MooViE

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 angle_helper Namespace Reference

Functions

- double deg_to_rad (double deg)

    deg_to_rad
- double rad_to_deg (double rad)

    rad_to_deg
- double **rad_dist** (double rad0, double rad1)

4.1.1 Detailed Description

A namespace for converter functions.

4.1.2 Function Documentation

4.1.2.1 deg_to_rad()

```
double angle_helper::deg_to_rad (
             double deg )  [inline]
```

deg_to_rad

Converts degree to radian value.

Parameters

deg	the degree value to be converted

Returns

the matching radian value

4.1.2.2 rad_to_deg()

```
double angle_helper::rad_to_deg (
            double rad )  [inline]
```

rad_to_deg

Converts radian to degree value.

Parameters

rad	the radian value to be converted

Returns

the matching degree value

Chapter 5

Class Documentation

5.1 Angle Class Reference

The Angle class.

```
#include <Coordinates.h>
```

Public Member Functions

- Angle (double angle)
  Angle.
- double value () const
  get
- double operator= (const double &angle)
  this = rhs
- bool operator== (const Angle &rhs) const
  this == rhs
- bool operator< (const Angle &rhs) const
  this < rhs
- bool operator<= (const Angle &rhs) const
  this <= rhs
- bool operator> (const Angle &rhs) const
  this == rhs
- bool operator>= (const Angle &rhs) const
  operator >=
- Angle & operator+= (const Angle &rhs)
  this += rhs
- Angle operator+ (const Angle &rhs) const
  this + rhs
- Angle & operator-= (const Angle &rhs)
  this -= rhs
- Angle operator- (const Angle &rhs) const
  this - rhs
- Angle & operator*= (double val)
  *this *= val*
- Angle operator* (double val) const
  *operator this * val*
- Angle & operator/= (double val)
  this /= val
- Angle operator/ (double val)
  this / val

Static Public Member Functions

- static Angle interpolate (const Angle &a1, const Angle &a2, double p)

    interpolate
- static Angle center (const Angle &a1, const Angle &a2)

    center

5.1.1 Detailed Description

The Angle class.

Angle is a wrapper class for angle values. Angles are stored as radian values. For consistence, its value needs to be in [0,2∗pi].

Author

beyss

Date

03.07.2017

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Angle()

```
Angle::Angle (
            double angle )  [inline]
```

Angle.

Creates a Angle from an angle value. If necessary, the value is corrected to be consistent.

Parameters

angle	the angle value

5.1.3 Member Function Documentation

5.1.3.1 center()

```
static Angle Angle::center (
            const Angle & a1,
            const Angle & a2 )  [inline], [static]
```

center

Returns the Angle in the center of two given Angles.

Parameters

a1	the first Angle
a2	the second Angle

Returns

the centered Angle

5.1.3.2 interpolate()

```
static Angle Angle::interpolate (
             const Angle & a1,
             const Angle & a2,
             double p )  [inline], [static]
```

interpolate

Returns an Angle that is (1-p) percent of a1 and p percent of a2. To be consistent, p should be in [0,1].

Parameters

a1	the first angle
a2	the second angle
p	the percentage

Returns

the interpolated Angle

5.1.3.3 operator∗()

```
Angle Angle::operator* (
             double val ) const  [inline]
```

operator this ∗ val

Multiplication operator returning an Angle with the value of adjusted this ∗ val.

Parameters

val	the factor

Returns

a new Angle equal to this ∗ val

5.1.3.4 operator∗=()

```
Angle& Angle::operator*= (
             double val )  [inline]
```

this ∗= val

Multiplication assignment operator multiplying this Angle's value with the given double value. If necessary, the value is corrected to be consistent.

Parameters

rhs	the factor

Returns

a reference to this angle

5.1.3.5 operator+()

```
Angle Angle::operator+ (
             const Angle & rhs ) const  [inline]
```

this + rhs

Friend addition operator returning an Angle equal to the return of this += rhs. It operates on a copy of lhs so that the original object is not changed.

Parameters

rhs	the right operand Angle

Returns

a new Angle equal to this + rhs

5.1.3.6 operator+=()

```
Angle& Angle::operator+= (
             const Angle & rhs )  [inline]
```

this += rhs

Addition assignment operator increasing this Angle's value by the other Angle's value. If necessary, the value is corrected to be consistent.

Parameters

rhs	the other Angle

Returns

a reference to this angle

5.1.3.7 operator-()

```
Angle Angle::operator- (
            const Angle & rhs ) const  [inline]
```

this - rhs

Friend addition operator returning an Angle equal to the return of this - rhs. It operates on a copy of lhs so that the original object is not changed.

Parameters

rhs	the right operand Angle

Returns

a new Angle equal to this - rhs

5.1.3.8 operator-=()

```
Angle& Angle::operator-= (
            const Angle & rhs )  [inline]
```

this -= rhs

Subtraction assignment operator decreasing this Angle's value by the other Angle's value. If necessary, the value is corrected to be consistent.

Parameters

rhs	the other angle

Returns

a reference to this angle

5.1.3.9 operator/()

```
Angle Angle::operator/ (
            double val )  [inline]
```

this / val

Division operator returning an Angle with the value of adjusted this / val.

Parameters

val	the dividend

Returns

a new Angle equal to this / val

5.1.3.10 operator/=()

```
Angle& Angle::operator/= (
            double val )  [inline]
```

this /= val

Division assignment operator divides this Angle's value by the given double value. If necessary, the value is corrected to be consistent.

Parameters

val	the dividend

Returns

a reference to this angle

5.1.3.11 operator<()

```
bool Angle::operator< (
            const Angle & rhs ) const  [inline]
```

this $<$ rhs

Smaller than operator checking wether this Angle's value is smaller than the other Angle's value.

Parameters

rhs	the other Angle

Returns

if smaller than or not

5.1.3.12 operator<=()

```
bool Angle::operator<= (
             const Angle & rhs ) const  [inline]
```

this <= rhs

Smaller than or equal to operator checking wether this Angle's value is smaller than or equal to the other Angle's value.

Parameters

rhs	the other Angle

Returns

if smaller than or equal or not

5.1.3.13 operator=()

```
double Angle::operator= (
             const double & angle )  [inline]
```

this = rhs

Assignment operator setting this Angle's value. If necessary, the value is corrected to be consistent.

Parameters

angle	

Returns

5.1.3.14 operator==()

```
bool Angle::operator== (
            const Angle & rhs ) const  [inline]
```

this == rhs

Equal to operator checking wether this Angle's value is equal to the other Angle's value.

Parameters

rhs	the other Angle

Returns

if equal or not

5.1.3.15 operator>()

```
bool Angle::operator> (
            const Angle & rhs ) const  [inline]
```

this == rhs

Greater than operator checking wether this Angle's value is greater than the other Angle's value.

Parameters

rhs	the other Angle

Returns

if greater than or not

5.1.3.16 operator>=()

```
bool Angle::operator>= (
            const Angle & rhs ) const  [inline]
```

operator >=

Greater than or equal to operator checking wether this Angle's value is smaller than or equal to the other Angle's value.

Parameters

rhs	the other Angle

Returns

if greater than or equal or not

5.1.3.17 value()

```
double Angle::value ( ) const  [inline]
```

get

Returns the value of this angle.

Returns

the angle value

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

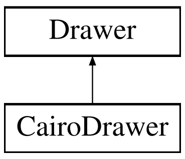
- include/Coordinates.h

5.2 CairoDrawer Class Reference

CairoDrawer draws on a SVG surface and stores it to a file.

`#include <CairoDrawer.h>`

Inheritance diagram for CairoDrawer:



Public Member Functions

- **CairoDrawer** (const std::string &fpath, int width, int height, std::size_t _num_inputs)
- virtual void change_surface (const std::string &fpath, int width, int height)

  changes the underlying surface by the given parameters
- virtual void draw_codomain_grid (const CodomainGrid &grid)

  draws a CodomainGrid
- virtual void draw_domain_axis (const DomainAxis &axis)

  draws a DomainAxis
- virtual void draw_relation_element (const RelationElement &link)

  draws a RelationElement
- virtual void finish ()

  save results

Static Public Attributes

- static const double **RADIAL_TEXT_FACTOR**
- static const double **COORDGRID_ADJUSTMENT**
- static const double **COORDPOINT_ANGLE**
- static const double **COORDGRID_DESCRIPTION_ANGLE**
- static const double **END_RADIUS_MAJOR_FACTOR**
- static const double **END_RADIUS_MINOR_FACTOR**
- static const double **RADIUS_TICK_LABEL_FACTOR**
- static const double **DATA_LINK_LINE_WIDTH**
- static const double **CONNECTOR_ARROW_HEIGHT**
- static const double **RADIUS_LABEL_DELTA**
- static const double **RADIUS_HISTOGRAM_DELTA**
- static const double **CONNECTOR_DELTA**
- static const double **TEXT_DELTA**
- static const double **ANGLE_DELTA_SMALL**
- static const double **ANGLE_DELTA_MEDIUM**
- static const double **ANGLE_DELTA_LARGE**
- static const double **RADIUS_DELTA**
- static const double **OUTPUT_EXTREME_RADIUS_DELTA**
- static const double **OUTPUT_LABEL_LINE_END_DELTA**
- static const double **OUTPUT_LABEL_RADIUS_DELTA**

Protected Member Functions

- virtual void set_surface (const std::string &fpath, int width, int height)

  hard-sets the underlying surface by the given parameters
- virtual void draw_histogram (const DomainAxis::Histogram &histogram, double radius, const Angle &start, const Angle &end)

  draws a Histogram
- virtual void draw_link (const Polar &origin1, const Polar &origin2, const Polar &target1, const Polar &target2, const DrawerProperties<> &prop)

  draws a link
- virtual void draw_connector (const Polar &from, const Polar &to, const DrawerProperties<> &prop)

  draws a connector
- virtual void draw_segment_axis (double inner_radius, double thickness, const Angle &start, const Angle &end, const DrawerProperties< std::array< Color, 10 >> &prop, Direction dir)

  draws a split axis
- virtual void draw_output_label (const Label &output_label, double radius_label, double radius_output, const Angle &begin, const Angle &end)

  draws an output label
- virtual void draw_arrow (const Polar &start, const DrawerProperties<> &prop)

  draws arrow
- virtual void draw_ring_segment (double radius, double thickness, const Angle &begin, const Angle &end, const DrawerProperties<> &prop, Direction dir)

  draws a ring segment
- virtual void draw_connector_segment (double begin_radius, double begin_angle, double end_radius, double end_angle, const DrawerProperties<> &prop)

  draws a connector Bezier curve
- virtual void draw_line (const Polar &from, const Polar &to, const DrawerProperties<> &prop)

  draws a simple line
- virtual void draw_arc (double inner_radius, const Angle &start, const Angle &end, Direction dir)

draws an arc
- virtual void draw_coord_point (const Polar &coord, const Angle &width, double height, const Drawer←Properties<> &prop)

  draws an error box
- virtual void draw_text_parallel (const Label &label, const Polar &start, const TextAlignment &alignment=Text←Alignment::CENTERED)

  draws a Label on a line to the middle
- virtual void draw_text_orthogonal (const Label &label, const Polar &start, const TextAlignment &alignment=TextAlignment::CENTERED)

  draws a Label orthogonal to a line to the middle
- void set_font_face (const Label &label)

  set font style
- Cairo::TextExtents **get_text_extents** (const Label &label) const
- Angle get_cairo_angle (const Angle &angle)

Additional Inherited Members

5.2.1 Detailed Description

CairoDrawer draws on a SVG surface and stores it to a file.

CairoDrawer is a wrapper class for MooViE's basic drawing abilities which are realized using Cairo.

Author

beyss

Date

05.07.2017

5.2.2 Member Function Documentation

5.2.2.1 change_surface()

```
virtual void CairoDrawer::change_surface (
             const std::string & fpath,
             int width,
             int height )  [virtual]
```

changes the underlying surface by the given parameters

Alters the surface of this Drawer in with, height and storage path. All unsafed changes will be stored and all kept resources freed correctly.

Parameters

fpath	a string containing an valid existing or accessible not existing path
width	an integer between 0 and MAX_INT
height	an integer between 0 and MAX_INT

Implements Drawer.

