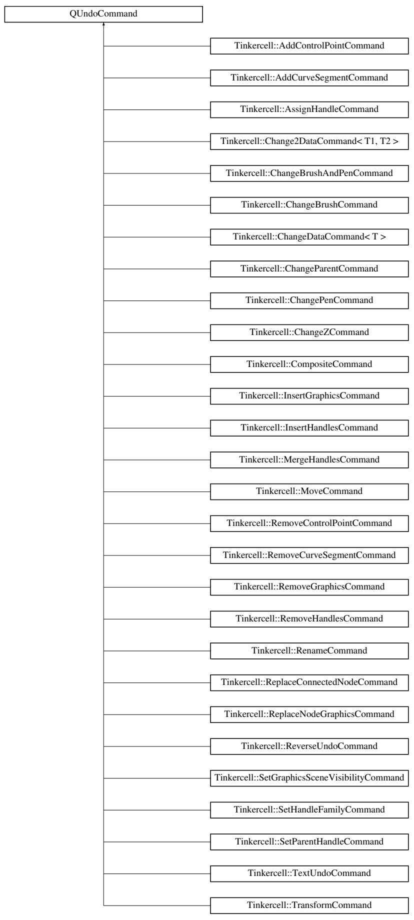
InterpreterThread

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

- PythonInterpreterThread.h
- PythonInterpreterThread.cpp

6.71 QUndoCommand Class Reference

Inheritance diagram for QUndoCommand:


QUndoCommand
Tinkercell::AddControlPointCommand
Tinkercell::AddCurveSegmentCommand
Tinkercell::AssignHandleCommand
Tinkercell::Change2DataCommand< T1, T2 >
Tinkercell::ChangeBrushAndPenCommand
Tinkercell::ChangeBrushCommand
Tinkercell::ChangeDataCommand< T >
Tinkercell::ChangeParentCommand
Tinkercell::ChangePenCommand
Tinkercell::ChangeZCommand
Tinkercell::CompositeCommand
Tinkercell::InsertGraphicsCommand
Tinkercell::InsertHandlesCommand
Tinkercell::MergeHandlesCommand
Tinkercell::MoveCommand
Tinkercell::RemoveControlPointCommand
Tinkercell::RemoveCurveSegmentCommand
Tinkercell::RemoveGraphicsCommand
Tinkercell::RemoveHandlesCommand
Tinkercell::RenameCommand
Tinkercell::ReplaceConnectedNodeCommand
Tinkercell::ReplaceNodeGraphicsCommand
Tinkercell::ReverseUndoCommand
Tinkercell::SetGraphicsSceneVisibilityCommand
Tinkercell::SetHandleFamilyCommand
Tinkercell::SetParentHandleCommand
Tinkercell::TextUndoCommand
Tinkercell::TransformCommand

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

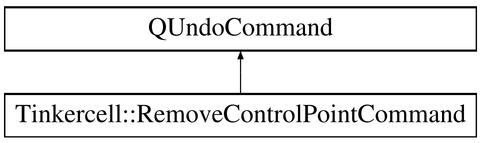
- DataTable.h

6.72 Tinkercell::RemoveControlPointCommand Class Reference

A command that removed control points. Allows undo and redo.

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::RemoveControlPointCommand:



Public Member Functions

- RemoveControlPointCommand (const QString &name, GraphicsScene ∗scene, ConnectionGraphicsItem::ControlPoint ∗item)

    constructor that makes the command. If added to history stack, also does redo

- RemoveControlPointCommand (const QString &name, GraphicsScene ∗scene, QList< ConnectionGraphicsItem::ControlPoint ∗ > items)

    constructor that makes the command. If added to history stack, also does redo

- void redo ()

    Remove new control points. Control points were set in the constructor.

- void undo ()

    Add new control points. Control points were set in the constructor.

Public Attributes

- QList< ConnectionGraphicsItem::ControlPoint ∗ > graphicsItems

    control points that were added

- GraphicsScene ∗ graphicsScene

    graphics scene to which control points were added

- QList< int > listK1

    the poisition(s) at which the control points were added

- QList< int > **listK2**

6.72.1 Detailed Description

A command that removed control points. Allows undo and redo.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 Tinkercell::RemoveControlPointCommand::RemoveControlPointCommand (const QString & *name*, GraphicsScene ∗ *scene*, ConnectionGraphicsItem::ControlPoint ∗ *item*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.72.2.2 Tinkercell::RemoveControlPointCommand::RemoveControlPointCommand (const QString & *name*, GraphicsScene ∗ *scene*, QList< ConnectionGraphicsItem::ControlPoint ∗ > *items*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.72.3 Member Function Documentation

6.72.3.1 void Tinkercell::RemoveControlPointCommand::redo ()

Remove new control points. Control points were set in the constructor.

Parameters

void

Returns

void

6.72.3.2 void Tinkercell::RemoveControlPointCommand::undo ()

Add new control points. Control points were set in the constructor.

Parameters

void

Returns

void

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

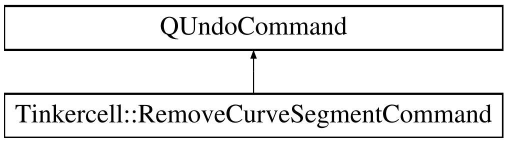
- UndoCommands.h
- UndoCommands.cpp

6.73 Tinkercell::RemoveCurveSegmentCommand Class Reference

A command that removed control points. Allows undo and redo.

```
#include <UndoCommands.h>
```

Inheritance diagram for Tinkercell::RemoveCurveSegmentCommand:



Public Member Functions

- RemoveCurveSegmentCommand (const QString &name, GraphicsScene ∗scene, ConnectionGraphicsItem::ControlPoint ∗item)

  constructor that makes the command. If added to history stack, also does redo

- RemoveCurveSegmentCommand (const QString &name, GraphicsScene ∗scene, ConnectionGraphicsItem ∗connection, QList< ConnectionGraphicsItem::ControlPoint ∗ > items)

  constructor that makes the command. If added to history stack, also does redo

- void redo ()

  Remove new control points. Control points were set in the constructor.

- void undo ()

  Add new control points. Control points were set in the constructor.

Public Attributes

- QList< ConnectionGraphicsItem::CurveSegment > curveSegments

  vector of control points that were added

- GraphicsScene ∗ graphicsScene

  graphics scene from which control points were removed

- ConnectionGraphicsItem ∗ connectionItem

  graphics item from which control points were removed

- QList< QGraphicsItem ∗ > parentsAtStart

  the nodes belonging with the control point vectors

- QList< QGraphicsItem ∗ > **parentsAtEnd**

6.73.1 Detailed Description

A command that removed control points. Allows undo and redo.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 Tinkercell::RemoveCurveSegmentCommand::RemoveCurveSegmentCommand (const QString & *name*, GraphicsScene ∗ *scene*, ConnectionGraphicsItem::ControlPoint ∗ *item*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.73.2.2 Tinkercell::RemoveCurveSegmentCommand::RemoveCurveSegmentCommand (const QString & *name*, GraphicsScene ∗ *scene*, ConnectionGraphicsItem ∗ *connection*, QList< ConnectionGraphicsItem::ControlPoint ∗ > *items*)

constructor that makes the command. If added to history stack, also does redo

Parameters

name

graphics scene

control point(s) that have been added

Returns

void

6.73.3 Member Function Documentation

6.73.3.1 void Tinkercell::RemoveCurveSegmentCommand::redo ()

Remove new control points. Control points were set in the constructor.

