MooViE - Multi-objective optimization Visualization Engine

v0.2

Generated by Doxygen 1.8.13

Contents

1 Namespace Index **1**

1.1 Namespace List 1

2 Hierarchical Index **3**

2.1 Class Hierarchy 3

3 Class Index **5**

3.1 Class List 5

4 Namespace Documentation **7**

4.1 angle_helper Namespace Reference 7

4.1.1 Detailed Description 7

4.1.2 Function Documentation 7

4.1.2.1 deg_to_rad() 7

4.1.2.2 rad_to_deg() 8

5 Class Documentation **9**

5.1 Angle Class Reference 9

5.1.1 Detailed Description 10

5.1.2 Constructor & Destructor Documentation 10

5.1.2.1 Angle() 10

5.1.3 Member Function Documentation 10

5.1.3.1 center() 10

5.1.3.2 interpolate() 11

5.1.3.3 operator∗() 11

5.1.3.4 operator∗=() 11
5.1.3.5 operator+() 12
5.1.3.6 operator+=() 12
5.1.3.7 operator-() 13
5.1.3.8 operator-=() 13
5.1.3.9 operator/() 13
5.1.3.10 operator/=() 14
5.1.3.11 operator<() 14
5.1.3.12 operator<=() 14
5.1.3.13 operator=() 15
5.1.3.14 operator==() 15
5.1.3.15 operator>() 15
5.1.3.16 operator>=() 17
5.1.3.17 value() 17
5.2 CairoDrawer Class Reference 18
5.2.1 Detailed Description 19
5.2.2 Member Function Documentation 19
5.2.2.1 change_surface() 20
5.2.2.2 draw_arc() 20
5.2.2.3 draw_arrow() 20
5.2.2.4 draw_connector() 21
5.2.2.5 draw_connector_segment() 21
5.2.2.6 draw_coord_point() 22
5.2.2.7 draw_histogram() 22
5.2.2.8 draw_input_axis() 22
5.2.2.9 draw_io_vector() 23
5.2.2.10 draw_line() 23
5.2.2.11 draw_link() 23
5.2.2.12 draw_output_grid() 24
5.2.2.13 draw_output_label() 24

5.2.2.14 draw_ring_segment() 25
5.2.2.15 draw_segment_axis() 25
5.2.2.16 draw_text_orthogonal() 26
5.2.2.17 draw_text_parallel() 26
5.2.2.18 finish() 27
5.2.2.19 get_cairo_angle() 27
5.2.2.20 set_font_face() 27
5.2.2.21 set_surface() 27
5.3 Cartesian Class Reference 29
5.3.1 Detailed Description 29
5.3.2 Constructor & Destructor Documentation 30
5.3.2.1 Cartesian() 30
5.3.3 Member Function Documentation 30
5.3.3.1 center() 30
5.3.3.2 interpolate() 30
5.3.3.3 operator==() 31
5.3.3.4 x() [1/2] 31
5.3.3.5 x() [2/2] 32
5.3.3.6 y() [1/2] 32
5.3.3.7 y() [2/2] 32
5.4 DataSet< T >::Cell Struct Reference 32
5.4.1 Detailed Description 33
5.4.2 Constructor & Destructor Documentation 33
5.4.2.1 Cell() [1/2] 33
5.4.2.2 Cell() [2/2] 33
5.4.3 Member Data Documentation 33
5.4.3.1 null 34
5.4.3.2 value 34
5.5 Color Class Reference 34
5.5.1 Detailed Description 35

5.5.2 Constructor & Destructor Documentation 35
5.5.2.1 Color() 35
5.5.3 Member Function Documentation 35
5.5.3.1 a() 35
5.5.3.2 b() 36
5.5.3.3 g() 36
5.5.3.4 operator"!=() 36
5.5.3.5 operator==() 36
5.5.3.6 r() 37
5.5.3.7 set_alpha() 37
5.5.3.8 set_blue() 37
5.5.3.9 set_green() 38
5.5.3.10 set_red() 38
5.5.4 Friends And Related Function Documentation 38
5.5.4.1 operator<< 38
5.5.5 Member Data Documentation 38
5.5.5.1 BLACK 39
5.6 Configuration Class Reference 39
5.6.1 Detailed Description 40
5.6.2 Member Function Documentation 41
5.6.2.1 get_grid_size() 41
5.6.2.2 get_height() 41
5.6.2.3 get_histogram_background() 41
5.6.2.4 get_histogram_fill() 41
5.6.2.5 get_histogram_height() 42
5.6.2.6 get_input_file() 42
5.6.2.7 get_input_inner_radius() 42
5.6.2.8 get_input_separation_angle() 42
5.6.2.9 get_input_thickness() 43
5.6.2.10 get_instance() 43

5.6.2.11 get_num_histogram_classes() 43
5.6.2.12 get_num_major_sections_axis() 43
5.6.2.13 get_num_major_sections_grid() 44
5.6.2.14 get_num_minor_sections_axis() 44
5.6.2.15 get_num_minor_sections_grid() 44
5.6.2.16 get_output_angle_span() 44
5.6.2.17 get_output_file() 45
5.6.2.18 get_output_inner_radius() 45
5.6.2.19 get_output_thickness() 45
5.6.2.20 get_prop_axis_label() 45
5.6.2.21 get_prop_scale_label() 46
5.6.2.22 get_prop_thick() 46
5.6.2.23 get_prop_thin() 46
5.6.2.24 get_relevant_places() 46
5.6.2.25 get_width() 47
5.6.2.26 initialize() [1/2] 47
5.6.2.27 initialize() [2/2] 47
5.6.2.28 is_histograms_enabled() 47
5.6.2.29 save_to_file() 48
5.6.2.30 set_grid_size() 48
5.6.2.31 set_height() 48
5.6.2.32 set_histogram_background() 48
5.6.2.33 set_histogram_fill() 49
5.6.2.34 set_histogram_height() 49
5.6.2.35 set_histograms_enabled() 49
5.6.2.36 set_input_inner_radius() 50
5.6.2.37 set_input_separation_angle() 50
5.6.2.38 set_input_thickness() 50
5.6.2.39 set_num_histogram_classes() 50
5.6.2.40 set_num_major_sections_axis() 51

5.6.2.41 set_num_major_sections_grid() 51
5.6.2.42 set_num_minor_sections_axis() 51
5.6.2.43 set_num_minor_sections_grid() 52
5.6.2.44 set_output_angle_span() 52
5.6.2.45 set_output_file() 52
5.6.2.46 set_output_inner_radius() 52
5.6.2.47 set_output_thickness() 53
5.6.2.48 set_prop_axis_label() 53
5.6.2.49 set_prop_scale_label() 53
5.6.2.50 set_prop_thick() 54
5.6.2.51 set_prop_thin() 54
5.6.2.52 set_relevant_places() 54
5.6.2.53 set_width() 54
5.6.3 Member Data Documentation 55
5.6.3.1 GLOW_10 55
5.6.3.2 SET2_3_1 55
5.6.3.3 SET3 55
5.7 DataSet< T >::DataRow::const_iterator Class Reference 55
5.8 DataSet< T >::const_iterator Class Reference 56
5.9 CoordinateConverter Class Reference 56
5.9.1 Detailed Description 57
5.9.2 Constructor & Destructor Documentation 57
5.9.2.1 CoordinateConverter() 57
5.9.3 Member Function Documentation 57
5.9.3.1 convert() [1/2] 57
5.9.3.2 convert() [2/2] 58
5.9.3.3 get_center_x() 58
5.9.3.4 get_center_y() 58
5.10 DataSet< T >::DataColumn Struct Reference 59
5.10.1 Detailed Description 59