5.2.2.2 draw_arc()

```
virtual void CairoDrawer::draw_arc (
             double inner_radius,
             const Angle & start,
             const Angle & end,
             Direction dir )  [protected], [virtual]
```

draws an arc

Draws a simple edge segment around the center of its coordinate system between the two given Angles and with the given radius.

Parameters

inner_radius	the inner radius
start	the start Angle
end	the end Angle
dir	the direction

Implements Drawer.

5.2.2.3 draw_arrow()

```
virtual void CairoDrawer::draw_arrow (
             const Polar & start,
             const DrawerProperties<> & prop )  [protected], [virtual]
```

draws arrow

Draws a arrow head from a given start pointing.

Parameters

start	the start of the arrow head
prop	DrawerProperties for the arrow head

Implements Drawer.

5.2.2.4 draw_codomain_grid()

```
virtual void CairoDrawer::draw_codomain_grid (
            const CodomainGrid & grid )  [virtual]
```

draws a CodomainGrid

Draws a CodomainGrid using its radius and angles. For thin or thick lines the properties given by the Configuration instance are used. On

Parameters

grid	the CodomainGrid to draw

Implements Drawer.

5.2.2.5 draw_connector()

```
virtual void CairoDrawer::draw_connector (
            const Polar & from,
            const Polar & to,
            const DrawerProperties<> & prop )  [protected], [virtual]
```

draws a connector

Draws a connection between to given polar coordinates. The connection is a bezier curve which is controlled by automatically generated control points.

Parameters

from	the start Polar
to	the end Polar
prop	the DrawerProperties

Implements Drawer.

5.2.2.6 draw_connector_segment()

```
virtual void CairoDrawer::draw_connector_segment (
            double start_radius,
            double start_angle,
            double end_radius,
            double end_angle,
            const DrawerProperties<> & prop )  [protected], [virtual]
```

draws a connector Bezier curve

Draws a Bezier curve from Polar(start_radius, start_angle) to Polar(end_radius, end_angle) which approximately behaves like Archimedean spiral. If the smaller difference angle between start_angle and end_angle is bigger than PI, the spiral will be approximated by two Bezier curves.

Parameters

start_radius	the radius of the starting point
start_angle	the angle of the starting point
end_radius	the radius of the end point
end_angle	the angle of the end point
prop	the DrawerProperties for the segment

Implements Drawer.

5.2.2.7 draw_coord_point()

```
virtual void CairoDrawer::draw_coord_point (
             const Polar & coord,
             const Angle & width,
             double height,
             const DrawerProperties<> & prop )  [protected], [virtual]
```

draws an error box

Draws a coordinate point with given height and with.

Parameters

coord	the polar coordinate to draw
width	the width
height	the height
prop	the DrawerProperties

Implements Drawer.

5.2.2.8 draw_domain_axis()

```
virtual void CairoDrawer::draw_domain_axis (
             const DomainAxis & axis )  [virtual]
```

draws a DomainAxis

Draws a DomainAxis using its radius and angles. For thin or thick lines the properties given by the Configuration instance are used.

Parameters

axis	the DomainAxis to draw

Implements Drawer.

5.2.2.9 draw_histogram()

```
virtual void CairoDrawer::draw_histogram (
             const DomainAxis::Histogram & histogram,
             double radius,
             const Angle & start,
             const Angle & end )  [protected], [virtual]
```

draws a Histogram

Draws a Histogram from the given radius, between begin and end Angle. For the histogram height, thin or thick lines the properties given by the Configuration instance are used.

Parameters

histogram	the Histogram to draw
radius	the start radius of the Histogram
start	the starting angle of the Histogram
end	the end angle of the Histogram

Implements Drawer.

5.2.2.10 draw_line()

```
virtual void CairoDrawer::draw_line (
             const Polar & from,
             const Polar & to,
             const DrawerProperties<> & prop )  [protected], [virtual]
```

draws a simple line

Draws a line from a given starting vertice to a given end vertice.

Parameters

from	the starting coordinates
to	the end coordinates
prop	the DrawerProperties to use

Implements Drawer.

5.2.2.11 draw_link()

```
virtual void CairoDrawer::draw_link (
             const Polar & origin1,
             const Polar & origin2,
             const Polar & target1,
             const Polar & target2,
             const DrawerProperties<> & prop )  [protected], [virtual]
```

draws a link

Draws a bold line between the lines origin1-origin2 and target1-target2. This is realized by drawing Bezier curves from origin1 to target1 and from origin2 to target2 and filling the so created surface.

Parameters

origin1	first origin coordinate
origin2	second origin coordinate
target1	first target coordinate
target2	second target coordinate
prop	DrawerProperties for the link

Implements Drawer.

5.2.2.12 draw_output_label()

```
virtual void CairoDrawer::draw_output_label (
             const Label & output_label,
             double radius_label,
             double radius_output,
             const Angle & begin,
             const Angle & end )  [protected], [virtual]
```

draws an output label

Draws the given Label output_label with the radius radius_label and a descriptive path that connects the output label with the associated output. The path consists of an arc segment and a line.

Parameters

output_label	the output label to draw
radius_label	the radius of the output label
radius_output	the radius of the associated output
begin	the angle at which the output ends
end	the angle at which the arc ends

Implements Drawer.

5.2.2.13 draw_relation_element()

```
virtual void CairoDrawer::draw_relation_element (
             const RelationElement & elem )  [virtual]
```

draws a RelationElement

Draws a RelationElement using its coordinates.

Parameters

elem	the RelationElement to draw

Implements Drawer.

5.2.2.14 draw_ring_segment()

```
virtual void CairoDrawer::draw_ring_segment (
             double radius,
             double thickness,
             const Angle & start,
             const Angle & end,
             const DrawerProperties<> & prop,
             Direction dir )  [protected], [virtual]
```

draws a ring segment

Draws a filled ring segment around the center of its coordinate system between the two given Angles and with the given radius.

Parameters

radius	the radius
thickness	the thinkness of the edge segment
begin	the begin Angle
end	the end Angle
prop	the CairoDrawer properties
dir	the direction

Implements Drawer.

5.2.2.15 draw_segment_axis()

```
virtual void CairoDrawer::draw_segment_axis (
                double inner_radius,
                double thickness,
                const Angle & begin,
                const Angle & end,
                const DrawerProperties< std::array< Color, 10 >> & prop,
                Direction dir )  [protected], [virtual]
```

draws a split axis

Draws a circle segment which is itself divided in colored segments.

Parameters

inner_radius	inner radius of the split axis
thickness	width of the split axis
begin	angle of the segments begin
end	angle of the segments end
prop	color
dir	direction of the split axis' colors

Implements Drawer.

5.2.2.16 draw_text_orthogonal()

```
virtual void CairoDrawer::draw_text_orthogonal (
                const Label & label,
                const Polar & start,
                const TextAlignment & alignment = TextAlignment::CENTERED )  [protected], [virtual]
```

draws a Label orthogonal to a line to the middle

Draws the given label orthogonal to the angle of the given coordinate's angle.

Parameters

label	the label to draw
start	the coordinate to adjust to

Implements Drawer.

5.2.2.17 draw_text_parallel()

```
virtual void CairoDrawer::draw_text_parallel (
                const Label & label,
                const Polar & start,
                const TextAlignment & alignment = TextAlignment::CENTERED )  [protected], [virtual]
```

draws a Label on a line to the middle

Draws the given label with the same angle like the given coordinate.

Parameters

label	the label to draw
start	the coordinate to adjust to

Implements Drawer.

5.2.2.18 finish()

```
virtual void CairoDrawer::finish ( )  [virtual]
```

save results

Save the Drawer's result to the given file.

Implements Drawer.

5.2.2.19 get_cairo_angle()

```
Angle CairoDrawer::get_cairo_angle (
             const Angle & angle )  [inline], [protected]
```

Cairo uses an non-standard way to define angles. The angle of 0 is on the positive X axis, but the angle of pi/2 or 90ř is on the negative Y axis (the common model uses the positive Y axis).

Parameters

angle	

Returns

the cairo angle

5.2.2.20 set_font_face()

```
void CairoDrawer::set_font_face (
             const Label & label )  [protected]
```

set font style

Sets the font face according to the TextProperties of the given Label.

Parameters

label	the Label whose properties to set

5.2.2.21 set_surface()

```
virtual void CairoDrawer::set_surface (
             const std::string & fpath,
             int width,
             int height )  [protected], [virtual]
```

hard-sets the underlying surface by the given parameters

Alters the surface of this Drawer in with, height and storage path.

Parameters

fpath	a string containing an valid or accessible path
width	an integer between 0 and MAX_INT
height	an integer between 0 and MAX_INT

Implements Drawer.

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

- include/CairoDrawer.h

5.3 Cartesian Class Reference

The Cartesian class.

`#include <Coordinates.h>`

Public Member Functions

- Cartesian (double x=0, double y=0)

  Cartesian.
- bool operator== (const Cartesian &rhs) const

  operator ==
- const double & x () const

  x
- double & x ()

  x
- const double & y () const

  y
- double & y ()

  y

Static Public Member Functions

- static Cartesian interpolate (const Cartesian &p1, const Cartesian &p2, double p)

  interpolate
- static Cartesian center (const Cartesian &p1, const Cartesian &p2)

  center

5.3.1 Detailed Description

The Cartesian class.

Cartesian represents a tupel from the Rš as cartesian coordinate.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Cartesian()

```
Cartesian::Cartesian (
             double x = 0,
             double y = 0 )  [inline]
```

Cartesian.

Creates a cartesian coordinate from given x and y value.

Parameters

x	the x value
y	the y value

5.3.3 Member Function Documentation

5.3.3.1 center()

```
static Cartesian Cartesian::center (
             const Cartesian & p1,
             const Cartesian & p2 )  [inline], [static]
```

center

Returns a Cartesian centered between two given Cartesian.

Parameters

p1	the first Cartesian
p2	the second Cartesian

Returns

the centered Cartesian

5.3.3.2 interpolate()

```
static Cartesian Cartesian::interpolate (
            const Cartesian & p1,
            const Cartesian & p2,
            double p )  [inline], [static]
```

interpolate

Returns an Cartesian whose radius and Angle are (1-p) percent of p1's and p percent of p2's radius and Angle. To be consistent, p should be in [0,1].

Parameters

p1	the first Cartesian
p2	the second Cartesian
p	the percentage

Returns

the interpolated Cartesian

5.3.3.3 operator==()

```
bool Cartesian::operator== (
            const Cartesian & rhs ) const  [inline]
```

operator ==

Equal to operator checking for equality of x and y.

Parameters

rhs	the other Cartesian

Returns

if equal or not

5.3.3.4 x() [1/2]

```
const double& Cartesian::x ( ) const  [inline]
```

x

Access function for this Cartesian's x value as readonly.

Returns

a constant reference to this Cartesians x value

5.3.3.5 x() [2/2]

```
double& Cartesian::x ( )  [inline]
```

x

Access function for this Cartesian's x value.

Returns

a reference to this Cartesians x value

5.3.3.6 y() [1/2]

```
const double& Cartesian::y ( ) const  [inline]
```

y

Access function for this Cartesian's y value as readonly.

Returns

a constant reference to this Cartesians y value

5.3.3.7 y() [2/2]

```
double& Cartesian::y ( )  [inline]
```

y

Access function for this Cartesian's y value.

Returns

a reference to this Cartesians y value

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

- include/Coordinates.h

5.4 DataSet< T >::Cell Struct Reference

the Cell struct

```
#include <DataSet.h>
```

Public Member Functions

- Cell ()
- Cell (T value_)

Public Attributes

- const bool null
- const T value

5.4.1 Detailed Description

template<typename T>
struct DataSet< T >::Cell

the Cell struct

Stores the value of a cell. The value is 0 if the Cell is a null cell.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Cell() [1/2]

```
template<typename T >
DataSet< T >::Cell::Cell ( )  [inline]
```

Creates a new null Cell.

