Plot

Generated by Doxygen 1.9.3

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Axis< T > Class Template Reference

Public Member Functions

- **Axis** (const Plot2D< T > &associatedPlot, const std::size_t width, const std::size_t height, const ImageCoordinate startIndex)
- const Pixel< T, std::string > & **at** (const ImageCoordinate &idx) const
- const std::vector< ImageCoordinate > & **getAxisCoordinates** () const
- const std::vector< ImageCoordinate > & **getLabelCoordinates** () const
- const std::vector< ImageCoordinate > & **getCoordinates** () const
- const Pixel< T, std::string > & **getAxisElement** () const
- const std::size_t **getLabelOffset** () const
- void **setAxisElement** (const Pixel< T, std::string > &axisElement)
- void **setAxisColor** (const T &color)
- void **setAxisSymbol** (const std::string &axisSymbol)
- void **setLabelOffset** (std::size_t labelOffset)
- void **addLabel** (const std::string &label, float position, bool leftOrBottom=true, bool rotateLabel=false)
- void **addTicks** (const std::vector< Tick > &ticks, bool leftOrBottom=true)

Private Member Functions

- std::pair< int, int > **mAddAttribute** (const std::vector< Pixel< T, std::string > > &attribute, float position, std::size_t offsetFromAxis, bool leftOrBottom, std::vector< ImageCoordinate > &coordinateVector, bool rotate=false)
- void **mAddRangeToCoordinateVector** (std::vector< ImageCoordinate > &coordinateVector, const ImageCoordinate &firstNewCoordinate, std::size_t hRange, std::size_t vRange)
- void **mShift** (int horizontalShift, int verticalShift, std::vector< ImageCoordinate > &coordinateVector)
- void **mUpdateAllCoordinates** ()

Private Attributes

- const Plot2D< T > & **mPlot**
- ImageCoordinate **mUpperLeftIndex**
- ImageCoordinate **mLowerRightIndex**
- bool **mIsHorizontal**
- Pixel< T, std::string > **mAxisElement**
- std::vector< Pixel< T, std::string > > **mLabel**
- std::vector< Pixel< T, std::string > > **mTicks**
- std::size_t **mLabelOffset** = 1
- std::vector< ImageCoordinate > **mAllCoordinates**
- std::vector< ImageCoordinate > **mAxisCoordinates**
- std::vector< ImageCoordinate > **mLabelCoordinates**
- std::vector< ImageCoordinate > **mTicksCoordinates**

Static Private Attributes

- static const Pixel< T, std::string > **emptyPixel** = Pixel<T,std::string>(T()," ")

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

- header/axis.h
- src/axis.cpp

4.2 Image< T1, T2 > Class Template Reference

`#include <image.h>`

Public Member Functions

- Image ()
- Image (std::size_t width, std::size_t height)
- Image (std::size_t width, std::size_t height, const Pixel< T1, T2 > &p)
- virtual float **getAspectRatio** () const
- virtual void **setAspectRatio** (float aspectRatio)
- virtual const Pixel< T1, T2 > & **at** (std::size_t i, std::size_t j) const
- virtual Pixel< T1, T2 > & **at** (std::size_t i, std::size_t j)
- virtual const std::vector< std::vector< Pixel< T1, T2 > > > & **getPixels** () const
- virtual void **setPixels** (const Pixel< T1, T2 > &p)
- virtual bool **getFillSpaces** () const
- virtual void **setFillSpaces** (bool fillSpaces)
- virtual std::size_t **getHeight** () const
- virtual void **setHeight** (std::size_t height)
- virtual std::size_t **getWidth** () const
- virtual void **setWidth** (std::size_t width)
- virtual void **show** () const

Private Member Functions

- virtual void **verifyCoordinate** (std::size_t i, std::size_t j) const

Private Attributes

- float **mAspectRatio** = 1.
- bool **mFillSpaces** = false
- std::vector< std::vector< Pixel< T1, T2 > > > **mPixels**
- int **mWidth**
- int **mHeight**