Parameters

void

Returns

void

6.73.3.2 void Tinkercell::RemoveCurveSegmentCommand::undo ()

Add new control points. Control points were set in the constructor.

Parameters

void

Returns

void

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

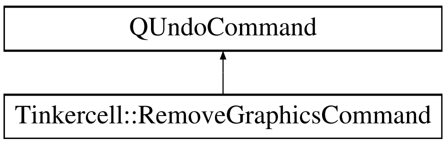
- UndoCommands.h
- UndoCommands.cpp

6.74 Tinkercell::RemoveGraphicsCommand Class Reference

this command performs an removal and allows redo/undo of that removal

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::RemoveGraphicsCommand:



Public Member Functions

- RemoveGraphicsCommand (const QString &name, QGraphicsItem ∗item, bool updataDataFields=true)

  constructor

- RemoveGraphicsCommand (const QString &name, const QList< QGraphicsItem ∗ > &items, bool updateDataFields=true)

  constructor

- void redo ()

  redo the change

- void undo ()

  undo the change

6.74.1 Detailed Description

this command performs an removal and allows redo/undo of that removal

6.74.2 Constructor & Destructor Documentation

6.74.2.1 Tinkercell::RemoveGraphicsCommand::RemoveGraphicsCommand (const QString & *name*, QGraphicsItem ∗ *item*, bool *updataDataFields* = `true`)

constructor

Parameters

QString name of command

GraphicsScene∗ where change happened

QGraphicsItem∗ item that is removed

bool update data of other items where removed items might occur (default=true)

6.74.2.2 Tinkercell::RemoveGraphicsCommand::RemoveGraphicsCommand (const QString & *name*, const QList< QGraphicsItem ∗ > & *items*, bool *updateDataFields* = `true`)

constructor

Parameters

QString name of command

GraphicsScene∗ where change happened

QList<QGraphicsItem∗>& items that are removed

bool update data of other items where removed items might occur (default=true)

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

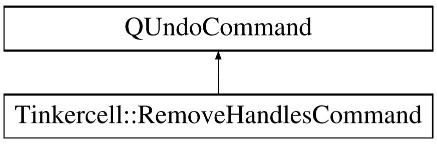
- UndoCommands.h
- UndoCommands.cpp

6.75 Tinkercell::RemoveHandlesCommand Class Reference

this command inserts new handles to a NetworkHandle

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::RemoveHandlesCommand:



Public Member Functions

- RemoveHandlesCommand (TextEditor ∗, const QList< ItemHandle ∗ > &, bool updateDataFields=true)

  constructor

- RemoveHandlesCommand (TextEditor ∗, ItemHandle ∗, bool updateDataFields=true)

  constructor

- void redo ()

  redo the change

- void undo ()

  undo the change

6.75.1 Detailed Description

this command inserts new handles to a NetworkHandle

6.75.2 Constructor & Destructor Documentation

6.75.2.1 Tinkercell::RemoveHandlesCommand::RemoveHandlesCommand (TextEditor ∗ *editor*, const QList< ItemHandle ∗ > & *list*, bool *updateDataFields* = `true`)

constructor

Parameters

TextEditor∗	window where items are deleted
QList<ItemHandle∗>	deleted items
bool	update data of other items where removed items might occur (default=true)

6.75.2.2 Tinkercell::RemoveHandlesCommand::RemoveHandlesCommand (TextEditor ∗ *editor*, ItemHandle ∗ *h*, bool *updateDataFields* = `true`)

constructor

Parameters

TextEditor∗ window where items are deleted

ItemHandle∗ deleted item

bool update data of other items where removed items might occur (default=true)

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

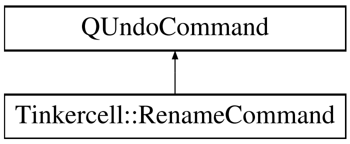
- UndoCommands.h
- UndoCommands.cpp

6.76 Tinkercell::RenameCommand Class Reference

this command changes the name of the handle of an item. important: use full name of the items!

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::RenameCommand:



Public Member Functions

- RenameCommand (const QString &name, NetworkHandle ∗, const QList< ItemHandle ∗ > &allItems, const QString &oldname, const QString &newname, bool forceUnique=true)

    constructor

- RenameCommand (const QString &name, NetworkHandle ∗, const QString &oldname, const QString &newname, bool forceUnique=true)

    constructor

- RenameCommand (const QString &name, NetworkHandle ∗, const QList< ItemHandle ∗ > &allItems, const QList< QString > &oldname, const QList< QString > &newname, bool forceUnique=true)

    constructor

- RenameCommand (const QString &name, NetworkHandle ∗, const QList< QString > &oldname, const QList< QString > &newname, bool forceUnique=true)

    constructor

- RenameCommand (const QString &name, NetworkHandle ∗, ItemHandle ∗itemHandle, const QString &newname, bool forceUnique=true)

    constructor

- RenameCommand (const QString &name, NetworkHandle ∗, const QList< ItemHandle ∗ > &allItems, ItemHandle ∗item, const QString &newname, bool forceUnique=true)

    constructor

- RenameCommand (const QString &name, NetworkHandle ∗, const QList< ItemHandle ∗ > &itemhandles, const QList< QString > &newnames, bool forceUnique=true)

    constructor

- RenameCommand (const QString &name, NetworkHandle ∗, const QList< ItemHandle ∗ > &allItems, const QList< ItemHandle ∗ > &itemhandles, const QList< QString > &newnames, bool forceUnique=true)

    constructor

- void **redo** ()
- void **undo** ()

Static Public Member Functions

- static void **findReplaceAllHandleData** (const QList< ItemHandle ∗ > &allItems, const QString &oldName, const QString &newName)
- static void **substituteString** (QString &targetValue, const QString &oldName, const QString &newName)

6.76.1 Detailed Description

this command changes the name of the handle of an item. important: use full name of the items!

6.76.2 Constructor & Destructor Documentation

6.76.2.1 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle ∗ *net*, const QList< ItemHandle ∗ > & *allItems*, const QString & *oldname*, const QString & *newname*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle ∗ network

QList affected items

QString old name

QString new name

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

6.76.2.2 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle ∗ *net*, const QString & *oldname*, const QString & *newname*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle ∗ network

QString old name

QString new name

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

6.76.2.3 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle ∗ *net*, const QList< ItemHandle ∗ > & *allItems*, const QList< QString > & *oldname*, const QList< QString > & *newname*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle ∗ network

QList affected items

QString old name

QString new name

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

6.76.2.4 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle ∗ *net*, const QList< QString > & *oldname*, const QList< QString > & *newname*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle ∗ network

QString old name

QString new name

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

6.76.2.5 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle ∗ *net*, ItemHandle ∗ *itemHandle*, const QString & *newname*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle ∗ network

ItemHandle∗ target item handle

QString new name

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

6.76.2.6 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle $*$ *net*, const QList< ItemHandle $*$ > & *allItems*, ItemHandle $*$ *item*, const QString & *newname*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle $*$ network

QList<ItemHandle$*$>& all the items to modify if they contain the new name

ItemHandle$*$ target item

QString new name

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

6.76.2.7 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle $*$ *net*, const QList< ItemHandle $*$ > & *itemhandles*, const QList< QString > & *newnames*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle $*$ network

QList<ItemHandle$*$>& target items

QList<QString> new names (one for each item)