5.4.2.2 Cell() [2/2]

```
template<typename T >
DataSet< T >::Cell::Cell (
             T value_ )  [inline]
```

Creates a new non-null Cell storing the value of T

5.4.3 Member Data Documentation

5.4.3.1 null

```
template<typename T >
const bool DataSet< T >::Cell::null
```

Null or not

5.4.3.2 value

```
template<typename T >
const T DataSet< T >::Cell::value
```

The value of the cell

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

- include/DataSet.h

5.5 CodomainGrid Class Reference

The CoordGrid class.

```
#include <CodomainGrid.h>
```

Public Member Functions

- CodomainGrid (const std::vector< DefVariable > &_output_vars, const Angle &_start, const Angle &_end, double _radius, double _height, Direction _dir)

  constructor
- const DefVariable & get_var (std::size_t num_output) const

  gets output variable
- std::size_t get_num_outputs () const

  gets number of outputs
- const Angle & get_start () const

  gets the start Angle
- void set_start (const Angle &_start)

  sets the start Angle
- const Angle & get_end () const

  gets the end Angle
- void set_end (const Angle &_end)

  gets the end Angle
- double get_radius () const

  gets the radius
- void set_radius (double _radius)

  sets the radius
- double get_height () const

  gets the height
- void set_height (double _height)

  sets the height
- Direction get_direction () const

  gets the Direction
- void set_direction (Direction _dir)

  sets the Direction
- const MultiScale & get_scale () const

  gets the MultiScale

5.5.1 Detailed Description

The CoordGrid class.

Representing a coordinate grid by its dimensional constraints. 4 outputs, 4 scale ticks

Author

beyss

Date

26.07.2017

5.5.2 Constructor & Destructor Documentation

5.5.2.1 CodomainGrid()

```
CodomainGrid::CodomainGrid (
             const std::vector< DefVariable > & _output_vars,
             const Angle & _start,
             const Angle & _end,
             double _radius,
             double _height,
             Direction _dir )
```

constructor

Creates a CodomainGrid presenting given variables and is drawn between given angles with given radius and height.

Parameters

_output_vars	a vector containing the output variables
_start	the start angle
_end	the end angle
_radius	the radius from the center
_height	the height beginning at the radius
_dir	the Direction the outputs values increase

5.5.3 Member Function Documentation

5.5.3.1 get_direction()

```
Direction CodomainGrid::get_direction ( ) const [inline]
```

gets the Direction

Returns the direction this CodomainGrid's output values increase. The Direction is either COUNTER_CLOCKWISE (with increasing Angle) or CLOCKWISE (with decreasing Angle).

Returns

the Direction

5.5.3.2 get_end()

```
const Angle& CodomainGrid::get_end ( ) const [inline]
```

gets the end Angle

Returns the end Angle of this CodomainGrid's drawing span.

Returns

the end Angle

5.5.3.3 get_height()

```
double CodomainGrid::get_height ( ) const  [inline]
```

gets the height

Returns the height measured from the radius.

Returns

the height

5.5.3.4 get_num_outputs()

```
std::size_t CodomainGrid::get_num_outputs ( ) const  [inline]
```

gets number of outputs

Returns the total number of stored output variables.

Returns

the number of outputs

5.5.3.5 get_radius()

```
double CodomainGrid::get_radius ( ) const  [inline]
```

gets the radius

Returns the radius measured from the center of the coordinate system.

Returns

the radius

5.5.3.6 get_scale()

```
const MultiScale& CodomainGrid::get_scale ( ) const  [inline]
```

gets the MultiScale

Returns the MultiScale of this CodomainGrid. This scale instance defines how the graphical scale will be drawn for each output.

Returns

the MultiScale

5.5.3.7 get_start()

```
const Angle& CodomainGrid::get_start ( ) const  [inline]
```

gets the start Angle

Returns the start Angle of this CodomainGrid's drawing span.

Returns

the start Angle

5.5.3.8 get_var()

```
const DefVariable& CodomainGrid::get_var (
             std::size_t num_output ) const
```

gets output variable

Returns the i-th output variable. If num_output >= num_outputs an exception is thrown.

Parameters

num_output	the number of the output to return

5.5.3.9 set_direction()

```
void CodomainGrid::set_direction (
             Direction _dir )  [inline]
```

sets the Direction

Sets the direction this CodomainGrid's output values increase. The Direction is either COUNTER_CLOCKWISE (with increasing Angle) or CLOCKWISE (with decreasing Angle).

Parameters

_dir	the Direction to set

5.5.3.10 set_end()

```
void CodomainGrid::set_end (
             const Angle & _end )  [inline]
```

gets the end Angle

Sets the end Angle of this CodomainGrid's drawing span.

Parameters

_end	the end Angle to set

5.5.3.11 set_height()

```
void CodomainGrid::set_height (
             double _height )  [inline]
```

sets the height

Sets the height measured from the radius.

Parameters

_height	the height to set

5.5.3.12 set_radius()

```
void CodomainGrid::set_radius (
             double _radius )  [inline]
```

sets the radius

Sets the radius measured from the center of the coordinate system.

Parameters

_radius	the radius to set

5.5.3.13 set_start()

```
void CodomainGrid::set_start (
             const Angle & _start )  [inline]
```

sets the start Angle

Starts the start Angle of this CodomainGrid's drawing span.

Parameters

_start	the start Angle to set

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

- include/CodomainGrid.h

5.6 Color Class Reference

The Color class.

```
#include <Color.h>
```

Public Member Functions

- Color (double r=0, double g=0, double b=0, double a=1)
  Color.
- **Color** (const Color &c, double a)
- const double & r () const
  r
- const double & g () const
  g
- const double & b () const
  b
- const double & a () const
  a
- bool operator== (const Color &color) const
  this == color
- bool operator!= (const Color &color) const
  this != color
- void set_red (double red)
  set red value
- void set_green (double green)
  set green value
- void set_blue (double blue)
  set blue value
- void set_alpha (double alpha)
  set alpha value

Static Public Attributes

- static const Color BLACK

Friends

- std::ostream & operator<< (std::ostream &o, const Color &c)
  ostream operator

5.6.1 Detailed Description

The Color class.

Color class represents a color by RGB and alpha value.

Author

beyss

Date

27.07.2017

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Color()

```
Color::Color (
             double r = 0,
             double g = 0,
             double b = 0,
             double a = 1 )  [inline]
```

Color.

Creates a Color from RGB and Alpha values.

Parameters

r	the red value
g	the green value
b	the blue value
a	the alpha value

5.6.3 Member Function Documentation

5.6.3.1 a()

```
const double& Color::a ( ) const  [inline]
```

a

Access function for the color's alpha value.

Returns

a reference to the colors alpha value

5.6.3.2 b()

```
const double& Color::b ( ) const  [inline]
```

b

Access function for the color's blue value.

Returns

a reference to the colors blue value

5.6.3.3 g()

```
const double& Color::g ( ) const  [inline]
```

g

Access function for the color's green value.

Returns

a reference to the colors green value

5.6.3.4 operator"!=()

```
bool Color::operator!= (
             const Color & color ) const  [inline]
```

this != color

Checks whether or not two colors are not equal. Two colors would be equal if their RGBA values were the same.

Parameters

color	the other color

Returns

not equal or not

5.6.3.5 operator==()

```
bool Color::operator== (
             const Color & color ) const  [inline]
```

this == color

Checks whether or not two colors are equal. This is the case if RGBA values are the same.

Parameters

color	the other color

Returns

equal or not

5.6.3.6 r()

```
const double& Color::r ( ) const  [inline]
```

r

Access function for the color's red value.

Returns

a reference to the colors red value

5.6.3.7 set_alpha()

```
void Color::set_alpha (
             double alpha )  [inline]
```

set alpha value

Sets the alpha value of this Color. Input values from 0 to 255 will be automatically corrected to values from [0,1].

Parameters

alpha	the alpha value to set

5.6.3.8 set_blue()

```
void Color::set_blue (
             double blue )  [inline]
```

set blue value

Sets the blue value of this Color. Input values from 0 to 255 will be automatically corrected to values from [0,1].

Parameters

blue	the blue value to set

5.6.3.9 set_green()

```
void Color::set_green (
             double green )  [inline]
```

set green value

Sets the green value of this Color. Input values from 0 to 255 will be automatically corrected to values from [0,1].

Parameters

green	the green value to set

5.6.3.10 set_red()

```
void Color::set_red (
             double red )  [inline]
```

set red value

Sets the red value of this Color. Input values from 0 to 255 will be automatically corrected to values from [0,1].

Parameters

red	the red value to set

5.6.4 Friends And Related Function Documentation

5.6.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & o,
            const Color & c )  [friend]
```

ostream operator

Puts string representation of Color c to the output stream o.

Parameters

o	the ostream to put into
c	the color to put

5.6.5 Member Data Documentation

5.6.5.1 BLACK

```
const Color Color::BLACK  [static]
```

A Color constant representing black (0,0,0,1)

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

- include/Color.h

5.7 Configuration Class Reference

```
#include <Configuration.h>
```

Public Member Functions

- const std::string & get_input_file () const
- const std::string & get_output_file () const
- void set_output_file (const std::string &_output_file)
- int get_width () const
- void set_width (int _width)
- int get_height () const
- void set_height (int _height)
- double get_output_angle_span () const
- void set_output_angle_span (double _output_angle_span)
- double get_output_inner_radius () const
- void set_output_inner_radius (double _output_inner_radius)
- double get_output_thickness () const
- void set_output_thickness (double _output_thickness)
- double get_grid_size () const
- void set_grid_size (double _grid_size)
- int get_num_major_sections_grid () const
- void set_num_major_sections_grid (int major_sections)
- int get_num_minor_sections_grid () const
- void set_num_minor_sections_grid (int minor_sections)
- double get_input_inner_radius () const
- void set_input_inner_radius (double _input_inner_radius)
- double get_input_separation_angle () const
- void set_input_separation_angle (double _input_separation_angle)
- double get_input_thickness () const
- void set_input_thickness (double _input_thickness)
- int get_num_major_sections_axis () const
- void set_num_major_sections_axis (int major_sections)
- int get_num_minor_sections_axis () const
- void set_num_minor_sections_axis (int minor_sections)
- bool is_histograms_enabled () const
- void set_histograms_enabled (bool _histograms_enabled)
- int get_num_histogram_classes () const
- void set_num_histogram_classes (int _num_histogram_classes)
- double get_histogram_height () const
- void set_histogram_height (double _histogram_height)
- const Color & get_histogram_background () const
- void set_histogram_background (const Color &_histogram_background)
- const Color & get_histogram_fill () const
- void set_histogram_fill (const Color &_histogram_fill)
- double get_connector_arc_ratio () const
- void set_connector_arc_ratio (double _connector_arc_ratio)
- const DrawerProperties & get_prop_thick () const
- void set_prop_thick (const DrawerProperties<> &_prop_thick)
- const DrawerProperties & get_prop_thin () const
- void set_prop_thin (const DrawerProperties<> &_prop_thin)
- const TextProperties & get_prop_scale_label () const
- void set_prop_scale_label (const TextProperties &_prop_scale_label)
- const TextProperties & get_prop_axis_label () const
- void set_prop_axis_label (const TextProperties &_prop_axis_label)

Static Public Member Functions

- static Configuration & get_instance ()
- static void initialize (const std::string &fname, const std::string &cpath)
- static void initialize (const std::string &fname)

Static Public Attributes

- static const std::array< Color, 10 > GLOW_10
- static const Triangle< Color, 12 > SET3
- static const Color SET2_3_1
- static const Color **SET2_3_2**
- static const Color **SET2_3_3**

5.7.1 Detailed Description

A class wrapping the settings and information that is necessary for a MooViE run. Configuration is implemented as a singelton. Before calling Configuration::get_instance to get the singleton instance Configuration::initialize need to be called once.

Author

stratmann

Date

16.01.2018

5.7.2 Member Function Documentation

5.7.2.1 get_connector_arc_ratio()

```
double Configuration::get_connector_arc_ratio ( ) const  [inline]
```

Returns the ratio of the radial distance between two data points that will be drawn as connector.

Returns

the connector arc ratio

5.7.2.2 get_grid_size()

```
double Configuration::get_grid_size ( ) const  [inline]
```

Returns the size of actual grid that is a part of the CodomainGrid.

Returns

the grid_size

5.7.2.3 get_height()

```
int Configuration::get_height ( ) const  [inline]
```

Returns the height of the MooViE scene

Returns

the height

5.7.2.4 get_histogram_background()

```
const Color& Configuration::get_histogram_background ( ) const  [inline]
```

Returns the background color that each histogram has.

Returns

the histogram background color

5.7.2.5 get_histogram_fill()

```
const Color& Configuration::get_histogram_fill ( ) const  [inline]
```

Returns the fill color of each histogram's bars.

Returns

the histogram fill color

5.7.2.6 get_histogram_height()