5.10.2 Constructor & Destructor Documentation 59

5.10.2.1 DataColumn() 59

5.10.3 Member Data Documentation 60

5.10.3.1 cells 60

5.10.3.2 type 60

5.10.3.3 var 60

5.11 DataSet< T >::DataRow Class Reference 60

5.11.1 Detailed Description 61

5.11.2 Constructor & Destructor Documentation 61

5.11.2.1 DataRow() 61

5.11.3 Member Function Documentation 61

5.11.3.1 begin() 61

5.11.3.2 end() 62

5.11.3.3 is_enabled() 62

5.11.3.4 operator[]() 62

5.11.3.5 set_enabled() 62

5.11.3.6 size() 63

5.12 DataSet< T > Class Template Reference 63

5.12.1 Detailed Description 64

5.12.2 Member Enumeration Documentation 64

5.12.2.1 ColumnType 64

5.12.3 Constructor & Destructor Documentation 65

5.12.3.1 DataSet() `[1/2]` 65

5.12.3.2 DataSet() `[2/2]` 65

5.12.4 Member Function Documentation 65

5.12.4.1 begin() 65

5.12.4.2 end() 65

5.12.4.3 get_num_active_cols() 66

5.12.4.4 get_num_active_inputs() 66

5.12.4.5 get_num_active_outputs() 66

5.12.4.6 get_num_cols() 66
5.12.4.7 get_num_inputs() 67
5.12.4.8 get_num_outputs() 67
5.12.4.9 get_num_rows() 67
5.12.4.10 input_variables() 67
5.12.4.11 operator[]() 68
5.12.4.12 output_variables() 68
5.12.4.13 parse_from_csv() 68
5.12.4.14 restrict_column() 69
5.12.4.15 swap_columns() 69
5.12.4.16 toggle_column() 69
5.13 Drawer Class Reference 70
5.13.1 Detailed Description 71
5.13.2 Constructor & Destructor Documentation 72
5.13.2.1 Drawer() 72
5.13.3 Member Function Documentation 72
5.13.3.1 change_surface() 72
5.13.3.2 create_link_control_point() 72
5.13.3.3 draw_arc() 73
5.13.3.4 draw_arrow() 73
5.13.3.5 draw_connector() 74
5.13.3.6 draw_connector_segment() 74
5.13.3.7 draw_coord_point() 74
5.13.3.8 draw_histogram() 75
5.13.3.9 draw_input_axis() 75
5.13.3.10 draw_io_vector() 76
5.13.3.11 draw_line() 76
5.13.3.12 draw_link() 76
5.13.3.13 draw_output_grid() 77
5.13.3.14 draw_output_label() 77

5.13.3.15 draw_ring_segment() 78
5.13.3.16 draw_segment_axis() 78
5.13.3.17 draw_text_orthogonal() 79
5.13.3.18 draw_text_parallel() 79
5.13.3.19 finish() 79
5.13.3.20 get_connector_end() 80
5.13.3.21 get_connector_start() 80
5.13.3.22 set_surface() 80
5.13.4 Member Data Documentation 81
5.13.4.1 m_coord_converter 81
5.13.4.2 m_num_inputs 81
5.14 DrawerProperties< FillT > Struct Template Reference 81
5.14.1 Detailed Description 82
5.14.2 Constructor & Destructor Documentation 82
5.14.2.1 DrawerProperties() 82
5.14.3 Member Data Documentation 82
5.14.3.1 fill_color 82
5.14.3.2 line_color 83
5.14.3.3 line_width 83
5.15 InputAxis::Histogram Class Reference 83
5.15.1 Constructor & Destructor Documentation 83
5.15.1.1 Histogram() 83
5.15.2 Member Function Documentation 84
5.15.2.1 calculate() 84
5.15.2.2 get_num_intervals() 84
5.15.2.3 get_section_frequency() 84
5.15.2.4 set_num_intervals() 85
5.16 InputAxis Class Reference 85
5.16.1 Detailed Description 86
5.16.2 Constructor & Destructor Documentation 86

5.16.2.1 InputAxis() 86
5.16.3 Member Function Documentation 86
5.16.3.1 calculate_histogram() 87
5.16.3.2 get_end() 87
5.16.3.3 get_height() 87
5.16.3.4 get_histogram() 87
5.16.3.5 get_prop() 88
5.16.3.6 get_radius() 88
5.16.3.7 get_scale() 88
5.16.3.8 get_start() 88
5.16.3.9 get_var() 89
5.16.3.10 make_label() 89
5.16.3.11 set_end() 89
5.16.3.12 set_height() 89
5.16.3.13 set_prop() 90
5.16.3.14 set_radius() 90
5.16.3.15 set_start() 90
5.17 IOVector Class Reference 90
5.17.1 Detailed Description 91
5.17.2 Member Function Documentation 91
5.17.2.1 emplace_back() 91
5.17.2.2 operator[]() 91
5.17.2.3 size() 92
5.18 IOVectorFactory Class Reference 92
5.18.1 Detailed Description 92
5.18.2 Constructor & Destructor Documentation 92
5.18.2.1 IOVectorFactory() 92
5.18.3 Member Function Documentation 93
5.18.3.1 create() 93
5.19 Label Class Reference 93

5.19.1 Detailed Description 94
5.19.2 Constructor & Destructor Documentation 94
5.19.2.1 Label() 94
5.19.3 Member Function Documentation 94
5.19.3.1 get_properties() 94
5.19.3.2 get_text() 95
5.20 Mapper Class Reference 95
5.20.1 Detailed Description 95
5.20.2 Constructor & Destructor Documentation 95
5.20.2.1 Mapper() 95
5.20.3 Member Function Documentation 96
5.20.3.1 inverse() 96
5.20.3.2 map() 96
5.21 DataSet< T >::MockColumn Class Reference 97
5.21.1 Detailed Description 97
5.21.2 Constructor & Destructor Documentation 97
5.21.2.1 MockColumn() 97
5.21.3 Member Function Documentation 98
5.21.3.1 get_type() 98
5.21.3.2 get_var() 98
5.21.3.3 is_enabled() 98
5.21.3.4 operator[]() 98
5.21.3.5 set_enabled() 99
5.21.3.6 size() 99
5.21.3.7 swap() 99
5.22 MultiScale Class Reference 100
5.22.1 Detailed Description 100
5.22.2 Constructor & Destructor Documentation 100
5.22.2.1 MultiScale() 100
5.22.3 Member Function Documentation 101

5.22.3.1 add_scale() 101
5.22.3.2 get_extremes() 101
5.22.3.3 get_scale_number() 101
5.22.3.4 make_labels() 102
5.23 OutputGrid Class Reference 102
5.23.1 Detailed Description 102
5.23.2 Constructor & Destructor Documentation 103
5.23.2.1 OutputGrid() 103
5.23.3 Member Function Documentation 103
5.23.3.1 get_direction() 103
5.23.3.2 get_end() 103
5.23.3.3 get_height() 104
5.23.3.4 get_num_outputs() 104
5.23.3.5 get_radius() 104
5.23.3.6 get_scale() 104
5.23.3.7 get_start() 105
5.23.3.8 get_var() 105
5.23.3.9 set_direction() 105
5.23.3.10 set_end() 105
5.23.3.11 set_height() 106
5.23.3.12 set_radius() 106
5.23.3.13 set_start() 106
5.24 Point Struct Reference 107
5.24.1 Detailed Description 107
5.24.2 Constructor & Destructor Documentation 107
5.24.2.1 Point() 107
5.24.3 Member Data Documentation 108
5.24.3.1 coord 108
5.24.3.2 prop 108
5.25 Polar Class Reference 108

5.25.1 Detailed Description 109
5.25.2 Constructor & Destructor Documentation 109
5.25.2.1 Polar() 109
5.25.3 Member Function Documentation 109
5.25.3.1 angle() [1/2] 109
5.25.3.2 angle() [2/2] 110
5.25.3.3 center() 110
5.25.3.4 interpolate() 110
5.25.3.5 operator==() 111
5.25.3.6 radius() [1/2] 111
5.25.3.7 radius() [2/2] 111
5.26 Scale Class Reference 112
5.26.1 Detailed Description 112
5.26.2 Constructor & Destructor Documentation 112
5.26.2.1 Scale() 112
5.26.3 Member Function Documentation 113
5.26.3.1 get_major_intersections() 113
5.26.3.2 get_minor_intersections() 113
5.27 Scene Class Reference 113
5.27.1 Detailed Description 114
5.27.2 Constructor & Destructor Documentation 114
5.27.2.1 Scene() 114
5.27.3 Member Function Documentation 114
5.27.3.1 get_input_variables() 114
5.27.3.2 get_output_variables() 115
5.27.3.3 restrict_input() 115
5.27.3.4 restrict_output() 115
5.27.3.5 swap_inputs() 116
5.27.3.6 swap_outputs() 116
5.27.3.7 toggle_input() 116