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

6.76.2.8 Tinkercell::RenameCommand::RenameCommand (const QString & *name*, NetworkHandle $*$ *net*, const QList< ItemHandle $*$ > & *allItems*, const QList< ItemHandle $*$ > & *itemhandles*, const QList< QString > & *newnames*, bool *forceUnique* = `true`)

constructor

Parameters

QString name of command

NetworkHandle $*$ network

QList<ItemHandle$*$>& all the items to modify if they contain the new name

QList<ItemHandle$*$>& target items

QList<QString> new names (one for each item)

bool make sure that the new names are unique (default = true). Use false if you already made this check or want to rename to something that already exists

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

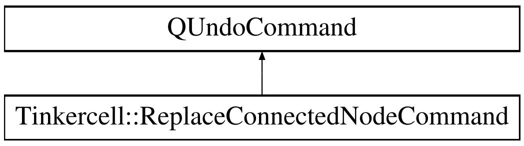
- UndoCommands.h
- UndoCommands.cpp

6.77 Tinkercell::ReplaceConnectedNodeCommand Class Reference

this command replaces one node item in a connection item with another

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ReplaceConnectedNodeCommand:



Public Member Functions

- ReplaceConnectedNodeCommand (const QString &name, ConnectionGraphicsItem ∗, NodeGraphicsItem ∗oldNode, NodeGraphicsItem ∗newNode)

  constructor

- void **redo** ()
- void **undo** ()

6.77.1 Detailed Description

this command replaces one node item in a connection item with another

6.77.2 Constructor & Destructor Documentation

6.77.2.1 Tinkercell::ReplaceConnectedNodeCommand::ReplaceConnectedNodeCommand (const QString & *name*, ConnectionGraphicsItem ∗ *c*, NodeGraphicsItem ∗ *oldNode*, NodeGraphicsItem ∗ *newNode*)

constructor

Parameters

QString	name of command
ConnectionGraphicsItem∗	connection where the nodes will be swapped
NodeGraphicsItem∗	node to replace (old node)
NodeGraphicsItem∗	new node

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

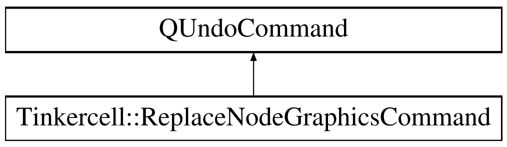
- UndoCommands.h
- UndoCommands.cpp

6.78 Tinkercell::ReplaceNodeGraphicsCommand Class Reference

this command can be used to replace the graphical representation of a node from an xml file

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ReplaceNodeGraphicsCommand:



Public Member Functions

- ReplaceNodeGraphicsCommand (const QString &, NodeGraphicsItem ∗, const QString &, bool transform=true)

  constructor

- ReplaceNodeGraphicsCommand (const QString &, const QList< NodeGraphicsItem ∗ > &, const QList< QString > &, bool transform=true)

  constructor

- void **undo** ()
- void **redo** ()

6.78.1 Detailed Description

this command can be used to replace the graphical representation of a node from an xml file

6.78.2 Constructor & Destructor Documentation

6.78.2.1 Tinkercell::ReplaceNodeGraphicsCommand::ReplaceNodeGraphicsCommand (const QString & *text*, NodeGraphicsItem ∗ *node*, const QString & *filename*, bool *transform* = `true`)

constructor

Parameters

QString	name of command
NodeGraphicsItem∗	the target node
QString	xml file name
bool	whether or not to transform the new graphics item to the original item's angle and size

6.78.2.2 Tinkercell::ReplaceNodeGraphicsCommand::ReplaceNodeGraphicsCommand (const QString & *text*, const QList< NodeGraphicsItem * > & *nodes*, const QList< QString > & *filenames*, bool *transform* = `true`)

constructor

Parameters

QString name of command

QList<NodeGraphicsItem*> the target nodes

QStringList xml file names

bool whether or not to transform the new graphics item to the original item's angle and size

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

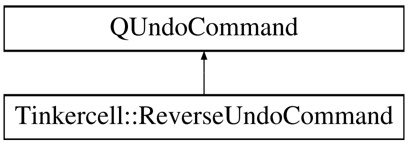
- UndoCommands.h
- UndoCommands.cpp

6.79 Tinkercell::ReverseUndoCommand Class Reference

this command can be used to invert another undo command (i.e. flip the redo/undo)

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::ReverseUndoCommand:



Public Member Functions

- ReverseUndoCommand (const QString &, QUndoCommand ∗, bool deleteCommand=true)

  constructor

- void **redo** ()
- void **undo** ()

Public Attributes

- QUndoCommand ∗ **command**
- bool **deleteCommand**

6.79.1 Detailed Description

this command can be used to invert another undo command (i.e. flip the redo/undo)

6.79.2 Constructor & Destructor Documentation

6.79.2.1 Tinkercell::ReverseUndoCommand::ReverseUndoCommand (const QString & *name*, QUndoCommand ∗ *cmd*, bool *deleteCommand* = `true`)

constructor

Parameters

QString	name of command
QList<QUndoCommand∗>&	the command to invert
bool	whether or not to delete the inverted command (true = DO delete)

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

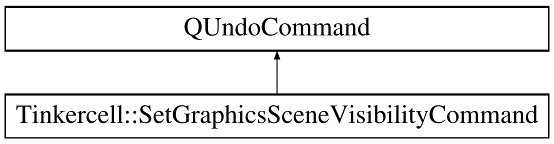
- UndoCommands.h
- UndoCommands.cpp

6.80 Tinkercell::SetGraphicsSceneVisibilityCommand Class Reference

this command is used to hide graphics items. Hidden graphics items will be part (unless their handles are also hidden) of the network but not visible on the screen.

```
#include <UndoCommands.h>
```

Inheritance diagram for Tinkercell::SetGraphicsSceneVisibilityCommand:



Public Member Functions

- SetGraphicsSceneVisibilityCommand (const QString &name, const QList< QGraphicsItem ∗ > &, const QList< bool > &)

  constructor
- SetGraphicsSceneVisibilityCommand (const QString &name, QGraphicsItem ∗, bool)

  constructor
- SetGraphicsSceneVisibilityCommand (const QString &name, const QList< QGraphicsItem ∗ > &, bool)

  constructor
- void redo ()

  redo parent change
- void undo ()

  undo parent change

6.80.1 Detailed Description

this command is used to hide graphics items. Hidden graphics items will be part (unless their handles are also hidden) of the network but not visible on the screen.

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

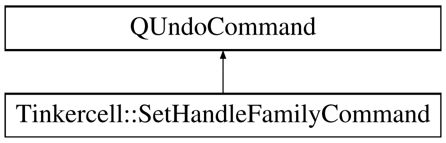
- UndoCommands.h
- UndoCommands.cpp

6.81 Tinkercell::SetHandleFamilyCommand Class Reference

this command is used to hide graphics items. Hidden graphics items will be part (unless their handles are also hidden) of the network but not visible on the screen.

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::SetHandleFamilyCommand:



Public Member Functions

- SetHandleFamilyCommand (const QString &name, const QList< ItemHandle ∗ > &, const QList< ItemFamily ∗ > &)

    constructor

- SetHandleFamilyCommand (const QString &name, ItemHandle ∗, ItemFamily ∗)

    constructor

- void redo ()

    redo parent change

- void undo ()

    undo parent change

Friends

- class **NetworkHandle**

6.81.1 Detailed Description

this command is used to hide graphics items. Hidden graphics items will be part (unless their handles are also hidden) of the network but not visible on the screen.