4.2.1 Detailed Description

template<typename T1 = Mono, typename T2 = char>
class Image< T1, T2 >

Image is an image that can be printed onto the screen, e.g. to the console output

- Pixel values are represented by a PxQ "matrix" (vectors) of "Pixel" objects for a PxQ image (P,Q: no. of pixels).
- Data can be loaded into the image

The image can be printed to the set output using the method show()

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Image() [1/3]

```
template<typename T1 , typename T2 >
Image< T1, T2 >::Image
```

Constructs an empty image of size 1 x 1.

4.2.2.2 Image() [2/3]

```
template<typename T1 , typename T2 >
Image< T1, T2 >::Image (
            std::size_t width,
            std::size_t height )
```

Constructs an empty image of size width x height.

4.2.2.3 Image() [3/3]

```
template<typename T1 , typename T2 >
Image< T1, T2 >::Image (
            std::size_t width,
            std::size_t height,
            const Pixel< T1, T2 > & p )
```

Constructs an image of size width x height with all pixels set equal to p.

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

- header/image.h
- src/image.cpp

4.3 Mono Struct Reference

Public Attributes

- bool **b** = 1

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

- header/color_models.h

4.4 Pixel< T1, T2 > Class Template Reference

Public Member Functions

- **Pixel** (T1 color, T2 symbol)
- T1 **getColor** () const
- void **setColor** (T1 color)
- T2 **getSymbol** () const
- void **setSymbol** (T2 symbol)

Private Member Functions

- void **verifyCoordinate** (std::size_t i, std::size_t j) const

Private Attributes

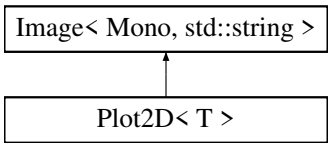
- T1 **mColor**
- T2 **mSymbol**

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

- header/pixel.h
- src/pixel.cpp

4.5 Plot2D< T > Class Template Reference

Inheritance diagram for Plot2D< T >:



Public Member Functions

- **Plot2D** (std::size_t width, std::size_t height)
- void **setXAxisScaling** (const std::function< double(double)> &xScalingFunction)
- void **setYAxisScaling** (const std::function< double(double)> &yScalingFunction)
- void **setPlotRange** (const std::pair< double, double > &xPlotRange, const std::pair< double, double > &yPlotRange)
- void addVerticalAxis (std::size_t hPos, std::size_t vPosStart, std::size_t vPosEnd, T color)
- void addHorizontalAxis (std::size_t vPos, std::size_t hPosStart, std::size_t hPosEnd, T color)
- void addVerticalAxis (std::size_t hPos, std::size_t vPosStart, std::size_t vPosEnd, T color, const std::string &label, float relativeLabelPosition=0.5, bool labelLeft=true, bool rotateLabel=false)
- void addHorizontalAxis (std::size_t vPos, std::size_t hPosStart, std::size_t hPosEnd, T color, const std::string &label, float relativeLabelPosition=0.5, bool labelBelow=true)
- void **addText** (std::size_t hPos, std::size_t vPos, const std::string &text)
- void **addDataSet** (std::shared_ptr< const DataSet > dataSet, Pixel< T, std::string > plotMarker)

Private Member Functions

- void **mClear** ()
- void **mDataSetToCoordinates** (std::size_t jDataSet)
- void **mAllDataSetsToCoordinates** ()
- ImageCoordinate **mDataPointToCoordinate** (const DataPointXY &dataPoint)