```
double Configuration::get_histogram_height ( ) const  [inline]
```

Returns the height that each histogram has.

Returns

the histogram height

5.7.2.7 get_input_file()

```
const std::string& Configuration::get_input_file ( ) const  [inline]
```

Returns the path to the input file.

Returns

the input file path

5.7.2.8 get_input_inner_radius()

```
double Configuration::get_input_inner_radius ( ) const  [inline]
```

Returns the inner radius of an input, the radius where the DomainAxis start.

Returns

the input inner radius

5.7.2.9 get_input_separation_angle()

```
double Configuration::get_input_separation_angle ( ) const  [inline]
```

Returns the seperation angle between inputs.

Returns

the input separation angle

5.7.2.10 get_input_thickness()

```
double Configuration::get_input_thickness ( ) const  [inline]
```

Returns the thickness of the colored ring of the DomainAxis.

Returns

the input thickness

5.7.2.11 get_instance()

```
static Configuration& Configuration::get_instance ( )  [inline], [static]
```

Returns a reference to the singleton instance of Configuration. Configuration::initialize needs to be called at least once before.

Returns

the reference to the Configuration instance

Exceptions

bad_function_call	if instance was not initialized

5.7.2.12 get_num_histogram_classes()

```
int Configuration::get_num_histogram_classes ( ) const  [inline]
```

Returns the number of classes that each histogram consists of.

Returns

the number of histogram classes

5.7.2.13 get_num_major_sections_axis()

```
int Configuration::get_num_major_sections_axis ( ) const  [inline]
```

Returns the number of bold sections of the scale of the DomainAxis.

Returns

the number of major sections

5.7.2.14 get_num_major_sections_grid()

```
int Configuration::get_num_major_sections_grid ( ) const  [inline]
```

Returns the number of bold sections of the scale of the CodomainGrid.

Returns

the number of major sections

5.7.2.15 get_num_minor_sections_axis()

```
int Configuration::get_num_minor_sections_axis ( ) const  [inline]
```

Returns the number of narrow sections of the scale of the DomainAxis.

Returns

the number of minor sections

5.7.2.16 get_num_minor_sections_grid()

```
int Configuration::get_num_minor_sections_grid ( ) const  [inline]
```

Returns the number of narrow sections of the scale of the CodomainGrid.

Returns

the number of minor sections

5.7.2.17 get_output_angle_span()

```
double Configuration::get_output_angle_span ( ) const  [inline]
```

Returns the output angle span, the angle span for the CodomainGrid.

Returns

the output angle span

5.7.2.18 get_output_file()

```
const std::string& Configuration::get_output_file ( ) const  [inline]
```

Returns the path to the output file.

Returns

the output file path

5.7.2.19 get_output_inner_radius()

```
double Configuration::get_output_inner_radius ( ) const  [inline]
```

Returns the inner radius of the output, the radius at which the CodomainGrid starts.

Returns

the output inner radius

5.7.2.20 get_output_thickness()

```
double Configuration::get_output_thickness ( ) const  [inline]
```

Returns the thickness of the outputs colored segmented ring.

Returns

the output thickness

5.7.2.21 get_prop_axis_label()

```
const TextProperties& Configuration::get_prop_axis_label ( ) const  [inline]
```

Returns MooViEs TextProperties for DomainAxis labels.

Returns

the TextProperties for DomainAxis labels

5.7.2.22 get_prop_scale_label()

```
const TextProperties& Configuration::get_prop_scale_label ( ) const  [inline]
```

Returns MooViEs TextProperties for Scale labels.

Returns

the TextProperties for Scale labels

5.7.2.23 get_prop_thick()

```
const DrawerProperties& Configuration::get_prop_thick ( ) const  [inline]
```

Returns MooViEs DrawerProperties for thick black lines.

Returns

the DrawerProperties for thick lines

5.7.2.24 get_prop_thin()

```
const DrawerProperties& Configuration::get_prop_thin ( ) const  [inline]
```

Returns MooViEs DrawerProperties for thin black lines.

Returns

the DrawerProperties for thin lines

5.7.2.25 get_width()

```
int Configuration::get_width ( ) const  [inline]
```

Returns the width of the MooViE scene

Returns

the width

5.7.2.26 initialize() [1/2]

```
static void Configuration::initialize (
             const std::string & fname,
             const std::string & cpath )  [static]
```

Initializes the singleton instance with the given input file path and the information given by the configuration file located under the given configuration file path.

Parameters

fname	the path to the input file
cpath	the path to the configuration file

5.7.2.27 initialize() [2/2]

```
static void Configuration::initialize (
             const std::string & fname )  [static]
```

Initializes the singleton instance with the given input file path and the standard configuration information.

Parameters

fname	the path to the input file

5.7.2.28 is_histograms_enabled()

```
bool Configuration::is_histograms_enabled ( ) const  [inline]
```

Returns whether or not histograms should be drawn.

Returns

histograms enabled or not

5.7.2.29 set_connector_arc_ratio()

```
void Configuration::set_connector_arc_ratio (
             double _connector_arc_ratio )  [inline]
```

Sets the ratio of the radial distance between two data points that will be drawn as connector.

Parameters

_ratio_connector_arc	the connector arc ratio to set

5.7.2.30 set_grid_size()

```
void Configuration::set_grid_size (
             double _grid_size )  [inline]
```

Sets the size of actual grid that is a part of the CodomainGrid.

Parameters

grid_size	the grid_size to set

5.7.2.31 set_height()

```
void Configuration::set_height (
             int _height )  [inline]
```

Sets the height of a MooViE scene.

Parameters

height	the height to set

5.7.2.32 set_histogram_background()

```
void Configuration::set_histogram_background (
             const Color & _histogram_background )  [inline]
```

Sets the background color that each histogram has.

Parameters

_histogram_background	the histogram background color to set

5.7.2.33 set_histogram_fill()

```
void Configuration::set_histogram_fill (
             const Color & _histogram_fill )  [inline]
```

Sets the fill color of each histogram's bars.

Parameters

_histogram↩ _fill	the histogram fill color to set

5.7.2.34 set_histogram_height()

```
void Configuration::set_histogram_height (
            double _histogram_height )  [inline]
```

Sets the height that each histogram has.

Parameters

_histogram_height	the histogram height to set

5.7.2.35 set_histograms_enabled()

```
void Configuration::set_histograms_enabled (
            bool _histograms_enabled )  [inline]
```

Sets whether or not histograms should be drawn.

Parameters

_histograms_enabled	histograms enabled or not

5.7.2.36 set_input_inner_radius()

```
void Configuration::set_input_inner_radius (
            double _input_inner_radius )  [inline]
```

Sets the inner radius of an input, the radius where the DomainAxis start.

Parameters

input_inner_radius	the input inner radius to set

5.7.2.37 set_input_separation_angle()

```
void Configuration::set_input_separation_angle (
             double _input_separation_angle )  [inline]
```

Sets the seperation angle between inputs.

Parameters

input_separation_angle	the input separation angle to set

5.7.2.38 set_input_thickness()

```
void Configuration::set_input_thickness (
             double _input_thickness )  [inline]
```

Sets the thickness of the colored ring of the DomainAxis.

Parameters

_input_thickness	the input thickness to set

5.7.2.39 set_num_histogram_classes()

```
void Configuration::set_num_histogram_classes (
             int _num_histogram_classes )  [inline]
```

Sets the number of classes that each histogram consists of.

Parameters

_num_histogram_classes	the number of histogram classes to set

5.7.2.40 set_num_major_sections_axis()

```
void Configuration::set_num_major_sections_axis (
             int major_sections )  [inline]
```

Sets the number of bold sections of the scale of the DomainAxis.

Parameters

major_sections	the number of major sections to set

5.7.2.41 set_num_major_sections_grid()

```
void Configuration::set_num_major_sections_grid (
             int major_sections )  [inline]
```

Sets the number of bold sections of the scale of the CodomainGrid.

Parameters

major_sections	the number of major sections to set

5.7.2.42 set_num_minor_sections_axis()

```
void Configuration::set_num_minor_sections_axis (
             int minor_sections )  [inline]
```

Sets the number of narrow sections of the scale of the DomainAxis.

Parameters

minor_sections	the number minor sections to set

5.7.2.43 set_num_minor_sections_grid()

```
void Configuration::set_num_minor_sections_grid (
             int minor_sections )  [inline]
```

Sets the number of narrow sections of the scale of the CodomainGrid.

Parameters

minor_sections	the number of minor sections to set

5.7.2.44 set_output_angle_span()

```
void Configuration::set_output_angle_span (
            double _output_angle_span )  [inline]
```

Sets the output angle span, the angle span for the CodomainGrid.

Parameters

output_angle_span	the output angle span to set

5.7.2.45 set_output_file()

```
void Configuration::set_output_file (
            const std::string & _output_file )  [inline]
```

Sets the path to the output file.

Parameters

output_file	the output file path to set

5.7.2.46 set_output_inner_radius()

```
void Configuration::set_output_inner_radius (
            double _output_inner_radius )  [inline]
```

Sets the inner radius of the output, the radius at which the CodomainGrid starts.

Parameters

output_inner_radius	the output inner radius to set

5.7.2.47 set_output_thickness()

```
void Configuration::set_output_thickness (
            double _output_thickness )  [inline]
```

Sets the thickness of the outputs colored segmented ring.

Parameters

output_thickness	the output_thickness to set

5.7.2.48 set_prop_axis_label()

```
void Configuration::set_prop_axis_label (
             const TextProperties & _prop_axis_label )  [inline]
```

Sets MooViEs TextProperties for DomainAxis labels.

Parameters

_prop_axis_label	the TextProperties to set

5.7.2.49 set_prop_scale_label()

```
void Configuration::set_prop_scale_label (
             const TextProperties & _prop_scale_label )  [inline]
```

Sets MooViEs TextProperties for Scale labels.

Parameters

_prop_scale_label	the TextProperties to set

5.7.2.50 set_prop_thick()

```
void Configuration::set_prop_thick (
             const DrawerProperties<> & _prop_thick )  [inline]
```

Sets MooViEs DrawerProperties for thick black lines.

Parameters

_prop_thick	the DrawerProperties to set

5.7.2.51 set_prop_thin()

```
void Configuration::set_prop_thin (
             const DrawerProperties<> & _prop_thin )  [inline]
```

Sets MooViEs DrawerProperties for thin black lines.

Parameters

_prop_thin	the DrawerProperties to set

5.7.2.52 set_width()

```
void Configuration::set_width (
             int _width )  [inline]
```

Sets the width of a MooViE scene.

Parameters

width	the width to set

5.7.3 Member Data Documentation

5.7.3.1 GLOW_10

```
const std::array<Color, 10> Configuration::GLOW_10  [static]
```

An array of Colors

5.7.3.2 SET2_3_1

```
const Color Configuration::SET2_3_1  [static]
```

Further color constants

5.7.3.3 SET3

```
const Triangle<Color, 12> Configuration::SET3  [static]
```

A Triangular storage which contains i+1 matching colors at the i-th index.

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

- include/Configuration.h

5.8 CoordinateConverter Class Reference

The PolarCartesian class.