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

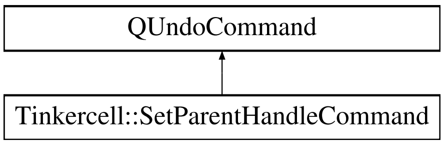
- UndoCommands.h
- UndoCommands.cpp

6.82 Tinkercell::SetParentHandleCommand Class Reference

this command assigns parent(s) to one or more handles

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::SetParentHandleCommand:



Public Member Functions

- SetParentHandleCommand (const QString &name, NetworkHandle ∗, ItemHandle ∗child, ItemHandle ∗parent)

  constructor

- SetParentHandleCommand (const QString &name, NetworkHandle ∗, const QList< ItemHandle ∗ > &children, ItemHandle ∗parent)

  constructor

- SetParentHandleCommand (const QString &name, NetworkHandle ∗, const QList< ItemHandle ∗ > &children, const QList< ItemHandle ∗ > &parents)

  constructor

- ∼SetParentHandleCommand ()

  destructor

- void redo ()

  redo parent change

- void undo ()

  undo parent change

Friends

- class **NetworkHandle**

6.82.1 Detailed Description

this command assigns parent(s) to one or more handles

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

- UndoCommands.h
- UndoCommands.cpp

6.83 Tinkercell::NodeGraphicsItem::Shape Class Reference

A closed polygon path made from arcs, lines, and beziers.

`#include <NodeGraphicsItem.h>`

Public Types

- enum { **Type** = UserType + 3 }

  for enabling dynamic_cast

Public Member Functions

- Shape (NodeGraphicsItem ∗idrawable_ptr=0, QGraphicsItem ∗parent=0, QGraphicsScene ∗scene=0)
- Shape (const Shape &copy)
- virtual Shape & operator= (const Shape &copy)
- void refresh ()

  Generates a new polygon using the points and types vectors Precondition: points.size > 1 Postcondition: NA.

- bool isClosed () const

  Checks if the polygon is closed.

- virtual QPainterPath shape () const

  gets a path that represents this shape

- virtual QRectF boundingRect () const

  bounding rect

- virtual int type () const

  for enabling dynamic_cast

Public Attributes

- QBrush defaultBrush

  permanent brush for this control point

- QPen defaultPen

  permanent pen for this control point

- NodeGraphicsItem ∗ nodeItem

  paint method. Call's parent's paint after setting antialiasing to true

- bool negative
- QVector< ControlPoint ∗ > controlPoints

  control points defining this shape

- QVector< qreal > parameters

  thinckness, arc angles, etc.

- QVector< ShapeType > types

  types of shapes to draw using the control points

- QPolygonF polygon

  the polygon constructed from controls and types vectors

- QPainterPath path

  the path constructed from controls and types vectors

- QPair< QPointF, QPointF > gradientPoints

  start and stop coordinates for gradient fill

Protected Member Functions

- virtual void recomputeBoundingRect ()

  reconstruct bounding rect

Protected Attributes

- QRectF boundingRectangle

  bounding reactangle for this shape

6.83.1 Detailed Description

A closed polygon path made from arcs, lines, and beziers.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 Tinkercell::NodeGraphicsItem::Shape::Shape (NodeGraphicsItem ∗ *idrawable_ptr* = 0, QGraphicsItem ∗ *parent* = 0, QGraphicsScene ∗ *scene* = 0)

Constructor: sets angle to 0 and scale to 1

6.83.2.2 Tinkercell::NodeGraphicsItem::Shape::Shape (const Shape & *copy*)

Copy Constructor

Copy Constructor : shallow copy of all vectors

6.83.3 Member Function Documentation

6.83.3.1 QRectF Tinkercell::NodeGraphicsItem::Shape::boundingRect () const `[virtual]`

bounding rect

bounding rectangle

6.83.3.2 NodeGraphicsItem::Shape & Tinkercell::NodeGraphicsItem::Shape::operator= (const Shape & *copy*) `[virtual]`

Copy operator

operator = shallow copy of all vectors

6.83.3.3 void Tinkercell::NodeGraphicsItem::Shape::refresh ()

Generates a new polygon using the points and types vectors Precondition: points.size > 1 Postcondition: NA.

paint method. Call's parent's paint after setting antialiasing to true

Parameters

void

Returns

void

Generates a new polygon using the points and types vectors Precondition: controlPoints.size > 1 Postcondition: NA

Parameters

void

Returns

void

6.83.3.4 QPainterPath Tinkercell::NodeGraphicsItem::Shape::shape () const `[virtual]`

gets a path that represents this shape

gets a path that represents this graphicsItem

6.83.4 Member Data Documentation

6.83.4.1 bool Tinkercell::NodeGraphicsItem::Shape::negative

is this a negative (clip out) shape

6.83.4.2 NodeGraphicsItem∗ Tinkercell::NodeGraphicsItem::Shape::nodeItem

paint method. Call's parent's paint after setting antialiasing to true

the NodeGraphicsItem that this shape belongs in

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

- NodeGraphicsItem.h
- NodeGraphicsItem.cpp

6.84 Tinkercell::ShowHideLegendItemsWidget Class Reference

Public Member Functions

- **ShowHideLegendItemsWidget** (DataPlot ∗plot, QWidget ∗parent)

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

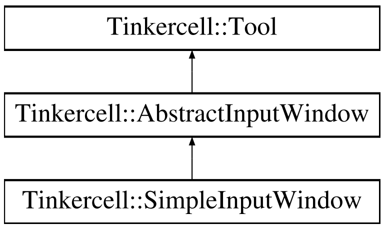
- Plot2DWidget.h
- Plot2DWidget.cpp

6.85 Tinkercell::SimpleInputWindow Class Reference

Used to create an input window that can receive user inputs for C plugins.

`#include <AbstractInputWindow.h>`

Inheritance diagram for Tinkercell::SimpleInputWindow:


Tinkercell::Tool
Tinkercell::AbstractInputWindow
Tinkercell::SimpleInputWindow

Public Slots

- virtual void exec ()

  Executes the CThread.

Static Public Member Functions

- static void CreateWindow (MainWindow ∗main, const QString &title, const QString &libraryFile, const QString &funcName, const DataTable< qreal > &)

  Create a simple input window to run a CThread. The window can be used to fill in an input matrix.

- static void CreateWindow (CThread ∗cthread, const QString &title, void(∗f)(tc_matrix), const DataTable< qreal > &)

  creates a docking window in Tinkercell's mainwindow that can receive inputs from user and run a function in a separate thread

- static void AddOptions (const QString &title, int i, int j, const QStringList &options)

  add a list of options (combo box) to an existing input window

- static void AddOptions (const QString &title, int i, int j)

  add a check box to an existing input window

Protected Slots

- virtual void dataChanged (int, int)

  updates the input matrix when user changes the table

- virtual void addRow ()

  add a row to the input matrix

- virtual void removeRow ()

remove a row from the input matrix

- virtual void comboBoxChanged (int)

  updates the input matrix when user changes the combo boxes

Protected Member Functions

- SimpleInputWindow (MainWindow ∗main, const QString &title, const QString &dllName, const QString &funcName, const DataTable< qreal > &)

  constructor that creates a docking window in Tinkercell's mainwindow that can receive inputs from user and run a function in a separate thread

- SimpleInputWindow (CThread ∗thread, const QString &title, void(∗f)(tc_matrix), const DataTable< qreal > &)

  constructor that creates a docking window in Tinkercell's mainwindow that can receive inputs from user and run a function in a separate thread

- SimpleInputWindow ()

  constructor -- does nothing

- SimpleInputWindow (const SimpleInputWindow &)

  copy constructor

- virtual void setupDisplay (const DataTable< qreal > &)

  reinitialize the contents on the input window

- void leaveEvent (QEvent ∗event)

  make the window transparent when mouse exits the window

- void enterEvent (QEvent ∗event)

  make the window transparent when mouse exits the window

Protected Attributes

- DataTable< qreal > dataTable

  the input matix

- QTableWidget tableWidget

  the table displaying the input matrix

- QList< QComboBox ∗ > comboBoxes

  combo boxes used in input window

- PopupListWidgetDelegate delegate

  the item delegate that is used to change values in the input window

Static Protected Attributes

- static QHash< QString, SimpleInputWindow ∗ > inputWindows

  the set of all simple input windows

6.85.1 Detailed Description

Used to create an input window that can receive user inputs for C plugins.