Private Attributes

- std::size_t **mWidth**
- std::size_t **mHeight**
- std::vector< Axis< T > > **mHorizontalAxes**
- std::vector< Axis< T > > **mVerticalAxes**
- std::vector< std::pair< std::string, ImageCoordinate > > **mText**
- ImageCoordinate **mDataFrameStart**
- ImageCoordinate **mDataFrameEnd**
- std::pair< double, double > **mXPlotRange**
- std::pair< double, double > **mYPlotRange**
- std::function< double(double)> **mXScalingFunction** = [ ](double x){ return x; }
- std::function< double(double)> **mYScalingFunction** = [ ](double y){ return y; }
- std::vector< std::shared_ptr< const DataSet > > **mDataSets**
- std::vector< std::vector< ImageCoordinate > > **mCoordinatesOfDataPoints**
- std::vector< Pixel< T, std::string > > **mPlotMarkers**

Static Private Attributes

- static const Pixel< T, std::string > **emptyPixel** = Pixel<T,std::string>(T()," ")

4.5.1 Member Function Documentation

4.5.1.1 addHorizontalAxis() [1/2]

```
template<typename T >
void Plot2D< T >::addHorizontalAxis (
            std::size_t vPos,
            std::size_t hPosStart,
            std::size_t hPosEnd,
            T color )
```

Adds a horizontal axis to the plot that extends from (hPosStart,vPos) to (hPosEnd,vPos).

Parameters

vPos	vertical coordinate of the axis
hPosStart	Horizontal coordinate of the start point of the axis
hPosEnd	Horizontal coordinate of the end of the axis
color	Color of the axis

4.5.1.2 addHorizontalAxis() [2/2]

```
template<typename T >
void Plot2D< T >::addHorizontalAxis (
            std::size_t vPos,
            std::size_t hPosStart,
            std::size_t hPosEnd,
            T color,
            const std::string & label,
            float relativeLabelPosition = 0.5,
            bool labelBelow = true )
```

Adds a labelled horizontal axis to the plot that extends from (hPosStart,vPos) to (hPosEnd,vPos). The position of the axis will be changed to make space for the label.

Parameters

label	Label that is added to the axis
relativeLabelPosition	Position of the first character of label along the axis; has to be between 0.0 and 1.0 (otherwise set to 0.0)
labelBelow	Placement relative to the axis (true: below, false: above)

See also

addHorizontalAxis(std::size_t vPos, std::size_t hPosStart, std::size_t hPosEnd, T color)

4.5.1.3 addVerticalAxis() [1/2]

```
template<typename T >
void Plot2D< T >::addVerticalAxis (
            std::size_t hPos,
            std::size_t vPosStart,
            std::size_t vPosEnd,
            T color )
```

Adds a vertical axis to the plot that extends from (hPos,vPosStart) to (hPos,vPosEnd)

Parameters

hPos	Horizontal coordinate of the axis
vPosStart	Vertical coordinate of the start point of the axis
vPosEnd	Vertical coordinate of the end of the axis
color	Color of the axis

4.5.1.4 addVerticalAxis() [2/2]

```
template<typename T >
void Plot2D< T >::addVerticalAxis (
            std::size_t hPos,
            std::size_t vPosStart,
            std::size_t vPosEnd,
            T color,
            const std::string & label,
            float relativeLabelPosition = 0.5,
            bool labelLeft = true,
            bool rotateLabel = false )
```

Adds a labelled vertical axis to the plot that extends from (hPos,vPosStart) to (hPos,vPosEnd). The position of the axis will be changed to make space for the label.

Parameters

label	Label that is added to the axis
relativeLabelPosition	Position of the first character of label along the axis; has to be between 0.0 and 1.0 (otherwise set to 0.0)
labelLeft	Placement relative to the axis (true: left, false: right)
rotateLabel	If true: rotate the label by 90 degrees clockwise. This can be useful to save space in the plot.