```
#include <Coordinates.h>
```

Public Member Functions

- CoordinateConverter (size_t width, size_t height)

  a converter for coordinates
- void convert (const Cartesian &from, Polar &to) const

  convert Cartesian to Polar
- void convert (const Polar &from, Cartesian &to) const

  convert Polar to Cartesian
- double get_center_x () const

  center x value
- double get_center_y () const

  center y value

5.8.1 Detailed Description

The PolarCartesian class.

CoordinateConverter simulates a fixed width/height coordinate system. It can convert polar and cartesian coordinates.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 CoordinateConverter()

```
CoordinateConverter::CoordinateConverter (
              size_t width,
              size_t height )  [inline]
```

a converter for coordinates

Creates a new coordinate system with given width and height. The center coordinate is at (width / 2, height / 2).

Parameters

width	the coordinate system width
height	the coordinate system system

5.8.3 Member Function Documentation

5.8.3.1 convert() [1/2]

```
void CoordinateConverter::convert (
             const Cartesian & from,
             Polar & to ) const  [inline]
```

convert Cartesian to Polar

Converts a Cartesian coordinate to a Polar coordinate.

Parameters

from	the Cartesian to convert
to	the Polar to store

5.8.3.2 convert() [2/2]

```
void CoordinateConverter::convert (
             const Polar & from,
             Cartesian & to ) const  [inline]
```

convert Polar to Cartesian

Converts a Polar coordinate to a Cartesian coordinate.

Parameters

from	the Polar to convert
to	the Polar to store

5.8.3.3 get_center_x()

```
double CoordinateConverter::get_center_x ( ) const  [inline]
```

center x value

Returns the x value of the center coordinate.

Returns

the center's x value

5.8.3.4 get_center_y()

```
double CoordinateConverter::get_center_y ( ) const  [inline]
```

center y value

Returns the y value of the center coordinate.

Returns

the center's y value

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

- include/Coordinates.h

5.9 DataSet< T > Class Template Reference

the DataSet class

```
#include <DataSet.h>
```

Classes

- struct Cell

    the Cell struct
- class iterator
- struct Variable

    The Var struct.

Public Types

- typedef std::vector< Cell > DataRow
- typedef const iterator const_iterator

Public Member Functions

- std::size_t cols () const
- std::size_t rows () const
- const DataRow & operator[ ] (std::size_t i) const
- const std::vector< Variable > & input_variables (void) const
- const std::vector< Variable > & output_variables (void) const
- const_iterator begin () const
- const_iterator end () const

Static Public Member Functions

- static DataSet ∗ parse_from_csv (const std::string &cont, std::string separator=",", std::string comment="#", std::string newline="\)

5.9.1 Detailed Description

template<typename T>
class DataSet< T >

the DataSet class

DataSet stores data. It is accessible via iterator, but cannot be modified.

Author

stratmann

Date

28.11.2017

5.9.2 Member Typedef Documentation

5.9.2.1 const_iterator

```
template<typename T >
typedef const iterator DataSet< T >::const_iterator
```

Renaming to simplify the use of iterators

5.9.2.2 DataRow

```
template<typename T >
typedef std::vector<Cell> DataSet< T >::DataRow
```

Renaming to ease the handling of rows

5.9.3 Member Function Documentation

5.9.3.1 begin()

```
template<typename T >
const_iterator DataSet< T >::begin ( ) const  [inline]
```

Returns a constant iterator pointing to the first DataRow.

Returns

a const_iterator

5.9.3.2 cols()

```
template<typename T >
std::size_t DataSet< T >::cols ( ) const  [inline]
```

Returns the number of columns in this table.

Returns

the number of columns

5.9.3.3 end()

```
template<typename T >
const_iterator DataSet< T >::end ( ) const  [inline]
```

Returns a constant iterator pointing to the end element of the DataRow storage.

Returns

a const_iterator

5.9.3.4 input_variables()

```
template<typename T >
const std::vector<Variable>& DataSet< T >::input_variables (
             void  ) const  [inline]
```

Returns a constant vector containing row (referred to as variables) information like the name and min/max values of the selected row.

Returns

the input variables

5.9.3.5 operator[]()

```
template<typename T >
const DataRow& DataSet< T >::operator[] (
             std::size_t i ) const  [inline]
```

Returns the row at position i in the table (starting at 0). DataRow can be used like a vector from the given type.

Returns

the DataRow object

5.9.3.6 output_variables()

```
template<typename T >
const std::vector<Variable>& DataSet< T >::output_variables (
            void  ) const  [inline]
```

Returns a constant vector containing column (referred to as variables) information like the name and min/max values of the selected row.

Returns

the output variables

5.9.3.7 parse_from_csv()

```
template<typename T >
DataSet< T > * DataSet< T >::parse_from_csv (
            const std::string & cont,
            std::string separator = ",",
            std::string comment = "#",
            std::string newline = "\n" )  [static]
```

Returns a data table parsed from a csv encoded string and encapsulated in a DataSet object. The table must have the form: input1 ... inputN output1 ... outputM ... ...

Parameters

cont	the csv encoded string
num_ins	the number of input variables
separator	the column seperator used in this csv string
comment	the comment indicator used in this csv string
newline	the newline indicator used in this csv string

Returns

the DataSet object

5.9.3.8 rows()

```
template<typename T >
std::size_t DataSet< T >::rows ( ) const  [inline]
```

Returns the number of rows in this table.

Returns

the number of rows

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

- include/DataSet.h

5.10 DomainAxis Class Reference

```
#include <DomainAxis.h>
```

Classes

- class Histogram

Public Member Functions

- DomainAxis (DefVariable _var, const Angle &_start, const Angle &_end, double _radius, double _height, const DrawerProperties<> &_prop)

  constructor
- const DefVariable & get_var () const

  gets the Var
- const Histogram & get_histogram () const

  gets the Histogram
- const Angle & get_start () const

  gets the start Angle
- void set_start (const Angle &_start)

  sets the start Angle
- const Angle & get_end () const

  gets the end Angle
- void set_end (const Angle &_end)

  gets the end Angle
- double get_radius () const

  gets the radius
- void set_radius (double _radius)

  sets the radius
- double get_height () const

  gets the height
- void set_height (double _height)

  sets the height
- const DrawerProperties & get_prop () const

  gets the DrawerProperties
- void set_prop (const DrawerProperties<> &_prop)

  sets the DrawerProperties
- const SimpleScale & get_scale () const

  gets the SimpleScale
- Label make_label (const TextProperties &_prop) const

  makes a label for this DomainAxis
- void calculate_histogram (const std::vector< double > &data)

  calculates Histogram frequencies

5.10.1 Detailed Description

A DomainAxis is an axis which displays the possible values of a input variable. It is visualized as a ring segment with a distinct color and has ticks for better readability.

Author

stratmann

Date

12.12.2017

5.10.2 Constructor & Destructor Documentation

5.10.2.1 DomainAxis()

```
DomainAxis::DomainAxis (
            DefVariable _var,
            const Angle & _start,
            const Angle & _end,
            double _radius,
            double _height,
            const DrawerProperties<> & _prop )
```

constructor

Creates a DomainAxis presenting a given variable and is drawn between given angles with given radius, height and properties.

Parameters

_var	the variable to present
_start	the start angle
_end	the end angle
_radius	the radius from the center
_height	the height beginning at the radius
_prop	the DrawerProperties

5.10.3 Member Function Documentation

5.10.3.1 calculate_histogram()

```
void DomainAxis::calculate_histogram (
            const std::vector< double > & data )
```

calculates Histogram frequencies

Calculates the frequencies of the Histogram.

Parameters

data	the data used

5.10.3.2 get_end()

```
const Angle& DomainAxis::get_end ( ) const  [inline]
```

gets the end Angle

Returns the end Angle of this DomainAxis' drawing span.

Returns

the end Angle

5.10.3.3 get_height()

```
double DomainAxis::get_height ( ) const  [inline]
```

gets the height

Returns the height measured from the radius.

Returns

the height

5.10.3.4 get_histogram()

```
const Histogram& DomainAxis::get_histogram ( ) const  [inline]
```

gets the Histogram

Returns a reference to its histogram. The DomainAxis::calculate_histogram function has to called before drawing the histogram because it is empty by default.

Returns

the Histogram

5.10.3.5 get_prop()

```
const DrawerProperties& DomainAxis::get_prop ( ) const  [inline]
```

gets the DrawerProperties

Returns the DrawerProperties that will be used to draw this DomainAxis.

Returns

the DrawerProperties

5.10.3.6 get_radius()

```
double DomainAxis::get_radius ( ) const  [inline]
```

gets the radius

Returns the radius measured from the center of the coordinate system.

Returns

the radius

5.10.3.7 get_scale()

```
const SimpleScale& DomainAxis::get_scale ( ) const  [inline]
```

gets the SimpleScale

Returns the SimpleScale of this DomainAxis. This scale instance defines how the graphical scale will be drawn.

Returns

the SimpleScale

5.10.3.8 get_start()

```
const Angle& DomainAxis::get_start ( ) const  [inline]
```

gets the start Angle

Returns the start Angle of this DomainAxis' drawing span.

Returns

the start Angle

5.10.3.9 get_var()

```
const DefVariable& DomainAxis::get_var ( ) const  [inline]
```

gets the Var

Returns a const reference to the variable this DomainAxis presents.

Returns

the Var

5.10.3.10 make_label()

```
Label DomainAxis::make_label (
              const TextProperties & _prop ) const  [inline]
```

makes a label for this DomainAxis

Constructs a label using the given TextProperties' style and this DomainAxis' variable name.

Parameters

_prop	

5.10.3.11 set_end()

```
void DomainAxis::set_end (
              const Angle & _end )  [inline]
```

gets the end Angle

Sets the end Angle of this DomainAxis' drawing span.

Parameters

_end	the end Angle to set

5.10.3.12 set_height()

```
void DomainAxis::set_height (
              double _height )  [inline]
```

sets the height

Sets the height measured from the radius.

Parameters

_height	the height to set

5.10.3.13 set_prop()

```
void DomainAxis::set_prop (
             const DrawerProperties<> & _prop )  [inline]
```

sets the DrawerProperties

Sets the DrawerProperties that will be used to draw this DomainAxis.

Parameters

_prop	the DrawerProperties to set

5.10.3.14 set_radius()

```
void DomainAxis::set_radius (
             double _radius )  [inline]
```

sets the radius

Sets the radius measured from the center of the coordinate system.

Parameters

_radius	the radius to set

5.10.3.15 set_start()

```
void DomainAxis::set_start (
             const Angle & _start )  [inline]
```

sets the start Angle

Starts the start Angle of this DomainAxis' drawing span.

Parameters

_start	the start Angle to set

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

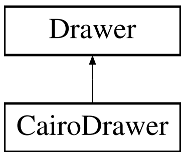
- include/DomainAxis.h

5.11 Drawer Class Reference

an abstract MooViE Drawer

```
#include <Drawer.h>
```

Inheritance diagram for Drawer:



Classes

- struct TextAlignment

  an text alignment representation

Public Member Functions

- Drawer (int width, int height, std::size_t _num_inputs)

  Drawer constructor.
- virtual void change_surface (const std::string &fpath, int width, int height)=0

  changes the underlying surface by the given parameters
- virtual void draw_codomain_grid (const CodomainGrid &grid)=0

  draws a CodomainGrid
- virtual void draw_domain_axis (const DomainAxis &axis)=0

  draws a DomainAxis
- virtual void draw_relation_element (const RelationElement &elem)=0

  draws a RelationElement
- virtual void finish ()=0

  save results

Static Public Attributes

- static constexpr double **LINK_CONTROL_STRENGTH** = 100

Protected Member Functions

- virtual void set_surface (const std::string &fpath, int width, int height)=0

    hard-sets the underlying surface by the given parameters
- virtual void draw_histogram (const DomainAxis::Histogram &histogram, double radius, const Angle &start, const Angle &end)=0

    draws a Histogram
- virtual void draw_link (const Polar &origin1, const Polar &origin2, const Polar &target1, const Polar &target2, const DrawerProperties<> &prop)=0

    draws a link
- virtual void draw_connector (const Polar &from, const Polar &to, const DrawerProperties<> &prop)=0

    draws a connector
- virtual void draw_segment_axis (double inner_radius, double thickness, const Angle &begin, const Angle &end, const DrawerProperties< std::array< Color, 10 >> &prop, Direction dir)=0

    draws a split axis
- virtual void draw_output_label (const Label &output_label, double radius_label, double radius_output, const Angle &begin, const Angle &end)=0

    draws an output label
- virtual void draw_arrow (const Polar &start, const DrawerProperties<> &prop)=0

    draws arrow
- virtual void draw_ring_segment (double radius, double thickness, const Angle &start, const Angle &end, const DrawerProperties<> &prop, Direction dir)=0

    draws a ring segment
- virtual void draw_connector_segment (double start_radius, double start_angle, double end_radius, double end_angle, const DrawerProperties<> &prop)=0

    draws a connector Bezier curve
- virtual void draw_line (const Polar &from, const Polar &to, const DrawerProperties<> &prop)=0

    draws a simple line
- virtual void draw_arc (double inner_radius, const Angle &start, const Angle &end, Direction dir)=0

    draws an arc
- virtual void draw_coord_point (const Polar &coord, const Angle &width, double height, const Drawer↩Properties<> &prop)=0

    draws an error box
- virtual void draw_text_parallel (const Label &label, const Polar &start, const TextAlignment &alignment=Text↩Alignment::CENTERED)=0

    draws a Label on a line to the middle
- virtual void draw_text_orthogonal (const Label &label, const Polar &start, const TextAlignment &alignment=TextAlignment::CENTERED)=0

    draws a Label orthogonal to a line to the middle
- Polar get_connector_start (const Polar &from, const Polar &to)
- Polar get_connector_end (const Polar &from, const Polar &to)
- Cartesian create_link_control_point (const Polar &point) const

    creates link control point

Protected Attributes

- const CoordinateConverter coord_converter
- std::size_t num_inputs

5.11.1 Detailed Description

an abstract MooViE Drawer

An abstract Drawer class that can be used to draw MooViE elements. Drawer is supposed to cover the strategy that is used to actually draw an image with a MooViE scene. It provides the implementation with a CoordinateConverter, TextAlignment wrapper and basic calculation functions for points.

Author

stratmann

Date

27.04.2018

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Drawer()