5.27.3.8 toggle_output() 117
5.27.3.9 update() 117
5.28 SimpleScale Class Reference 117
5.28.1 Detailed Description 118
5.28.2 Constructor & Destructor Documentation 118
5.28.2.1 SimpleScale() 118
5.28.3 Member Function Documentation 119
5.28.3.1 get_extremes() 119
5.28.3.2 make_labels() 119
5.29 Drawer::TextAlignment Struct Reference 119
5.29.1 Detailed Description 120
5.30 TextProperties Struct Reference 120
5.30.1 Detailed Description 120
5.30.2 Constructor & Destructor Documentation 121
5.30.2.1 TextProperties() 121
5.30.3 Member Data Documentation 121
5.30.3.1 bold 121
5.30.3.2 color 121
5.30.3.3 font_name 121
5.30.3.4 font_size 122
5.30.3.5 italic 122
5.31 Triangle< T, dim > Class Template Reference 122
5.31.1 Detailed Description 122
5.31.2 Constructor & Destructor Documentation 123
5.31.2.1 Triangle() [1/2] 123
5.31.2.2 Triangle() [2/2] 123
5.31.3 Member Function Documentation 123
5.31.3.1 at() [1/2] 123
5.31.3.2 at() [2/2] 124
5.32 DataSet< T >::Variable Struct Reference 124
5.32.1 Detailed Description 124
5.32.2 Constructor & Destructor Documentation 125
5.32.2.1 Variable() 125
5.32.3 Member Data Documentation 125
5.32.3.1 max 125
5.32.3.2 min 125
5.32.3.3 name 125
5.32.3.4 unit 126

Chapter 1

Namespace Index

1.1 Namespace List

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

angle_helper 7

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

- Angle . . . 9
- Cartesian . . . 29
- DataSet< T >::Cell . . . 32
- Color . . . 34
- Configuration . . . 39
- CoordinateConverter . . . 56
- DataSet< T >::DataColumn . . . 59
- DataSet< T >::DataRow . . . 60
- DataSet< T > . . . 63
- DataSet< double > . . . 63
- Drawer . . . 70
  - CairoDrawer . . . 18
- DrawerProperties< FillT > . . . 81
- InputAxis::Histogram . . . 83
- InputAxis . . . 85
- IOVector . . . 90
- IOVectorFactory . . . 92
- iterator
  - DataSet< T >::const_iterator . . . 56
  - DataSet< T >::DataRow::const_iterator . . . 55
- Label . . . 93
- Mapper . . . 95
- DataSet< T >::MockColumn . . . 97
- OutputGrid . . . 102
- Point . . . 107
- Polar . . . 108
- Scale . . . 112
  - MultiScale . . . 100
  - SimpleScale . . . 117
- Scene . . . 113
- Drawer::TextAlignment . . . 119
- TextProperties . . . 120
- Triangle< T, dim > . . . 122
- Triangle< Color, 12 > . . . 122
- DataSet< T >::Variable . . . 124

Chapter 3

Class Index

3.1 Class List

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

Angle
    Mathmatical angle representation 9
CairoDrawer
    SVG surface drawer for MooViE 18
Cartesian
    The Cartesian class 29
DataSet< T >::Cell
    Cell of a data table 32
Color
    RGB color representation 34
Configuration
    Configuration for a MooViE run 39
DataSet< T >::DataRow::const_iterator 55
DataSet< T >::const_iterator 56
CoordinateConverter
    Converter between polar and cartesian coordinates 56
DataSet< T >::DataColumn
    Column of a data table 59
DataSet< T >::DataRow
    Row of a data table 60
DataSet< T >
    Table of data 63
Drawer
    Abstract Drawer for MooViE scenes 70
DrawerProperties< FillT >
    Properties to modify a MooViE drawers behavior 81
InputAxis::Histogram 83
InputAxis
    InputAxis MooViE component representation 85
IOVector
    IOVector MooViE component representation 90
IOVectorFactory 92
Label
    Text label MooViE component representation 93
Mapper
    Mapper is a bijective function f: [a,b] -> [c,d] 95

DataSet< T >::MockColumn
Technical column for internal use 97
MultiScale 100
OutputGrid
OutputGrid MooViE component representation 102
Point
Styled polar coordinate 107
Polar
Polar coordinate representation 108
Scale
Ticked scale 112
Scene
MooViE scene 113
SimpleScale 117
Drawer::TextAlignment
Text alignment representation 119
TextProperties
Properties to modify a MooViE Drawers text style 120
Triangle< T, dim >
Triangular set storage 122
DataSet< T >::Variable
Header description 124

Chapter 4

Namespace Documentation

4.1 angle_helper Namespace Reference

Functions

- double deg_to_rad (double deg)
- double rad_to_deg (double rad)
- 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]
```

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]
```

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

Mathmatical angle representation.

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

Public Member Functions

- Angle (double angle)

  constructor
- double value () const
- double operator= (double angle)
- bool operator== (const Angle &rhs) const
- bool operator< (const Angle &rhs) const
- bool operator<= (const Angle &rhs) const
- bool operator> (const Angle &rhs) const
- bool operator>= (const Angle &rhs) const
- Angle & operator+= (const Angle &rhs)
- Angle operator+ (const Angle &rhs) const

  this + rhs
- Angle & operator-= (const Angle &rhs)
- Angle operator- (const Angle &rhs) const

  this - rhs
- Angle & operator*= (double val)
- Angle operator* (double val) const
- Angle & operator/= (double val)
- Angle operator/ (double val)

Static Public Member Functions

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

5.1.1 Detailed Description

Mathmatical angle representation.

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]
```

constructor

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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= (
             double angle )  [inline]
```

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

Parameters

angle	the angle value in radians

Returns

the adjusted angle value

5.1.3.14 operator==()

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

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]
```

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]
```

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]
```

Returns the value of this angle.

Returns

the angle value

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

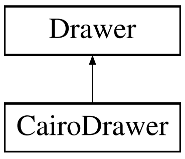
- include/Coordinates.h

5.2 CairoDrawer Class Reference

SVG surface drawer for MooViE.

```
#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, std::size_t num_inputs_)
- virtual void draw_output_grid (const OutputGrid &grid)
- virtual void draw_input_axis (const InputAxis &axis)
- virtual void draw_io_vector (const IOVector &iov)
- virtual void finish ()

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 **IO_LINE_WIDTH**
- static const double **CONNECTOR_ARC_RATIO**
- static const double **CONNECTOR_ARROW_HEIGHT**
- 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_FONT_FACTOR**
- static const double **INPUT_AXIS_FONT_FACTOR**
- static const double **INPUT_TICK_FONT_FACTOR**

Protected Member Functions

- virtual void set_surface (const std::string &fpath, int width, int height)
- virtual void draw_histogram (const InputAxis::Histogram &histogram, double radius, const Angle &start, const Angle &end)
- virtual void draw_link (const Polar &origin1, const Polar &origin2, const Polar &target1, const Polar &target2, const DrawerProperties<> &prop)
- virtual void draw_connector (const Polar &from, const Polar &to, const DrawerProperties<> &prop)
- 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)
- virtual void draw_output_label (const Label &output_label, double radius_label, double radius_output, const Angle &begin, const Angle &end)
- virtual void draw_arrow (const Polar &start, const DrawerProperties<> &prop)
- virtual void draw_ring_segment (double radius, double thickness, const Angle &begin, const Angle &end, const DrawerProperties<> &prop, Direction dir)
- virtual void draw_connector_segment (double begin_radius, double begin_angle, double end_radius, double end_angle, const DrawerProperties<> &prop)
- virtual void draw_line (const Polar &from, const Polar &to, const DrawerProperties<> &prop)
- virtual void draw_arc (double inner_radius, const Angle &start, const Angle &end, Direction dir)
- virtual void draw_coord_point (const Polar &coord, const Angle &width, double height, const Drawer←Properties<> &prop)
- virtual void draw_text_parallel (const Label &label, const Polar &start, const TextAlignment &alignment=Text←Alignment::CENTERED)
- virtual void draw_text_orthogonal (const Label &label, const Polar &start, const TextAlignment &alignment=TextAlignment::CENTERED)
- 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

SVG surface drawer for MooViE.