See also

addVerticalAxis(std::size_t hPos, std::size_t vPosStart, std::size_t vPosEnd, T color)

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

- header/axis.h
- header/plot2d.h
- src/plot2d.cpp

4.6 RGB Struct Reference

Public Attributes

- uint8_t **r** = 255
- uint8_t **g** = 255
- uint8_t **b** = 255

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

- header/color_models.h

Chapter 5

File Documentation

5.1 axis.h

```
1 #pragma once
2
3 /* Axis represents an axis in a plot associated to it */
4
5 #include <algorithm> //std::find, std::max
6 #include <string>
7 #include <utility> //std::pair
8
9 #include "../header/color_models.h"
10 #include "../header/pixel.h"
11
12 //Forward declaration of Plot2D, since it is needed as reference member in the Axis class
13 template <typename T>
14 class Plot2D;
15
16 using ImageCoordinate = std::pair<std::size_t,std::size_t>; //Stores a pair (jh,jv) of horizontal and
       vertical indices in the base image of the plot
17 using Tick = std::pair<std::string,float>; //Stores a "Tick" next to the axis in the form (TickLabel,
       relativePosition)
18
19 //Declaration of the Axis class
20 template <typename T = Mono>
21 class Axis
22 {
23 private:
24     inline static const Pixel<T,std::string> emptyPixel = Pixel<T,std::string>(T()," ");
25     const Plot2D<T> &mPlot; //The axis is part of this plot
26
27     ImageCoordinate mUpperLeftIndex; //Upper left index of the rectangle holding the axis including ticks
       and labels
28     ImageCoordinate mLowerRightIndex; //Upper left index of the rectangle holding the axis including
       ticks and labels
29
30     bool mIsHorizontal; //0: horizontal axis, 1: vertical axis
31     Pixel<T,std::string> mAxisElement; //Pixel representing a single element of the axis
32     std::vector< Pixel<T,std::string> > mLabel;
33     std::vector< Pixel<T,std::string> > mTicks;
34     std::size_t mLabelOffset = 1; //Space (in Pixels) between label and axis
35
36     //TODO: The following four should possibly better be maps or sets in order to improve lookups!
37     std::vector<ImageCoordinate> mAllCoordinates;
38     std::vector<ImageCoordinate> mAxisCoordinates; //Vector of index pairs (jh,jv) in the base Image of
       the associated Plot2D that are showing vertical axes elements
39     std::vector<ImageCoordinate> mLabelCoordinates; //Vector of index pairs (jh,jv) in the base Image of
       the associated Plot2D that are showing the label
40     std::vector<ImageCoordinate> mTicksCoordinates; //Vector of index pairs (jh,jv) in the base Image of
       the associated Plot2D that are showing empty pixels
41
42     std::pair<int,int> mAddAttribute( const std::vector< Pixel<T,std::string> >& attribute, float
       position, std::size_t offsetFromAxis, bool leftOrBottom, std::vector<ImageCoordinate>&
       coordinateVector , bool rotate = false );
43     void mAddRangeToCoordinateVector( std::vector<ImageCoordinate>& coordinateVector, const
       ImageCoordinate& firstNewCoordinate, std::size_t hRange, std::size_t vRange );
44     void mShift( int horizontalShift, int verticalShift, std::vector<ImageCoordinate>& coordinateVector
       ); //Shift all indices in all the indexVectors horizontally by horizontalShift and vertically by
       verticalShift
45     void mUpdateAllCoordinates(); //Updates the data member mAllCoordinates (e.g. after mAxisCoordinates
       etc. have been changed)
```

```
46
47 public:
48     /*Constructors and destructors*/
49     //Axis() = delete; //No default constructor because an Axis always needs an associated Plot2D in
       which it lives
50     Axis( const Plot2D<T>& associatedPlot, const std::size_t width, const std::size_t height, const
       ImageCoordinate startIndex );
51
52     /*---Getters and Setters----*/
53     const Pixel<T,std::string>& at( const ImageCoordinate& idx ) const;
54     const std::vector<ImageCoordinate>& getAxisCoordinates() const;
55     const std::vector<ImageCoordinate>& getLabelCoordinates() const;
56     const std::vector<ImageCoordinate>& getCoordinates() const;
57     const Pixel<T,std::string>& getAxisElement() const;
58     const std::size_t getLabelOffset() const;
59
60     void setAxisElement( const Pixel<T,std::string>& axisElement);
61     void setAxisColor( const T& color );
62     void setAxisSymbol( const std::string& axisSymbol );
63
64     void setLabelOffset( std::size_t labelOffset );
65     /*--------------------------*/
66
67     /*---------Modifiers--------*/
68     void addLabel( const std::string& label, float position, bool leftOrBottom = true, bool rotateLabel =
       false ); //adds a label at position (0<=position<=1) relative to leftmost (horizontal axis) or top
       (vertical axis) pixel
69     void addTicks( const std::vector<Tick>& ticks, bool leftOrBottom = true ); //adds the ticks
       "ticks.at(j).first" at relative positions "ticks.at(j).second"
70
71     /*--------------------------*/
72
73     /*----------Operators----------*/
74
75     /*-----------------------------*/
76 };
77
78 /*Include method definitions (needed here for the compiler because it is a class template!)*/
79 #include "../src/axis.cpp"
```