6.85.2 Constructor & Destructor Documentation

6.85.2.1 Tinkercell::SimpleInputWindow::SimpleInputWindow (MainWindow ∗ *main*, const QString & *title*, const QString & *dllName*, const QString & *funcName*, const DataTable< qreal > & *data*) `[protected]`

constructor that creates a docking window in Tinkercell's mainwindow that can receive inputs from user and run a function in a separate thread

Parameters

MainWindow	
QString	title
QString	dynamic library file
QString	function to run inside library
QDataTable<qreal>	input table and its default values

6.85.2.2 Tinkercell::SimpleInputWindow::SimpleInputWindow (CThread ∗ *thread*, const QString & *title*, void(∗)(tc_matrix) *f*, const DataTable< qreal > & *data*) `[protected]`

constructor that creates a docking window in Tinkercell's mainwindow that can receive inputs from user and run a function in a separate thread

Parameters

CThread	∗ existing thread with the library containing the function
QString	title
inputtc_matrixFunction∗	function that is triggered by the run button in the input window
QDataTable<qreal>	input table and its default values

6.85.3 Member Function Documentation

6.85.3.1 void Tinkercell::SimpleInputWindow::AddOptions (const QString & *title*, int *i*, int *j*) `[static]`

add a check box to an existing input window

Parameters

QString title

int row

int column

6.85.3.2 void Tinkercell::SimpleInputWindow::AddOptions (const QString & *title*, int *i*, int *j*, const QStringList & *options*) `[static]`

add a list of options (combo box) to an existing input window

Parameters

QString title

int row

int column

QStringList options

6.85.3.3 void Tinkercell::SimpleInputWindow::CreateWindow (CThread ∗ *cthread*, const QString & *title*, void(∗)(tc_matrix) *f*, const DataTable< qreal > & *data*) `[static]`

creates a docking window in Tinkercell's mainwindow that can receive inputs from user and run a function in a separate thread

Parameters

CThread ∗ existing thread with the library containing the function

QString title

inputtc_matrixFunction∗ function that is triggered by the run button in the input window

QDataTable<qreal> input table and its default values

6.85.3.4 void Tinkercell::SimpleInputWindow::CreateWindow (MainWindow ∗ *main*, const QString & *title*, const QString & *libraryFile*, const QString & *funcName*, const DataTable< qreal > & *data*) `[static]`

Create a simple input window to run a CThread. The window can be used to fill in an input matrix.

Parameters

MainWindow

QString title

QString dynamic library file (will first search if already loaded in MainWindow)

QString function to run inside library

DataTable<double> inputs

QList<QStringList> options for the inputs (optional)

6.85.3.5 void Tinkercell::SimpleInputWindow::exec () `[virtual, slot]`

Executes the CThread.

See also

CThread

Reimplemented from Tinkercell::AbstractInputWindow.

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

- AbstractInputWindow.h
- AbstractInputWindow.cpp

6.86 Tinkercell::Plot3DWidget::StandardColor Class Reference

Public Member Functions

- **StandardColor** (double, const QColor &, double, const QColor &)
- Qwt3D::RGBA **operator()** (double x, double y, double z) const
- Qwt3D::RGBA **operator()** (Qwt3D::Triple const &t) const
- Qwt3D::ColorVector & **createVector** (Qwt3D::ColorVector &vec)

Public Attributes

- QColor **start**
- QColor **end**
- double **minZ**
- double **maxZ**

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

- Plot3DWidget.h
- Plot3DWidget.cpp

6.87 Tinkercell::SymbolsTable Class Reference

The symbols table is updated every time the scene or text editor changes. The symbols table contains the list of item names and ItemHandle pointers as well as names and pointers to each data entry in each item.

```
#include <SymbolsTable.h>
```

Public Member Functions

- SymbolsTable (NetworkHandle ∗)

  constructor

- virtual void update ()

  update the symbols table

- virtual bool isValidPointer (void ∗) const

  checks whether the given item handle pointer is valid

- virtual QList< ItemHandle ∗ > allHandlesSortedByFamily () const

  get list of all items sorted according to family

- virtual QList< ItemHandle ∗ > allHandlesSortedByName () const

  get list of all items sorted according to their full name

Public Attributes

- QHash< QString, ItemHandle ∗ > uniqueHandlesWithDot

  handle names and the corresponsing handles. This hash stores the unique full names, such a M.A and M_A

- QHash< QString, ItemHandle ∗ > **uniqueHandlesWithUnderscore**
- QHash< QString, ItemHandle ∗ > nonuniqueHandles

  handle names and the corresponsing handles. This hash stores the the non-unique names, such as A. Therefore the hash may contain multiple values for the same key (see QHash documentation)

- QHash< QString, QPair< ItemHandle ∗, QString > > uniqueDataWithDot

  row or column name and the corresponding handle and tool in which the row or column name belongs. Stores full names only. For example, if A.k0 is a data item, then this table will contain A.k0 and A_k0. All entries are unique.

- QHash< QString, QPair< ItemHandle ∗, QString > > **uniqueDataWithUnderscore**
- QHash< QString, QPair< ItemHandle ∗, QString > > nonuniqueData

  row or column name and the corresponding handle and tool in which the row or column name belongs. Stores just the row or column name. For example, if A.k0 is a data item, then this table will contain k0. The individual, non-unique, names such as k0 may have multiple hash values for the same hash key (see QHash documentation).

- QHash< QString, ItemHandle ∗ > handlesByFamily

  this hash contains all the list of items belonging in each family. The items are listed under their family only and not under their parent families. For example, you will not find an item of family "Elephant" under the "Mammals" key. You will have to specifically search under "Elephant" and use ItemFamily's isA method to find out that it is also a "Mammal"

Protected Member Functions

- virtual void update (const QList< ItemHandle $*$ > &)

  update the symbols table

Protected Attributes

- NetworkHandle $*$ network

  the network that this symbols table belongs with

- ItemHandle globalHandle

  This is a special item handle that does not represent any item on the scene. It is used to store "global" data.

- QHash< void $*$, QString > handlesAddress

  addresses of all handles

Friends

- class **NetworkHandle**

6.87.1 Detailed Description

The symbols table is updated every time the scene or text editor changes. The symbols table contains the list of item names and ItemHandle pointers as well as names and pointers to each data entry in each item.