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

Authors

beyss, stratmann

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,
             std::size_t num_inputs )  [virtual]
```

Alters the surface of this Drawer in number of inputs, width, 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
num_inputs	the number of inputs

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 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 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_connector()

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

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.

5.2.2.5 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 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.

5.2.2.6 draw_coord_point()

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

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.

5.2.2.7 draw_histogram()

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

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.

5.2.2.8 draw_input_axis()

```
virtual void CairoDrawer::draw_input_axis (
            const InputAxis & axis )  [virtual]
```

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

Parameters

axis	the InputAxis to draw

Implements Drawer.

5.2.2.9 draw_io_vector()

```
virtual void CairoDrawer::draw_io_vector (
            const IOVector & elem )  [virtual]
```

Draws a IOVector using its coordinates.

Parameters

elem	the IOVector to draw

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 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 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_grid()

```
virtual void CairoDrawer::draw_output_grid (
                const OutputGrid & grid )  [virtual]
```

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

Parameters

grid	the OutputGrid to draw

Implements Drawer.

5.2.2.13 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 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

Parameters

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.

5.2.2.14 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 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 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 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 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 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 90° 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]
```

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 R^2 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

Cell of a data table.

```
#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

Cell of a data table.

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 Color Class Reference

RGB color representation.

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

Public Member Functions

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

Static Public Attributes

- static const Color BLACK

Friends

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

5.5.1 Detailed Description

RGB color representation.

Color class represents a color by RGB and alpha value.

Authors

beyss, stratmann

Date

27.07.2017

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Color()

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

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

5.5.3 Member Function Documentation

5.5.3.1 a()

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

Access function for the color's alpha value.

Returns

a reference to the colors alpha value

5.5.3.2 b()

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

Access function for the color's blue value.

Returns

a reference to the colors blue value

5.5.3.3 g()

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

Access function for the color's green value.

Returns

a reference to the colors green value

5.5.3.4 operator"!=()

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

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 equal

5.5.3.5 operator==()

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

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

5.5.3.6 r()

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

Access function for the color's red value.

Returns

a reference to the colors red value

5.5.3.7 set_alpha()

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

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

5.5.3.8 set_blue()

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

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

5.5.3.9 set_green()

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

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

5.5.3.10 set_red()

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

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

5.5.4 Friends And Related Function Documentation

5.5.4.1 operator<<

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

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

Parameters

o	the ostream to put into
c	the color to put

5.5.5 Member Data Documentation

5.5.5.1 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

5.6 Configuration Class Reference

Configuration for a MooViE run.

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

Public Member Functions

- const std::string & get_input_file () const
- void **set_input_file** (const std::string &input_file)
- 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)
- int get_relevant_places () const
- void set_relevant_places (int relevant_places)
- 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 void save_to_file (const std::string &cpath)

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**

5.6.1 Detailed Description

Configuration for a MooViE run.

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

5.6.2 Member Function Documentation

5.6.2.1 get_grid_size()

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

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

Returns

the m_grid_size

5.6.2.2 get_height()

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

Returns the height of the MooViE scene

Returns

the height

5.6.2.3 get_histogram_background()

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

Returns the background color that each histogram has.

Returns

the histogram background color

5.6.2.4 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

5.6.2.5 get_histogram_height()

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

Returns the height that each histogram has.

Returns

the histogram height

5.6.2.6 get_input_file()

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

Returns the path to the input file.

Returns

the input file path

5.6.2.7 get_input_inner_radius()

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

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

Returns

the input inner radius

5.6.2.8 get_input_separation_angle()

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

Returns the seperation angle between inputs.

Returns

the input separation angle

5.6.2.9 get_input_thickness()

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

Returns the thickness of the colored ring of the InputAxis.

Returns

the input thickness

5.6.2.10 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

5.6.2.11 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

5.6.2.12 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 InputAxis.

Returns

the number of major sections

5.6.2.13 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 OutputGrid.

Returns

the number of major sections

5.6.2.14 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 InputAxis.

Returns

the number of minor sections

5.6.2.15 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 OutputGrid.

Returns

the number of minor sections

5.6.2.16 get_output_angle_span()

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

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

Returns

the output angle span

5.6.2.17 get_output_file()

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

Returns the path to the output file.

Returns

the output file path

5.6.2.18 get_output_inner_radius()

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

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

Returns

the output inner radius

5.6.2.19 get_output_thickness()

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

Returns the thickness of the outputs colored segmented ring.

Returns

the output thickness

5.6.2.20 get_prop_axis_label()

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

Returns MooViEs TextProperties for InputAxis labels.

Returns

the TextProperties for InputAxis labels

5.6.2.21 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

5.6.2.22 get_prop_thick()

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

Returns MooViEs DrawerProperties for thick black lines.

Returns

the DrawerProperties for thick lines

5.6.2.23 get_prop_thin()

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

Returns MooViEs DrawerProperties for thin black lines.

Returns

the DrawerProperties for thin lines

5.6.2.24 get_relevant_places()

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

Returns the decimal place that is every column cell is rounded to.

Returns

the decimal place

5.6.2.25 get_width()

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

Returns the width of the MooViE scene

Returns

the width

5.6.2.26 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

5.6.2.27 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

5.6.2.28 is_histograms_enabled()

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

Returns whether or not histograms should be drawn.

Returns

histograms enabled or not

5.6.2.29 save_to_file()

```
static void Configuration::save_to_file (
             const std::string & cpath )  [static]
```

Writes the current configuration instance to the specified file path.

Parameters

cpath	the path to save the configuration file to

5.6.2.30 set_grid_size()

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

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

Parameters

grid_size	the m_grid_size to set

5.6.2.31 set_height()

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

Sets the height of a MooViE scene.

Parameters

height	the height to set

5.6.2.32 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

5.6.2.33 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

5.6.2.34 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

5.6.2.35 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

5.6.2.36 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 InputAxis start.

Parameters

input_inner_radius	the input inner radius to set

5.6.2.37 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

5.6.2.38 set_input_thickness()

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

Sets the thickness of the colored ring of the InputAxis.

Parameters

input_thickness	the input thickness to set

5.6.2.39 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

5.6.2.40 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 InputAxis.

Parameters

major_sections	the number of major sections to set

5.6.2.41 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 OutputGrid.

Parameters

major_sections	the number of major sections to set

5.6.2.42 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 InputAxis.

Parameters

minor_sections	the number minor sections to set

5.6.2.43 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 OutputGrid.

Parameters

minor_sections	the number of minor sections to set

5.6.2.44 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 OutputGrid.

Parameters

output_angle_span	the output angle span to set

5.6.2.45 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

5.6.2.46 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 OutputGrid starts.

Parameters

output_inner_radius	the output inner radius to set

5.6.2.47 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

5.6.2.48 set_prop_axis_label()

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

Sets MooViEs TextProperties for InputAxis labels.

Parameters

prop_axis_label	the TextProperties to set

5.6.2.49 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

5.6.2.50 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

5.6.2.51 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

5.6.2.52 set_relevant_places()

```
void Configuration::set_relevant_places (
             int relevant_places )  [inline]
```

Sets the decimal place that is every column cell is rounded to.

Parameters

relevant_places	the epsilon places to set

5.6.2.53 set_width()

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

Sets the width of a MooViE scene.

Parameters

width	the width to set

5.6.3 Member Data Documentation

5.6.3.1 GLOW_10

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

An array of Colors

5.6.3.2 SET2_3_1

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

Further color constants

5.6.3.3 SET3

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

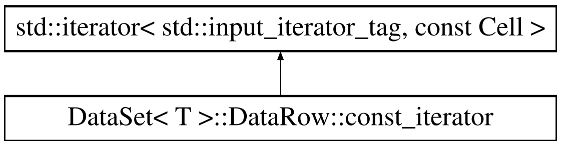
A Triangular storage which contains i+1 matching colors at the i-th index.

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

- include/Configuration.h

5.7 DataSet< T >::DataRow::const_iterator Class Reference

Inheritance diagram for DataSet< T >::DataRow::const_iterator:


std::iterator< std::input_iterator_tag, const Cell >
DataSet< T >::DataRow::const_iterator

Public Member Functions

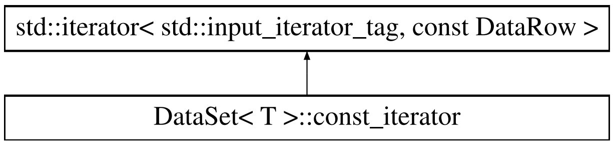
- **const_iterator** (const typename std::vector< MockColumn >::const_iterator &_it, const typename std↩ ::vector< MockColumn >::const_iterator &_end, std::size_t _offset)
- const_iterator & **operator++** ()
- const_iterator **operator++** (int)
- bool **operator==** (const const_iterator &other) const
- bool **operator!=** (const const_iterator &other) const
- const Cell & **operator*** () const

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

- include/DataSet.h

5.8 DataSet< T >::const_iterator Class Reference

Inheritance diagram for DataSet< T >::const_iterator:


std::iterator< std::input_iterator_tag, const DataRow >
DataSet< T >::const_iterator

Public Member Functions

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

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

- include/DataSet.h

5.9 CoordinateConverter Class Reference

Converter between polar and cartesian coordinates.

```
#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
- void convert (const Polar &from, Cartesian &to) const
- double get_center_x () const
- double get_center_y () const

5.9.1 Detailed Description

Converter between polar and cartesian coordinates.