5.2 color_models.h

```
1 /*Defines structs that represent color values for single pixels*/
2
3 #pragma once
4 #include <cstdint> //defines integral types int8_t etc.
5
6 //RGB (8-bit) color scale
7 struct RGB {
8     uint8_t r = 255;
9     uint8_t g = 255;
10     uint8_t b = 255;
11 };
12
13 std::string color_to_ansi( RGB color ) {
14     return "\033[38;2;" + std::to_string(color.r) + ";" + std::to_string(color.g) + ";" +
       std::to_string(color.b) + "m";
15 }
16
17 //Mono color scale (binary values)
18 struct Mono {
19     bool b = 1; //b=1: black, b=0: white
20 };
21
22 std::string color_to_ansi( Mono color ) {
23     if( color.b == 1 ) return "\033[30m";
24     if( color.b == 0 ) return "\033[37m";
25 }
```

5.3 image.h

```
1 #pragma once
2
3 #include <cstddef> //std::size_t
4 #include <iostream>
5 #include <stdexcept>
6 #include <utility> //std::as_const()
7 #include <vector>
8 #include "../header/color_models.h"
```

```
9 #include "../header/pixel.h"
10
17 template <typename T1 = Mono, typename T2 = char> //T1: color model, T2: type of each pixel (e.g. char)
18 class Image
19 {
20 public:
21     /*Constructors and destructors*/
22     Image();
23     Image(std::size_t width, std::size_t height);
24     Image(std::size_t width, std::size_t height, const Pixel<T1,T2> &p);
25     virtual ~Image() = default;
26
27     /*Getters and setters*/
28     virtual float getAspectRatio() const;
29     virtual void setAspectRatio( float aspectRatio );
30     virtual const Pixel<T1,T2>& at( std::size_t i, std::size_t j ) const; //Returns the Pixel at position
       (i,j) as const reference
31     virtual Pixel<T1,T2>& at( std::size_t i, std::size_t j ); //Returns the Pixel at position (i,j) as
       non-const reference
32     virtual const std::vector< std::vector< Pixel<T1,T2> > >& getPixels() const;//Return a const
       reference to the row major matrix of pixels
33     virtual void setPixels( const Pixel<T1,T2> &p ); //Set all pixels in the image equal to p
34     virtual bool getFillSpaces() const;
35     virtual void setFillSpaces( bool fillSpaces );
36     virtual std::size_t getHeight() const;
37     virtual void setHeight( std::size_t height );
38     virtual std::size_t getWidth() const;
39     virtual void setWidth( std::size_t width );
40
41     /*Output*/
42     virtual void show() const; //Show the image in the terminal
43
44 private:
45     float mAspectRatio = 1.; //AspectRatio of the figure (so far: rounded to nearest int!). The aspect
       ratio is only used in the output. Note: The real aspect ratio in the output depends on the terminal
       simulator in use.
46     bool mFillSpaces = false; //If true and mAspectRatio != 1.: spaces are filled with symbol to the
       left. Otherwise spaces are left blank.
47     std::vector< std::vector< Pixel<T1,T2> > > mPixels; //row major matrix of pixels: pixel =
       mPixels.at(iRow).at(iColumn)
48     int mWidth, mHeight; //Dimensions of the image in numbers of pixels (independent of mAspectRatio!)
49     virtual void verifyCoordinate(std::size_t i, std::size_t j) const; //Verify that (i,j) is within
       limits of the image size (throws std::out_of_range() if not)
50 };
51
52 /*Include method definitions (needed here for the compiler because it is a class template!)*/
53 #include "../src/image.cpp"
```