```
Drawer::Drawer (
             int width,
             int height,
             std::size_t _num_inputs )  [inline]
```

Drawer constructor.

Creates a Drawer which draws on a surface with the given width and height.

Parameters

width	the surface width
height	the surface height

5.11.3 Member Function Documentation

5.11.3.1 change_surface()

```
virtual void Drawer::change_surface (
             const std::string & fpath,
             int width,
             int height )  [pure virtual]
```

changes the underlying surface by the given parameters

Alters the surface of this Drawer in with, height and storage path. All unsafed changes will be stored and all kept resources freed correctly.

Parameters

fpath	a string containing an valid existing or accessible not existing path
width	an integer between 0 and MAX_INT
height	an integer between 0 and MAX_INT

Implemented in CairoDrawer.

5.11.3.2 create_link_control_point()

```
Cartesian Drawer::create_link_control_point (
             const Polar & point ) const  [inline], [protected]
```

creates link control point

Creates a control point for a Bezier curve approximating a link.

Parameters

point	coordinate to which the control point will be created

Returns

the control point

5.11.3.3 draw_arc()

```
virtual void Drawer::draw_arc (
             double inner_radius,
             const Angle & start,
             const Angle & end,
             Direction dir )  [protected], [pure virtual]
```

draws an arc

Draws a simple edge segment around the center of its coordinate system between the two given Angles and with the given radius.

Parameters

inner_radius	the inner radius
start	the start Angle
end	the end Angle
dir	the direction

Implemented in CairoDrawer.

5.11.3.4 draw_arrow()

```
virtual void Drawer::draw_arrow (
            const Polar & start,
            const DrawerProperties<> & prop )  [protected], [pure virtual]
```

draws arrow

Draws a arrow head from a given start pointing.

Parameters

start	the start of the arrow head
prop	DrawerProperties for the arrow head

Implemented in CairoDrawer.

5.11.3.5 draw_codomain_grid()

```
virtual void Drawer::draw_codomain_grid (
            const CodomainGrid & grid )  [pure virtual]
```

draws a CodomainGrid

Draws a CodomainGrid using its radius and angles. For thin or thick lines the properties given by the Configuration instance are used. On

Parameters

grid	the CodomainGrid to draw

Implemented in CairoDrawer.

5.11.3.6 draw_connector()

```
virtual void Drawer::draw_connector (
            const Polar & from,
            const Polar & to,
            const DrawerProperties<> & prop )  [protected], [pure virtual]
```

draws a connector

Draws a connection between to given polar coordinates. The connection is a bezier curve which is controlled by automatically generated control points.

Parameters

from	the start Polar
to	the end Polar
prop	the DrawerProperties

Implemented in CairoDrawer.

5.11.3.7 draw_connector_segment()

```
virtual void Drawer::draw_connector_segment (
             double start_radius,
             double start_angle,
             double end_radius,
             double end_angle,
             const DrawerProperties<> & prop )  [protected], [pure virtual]
```

draws a connector Bezier curve

Draws a Bezier curve from Polar(start_radius, start_angle) to Polar(end_radius, end_angle) which approximately behaves like Archimedean spiral. If the smaller difference angle between start_angle and end_angle is bigger than PI, the spiral will be approximated by two Bezier curves.

Parameters

start_radius	the radius of the starting point
start_angle	the angle of the starting point
end_radius	the radius of the end point
end_angle	the angle of the end point
prop	the DrawerProperties for the segment

Implemented in CairoDrawer.

5.11.3.8 draw_coord_point()

```
virtual void Drawer::draw_coord_point (
             const Polar & coord,
             const Angle & width,
             double height,
             const DrawerProperties<> & prop )  [protected], [pure virtual]
```

draws an error box

Draws a coordinate point with given height and with.

Parameters

coord	the polar coordinate to draw
width	the width
height	the height
prop	the DrawerProperties

Implemented in CairoDrawer.

5.11.3.9 draw_domain_axis()

```
virtual void Drawer::draw_domain_axis (
            const DomainAxis & axis )  [pure virtual]
```

draws a DomainAxis

Draws a DomainAxis using its radius and angles. For thin or thick lines the properties given by the Configuration instance are used.

Parameters

axis	the DomainAxis to draw

Implemented in CairoDrawer.

5.11.3.10 draw_histogram()

```
virtual void Drawer::draw_histogram (
            const DomainAxis::Histogram & histogram,
            double radius,
            const Angle & start,
            const Angle & end )  [protected], [pure virtual]
```

draws a Histogram

Draws a Histogram from the given radius, between begin and end Angle. For the histogram height, thin or thick lines the properties given by the Configuration instance are used.

Parameters

histogram	the Histogram to draw
radius	the start radius of the Histogram
start	the starting angle of the Histogram
end	the end angle of the Histogram

Implemented in CairoDrawer.

5.11.3.11 draw_line()

```
virtual void Drawer::draw_line (
             const Polar & from,
             const Polar & to,
             const DrawerProperties<> & prop )  [protected], [pure virtual]
```

draws a simple line

Draws a line from a given starting vertice to a given end vertice.

Parameters

from	the starting coordinates
to	the end coordinates
prop	the DrawerProperties to use

Implemented in CairoDrawer.

5.11.3.12 draw_link()

```
virtual void Drawer::draw_link (
             const Polar & origin1,
             const Polar & origin2,
             const Polar & target1,
             const Polar & target2,
             const DrawerProperties<> & prop )  [protected], [pure virtual]
```

draws a link

Draws a bold line between the lines origin1-origin2 and target1-target2. This is realized by drawing Bezier curves from origin1 to target1 and from origin2 to target2 and filling the so created surface.

Parameters

origin1	first origin coordinate
origin2	second origin coordinate
target1	first target coordinate
target2	second target coordinate
prop	DrawerProperties for the link

Implemented in CairoDrawer.

5.11.3.13 draw_output_label()

```
virtual void Drawer::draw_output_label (
             const Label & output_label,
             double radius_label,
             double radius_output,
             const Angle & begin,
             const Angle & end )  [protected], [pure virtual]
```

draws an output label

Draws the given Label output_label with the radius radius_label and a descriptive path that connects the output label with the associated output. The path consists of an arc segment and a line.

Parameters

output_label	the output label to draw
radius_label	the radius of the output label
radius_output	the radius of the associated output
begin	the angle at which the output ends
end	the angle at which the arc ends

Implemented in CairoDrawer.

5.11.3.14 draw_relation_element()

```
virtual void Drawer::draw_relation_element (
             const RelationElement & elem )  [pure virtual]
```

draws a RelationElement

Draws a RelationElement using its coordinates.

Parameters

elem	the RelationElement to draw

Implemented in CairoDrawer.

5.11.3.15 draw_ring_segment()

```
virtual void Drawer::draw_ring_segment (
             double radius,
             double thickness,
             const Angle & start,
             const Angle & end,
             const DrawerProperties<> & prop,
             Direction dir )  [protected], [pure virtual]
```

draws a ring segment

Draws a filled ring segment around the center of its coordinate system between the two given Angles and with the given radius.

Parameters

radius	the radius
thickness	the thinkness of the edge segment
begin	the begin Angle
end	the end Angle
prop	the CairoDrawer properties
dir	the direction

Implemented in CairoDrawer.

5.11.3.16 draw_segment_axis()

```
virtual void Drawer::draw_segment_axis (
             double inner_radius,
             double thickness,
             const Angle & begin,
             const Angle & end,
             const DrawerProperties< std::array< Color, 10 >> & prop,
             Direction dir )  [protected], [pure virtual]
```

draws a split axis

Draws a circle segment which is itself divided in colored segments.

Parameters

inner_radius	inner radius of the split axis
thickness	width of the split axis
begin	angle of the segments begin
end	angle of the segments end
prop	color
dir	direction of the split axis' colors

Implemented in CairoDrawer.

5.11.3.17 draw_text_orthogonal()

```
virtual void Drawer::draw_text_orthogonal (
             const Label & label,
             const Polar & start,
             const TextAlignment & alignment = TextAlignment::CENTERED )  [protected], [pure
virtual]
```

draws a Label orthogonal to a line to the middle

Draws the given label orthogonal to the angle of the given coordinate's angle.

Parameters

label	the label to draw
start	the coordinate to adjust to

Implemented in CairoDrawer.

5.11.3.18 draw_text_parallel()

```
virtual void Drawer::draw_text_parallel (
                const Label & label,
                const Polar & start,
                const TextAlignment & alignment = TextAlignment::CENTERED )  [protected], [pure
virtual]
```

draws a Label on a line to the middle

Draws the given label with the same angle like the given coordinate.

Parameters

label	the label to draw
start	the coordinate to adjust to

Implemented in CairoDrawer.

5.11.3.19 finish()

```
virtual void Drawer::finish ( )  [pure virtual]
```

save results

Save the Drawer's result to the given file.

Implemented in CairoDrawer.

5.11.3.20 get_connector_end()

```
Polar Drawer::get_connector_end (
            const Polar & from,
            const Polar & to )  [inline], [protected]
```

Calculates a Polar coordinate for the end of a connector between 'from' and 'to'. If the resulting coordinate is passed to a connector drawing function, the connector does not immediately end at to.

Parameters

from	the Polar coordinate to start the connector from
from	the Polar coordinate to draw the connector to

Returns

the modified connector end coordinate

5.11.3.21 get_connector_start()

```
Polar Drawer::get_connector_start (
            const Polar & from,
            const Polar & to )  [inline], [protected]
```

Calculates a Polar coordinate for the beginning of a connector between 'from' and 'to'. If the resulting coordinate is passed to a connector drawing function, the connector does not immediately start at from.

Parameters

from	the Polar coordinate to start the connector from
from	the Polar coordinate to draw the connector to

Returns

the modified connector start coordinate

5.11.3.22 set_surface()

```
virtual void Drawer::set_surface (
            const std::string & fpath,
            int width,
            int height )  [protected], [pure virtual]
```

hard-sets the underlying surface by the given parameters

Alters the surface of this Drawer in with, height and storage path.

Parameters

fpath	a string containing an valid or accessible path
width	an integer between 0 and MAX_INT
height	an integer between 0 and MAX_INT

Implemented in CairoDrawer.

5.11.4 Member Data Documentation

5.11.4.1 coord_converter

```
const CoordinateConverter Drawer::coord_converter  [protected]
```

Polar-Cartesian converting

5.11.4.2 num_inputs

```
std::size_t Drawer::num_inputs  [protected]
```

Number of input variables of the multi-objective data to draw

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

- include/Drawer.h

5.12 DrawerProperties< FillT > Struct Template Reference

The DrawerProperties class.