6.87.2 Constructor & Destructor Documentation

6.87.2.1 Tinkercell::SymbolsTable::SymbolsTable (NetworkHandle $*$ *net*)

constructor

Parameters

NetworkWindow*$*$* network that this symbol table belongs in

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

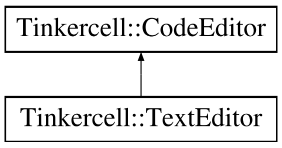
- SymbolsTable.h
- SymbolsTable.cpp

6.88 Tinkercell::TextEditor Class Reference

This is the window that allows used to construct networks using text, as opposed to graphics, which is done by GraphicsScene. The TextEditor requires a supporting tool that parses the text and calls the itemsInserted or itemsRemoved methods. Without a supporting parser tool, the TextEditor will not do anything.

```
#include <TextEditor.h>
```

Inheritance diagram for Tinkercell::TextEditor:



Public Slots

- virtual void popOut ()
  calls main window's popOut
- virtual void popIn ()
  calls main window's popIn
- virtual void undo ()
  undo last edit
- virtual void redo ()
  redo last undo
- virtual void selectAll ()
  select all text
- virtual void copy ()
  copy selected text
- virtual void cut ()
  cut selected text
- virtual void paste ()
  paste text from clipboard
- void find (const QString &)
  find specified text
- void replace (const QString &old_string, const QString &new_string)
  find and replace specified text
- virtual void print (QPrinter ∗printer)
  print text

Signals

- void textChanged (TextEditor ∗, const QString &, const QString &, const QString &)
  some text inside this editor has been changed
- void lineChanged (TextEditor ∗, int, const QString &)
  the cursor has moved to a different line
- void itemsInserted (NetworkHandle ∗, const QList< ItemHandle ∗ > &)
  signal that is emitted when items are inserted in this TextEditor.
- void itemsRemoved (NetworkHandle ∗, const QList< ItemHandle ∗ > &)
  signal that is emitted when items are removed from this TextEditor.
- void parse (TextEditor ∗)
  request to parse the text in the current text editor

Public Member Functions

- TextEditor (NetworkHandle ∗, QWidget ∗parent=0)
  default constructor
- ∼TextEditor ()
  destructor -- removes all the text items
- void insert (ItemHandle ∗)
  insert a text item
- void insert (const QList< ItemHandle ∗ > &)
  insert text items
- void remove (ItemHandle ∗)
  remove an item
- void remove (const QList< ItemHandle ∗ > &)
  remove text items
- void setItems (const QList< ItemHandle ∗ > &)
  clear existing items and insert new items
- QList< ItemHandle ∗ > & items ()
  all the items represented by the text in this TextEditor
- void push (QUndoCommand ∗)
  push a command to the undo/redo stack
- QString selectedText () const
  gets the selected text

- MainWindow ∗ mainWindow () const

  the main window containing this network

- ConsoleWindow ∗ console () const

  same as network->mainWindow->console()

- ItemHandle ∗ localHandle () const

  same as networkWindow->handle

- ItemHandle ∗ globalHandle () const

  same as network->globalHandle()

Public Attributes

- QMenu ∗ contextSelectionMenu

  the context menu that is shown during right-click event on a text editor with text selected. Plugins can add new actions to this menu.

- QMenu ∗ contextEditorMenu

  the context menu that is shown during right-click event on a text editor with no text selected. Plugins can add new actions to this menu.

- NetworkHandle ∗ network

  the network handle represented in this text editor

- NetworkWindow ∗ networkWindow

  the network window containing this text editor

Static Public Attributes

- static bool **SideBarEnabled** = true

Protected Member Functions

- virtual void keyPressEvent (QKeyEvent ∗event)

  listens to keyboard events in order to determine when the current line has changed

- virtual void mousePressEvent (QMouseEvent ∗event)

  listens to mouse events just to activate this window

- virtual void contextMenuEvent (QContextMenuEvent ∗event)

  creates context menu with actions in the contextMenu member

- virtual void mouseReleaseEvent (QMouseEvent ∗event)

  emits line changed and text changed if needed

Protected Attributes

- int prevBlockNumber

  previously accessed line number. This is to keep track of when a line is modified

- int changedBlockNumber

  current line number. This is to keep track of when a line is modified

- QString prevBlockText

  previously accessed line. This is to keep track of when a line is modified

- QString changedBlockText

  current line. This is to keep track of when a line is modified

- QString prevText

  current text. This is to keep track of when the text is modified

- QList< ItemHandle ∗ > allItems

  all the items represented by the text in this TextEditor

Friends

- class **TextUndoCommand**
- class **NetworkWindow**
- class **NetworkHandle**
- class **SymbolsTable**
- class **MainWindow**

6.88.1 Detailed Description

This is the window that allows used to construct networks using text, as opposed to graphics, which is done by GraphicsScene. The TextEditor requires a supporting tool that parses the text and calls the itemsInserted or itemsRemoved methods. Without a supporting parser tool, the TextEditor will not do anything.

6.88.2 Member Function Documentation

6.88.2.1 void Tinkercell::TextEditor::find (const QString & *s*) `[slot]`

find specified text

Parameters

QString text to find

6.88.2.2 void Tinkercell::TextEditor::insert (const QList< ItemHandle ∗ > & *list*)

insert text items

Parameters

QList<ItemHandle∗> the items

6.88.2.3 void Tinkercell::TextEditor::insert (ItemHandle ∗ *item*)

insert a text item

Parameters

ItemHandle∗ the item

6.88.2.4 void Tinkercell::TextEditor::itemsInserted (NetworkHandle ∗, const QList< ItemHandle ∗ > &) `[signal]`

signal that is emitted when items are inserted in this TextEditor.

Parameters

NetworkHandle∗

QList<ItemHandle∗> new item handles

6.88.2.5 void Tinkercell::TextEditor::itemsRemoved (NetworkHandle ∗, const QList< ItemHandle ∗ > &) `[signal]`

signal that is emitted when items are removed from this TextEditor.

Parameters

NetworkHandle∗

QList<ItemHandle∗> removed item handles

6.88.2.6 void Tinkercell::TextEditor::lineChanged (TextEditor ∗, int, const QString &) `[signal]`

the cursor has moved to a different line

Parameters

int index of the current line

QString current line text

6.88.2.7 void Tinkercell::TextEditor::parse (TextEditor ∗) `[signal]`

request to parse the text in the current text editor

Parameters

TextEditor∗ editor

6.88.2.8 void Tinkercell::TextEditor::popIn () `[virtual, slot]`

calls main window's popIn

Returns

void

6.88.2.9 void Tinkercell::TextEditor::popOut () `[virtual, slot]`

calls main window's popOut

Returns

void

6.88.2.10 void Tinkercell::TextEditor::print (QPrinter ∗ *printer*) `[virtual, slot]`

print text

Parameters

QPrinter

6.88.2.11 void Tinkercell::TextEditor::push (QUndoCommand ∗ *c*)

push a command to the undo/redo stack

Parameters

QUndoCommand∗

6.88.2.12 void Tinkercell::TextEditor::remove (const QList< ItemHandle ∗ > & *handles*)

remove text items

Parameters

QList<ItemHandle∗*>* the items

6.88.2.13 void Tinkercell::TextEditor::remove (ItemHandle ∗ *item*)

remove an item

Parameters

ItemHandle∗ the item

6.88.2.14 void Tinkercell::TextEditor::replace (const QString & *old_string*, const QString & *new_string*) `[slot]`

find and replace specified text

Parameters

QRegExp text to find

QString text to replace

6.88.2.15 void Tinkercell::TextEditor::setItems (const QList< ItemHandle ∗ > & *newItems*)

clear existing items and insert new items

Parameters

QList<ItemHandle∗> the new items

6.88.2.16 void Tinkercell::TextEditor::textChanged (TextEditor ∗, const QString &, const QString &, const QString &) `[signal]`

some text inside this editor has been changed

Parameters

QString old text

QString new text

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

- TextEditor.h
- TextEditor.cpp

6.89 Tinkercell::TextGraphicsItem Class Reference

editable text item

`#include <TextGraphicsItem.h>`

Public Types

- enum { **Type** = UserType + 8 }

  for enabling dynamic_cast

Public Member Functions

- virtual ItemHandle ∗ handle () const

  this text item's handle

- void setHandle (ItemHandle ∗)

  set this text item's handle

- TextGraphicsItem (const QString &text, QGraphicsItem ∗parent=0)

  Constructor.