5.4 main.h

```
1 int main( int argc, char** argv );
```

5.5 pixel.h

```
1 #pragma once
2
3 /*Pixel represents a single "pixel" in an image.
4 - The pixel is represented by a symbol of Type T2 (e.g. char 'x') and a color of color model T1 (e.g. Mono
      with b = 0 -> black)
5 */
6
7 #include <iostream>
8 #include <stdexcept>
9 #include "../header/color_models.h"
10
11 template <typename T1 = Mono, typename T2 = char> //T1: color model, T2: type of each pixel (e.g. char)
12 class Pixel
13 {
14 public:
15     /*Constructors and destructors*/
16     Pixel() = default; //Construct an empty image of size 1x1
17     Pixel( T1 color, T2 symbol ); //Construct an empty image of size 1x1
18
19     /*Getters and setters*/
20     T1 getColor() const;
21     void setColor( T1 color );
22     T2 getSymbol() const;
23     void setSymbol( T2 symbol );
24
```

```
25 private:
26     T1 mColor; //Color of the pixel
27     T2 mSymbol; //Symbol representing the pixel if displayed in the terminal
28     void verifyCoordinate(std::size_t i, std::size_t j) const; //Verify that (i,j) is within limits of
       the image size (throws std::out_of_range() if not)
29 };
30
31 template <typename T1, typename T2>
32 std::ostream& operator«( std::ostream& ostr, const Pixel<T1,T2>& p ); //Outputs the (colored) pixel p to
       the stream ostr
33
34 /*Include method definitions (needed here for the compiler because it is a class template!)*/
35 #include "../src/pixel.cpp"
```