```
#include <DrawerProperties.h>
```

Public Member Functions

- DrawerProperties (double _line_width, const Color &_line_color, const FillT &_fill_color)
  DrawerProperties.

Public Attributes

- double line_width
- Color line_color
- FillT fill_color

5.12.1 Detailed Description

template<typename FillT = Color>
struct DrawerProperties< FillT >

The DrawerProperties class.

DrawerProperties can be used to control the line thinkness, stroke and fill color of a Drawer.

Author

beyss

Date

05.07.2017

5.12.2 Constructor & Destructor Documentation

5.12.2.1 DrawerProperties()

```
template<typename FillT = Color>
DrawerProperties< FillT >::DrawerProperties (
             double _line_width,
             const Color & _line_color,
             const FillT & _fill_color )  [inline]
```

DrawerProperties.

Creates a DrawerProperties instance storing the given line thinkness, stroke and fill color of a Drawer.

Parameters

_line_width	the line width
_line_color	the line color
_fill_color	the fill color

5.12.3 Member Data Documentation

5.12.3.1 fill_color

```
template<typename FillT = Color>
FillT DrawerProperties< FillT >::fill_color
```

Fill color(s)

5.12.3.2 line_color

```
template<typename FillT = Color>
Color DrawerProperties< FillT >::line_color
```

Line color

5.12.3.3 line_width

```
template<typename FillT = Color>
double DrawerProperties< FillT >::line_width
```

The line width

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

- include/DrawerProperties.h

5.13 DomainAxis::Histogram Class Reference

Public Member Functions

- Histogram (DefVariable _var)

  constructor
- void calculate (const std::vector< double > &data)
- double get_section_frequency (std::size_t i) const

  frequency of the i-th section
- std::size_t get_num_intervals (void) const

  gets number of equidistant intervals
- void set_num_intervals (std::size_t _num_intervals)

  sets the number of equistant intervals

5.13.1 Constructor & Destructor Documentation

5.13.1.1 Histogram()

```
DomainAxis::Histogram::Histogram (
            DefVariable _var )
```

constructor

Creates an empty Histogram for this variable with the specified number of intervals.

Parameters

_var	the variable to present

5.13.2 Member Function Documentation

5.13.2.1 calculate()

```
void DomainAxis::Histogram::calculate (
            const std::vector< double > & data )
```

Calculates equidistant data sections and stores them.

Parameters

data	the input values of this variable

5.13.2.2 get_num_intervals()

```
std::size_t DomainAxis::Histogram::get_num_intervals (
            void  ) const  [inline]
```

gets number of equidistant intervals

Returns the number of equidistant intervals the domain of this Histogram's Variable is divided in.

Returns

the interval count

5.13.2.3 get_section_frequency()

```
double DomainAxis::Histogram::get_section_frequency (
            std::size_t i ) const
```

frequency of the i-th section

Returns the value of the histogram graph in this section. They are associated with the relative frequency of the equidistant intervals.

Parameters

i	index of the section

Returns

the height

5.13.2.4 set_num_intervals()

```
void DomainAxis::Histogram::set_num_intervals (
            std::size_t _num_intervals )  [inline]
```

sets the number of equistant intervals

Sets the histogram to have a given number of equidistant intervals. If values for an old number of intervals have been stored, all data from is deleted and the frequencies set to 0.

Parameters

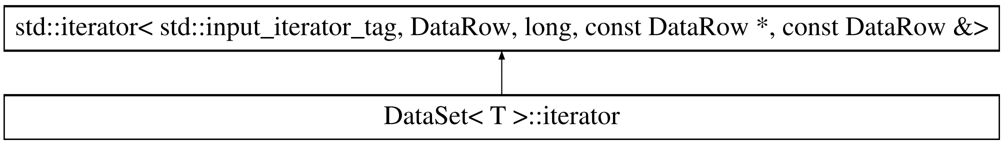
_num_interval	the new interval count

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

- include/DomainAxis.h

5.14 DataSet< T >::iterator Class Reference

Inheritance diagram for DataSet< T >::iterator:


std::iterator< std::input_iterator_tag, DataRow, long, const DataRow *, const DataRow &>
DataSet< T >::iterator

Public Member Functions

- **iterator** (const typename std::vector< DataRow >::const_iterator &it)
- iterator & **operator++** ()
- iterator **operator++** (int)
- bool **operator==** (const iterator &other) const
- bool **operator!=** (const iterator &other) const
- const DataRow & **operator*** () const

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

- include/DataSet.h

5.15 Label Class Reference

The Label class.

```
#include <Label.h>
```

Public Member Functions

- Label (const std::string &_text, const TextProperties &_prop)

  constructor
- const std::string & get_text () const

  gets text
- const TextProperties & get_properties () const

  gets TextProperties

5.15.1 Detailed Description

The Label class.

A Label is a formatted text that is stored as a text string and a TextProperties object.

Author

stratmann

Date

27.04.2018

5.15.2 Constructor & Destructor Documentation

5.15.2.1 Label()

```
Label::Label (
            const std::string & _text,
            const TextProperties & _prop )   [inline]
```

constructor

Creates a Label from given text and TextProperties.

Parameters

text	the text to be displayed
prop	the TextProperties to be used

5.15.3 Member Function Documentation

5.15.3.1 get_properties()

```
const TextProperties& Label::get_properties ( ) const  [inline]
```

gets TextProperties

Returns a const reference to this Labels TextProperties.

Returns

a reference to the TextProperties

5.15.3.2 get_text()

```
const std::string& Label::get_text ( ) const  [inline]
```

gets text

Returns a const reference to this Labels text.

Returns

a reference to the text

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

- include/Label.h

5.16 Mapper Class Reference

Mapper is a bijective function f: [a,b] -> [c,d].

```
#include <Mapper.h>
```

Public Member Functions

- Mapper (const std::pair< double, double > &_in, const std::pair< double, double > &_out)

  constructor
- double map (const double &out_val) const

  maps [a,b] -> [c,d]
- double inverse (const double &in_val) const

  maps [c,d] -> [a,b]

5.16.1 Detailed Description

Mapper is a bijective function f: [a,b] -> [c,d].

Mapper represent a mapping of from one interval to another: [a,b] -> [c,d]. It solves the linear equations

1. f(a) = r∗a + s = c
2. f(b) = r∗b + s = d for r and s so that it can determine f.

Author

beyss

Date

26.07.2017

5.16.2 Constructor & Destructor Documentation

5.16.2.1 Mapper()

```
Mapper::Mapper (
            const std::pair< double, double > & _in,
            const std::pair< double, double > & _out )  [inline]
```

constructor

Creates a Mapper from two given intervals.

Parameters

in	the first interval
out	the second interval

5.16.3 Member Function Documentation

5.16.3.1 inverse()

```
double Mapper::inverse (
            const double & in_val ) const  [inline]
```

maps [c,d] -> [a,b]

Returns the value associated to the given input using the inverse of its linear mapping function.

Parameters

in_val	the value to map

Returns

the mapped value

5.16.3.2 map()

```
double Mapper::map (
             const double & out_val ) const  [inline]
```

maps [a,b] -> [c,d]

Returns the value associated to the given input using its linear mapping function.

Parameters

out_val	the value to map

Returns

the mapped value

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

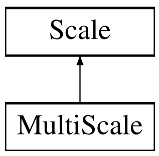
- include/Mapper.h

5.17 MultiScale Class Reference

a n-dimensional scale

```
#include <Scale.h>
```

Inheritance diagram for MultiScale:



Public Member Functions

- MultiScale (size_t ticks_major, size_t ticks_minor, const TextProperties &label_prop, const std::string &label↩ _suffix="")

  constructor
- void add_scale (const std::pair< double, double > &extremes)

  adds scale
- size_t get_scale_number (void) const

  gets the number of scales
- const std::pair< double, double > get_extremes (size_t i) const

  gets the i-th extremes
- std::vector< Label > make_labels (size_t i) const

  make description labels

Additional Inherited Members

5.17.1 Detailed Description

a n-dimensional scale

A Scale that represents a graphical axis that can display data from the R^n with two given extremes for each entry.

Author

stratmann

Date

15.05.2018

5.17.2 Constructor & Destructor Documentation

5.17.2.1 MultiScale()

```
MultiScale::MultiScale (
             size_t ticks_major,
             size_t ticks_minor,
             const TextProperties & label_prop,
             const std::string & label_suffix = "" )  [inline]
```

constructor

Creates a new MultiScale from major (big) and minor intersections, label properties, label suffix (unit) and extreme values. To use MultiScale, extreme values of each entry need to be added.

Parameters

major_intersections	number of big intersection lines
minor_intersections	number of small intersection lines
label_prop	the style of the label text
label_suffix	the unit of the presented data

5.17.3 Member Function Documentation

5.17.3.1 add_scale()

```
void MultiScale::add_scale (
             const std::pair< double, double > & extremes )  [inline]
```

adds scale

Adds extreme value of another scalable entry to this MultiScale.

Parameters

extremes	the extreme values

5.17.3.2 get_extremes()

```
const std::pair<double, double> MultiScale::get_extremes (
             size_t i ) const  [inline]
```

gets the i-th extremes

Returns the extreme values of the i-th entry.

Returns

the extremes

5.17.3.3 get_scale_number()

```
size_t MultiScale::get_scale_number (
             void  ) const  [inline]
```

gets the number of scales

Returns the number of scales of this MultiScale.

Returns

number of scales

5.17.3.4 make_labels()

```
std::vector<Label> MultiScale::make_labels (
              size_t i ) const
```

make description labels

Constructs description labels from the

Returns

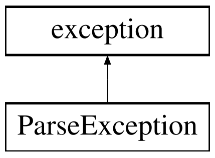
the labels

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

- include/Scale.h

5.18 ParseException Class Reference

Inheritance diagram for ParseException:



Public Member Functions

- **ParseException** (const std::string &msg)
- virtual char const ∗ **what** ()

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

- include/Utils.h

5.19 Point Struct Reference

a coordinate with drawing information

```
#include <RelationElement.h>
```

Public Member Functions

- Point (Polar &&_coord, const DrawerProperties<> &_prop)
  constructor

Public Attributes

- const Polar **coord**
- const DrawerProperties **prop**

5.19.1 Detailed Description

a coordinate with drawing information

A point in a polar coordinate system. The point has additional properties specifying how a curve starting from its coordinate should be styled.

Author

stratmann

Date

07.03.2018

5.19.2 Constructor & Destructor Documentation

5.19.2.1 Point()

```
Point::Point (
            Polar && _coord,
            const DrawerProperties<> & _prop )  [inline]
```

constructor

Creates a Point using a given Polar and DrawerProperties.

Parameters

_coord	the coordinate
_prop	the DrawerProperties

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

- include/RelationElement.h

5.20 Polar Class Reference

The Polar class.

```
#include <Coordinates.h>
```

Public Member Functions

- Polar (double radius=0, Angle angle=0)

  Polar.
- bool operator== (const Polar &rhs) const

  this == rhs
- const double & radius () const

  r
- double & radius ()

  r
- const Angle & angle () const

  phi
- Angle & angle ()

  phi

Static Public Member Functions

- static Polar interpolate (const Polar &p1, const Polar &p2, double p)

  interpolate
- static Polar center (const Polar &p1, const Polar &p2)

  center

5.20.1 Detailed Description

The Polar class.

Polar represents a tupel from the Rš in polar coordinate form.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 Polar()

```
Polar::Polar (
             double radius = 0,
             Angle angle = 0 )   [inline]
```

Polar.

Creates a Polar coordinate from a given radius and angle.

Parameters

r	the radius
phi	the angle

5.20.3 Member Function Documentation

5.20.3.1 angle() [1/2]

```
const Angle& Polar::angle ( ) const  [inline]
```

phi

Access function for this Polar's angle readonly.

Returns

a constant reference to the Angle

5.20.3.2 angle() [2/2]

```
Angle& Polar::angle ( )  [inline]
```

phi

Access function for this Polar's angle.

Returns

a reference to the Angle

5.20.3.3 center()

```
static Polar Polar::center (
             const Polar & p1,
             const Polar & p2 )  [inline], [static]
```

center

Returns a Polar centered between two given Polars.

Parameters

p1	the first Polar
p2	the second Polar

Returns

the centered Polar

5.20.3.4 interpolate()

```
static Polar Polar::interpolate (
             const Polar & p1,
             const Polar & p2,
             double p )  [inline], [static]
```

interpolate

Returns an Polar whose radius and Angle are (1-p) percent of p1's and p percent of p2's radius and Angle. To be consistent, p should be in [0,1].

Parameters

p1	the first Polar
p2	the second Polar
p	the percentage

Returns

the interpolated Polar

5.20.3.5 operator==()

```
bool Polar::operator== (
             const Polar & rhs ) const  [inline]
```

this == rhs

Equal to operator checking for equality of radius and angle.

Parameters

rhs	the other Polar

Returns

if equal or not

5.20.3.6 radius() [1/2]

```
const double& Polar::radius ( ) const  [inline]
```

r

Access function for this Polar's radius as readonly.

Returns

a constant reference to this Polar's radius

5.20.3.7 radius() [2/2]