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

5.9.2 Constructor & Destructor Documentation

5.9.2.1 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

5.9.3 Member Function Documentation

5.9.3.1 convert() [1/2]

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

Converts a Cartesian coordinate to a Polar coordinate.

Parameters

from	the Cartesian to convert
to	the Polar to store

5.9.3.2 convert() [2/2]

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

Converts a Polar coordinate to a Cartesian coordinate.

Parameters

from	the Polar to convert
to	the Polar to store

5.9.3.3 get_center_x()

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

Returns the x value of the center coordinate.

Returns

the center's x value

5.9.3.4 get_center_y()

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

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

5.10 DataSet< T >::DataColumn Struct Reference

Column of a data table.

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

Public Member Functions

- DataColumn (ColumnType type_, Variable var_)

Public Attributes

- const ColumnType type
- Variable var
- std::vector< Cell > cells

5.10.1 Detailed Description

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

Column of a data table.

DataColumn represents a column of a data set. It has a type (INPUT, OUTPUT), a Variable and a set of cells

5.10.2 Constructor & Destructor Documentation

5.10.2.1 DataColumn()

```
template<typename T>
DataSet< T >::DataColumn::DataColumn (
            ColumnType type_,
            Variable var_ )  [inline]
```

Creates a new column with given ColumnType and Variable.

Parameters

*type_*	the ColumnType
*var_*	the Variable

5.10.3 Member Data Documentation

5.10.3.1 cells

```
template<typename T>
std::vector<Cell> DataSet< T >::DataColumn::cells
```

An array of the cells of this column

5.10.3.2 type

```
template<typename T>
const ColumnType DataSet< T >::DataColumn::type
```

The ColumnType

5.10.3.3 var

```
template<typename T>
Variable DataSet< T >::DataColumn::var
```

The header information about this column (name, unit, range)

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

- include/DataSet.h

5.11 DataSet< T >::DataRow Class Reference

Row of a data table.

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

Classes

- class const_iterator

Public Member Functions

- DataRow (const std::vector< MockColumn > &columns, const std::size_t &enabled_columns, std::size_↩
t offset)
- const Cell & operator[ ] (std::size_t i) const
- void set_enabled (bool enabled)
- std::size_t size () const
- bool is_enabled () const
- const_iterator begin () const
- const_iterator end () const

5.11.1 Detailed Description

template<typename T>
class DataSet< T >::DataRow

Row of a data table.

DataRow represents a row in this DataSet. A DataRow does not become invalid when column order is changed or a column is disabled. It might get invalid when restricting columns to a certain interval.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 DataRow()

```
template<typename T>
DataSet< T >::DataRow::DataRow (
             const std::vector< MockColumn > & columns,
             const std::size_t & enabled_columns,
             std::size_t offset )  [inline]
```

Creates a DataRow from given reference to the columns and to the number of enabled columns (needed for update) and the row number (aka column offset).

Parameters

columns	a reference to the column array
enabled_columns	a reference to the number of enabled columns
offset	the row offset

5.11.3 Member Function Documentation

5.11.3.1 begin()

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

Returns a const_iterator pointing to the first cell in this DataRow.

Returns

the iterator

5.11.3.2 end()

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

Returns a const_iterator pointing to the end of this DataRow

Returns

the iterator

5.11.3.3 is_enabled()

```
template<typename T>
bool DataSet< T >::DataRow::is_enabled ( ) const  [inline]
```

Returns the value of the enabled flag.

Returns

enabled or not

5.11.3.4 operator[]()

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

Accesses the i-th Cell of this DataRow.

Parameters

i	the index

Returns

the Cell

5.11.3.5 set_enabled()

```
template<typename T>
void DataSet< T >::DataRow::set_enabled (
              bool enabled )  [inline]
```

Sets the enabled flag of this MockColumn to the specified value.

Parameters

enabled	set enabled or not

5.11.3.6 size()

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

Returns the size of this MockColumn.

Returns

the size

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

- include/DataSet.h

5.12 DataSet< T > Class Template Reference

Table of data.

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

Classes

- struct Cell

  Cell of a data table.
- class const_iterator
- struct DataColumn

  Column of a data table.
- class DataRow

  Row of a data table.
- class MockColumn

  Technical column for internal use.
- struct Variable

  Header description.

Public Types

- enum ColumnType { **INPUT**, **OUTPUT** }

Public Member Functions

- DataSet ()
- DataSet (const std::string &fpath)
- void parse_from_csv (const std::string &cont, std::string separator=",", std::string comment="#", std::string newline="\)
- void toggle_column (std::size_t c, bool mode)
- void swap_columns (std::size_t c0, std::size_t c1)
- void restrict_column (std::size_t c, T l_restr, T u_restr)
- std::size_t get_num_active_cols () const
- std::size_t get_num_cols () const
- std::size_t get_num_rows () const
- std::size_t get_num_active_inputs () const
- std::size_t get_num_inputs () const
- std::size_t get_num_active_outputs () const
- std::size_t get_num_outputs () const
- const DataRow & operator[ ] (std::size_t i) const
- std::vector< Variable > input_variables (void) const
- std::vector< Variable > output_variables (void) const
- const_iterator begin () const
- const_iterator end () const

5.12.1 Detailed Description

template<typename T>
class DataSet< T >

Table of data.

A class for storing data of some type. It is accessible row-wise and not directly changeble, but columns can be swapped, toggled and restricted. Rows are divided in INPUTs and OUTPUTs, so they can be used in MooViE terms. It can also be parsed from a CSV file.

Author

stratmann

Date

28.11.2017

5.12.2 Member Enumeration Documentation

5.12.2.1 ColumnType

```
template<typename T>
enum DataSet::ColumnType
```

MooViE columns can either represent outputs or inputs. This is indicated by a member of the type ColumnType.

5.12.3 Constructor & Destructor Documentation

5.12.3.1 DataSet() [1/2]

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

Creates an empty DataSet.

5.12.3.2 DataSet() [2/2]

```
template<typename T>
DataSet< T >::DataSet (
            const std::string & fpath )  [inline]
```

Parses a DataSet from a given CSV file. The table must have the form: <input1>[<uniti1>], ... , <input↩
N>[<unitiN>], <output1>[<unito1>], ... , <outputM>[<unitoM>] <datai1>, ... , <dataiN>, <datao1>, ... ,
<dataoM>

Parameters

fpath	the CSV file path

5.12.4 Member Function Documentation

5.12.4.1 begin()

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

Returns a constant iterator pointing to the first DataRow.

Returns

a const_iterator

5.12.4.2 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

5.12.4.3 get_num_active_cols()

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

Returns the number of active columns in this table. For every toggled-off column this size decreases by 1.

Returns

the number of active columns

5.12.4.4 get_num_active_inputs()

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

Returns the number of active inputs in this table. For every toggled-off column this size decreases by 1.

Returns

the number of active inputs

5.12.4.5 get_num_active_outputs()

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

Returns the number of outputs in this table. For every toggled-off column this size decreases by 1.

Returns

the number of active outputs

5.12.4.6 get_num_cols()

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

Returns the total number of columns in this table. This includes toggled-off columns.

Returns

the number of total columns

5.12.4.7 get_num_inputs()

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

Returns the total number of inputs in this table. This includes toggled-off columns.

Returns

the total number of inputs

5.12.4.8 get_num_outputs()

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

Returns the total number of outputs in this table. This includes toggled-off columns.

Returns

the total number of outputs

5.12.4.9 get_num_rows()

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

Returns the number of rows in this table.

Returns

the number of rows

5.12.4.10 input_variables()

```
template<typename T>
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

5.12.4.11 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

5.12.4.12 output_variables()

```
template<typename T>
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

5.12.4.13 parse_from_csv()

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

Returns a data table parsed from a csv encoded string and encapsulated in a DataSet object. The table must have the form: <input1>[<uniti1>], ... , <inputN>[<unitiN>], <output1>[<unito1>], ... , <outputM>[<unitoM>] <datai1>, ... , <dataiN>, <datao1>, ... , <dataoM>

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

5.12.4.14 restrict_column()

```
template<typename T>
void DataSet< T >::restrict_column (
            std::size_t c,
            T l_restr,
            T u_restr )  [inline]
```

Restricts a column to values in the given interval. The DataRows that contain a Cell not fitting in this interval will be disabled.

Parameters

c	the column index
l_restr	lower restriction value
u_restr	upper restriction value

Exceptions

out←_of	range if c is incorrect

5.12.4.15 swap_columns()

```
template<typename T>
void DataSet< T >::swap_columns (
            std::size_t c0,
            std::size_t c1 )  [inline]
```

Swaps the two columns with the given index. The DataRows are changed accordingly.