5.6 plot2d.h

```
1 #pragma once
2
3 /* Plot2D is a special Image (i.e. derived from class Image) representing a plot of discrete
       two-dimensional data. It provides:
4     - Axes
5     - Labels
6     -
7     -
8
9 */
10
11 #include <algorithm> //std::swap
12 #include <memory> //Smart pointers
13 #include <string>
14 #include <utility> //std::pair
15
16 #include "../header/axis.h"
17 #include "../header/color_models.h"
18 #include "../header/image.h"
19 #include "../header/pixel.h"
20
21 using ImageCoordinate = std::pair<std::size_t,std::size_t>; //Pair (jh,jv) of horizontal and vertical
       indices in the base image
22 using DataPointXY = std::pair<double,double>; //Pair (x,y) of data values plotted in the image
23 using DataSet = std::vector<DataPointXY>; //DataSet corresponding to one "curve" in the plot
24
25 template <typename T = Mono>
26 class Plot2D : public Image<T,std::string>
27 {
28 private:
29     inline static const Pixel<T,std::string> emptyPixel = Pixel<T,std::string>(T()," ");
30
31     std::size_t mWidth;
32     std::size_t mHeight;
33
34     std::vector<Axis<T» mHorizontalAxes;
35     std::vector<Axis<T» mVerticalAxes;
36
37     std::vector<std::pair<std::string,ImageCoordinate» mText; //Stores text in the image together with
       the index pair (jh,jv) of its position
38
39     /*Index ranges for the part of the image showing the data (i.e. within the frame)*/
40     ImageCoordinate mDataFrameStart; //Index pair (jh,jv) of top left corner
41     ImageCoordinate mDataFrameEnd; //Index pair (jh,jv) of lower right corner
42     std::pair<double,double> mXPlotRange; //Plot range on the x axis
43     std::pair<double,double> mYPlotRange; //Plot range on the y axis
44
45     //Functions defining the scaling of axes (i.e. transformation from original values to relative
       position on axes)
46     std::function<double(double)> mXScalingFunction = [](double x){ return x; }; //Default: linear
47     std::function<double(double)> mYScalingFunction = [](double y){ return y; }; //Default: linear
48
49     //Vector of pointers to vectors holding the data shown in the plot for every data (i.e. for every
       curve).
50     //Every data set consists of a vector of pairs of doubles (x and y values)
51     std::vector< std::shared_ptr<const DataSet> > mDataSets;
52
53     //Vector of ImageCoordinates in the base image for Pixels of every data set shown in the same order
       as mDataSets (i.e. mDataSets.at(j) corresponds to mImageCoordinatesOfDataPoints.at(j))
54     std::vector< std::vector<ImageCoordinate> > mCoordinatesOfDataPoints;
55
56     //Markers for every curve in the same order as mData (i.e. mPlotMarkers.at(j) belongs to mData.at(j)
       and mDataPixels.at(j))
57     std::vector< Pixel<T,std::string> > mPlotMarkers;
58
59     void mClear(); //Clears the data frame, i.e. sets all pixels representing data points to emptyPixel
60     void mDataSetToCoordinates( std::size_t jDataSet ); //Set mDataPoints.at(jData) for the data in
       mData.at(jData), removes the previous pixels (if any) in the image and sets the new ones.
```

```
61      void mAllDataSetsToCoordinates(); //Set mDataPoints all data sets in mData.at(jData), removes the
        previous pixels (if any) in the image and sets the new ones.
62      ImageCoordinate mDataPointToCoordinate( const DataPointXY& dataPoint ); //Transforms a pair of
        original data values to a point (index pair) in the image, taking into account the scaling functions
        and data frame boundaries
63
64  public:
65      /*Constructors and destructors*/
66      //Plot2D() = default; //Constructs an empty plot with all values set to zero or equivalent
67      Plot2D( std::size_t width, std::size_t height );
68
69      /*---Getters and Setters----*/
70      void setXAxisScaling( const std::function<double(double)>& xScalingFunction );
71      void setYAxisScaling( const std::function<double(double)>& yScalingFunction );
72      void setPlotRange( const std::pair<double,double>& xPlotRange, const std::pair<double,double>&
        yPlotRange );
73      /*-------------------------*/
74
75      /*---------Modifiers--------*/
82      void addVerticalAxis( std::size_t hPos, std::size_t vPosStart, std::size_t vPosEnd, T color );
83
90      void addHorizontalAxis( std::size_t vPos, std::size_t hPosStart, std::size_t hPosEnd, T color );
91
99      void addVerticalAxis( std::size_t hPos, std::size_t vPosStart, std::size_t vPosEnd, T color, const
        std::string& label, float relativeLabelPosition = 0.5, bool labelLeft = true, bool rotateLabel =
        false ); //Adds a vertical axis with label
100
107      void addHorizontalAxis( std::size_t vPos, std::size_t hPosStart, std::size_t hPosEnd, T color, const
        std::string& label, float relativeLabelPosition = 0.5, bool labelBelow = true ); //Adds a horizontal
        axis with label
108      void addText( std::size_t hPos, std::size_t vPos, const std::string& text ); //Adds (horizontal)
        text starting at position (hPos,vPos)
109      void addDataSet( std::shared_ptr<const DataSet> dataSet, Pixel<T,std::string> plotMarker );
110
111      //"REMOVERS"
112      //void flush(); //Removes all data and data points
113
114      /*-------------------------*/
115
116      /*----------Operators----------*/
117
118      /*-----------------------------*/
119  };
120
121  /*Include method definitions (needed here for the compiler because it is a class template!)*/
122  #include "../src/plot2d.cpp"
```

Index