```
double& Polar::radius ( )  [inline]
```

r

Access function for this Polar's radius.

Returns

a reference to this Polar's radius

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

- include/Coordinates.h

5.21 RelationElement Class Reference

a row of input/output data

```
#include <RelationElement.h>
```

Public Member Functions

- const Point & operator[ ] (std::size_t i) const

  access i-th point
- std::size_t size (void) const

  the number of Point
- template<typename... Arg>
  void emplace_back (Arg &&... args)

  add Point from arguments

5.21.1 Detailed Description

a row of input/output data

An element of the relation R^n x R^m or a row of data consisting of n inputs and m outputs. It can be drawn using n links and m connectors using the style specified for each Point. It is necessary to know the index i=n-1 to draw a RelationElement.

Author

stratmann

Date

07.03.2018

5.21.2 Member Function Documentation

5.21.2.1 emplace_back()

```
template<typename... Arg>
void RelationElement::emplace_back (
             Arg &&... args ) [inline]
```

add Point from arguments

Constructs and adds Point in-place using the given arguments.

Parameters

args	the arguments (Polar, DrawerProperties)

5.21.2.2 operator[]()

```
const Point& RelationElement::operator[] (
             std::size_t i ) const [inline]
```

access i-th point

Returns a const-reference to the Point of the i-th position of this RelationElement. There is no boundry check so that the result for i > RelationElement::size is undefined.

Parameters

the	index of the Point

Returns

the Point

5.21.2.3 size()

```
std::size_t RelationElement::size (
            void ) const [inline]
```

the number of Point

Returns the total number of Points n+m of this RelationElement.

Returns

the size

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

- include/RelationElement.h

5.22 RelationElementFactory Class Reference

a factory for RelationElements

`#include <RelationElement.h>`

Public Member Functions

- RelationElementFactory (std::size_t num_data_rows, const CodomainGrid &grid, const std::vector< DomainAxis > &axis)

  constructor
- RelationElement create (const DefDataRow &row) const

  creates a new RelationElement

5.22.1 Detailed Description

a factory for RelationElements

A class for constructing RelationElements. It follows the factory pattern.

Author

stratmann

Date

07.03.2018

5.22.2 Constructor & Destructor Documentation

5.22.2.1 RelationElementFactory()

```
RelationElementFactory::RelationElementFactory (
            std::size_t num_data_rows,
            const CodomainGrid & grid,
            const std::vector< DomainAxis > & axis )
```

constructor

Creates a new RelationElement factory which needs the number of rows in the data set and the CodomainGrid and the DomainAxis' with wich the RelationElement will be drawn.

Parameters

num_data_rows	the number of rows of the data set
grid	the CodomainGrid
axis	the DomainAxis'

5.22.3 Member Function Documentation

5.22.3.1 create()

```
RelationElement RelationElementFactory::create (
            const DefDataRow & row ) const
```

creates a new RelationElement

Creates a new RelationElement from a given DefDataRow with

Parameters

row	the DefDataRow

Returns

the so created RelationElement

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

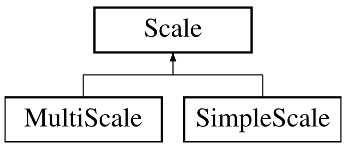
- include/RelationElement.h

5.23 Scale Class Reference

a scale

```
#include <Scale.h>
```

Inheritance diagram for Scale:



Public Member Functions

- Scale (size_t _major_intersections, size_t _minor_intersections, const TextProperties &_label_prop, const std::string &_label_suffix="")

    constructor
- size_t get_major_intersections (void) const

    number of big intersection lines
- size_t get_minor_intersections (void) const

    number of small intersection lines

Protected Attributes

- size_t **major_intersections**
- size_t **minor_intersections**
- TextProperties **label_prop**
- std::string **label_suffix**

5.23.1 Detailed Description

a scale

The Scale class represents a graphical scale of an axis by its extreme values and intersections counts.

Author

beyss

Date

22.08.2017

5.23.2 Constructor & Destructor Documentation

5.23.2.1 Scale()

```
Scale::Scale (
            size_t _major_intersections,
            size_t _minor_intersections,
            const TextProperties & _label_prop,
            const std::string & _label_suffix = "" )  [inline]
```

constructor

Creates a Scale from major (big) and minor intersections, label properties and a label suffix (unit).

Parameters

major_intersections	number of big intersection lines
minor_intersections	number of small intersection lines
label_prop	the style of the label text
label_suffix	the unit of the presented data

5.23.3 Member Function Documentation

5.23.3.1 get_major_intersections()

```
size_t Scale::get_major_intersections (
            void ) const  [inline]
```

number of big intersection lines

Returns the number of major intersection lines of this scale.

Returns

number of major intersections

5.23.3.2 get_minor_intersections()

```
size_t Scale::get_minor_intersections (
            void ) const  [inline]
```

number of small intersection lines

Returns the number of major intersection lines of this scale.

Returns

number of minor intersections

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

- include/Scale.h

5.24 Scene Class Reference

The Scene class.

```
#include <Scene.h>
```

Public Member Functions

- Scene ()

    Scene.
- void **update** (void)

5.24.1 Detailed Description

The Scene class.

Scene constructs a diagram that displays data vectors

Author

beyss

Date

28.08.2017

5.24.2 Constructor & Destructor Documentation

5.24.2.1 Scene()

```
Scene::Scene ( )
```

Scene.

Creates a new MooViE Scene.

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

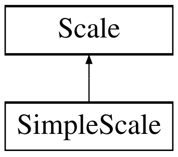
- include/Scene.h

5.25 SimpleScale Class Reference

a 1-dimensional scale

`#include <Scale.h>`

Inheritance diagram for SimpleScale:



Public Member Functions

- SimpleScale (size_t _major_intersections, size_t _minor_intersections, const std::pair< double, double > &_extremes, const TextProperties &_label_prop, const std::string &_label_suffix="")

    constructor
- const std::pair< double, double > & get_extremes () const

    extreme_vals
- std::vector< Label > make_labels (void) const

    make description labels

Additional Inherited Members

5.25.1 Detailed Description

a 1-dimensional scale

A Scale that represents a graphical axis that can display data from the real numbers with two given extremes.

Author

stratmann

Date

15.05.2018

5.25.2 Constructor & Destructor Documentation

5.25.2.1 SimpleScale()

```
SimpleScale::SimpleScale (
            size_t _major_intersections,
            size_t _minor_intersections,
            const std::pair< double, double > & _extremes,
            const TextProperties & _label_prop,
            const std::string & _label_suffix = "" )  [inline]
```

constructor

Creates a new SimpleScale from major (big) and minor intersections, label properties, label suffix (unit) and extreme values.

Parameters

major_intersections	number of big intersection lines
minor_intersections	number of small intersection lines
extremes	the extreme values of the scale
label_prop	the style of the label text
label_suffix	the unit of the presented data

5.25.3 Member Function Documentation

5.25.3.1 get_extremes()

```
const std::pair<double,double>& SimpleScale::get_extremes ( ) const  [inline]
```

extreme_vals

Access function for the Ticks extreme values.

Returns

a reference to the extreme values

5.25.3.2 make_labels()

```
std::vector<Label> SimpleScale::make_labels (
            void  ) const
```

make description labels

Constructs description labels from the

Returns

the labels

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

- include/Scale.h

5.26 Drawer::TextAlignment Struct Reference

an text alignment representation

```
#include <Drawer.h>
```

Public Member Functions

- **TextAlignment** (double ratio)

Public Attributes

- double **ratio**

Static Public Attributes

- static const TextAlignment **LEFT**
- static const TextAlignment **HALF_LEFT**
- static const TextAlignment **CENTERED**
- static const TextAlignment **HALF_RIGHT**
- static const TextAlignment **RIGHT**

5.26.1 Detailed Description

an text alignment representation

TextAlignment represents the alignment of MooViE Labels. It can be used for both horizontal and vertical alignment.

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

- include/Drawer.h

5.27 TextProperties Struct Reference

The TextProperties class.

```
#include <TextProperties.h>
```

Public Member Functions

- TextProperties (const std::string &font_name, double font_size, const Color &color=Color::BLACK, bool bold=false, bool italic=false)

    TextProperties.

Public Attributes

- std::string font_name
- double font_size
- Color color
- bool bold
- bool italic

5.27.1 Detailed Description

The TextProperties class.

TextProperties can be used to control font, size, color and style of a drawn text.

Author

beyss

Date

05.07.2017

5.27.2 Constructor & Destructor Documentation

5.27.2.1 TextProperties()

```
TextProperties::TextProperties (
            const std::string & font_name,
            double font_size,
            const Color & color = Color::BLACK,
            bool bold = false,
            bool italic = false )  [inline]
```

TextProperties.

Parameters

font_name	
font_size	
color	
bold	
italic	

5.27.3 Member Data Documentation

5.27.3.1 bold

```
bool TextProperties::bold
```

The boldness of the text

5.27.3.2 color

```
Color TextProperties::color
```

The text color

5.27.3.3 font_name

```
std::string TextProperties::font_name
```

The font name

5.27.3.4 font_size

```
double TextProperties::font_size
```

The font size

5.27.3.5 italic

```
bool TextProperties::italic
```

The skewness of the text

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

- include/TextProperties.h

5.28 Triangle< T, dim > Class Template Reference

Triangle stores matching Colors.

```
#include <Triangle.h>
```

Public Member Functions

- Triangle ()

  Triangle.
- Triangle (const std::vector< T > data)

  Triangle.
- const T & at (size_t i, size_t j) const

  at
- T & at (size_t i, size_t j)

  at

5.28.1 Detailed Description

template<typename T, size_t dim>
class Triangle< T, dim >

Triangle stores matching Colors.

Triangle stores sets who have a size equal to their their index + 1. The total storage of a Triangle instance is equal to the dim-th triangular number (starting with T_1 = 1). 0: Elem00 1: Elem10 Elem11 2: Elem20 Elem21 Elem22 ...

Author

beyss

Date

23.08.2017

5.28.2 Constructor & Destructor Documentation

5.28.2.1 Triangle() [1/2]

```
template<typename T, size_t dim>
Triangle< T, dim >::Triangle ( )  [inline]
```

Triangle.

Creates a Triangle with an empty storage.

5.28.2.2 Triangle() [2/2]

```
template<typename T, size_t dim>
Triangle< T, dim >::Triangle (
             const std::vector< T > data )  [inline]
```

Triangle.

Creates a Triangle from a given data vector whose size must be the dim-th triangular number.

Parameters

data	the data vector

5.28.3 Member Function Documentation

5.28.3.1 at() [1/2]

```
template<typename T, size_t dim>
const T& Triangle< T, dim >::at (
            size_t i,
            size_t j ) const  [inline]
```

at

Readonly access function for the j-th element of the i-th set.

Parameters

i	the "row"
j	the "column"

Returns

a constant reference to the storage element

5.28.3.2 at() [2/2]

```
template<typename T, size_t dim>
T& Triangle< T, dim >::at (
            size_t i,
            size_t j )  [inline]
```

at

Access function for the j-th element of the i-th set.

Parameters

i	the "row"
j	the "column"

Returns

a reference to the storage element

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

- include/Triangle.h

5.29 DataSet< T >::Variable Struct Reference

The Var struct.

```
#include <DataSet.h>
```

Public Member Functions

- Variable (T min_, T max_, const std::string &name_, const std::string &unit_="")
  Var.

Public Attributes

- T min
- T max
- std::string name
- std::string **unit**

5.29.1 Detailed Description

template<typename T>
struct DataSet< T >::Variable

The Var struct.

Var represents an entity attribute and stores its name, maximal and minimal value.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 Variable()

```
template<typename T >
DataSet< T >::Variable::Variable (
             T min_,
             T max_,
             const std::string & name_,
             const std::string & unit_ = "" )  [inline]
```

Var.

Creates a Variable with the given name, min and max value.

Parameters

min	the min value
max	the max value
name	the name

5.29.3 Member Data Documentation

5.29.3.1 max

```
template<typename T >
T DataSet< T >::Variable::max
```

Maximal value

5.29.3.2 min

```
template<typename T >
T DataSet< T >::Variable::min
```

Minimal value

5.29.3.3 name

```
template<typename T >
std::string DataSet< T >::Variable::name
```

Variable name

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

- include/DataSet.h