Parameters

c0	the index of the first column
c1	the index of the second column

Exceptions

out_of_bounds	if indices are incorrect

5.12.4.16 toggle_column()

```
template<typename T>
```

```
void DataSet< T >::toggle_column (
            std::size_t c,
            bool mode )  [inline]
```

Enables/disables a column. The DataRows now do not contain the affected Cell anymore.

Parameters

c	the column index
mode	set enabled or disabled

Exceptions

out_of_range	id c is incorrect

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

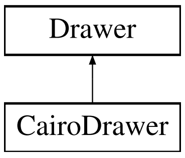
- include/DataSet.h

5.13 Drawer Class Reference

Abstract Drawer for MooViE scenes.

`#include <Drawer.h>`

Inheritance diagram for Drawer:



Classes

- struct TextAlignment

  Text alignment representation.

Public Member Functions

- Drawer (int width, int height, std::size_t num_inputs)
- virtual void change_surface (const std::string &fpath, int width, int height, std::size_t num_inputs)=0
- virtual void draw_output_grid (const OutputGrid &grid)=0
- virtual void draw_input_axis (const InputAxis &axis)=0
- virtual void draw_io_vector (const IOVector &elem)=0
- virtual void finish ()=0
- void **set_num_inputs** (std::size_t num_inputs)

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
- virtual void draw_histogram (const InputAxis::Histogram &histogram, double radius, const Angle &start, const Angle &end)=0
- virtual void draw_link (const Polar &origin1, const Polar &origin2, const Polar &target1, const Polar &target2, const DrawerProperties<> &prop)=0
- virtual void draw_connector (const Polar &from, const Polar &to, const DrawerProperties<> &prop)=0
- 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
- virtual void draw_output_label (const Label &output_label, double radius_label, double radius_output, const Angle &begin, const Angle &end)=0
- virtual void draw_arrow (const Polar &start, const DrawerProperties<> &prop)=0
- virtual void draw_ring_segment (double radius, double thickness, const Angle &start, const Angle &end, const DrawerProperties<> &prop, Direction dir)=0
- virtual void draw_connector_segment (double start_radius, double start_angle, double end_radius, double end_angle, const DrawerProperties<> &prop)=0
- virtual void draw_line (const Polar &from, const Polar &to, const DrawerProperties<> &prop)=0
- virtual void draw_arc (double inner_radius, const Angle &start, const Angle &end, Direction dir)=0
- virtual void draw_coord_point (const Polar &coord, const Angle &width, double height, const Drawer↩ Properties<> &prop)=0
- virtual void draw_text_parallel (const Label &label, const Polar &start, const TextAlignment &alignment=Text↩ Alignment::CENTERED)=0
- virtual void draw_text_orthogonal (const Label &label, const Polar &start, const TextAlignment &alignment=TextAlignment::CENTERED)=0
- 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

Protected Attributes

- CoordinateConverter m_coord_converter
- std::size_t m_num_inputs

5.13.1 Detailed Description

Abstract Drawer for MooViE scenes.

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

5.13.2 Constructor & Destructor Documentation

5.13.2.1 Drawer()

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

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

Parameters

width	the surface width
height	the surface height
num_inputs	the number of inputs

5.13.3 Member Function Documentation

5.13.3.1 change_surface()

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

Alters the surface of this Drawer in number of inputs, width, 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
num_inputs	the number of inputs

Implemented in CairoDrawer.

5.13.3.2 create_link_control_point()

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

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

5.13.3.3 draw_arc()

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

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.

5.13.3.4 draw_arrow()

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

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.

5.13.3.5 draw_connector()

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

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.

5.13.3.6 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 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.

5.13.3.7 draw_coord_point()

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

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.

5.13.3.8 draw_histogram()

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

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.

5.13.3.9 draw_input_axis()

```
virtual void Drawer::draw_input_axis (
            const InputAxis & axis )  [pure virtual]
```

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

Parameters

axis	the InputAxis to draw

Implemented in CairoDrawer.

5.13.3.10 draw_io_vector()

```
virtual void Drawer::draw_io_vector (
             const IOVector & elem )  [pure virtual]
```

Draws a IOVector using its coordinates.

Parameters

elem	the IOVector to draw

Implemented in CairoDrawer.

5.13.3.11 draw_line()

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

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.

5.13.3.12 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 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.

5.13.3.13 draw_output_grid()

```
virtual void Drawer::draw_output_grid (
             const OutputGrid & grid )  [pure virtual]
```

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

Parameters

grid	the OutputGrid to draw

Implemented in CairoDrawer.

5.13.3.14 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 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.

5.13.3.15 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 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.

5.13.3.16 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 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.

5.13.3.17 draw_text_orthogonal()

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

Draws 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.

5.13.3.18 draw_text_parallel()

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

Draws 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.

5.13.3.19 finish()

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

Save the Drawer's result to the given file.

Implemented in CairoDrawer.

5.13.3.20 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

5.13.3.21 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

5.13.3.22 set_surface()

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

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.

5.13.4 Member Data Documentation

5.13.4.1 m_coord_converter

```
CoordinateConverter Drawer::m_coord_converter  [protected]
```

Polar-Cartesian converting

5.13.4.2 m_num_inputs

```
std::size_t Drawer::m_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

5.14 DrawerProperties< FillT > Struct Template Reference

Properties to modify a MooViE drawers behavior.

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

Public Member Functions

- DrawerProperties (double line_width_, const Color &line_color_, const FillT &fill_color_)

Public Attributes

- double line_width
- Color line_color
- FillT fill_color

5.14.1 Detailed Description

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

Properties to modify a MooViE drawers behavior.

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

Author

beyss

Date

05.07.2017

5.14.2 Constructor & Destructor Documentation

5.14.2.1 DrawerProperties()

```
template<typename FillT = Color>
DrawerProperties< FillT >::DrawerProperties (
            double line_width_,
            const Color & line_color_,
            const FillT & fill_color_ )  [inline]
```

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

5.14.3 Member Data Documentation

5.14.3.1 fill_color

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

Fill color(s)

5.14.3.2 line_color

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

Line color

5.14.3.3 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

5.15 InputAxis::Histogram Class Reference

Public Member Functions

- Histogram (DefVariable var)
- void calculate (const std::vector< double > &data)
- double get_section_frequency (std::size_t i) const
- std::size_t get_num_intervals (void) const
- void set_num_intervals (std::size_t num_intervals)

5.15.1 Constructor & Destructor Documentation

5.15.1.1 Histogram()

```
InputAxis::Histogram::Histogram (
            DefVariable var )
```

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

Parameters

var	the variable to present

5.15.2 Member Function Documentation

5.15.2.1 calculate()

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

Calculates equidistant data sections and stores them.

Parameters

data	the input values of this variable

5.15.2.2 get_num_intervals()

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

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

Returns

the interval count

5.15.2.3 get_section_frequency()

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

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

5.15.2.4 set_num_intervals()

```
void InputAxis::Histogram::set_num_intervals (
            std::size_t num_intervals )  [inline]
```

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/InputAxis.h

5.16 InputAxis Class Reference

InputAxis MooViE component representation.

```
#include <InputAxis.h>
```

Classes

- class Histogram

Public Member Functions

- InputAxis (DefVariable variable, const Angle &start, const Angle &end, double radius, double height, const DrawerProperties<> &prop)

  constructor
- const DefVariable & get_var () const
- const Histogram & get_histogram () const
- const Angle & get_start () const
- void set_start (const Angle &start)
- const Angle & get_end () const
- void set_end (const Angle &end)
- double get_radius () const
- void set_radius (double radius)
- double get_height () const
- void set_height (double height)
- const DrawerProperties & get_prop () const
- void set_prop (const DrawerProperties<> &prop)
- const SimpleScale & get_scale () const
- Label make_label (const TextProperties &prop) const
- void calculate_histogram (const std::vector< double > &data)

5.16.1 Detailed Description

InputAxis MooViE component representation.

A InputAxis 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

5.16.2 Constructor & Destructor Documentation

5.16.2.1 InputAxis()