- TextGraphicsItem (QGraphicsItem ∗parent=0)

  Constructor.

- TextGraphicsItem (const TextGraphicsItem &copy)

  Copy Constructor.

- virtual TextGraphicsItem ∗ clone ()

  Clone this item.

- TextGraphicsItem (ItemHandle ∗handle, QGraphicsItem ∗parent=0)

  Copy Constructor.

- virtual ∼TextGraphicsItem ()

  Destructor.

- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

  Paint this text item with or without a border.

- virtual void showBorder (bool show=true)

  whether or not to paint this item with a border

- virtual QString text () const

  the string painted by this text graphics item. same as toPlainText

- virtual void setText (const QString &)

  set the string painted by this text graphics item. same as setPlainText

- int type () const

  for enabling dynamic_cast

Static Public Member Functions

- static TextGraphicsItem ∗ cast (QGraphicsItem ∗)

  cast a graphics item to a text item using qgraphicsitem_cast

Public Attributes

- QPair< QGraphicsItem ∗, QPointF > relativePosition

  relative position with a target item

- QString groupID

  for identifying which group this item belongs in

Protected Attributes

- QGraphicsRectItem ∗ boundingRectItem

  draws a border around the text item. hide or show using showBorder()

- ItemHandle ∗ itemHandle

  the handle in which this item belongs

6.89.1 Detailed Description

editable text item

6.89.2 Constructor & Destructor Documentation

6.89.2.1 Tinkercell::TextGraphicsItem::TextGraphicsItem (const QString & *text*, QGraphicsItem ∗ *parent* = 0)

Constructor.

Parameters

QString	text
QGraphicsItem∗	parent

Constructor: sets text edit interaction

6.89.2.2 Tinkercell::TextGraphicsItem::TextGraphicsItem (QGraphicsItem ∗ *parent* = 0)

Constructor.

Parameters

QGraphicsItem∗ parent

Constructor: sets text edit interaction

6.89.2.3 Tinkercell::TextGraphicsItem::TextGraphicsItem (const TextGraphicsItem & *copy*)

Copy Constructor.

Parameters

TextGraphicsItem∗ copy

Copy Constructor

6.89.2.4 Tinkercell::TextGraphicsItem::TextGraphicsItem (ItemHandle ∗ *handle*, QGraphicsItem ∗ *parent* = 0)

Copy Constructor.

Parameters

ItemHandle∗ handle to which this item belongs
QGraphicsItem∗ parent

Constructor: sets text edit interaction and name of handle

6.89.3 Member Function Documentation

6.89.3.1 TextGraphicsItem ∗ Tinkercell::TextGraphicsItem::cast (QGraphicsItem ∗ *q*) `[static]`

cast a graphics item to a text item using qgraphicsitem_cast

Parameters

QGraphicsItem graphics item

Returns

TextGraphicsItem this will be 0 if the cast is invalid

6.89.3.2 void Tinkercell::TextGraphicsItem::setText (const QString & *s*) `[virtual]`

set the string painted by this text graphics item. same as setPlainText

Parameters

QString

6.89.3.3 QString Tinkercell::TextGraphicsItem::text () const `[virtual]`

the string painted by this text graphics item. same as toPlainText

Returns

QString

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

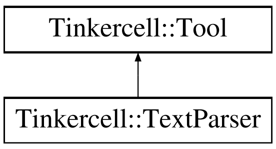
- TextGraphicsItem.h
- TextGraphicsItem.cpp

6.90 Tinkercell::TextParser Class Reference

TextParser is the parent class for all parsers. Parsers are classes that interpret the string in a TextEditor and insert items or modify items as needed. TinkerCell can support multiple parsers through the use of the TextParser interface.

`#include <TextParser.h>`

Inheritance diagram for Tinkercell::TextParser:



Public Slots

- virtual void activate ()

  set this parser as the current parser

- virtual void deactivate ()

  this parser is no longer the current parser

- virtual void parse (TextEditor ∗)

  this parser has been requested to parse the text inside the given text editor

- virtual void textChanged (TextEditor ∗, const QString &, const QString &, const QString &)

  some text inside this editor has been changed

- virtual void lineChanged (TextEditor ∗, int, const QString &)

  the cursor has moved to a different line

Signals

- void validSyntax (bool)

  invalid syntax

Public Member Functions

- TextParser (const QString &Name, QWidget ∗parent=0)

  constructor

Static Public Member Functions

- static void setParser (TextParser ∗)

  set the text parser for all text editors. The current text parser can be obtained using TextParser::currentParser();

- static TextParser ∗ currentParser ()

  The current text parser that is being used (can be 0 if none).

Public Attributes

- QPixmap icon

  icon for this class

6.90.1 Detailed Description

TextParser is the parent class for all parsers. Parsers are classes that interpret the string in a TextEditor and insert items or modify items as needed. TinkerCell can support multiple parsers through the use of the TextParser interface.

6.90.2 Constructor & Destructor Documentation

6.90.2.1 Tinkercell::TextParser::TextParser (const QString & *Name*, QWidget ∗ *parent* = 0)

constructor

Parameters

QString name

QWidget∗ parent

6.90.3 Member Function Documentation

6.90.3.1 void Tinkercell::TextParser::lineChanged (TextEditor ∗, int, const QString &) [virtual, slot]

the cursor has moved to a different line

Parameters

int index of the current line

QString current line text

6.90.3.2 void Tinkercell::TextParser::parse (TextEditor ∗) `[virtual, slot]`

this parser has been requested to parse the text inside the given text editor

Parameters

TextEditor∗ the text editor

6.90.3.3 void Tinkercell::TextParser::textChanged (TextEditor ∗, const QString &, const QString &, const QString &) `[virtual, slot]`

some text inside this editor has been changed

Parameters

TextEditor∗ the current editor

QString old text

QString new text

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

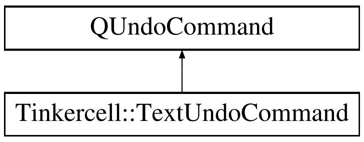
- TextParser.h
- TextParser.cpp

6.91 Tinkercell::TextUndoCommand Class Reference

this command performs a text change

`#include <TextEditor.h>`

Inheritance diagram for Tinkercell::TextUndoCommand:



Public Member Functions

- TextUndoCommand (TextEditor ∗, const QString &, const QString &)

  constructor

- void redo ()

  redo the change

- void undo ()

  undo the change

6.91.1 Detailed Description

this command performs a text change

6.91.2 Constructor & Destructor Documentation

6.91.2.1 Tinkercell::TextUndoCommand::TextUndoCommand (TextEditor ∗ *editor*, const QString & *oldText*, const QString & *newText*)

constructor

Parameters

***TextEditor*∗** editor where change happened

QString new text

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

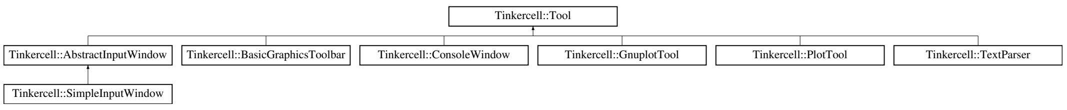
- TextEditor.h
- TextEditor.cpp

6.92 Tinkercell::Tool Class Reference

everything other than the main window is a tool

`#include <Tool.h>`

Inheritance diagram for Tinkercell::Tool:


Tinkercell::Tool
Tinkercell::AbstractInputWindow
Tinkercell::BasicGraphicsToolbar
Tinkercell::ConsoleWindow
Tinkercell::GnuplotTool
Tinkercell::PlotTool
Tinkercell::TextParser
Tinkercell::SimpleInputWindow

Public Slots

- virtual void select (int i=0)

  what happens when this tool is selected

- virtual void deselect (int i=0)

  what happens when this tool is deselected

- virtual void addAction (const QIcon &, const QString &text=QString(), const QString &tooltip=QString())

  add an action that will be displayed in the context menu when specific items with this tool in their tools list are selected

- virtual void addGraphicsItem (ToolGraphicsItem ∗)

  add a graphics item that will be displayed on the current scene when specific items with this tool in their tools list are selected

Signals

- void selected ()

  this tool is selected

- void deselected ()

  this tool is deselected

Public Member Functions

- Tool ()

  constructor

- ∼Tool ()

  destructor. removes graphicsItem and toolButton is not 0

- Tool (const QString &Name, const QString &category=QString(), QWidget ∗parent=0)

  constructor

- virtual bool setMainWindow (MainWindow ∗main)

    set the main window for this tool

- ConsoleWindow ∗ console ()

    console window (same as mainWindow->console())

- GraphicsScene ∗ currentScene () const

    the main window's current scene

- TextEditor ∗ currentTextEditor () const

    the main window's current text editor

- NetworkHandle ∗ currentNetwork () const

    the main window's current network

- NetworkWindow ∗ currentWindow () const

    the main window's current network's current window

- QPair< QList< ItemHandle ∗ >, QList< QGraphicsItem ∗ > > getItemsFromFile (const QString &filename)

    get the items inside a file. Some tool must implement this function and connect to the getItemsFromFile signal. The Core library does not implement a read file function.

Static Public Member Functions

- static QString homeDir ()

    same as MainWindow::homeDir

- static QString tempDir ()

    same as MainWindow::tempDir

Public Attributes

- QString name

    name of this tool

- QString category

    category that this tool belongs in

- QString description

    brief description of this tool

- MainWindow ∗ mainWindow

    main window for this tool

Protected Slots

- virtual void actionTriggered (QAction ∗action)

  context menu action triggered

Friends

- class **GraphicsScene**
- class **TextEditor**
- class **MainWindow**
- class **NetworkHandle**
- class **ToolGraphicsItem**

6.92.1 Detailed Description

everything other than the main window is a tool

6.92.2 Constructor & Destructor Documentation

6.92.2.1 Tinkercell::Tool::Tool (const QString & *Name*, const QString & *category* = `QString()`, QWidget ∗ *parent* = `0`)

constructor

Parameters

QString name

QString category (default = empty)

QWidget∗ parent (default = 0)

6.92.3 Member Function Documentation

6.92.3.1 NetworkHandle ∗ Tinkercell::Tool::currentNetwork () const

the main window's current network

Returns

NetworkHandle∗ current network handle

6.92.3.2 NetworkWindow ∗ Tinkercell::Tool::currentWindow () const

the main window's current network's current window

Returns

NetworkWindow∗ current network window

6.92.3.3 QPair< QList< ItemHandle ∗ >, QList< QGraphicsItem ∗ > > Tinkercell::Tool::getItemsFromFile (const QString & *filename*)

get the items inside a file. Some tool must implement this function and connect to the getItemsFromFile signal. The Core library does not implement a read file function.

Parameters

QString& file that is selected by user

Returns

QPair< QList<ItemHandle∗>, QList<QGraphicsItem∗> > list of handles and graphics items inside the file
void

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

- Tool.h
- Tool.cpp

6.93 Tinkercell::ToolGraphicsItem Class Reference

tools that are drawn on the scene instead of displayed as a window

```
#include <Tool.h>
```

Public Types

- enum { **Type** = UserType + 9 }

  for enabling dynamic_cast

Public Member Functions

- ToolGraphicsItem (Tool ∗)

  constructor must have an associated Tool

- virtual void select ()

  this item has been selected

- virtual void deselect ()

  this item has been deselected

- int type () const

  for enabling dynamic_cast

- virtual void visible (bool)

  show or hide this graphical tool. The graphical tool may choose whether or not to be visible based on other factors.

Static Public Member Functions

- static ToolGraphicsItem ∗ cast (QGraphicsItem ∗)

  cast a graphics item to a ToolGraphicsItem

Public Attributes

- Tool ∗ tool

  main window for this tool

6.93.1 Detailed Description

tools that are drawn on the scene instead of displayed as a window

6.93.2 Member Function Documentation

6.93.2.1 ToolGraphicsItem ∗ Tinkercell::ToolGraphicsItem::cast (QGraphicsItem ∗ *q*) `[static]`

cast a graphics item to a ToolGraphicsItem

Returns

ToolGraphicsItem∗ can be 0 if invalid cast

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

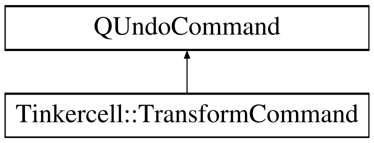
- Tool.h
- Tool.cpp

6.94 Tinkercell::TransformCommand Class Reference

this command changes the size, angle, and orientation of an item

`#include <UndoCommands.h>`

Inheritance diagram for Tinkercell::TransformCommand:



Public Member Functions

- TransformCommand (const QString &name, QGraphicsScene ∗scene, QGraphicsItem ∗item, const QPointF &sizechange, qreal anglechange, bool VFlip, bool HFlip)

  constructor

- TransformCommand (const QString &name, QGraphicsScene ∗scene, const QList< QGraphicsItem ∗ > &items, const QList< QPointF > &sizechange, const QList< qreal > &anglechange, bool VFlip, bool HFlip)

  constructor

- void **redo** ()
- void **undo** ()

6.94.1 Detailed Description

this command changes the size, angle, and orientation of an item

6.94.2 Constructor & Destructor Documentation

6.94.2.1 Tinkercell::TransformCommand::TransformCommand (const QString & *name*, QGraphicsScene ∗ *scene*, QGraphicsItem ∗ *item*, const QPointF & *sizechange*, qreal *anglechange*, bool *VFlip*, bool *HFlip*)

constructor

Parameters

QString	name of command
GraphicsScene∗	scene where change happened
QGraphicsItem∗	item that is affected
QPointF	change in size (w,h)
double	angle change
boolean	flip vertically
boolean	flip horizontally

6.94.2.2 Tinkercell::TransformCommand::TransformCommand (const QString & *name*, QGraphicsScene ∗ *scene*, const QList< QGraphicsItem ∗ > & *items*, const QList< QPointF > & *sizechange*, const QList< qreal > & *anglechange*, bool *VFlip*, bool *HFlip*)

constructor

Parameters

QString	name of command
GraphicsScene∗	scene where change happened
QList<QGraphicsItem	∗>& items that are affected
QList<QPointF>&	change in size (w,h)
QList<qreal>&	angle change
boolean	flip vertically (all items)
boolean	flip horizontally (all items)

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

- UndoCommands.h
- UndoCommands.cpp

6.95 Tinkercell::Unit Class Reference

A unit of measurement.

`#include <ItemFamily.h>`

Public Member Functions

- **Unit** (const QString &property, const QString &name)

Public Attributes

- QString **property**
- QString **name**

6.95.1 Detailed Description

A unit of measurement.

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

- ItemFamily.h
- ItemFamily.cpp

Index