```
InputAxis::InputAxis (
            DefVariable variable,
            const Angle & start,
            const Angle & end,
            double radius,
            double height,
            const DrawerProperties<> & prop )
```

constructor

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

Parameters

variable	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

5.16.3 Member Function Documentation

5.16.3.1 calculate_histogram()

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

Calculates the frequencies of the Histogram.

Parameters

data	the data used

5.16.3.2 get_end()

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

Returns the end Angle of this InputAxis' drawing span.

Returns

the end Angle

5.16.3.3 get_height()

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

Returns the height measured from the radius.

Returns

the height

5.16.3.4 get_histogram()

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

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

Returns

the Histogram

5.16.3.5 get_prop()

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

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

Returns

the DrawerProperties

5.16.3.6 get_radius()

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

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

Returns

the radius

5.16.3.7 get_scale()

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

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

Returns

the SimpleScale

5.16.3.8 get_start()

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

Returns the start Angle of this InputAxis' drawing span.

Returns

the start Angle

5.16.3.9 get_var()

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

Returns a const reference to the variable this InputAxis presents.

Returns

the Var

5.16.3.10 make_label()

```
Label InputAxis::make_label (
             const TextProperties & prop ) const  [inline]
```

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

Parameters

prop	

5.16.3.11 set_end()

```
void InputAxis::set_end (
             const Angle & end )  [inline]
```

Sets the end Angle of this InputAxis' drawing span.

Parameters

end	the end Angle to set

5.16.3.12 set_height()

```
void InputAxis::set_height (
             double height )  [inline]
```

Sets the height measured from the radius.

Parameters

height	the height to set

5.16.3.13 set_prop()

```
void InputAxis::set_prop (
             const DrawerProperties<> & prop )  [inline]
```

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

Parameters

prop	the DrawerProperties to set

5.16.3.14 set_radius()

```
void InputAxis::set_radius (
             double radius )  [inline]
```

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

Parameters

radius	the radius to set

5.16.3.15 set_start()

```
void InputAxis::set_start (
             const Angle & start )  [inline]
```

Starts the start Angle of this InputAxis' drawing span.

Parameters

start	the start Angle to set

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

- include/InputAxis.h

5.17 IOVector Class Reference

IOVector MooViE component representation.

`#include <IOVector.h>`

Public Member Functions

- const Point & operator[] (std::size_t i) const
- std::size_t size (void) const
- template<typename ... Arg>
  void emplace_back (Arg &&... args)

5.17.1 Detailed Description

IOVector MooViE component representation.

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 IOVector.

Author

stratmann

Date

07.03.2018

5.17.2 Member Function Documentation

5.17.2.1 emplace_back()

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

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

Parameters

args	the arguments (Polar, DrawerProperties)

5.17.2.2 operator[]()

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

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

Parameters

the	index of the Point

Returns

the Point

5.17.2.3 size()

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

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

Returns

the size

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

- include/IOVector.h

5.18 IOVectorFactory Class Reference

```
#include <IOVector.h>
```

Public Member Functions

- IOVectorFactory (std::size_t num_data_rows, const OutputGrid &grid, const std::vector< InputAxis > &axis)
- IOVector create (const DefDataRow &row) const

5.18.1 Detailed Description

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

Author

stratmann

Date

07.03.2018

5.18.2 Constructor & Destructor Documentation

5.18.2.1 IOVectorFactory()

```
IOVectorFactory::IOVectorFactory (
              std::size_t num_data_rows,
              const OutputGrid & grid,
              const std::vector< InputAxis > & axis )
```

Creates a new IOVector factory which needs the number of rows in the data set and the OutputGrid and the Input←Axis' with wich the IOVector will be drawn.

Parameters

num_data_rows	the number of rows of the data set
grid	the OutputGrid
axis	the InputAxis'

5.18.3 Member Function Documentation

5.18.3.1 create()

```
IOVector IOVectorFactory::create (
            const DefDataRow & row ) const
```

Creates a new IOVector from a given DefDataRow. If an input value is too close to zero (as defined by moovie.↩ epsilon_places), an invalid coordinate is added that needs to be ignored by the Drawer.

Parameters

row	the DefDataRow

Returns

the so created IOVector

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

- include/IOVector.h

5.19 Label Class Reference

Text label MooViE component representation.

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

Public Member Functions

- Label (const std::string &text, const TextProperties &prop)
- const std::string & get_text () const
- const TextProperties & get_properties () const

5.19.1 Detailed Description

Text label MooViE component representation.

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

Author

stratmann

Date

27.04.2018

5.19.2 Constructor & Destructor Documentation

5.19.2.1 Label()

```
Label::Label (
            const std::string & text,
            const TextProperties & prop )  [inline]
```

Creates a Label from given text and TextProperties.

Parameters

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

5.19.3 Member Function Documentation

5.19.3.1 get_properties()

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

Returns a const reference to this Labels TextProperties.

Returns

a reference to the TextProperties

5.19.3.2 get_text()

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

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

5.20 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)
- double map (const double &out_val) const
- double inverse (const double &in_val) const

5.20.1 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

5.20.2 Constructor & Destructor Documentation

5.20.2.1 Mapper()

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

Creates a Mapper from two given intervals.

Parameters

in	the first interval
out	the second interval

5.20.3 Member Function Documentation

5.20.3.1 inverse()

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

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

5.20.3.2 map()

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

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

5.21 DataSet< T >::MockColumn Class Reference

Technical column for internal use.

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

Public Member Functions

- MockColumn (DataColumn ∗column)
- const Cell & operator[ ] (std::size_t i) const
- void **set_range** (double l_restr, double u_restr)
- void set_enabled (bool enabled)
- ColumnType get_type () const
- Variable get_var () const
- std::size_t size () const
- bool is_enabled () const

Static Public Member Functions

- static void swap (MockColumn &m0, MockColumn &m1)

5.21.1 Detailed Description

template<typename T>
class DataSet< T >::MockColumn

Technical column for internal use.

A mock column that is supposed to hold a pointer to the column storage. The DataColumns can be swapped between the MockColumn. MockColumns can be enabled and disabled which alters the number of cells in the DataRows accordingly.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 MockColumn()

```
template<typename T>
DataSet< T >::MockColumn::MockColumn (
            DataColumn * column )  [inline]
```

Creates a MockColumn from a DataColumn. This MockColumn wraps and it and provides read-only access to all its components.

Parameters

_column	the DataColumn

5.21.3 Member Function Documentation

5.21.3.1 get_type()

```
template<typename T>
ColumnType DataSet< T >::MockColumn::get_type ( ) const  [inline]
```

Returns the ColumnType of this MockColumn. It is either INPUT or OUTPUT.

Returns

the ColumnType

5.21.3.2 get_var()

```
template<typename T>
Variable DataSet< T >::MockColumn::get_var ( ) const  [inline]
```

Returns the Variable of this MockColumn.

Returns

the Variable

5.21.3.3 is_enabled()

```
template<typename T>
bool DataSet< T >::MockColumn::is_enabled ( ) const  [inline]
```

Returns the value of the enabled flag.

Returns

enabled or not

5.21.3.4 operator[]()

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

Accesses the i-th Cell in the stored column.

Parameters

i	the row index

Returns

the Cell

5.21.3.5 set_enabled()

```
template<typename T>
void DataSet< T >::MockColumn::set_enabled (
             bool enabled )  [inline]
```

Sets the enabled flag of this MockColumn to the specified value.

Parameters

enabled	set enabled or not

5.21.3.6 size()

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

Returns the size of this MockColumn.

Returns

the size

5.21.3.7 swap()

```
template<typename T>
static void DataSet< T >::MockColumn::swap (
             MockColumn & m0,
             MockColumn & m1 )  [inline], [static]
```

Class function to swap the columns of two MockColumns.

Parameters

m0	the first MockColumn
m1	the second MockColumn

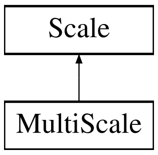
The documentation for this class was generated from the following file:

- include/DataSet.h

5.22 MultiScale Class Reference

```
#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="")
- void add_scale (const std::pair< double, double > &extremes)
- size_t get_scale_number (void) const
- const std::pair< double, double > get_extremes (size_t i) const
- std::vector< Label > make_labels (size_t i) const

Additional Inherited Members

5.22.1 Detailed Description

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

5.22.2 Constructor & Destructor Documentation

5.22.2.1 MultiScale()

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

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

5.22.3 Member Function Documentation

5.22.3.1 add_scale()

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

Adds extreme value of another scalable entry to this MultiScale.

Parameters

extremes	the extreme values

5.22.3.2 get_extremes()

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

Returns the extreme values of the i-th entry.

Returns

the extremes

5.22.3.3 get_scale_number()

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

Returns the number of scales of this MultiScale.

Returns

number of scales

5.22.3.4 make_labels()

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

Constructs description labels using the scale with the given index.

Returns

the labels

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

- include/Scale.h

5.23 OutputGrid Class Reference

OutputGrid MooViE component representation.

```
#include <OutputGrid.h>
```

Public Member Functions

- OutputGrid (const std::vector< DefVariable > &output_vars, const Angle &start, const Angle &end, double radius, double height, Direction dir)
- const DefVariable & get_var (std::size_t num_output) const
- std::size_t get_num_outputs () const
- const Angle & get_start () const
- void set_start (const Angle &start)
- const Angle & get_end () const
- void set_end (const Angle &end)
- double get_radius () const
- void set_radius (double radius)

  sets the radius
- double get_height () const
- void set_height (double height)
- Direction get_direction () const
- void set_direction (Direction direction)
- const MultiScale & get_scale () const

5.23.1 Detailed Description

OutputGrid MooViE component representation.

Representing a coordinate grid by its dimensional constraints.

Author

beyss

Date

26.07.2017

5.23.2 Constructor & Destructor Documentation

5.23.2.1 OutputGrid()

```
OutputGrid::OutputGrid (
            const std::vector< DefVariable > & output_vars,
            const Angle & start,
            const Angle & end,
            double radius,
            double height,
            Direction dir )
```

Creates a OutputGrid 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

5.23.3 Member Function Documentation

5.23.3.1 get_direction()

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

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

Returns

the Direction

5.23.3.2 get_end()

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

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

Returns

the end Angle

5.23.3.3 get_height()

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

Returns the height measured from the radius.

Returns

the height

5.23.3.4 get_num_outputs()

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

Returns the total number of stored output variables.

Returns

the number of outputs

5.23.3.5 get_radius()

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

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

Returns

the radius

5.23.3.6 get_scale()

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

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

Returns

the MultiScale

5.23.3.7 get_start()

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

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

Returns

the start Angle

5.23.3.8 get_var()

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

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

5.23.3.9 set_direction()

```
void OutputGrid::set_direction (
             Direction direction )  [inline]
```

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

Parameters

direction	the Direction to set

5.23.3.10 set_end()

```
void OutputGrid::set_end (
             const Angle & end )  [inline]
```

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

Parameters

end	the end Angle to set

5.23.3.11 set_height()

```
void OutputGrid::set_height (
             double height )  [inline]
```

Sets the height measured from the radius.

Parameters

height	the height to set

5.23.3.12 set_radius()

```
void OutputGrid::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

5.23.3.13 set_start()

```
void OutputGrid::set_start (
             const Angle & start )  [inline]
```

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

Parameters

start	the start Angle to set

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

- include/OutputGrid.h

5.24 Point Struct Reference

Styled polar coordinate.

```
#include <IOVector.h>
```

Public Member Functions

- Point (Polar &&coord_, const DrawerProperties<> &prop_)

    constructor

Public Attributes

- const Polar coord
- const DrawerProperties prop

5.24.1 Detailed Description

Styled polar coordinate.

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

5.24.2 Constructor & Destructor Documentation

5.24.2.1 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

5.24.3 Member Data Documentation

5.24.3.1 coord

```
const Polar Point::coord
```

The coordinate

5.24.3.2 prop

```
const DrawerProperties Point::prop
```

The property with which to draw

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

- include/IOVector.h

5.25 Polar Class Reference

Polar coordinate representation.

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

Public Member Functions

- Polar (double radius=0, Angle angle=0)
- bool operator== (const Polar &rhs) const
- const double & radius () const
- double & radius ()
- const Angle & angle () const
- Angle & angle ()

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

5.25.1 Detailed Description

Polar coordinate representation.

Polar represents a tupel from C in polar coordinate form.

Authors

beyss, stratmann

Date

03.07.2018

5.25.2 Constructor & Destructor Documentation

5.25.2.1 Polar()

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

Creates a Polar coordinate from a given radius and angle.

Parameters

r	the radius
phi	the angle

5.25.3 Member Function Documentation

5.25.3.1 angle() [1/2]

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

Access function for this Polar's m_angle readonly.

Returns

a constant reference to the Angle

5.25.3.2 angle() [2/2]

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

Access function for this Polar's m_angle.

Returns

a reference to the Angle

5.25.3.3 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

5.25.3.4 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

5.25.3.5 operator==()

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

Equal to operator checking for equality of radius and angle.

Parameters

rhs	the other Polar

Returns

if equal or not

5.25.3.6 radius() [1/2]

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

Access function for this Polar's radius as readonly.

Returns

a constant reference to this Polar's radius

5.25.3.7 radius() [2/2]

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

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:

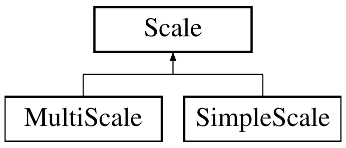
- include/Coordinates.h

5.26 Scale Class Reference

Ticked 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="")
- size_t get_major_intersections (void) const
- size_t get_minor_intersections (void) const

Protected Attributes

- size_t **m_major_intersections**
- size_t **m_minor_intersections**
- TextProperties **m_label_prop**
- std::string **m_label_suffix**

5.26.1 Detailed Description

Ticked scale.

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

Author

beyss

Date

22.08.2017

5.26.2 Constructor & Destructor Documentation

5.26.2.1 Scale()

```
Scale::Scale (
            size_t major_intersections,
            size_t minor_intersections,
            const TextProperties & label_prop,
            const std::string & label_suffix = "" )  [inline]
```

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

5.26.3 Member Function Documentation

5.26.3.1 get_major_intersections()

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

Returns the number of major intersection lines of this scale.

Returns

number of major intersections

5.26.3.2 get_minor_intersections()

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

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

5.27 Scene Class Reference

MooViE scene.

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

Public Member Functions

- Scene ()
- void update (void)
- std::vector< DefVariable > get_input_variables () const
- std::vector< DefVariable > get_output_variables () const
- void toggle_input (std::size_t index, bool mode)
- void toggle_output (std::size_t index, bool mode)
- void swap_inputs (std::size_t from_index, std::size_t to_index)
- void swap_outputs (std::size_t from_index, std::size_t to_index)
- void restrict_input (std::size_t index, double lower_restr, double upper_restr)
- void restrict_output (std::size_t index, double lower_restr, double upper_restr)

5.27.1 Detailed Description

MooViE scene.

Scene class represents a MooViE scene. It is initially drawn and every alteration will require an update to be persistent.

Author

beyss

Date

28.08.2017

5.27.2 Constructor & Destructor Documentation

5.27.2.1 Scene()

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

Creates a new MooViE Scene and draws it immediately. It is required to first initialize a Configuration.

5.27.3 Member Function Documentation

5.27.3.1 get_input_variables()

```
std::vector<DefVariable> Scene::get_input_variables ( ) const
```

Returns the current input variables of this MooViE scene. Needs to be called again if Scene was altered.

Returns

the input variables

5.27.3.2 get_output_variables()

```
std::vector<DefVariable> Scene::get_output_variables ( ) const
```

Returns the current output variables of this MooViE scene. Needs to be called again if Scene was altered.

Returns

the output variables

5.27.3.3 restrict_input()

```
void Scene::restrict_input (
            std::size_t index,
            double lower_restr,
            double upper_restr )
```

Restricts the input with given index to a given interval. Every row whose associated input value is not in the interval will be disabled. The scene needs to be updated afterwards.

Parameters

index	the input index
lower_restr	the lower bound
upper_restr	the upper bound

5.27.3.4 restrict_output()

```
void Scene::restrict_output (
            std::size_t index,
            double lower_restr,
            double upper_restr )
```

Restricts the output with given index to a given interval. Every row whose associated output value is not in the interval will be disabled. The scene needs to be updated afterwards.

Parameters

index	the output index
lower_restr	the lower bound
upper_restr	the upper bound

5.27.3.5 swap_inputs()

```
void Scene::swap_inputs (
            std::size_t from_index,
            std::size_t to_index )
```

Rearranges the order of inputs in this MooViE scene by swapping the inputs with the given indices. The scene needs to be updated afterwards.

Parameters

from_index	the first inputs index
to_index	the second inputs index

Exceptions

out_of_bounds	if indices are incorrect

5.27.3.6 swap_outputs()

```
void Scene::swap_outputs (
            std::size_t from_index,
            std::size_t to_index )
```

Rearranges the order of outputs in this MooViE scene by swapping the outputs with the given indices. The scene needs to be updated afterwards.

Parameters

from_index	the first outputs index
to_index	the second outputs index

Exceptions

out_of_bounds	if indices are incorrect

5.27.3.7 toggle_input()

```
void Scene::toggle_input (
            std::size_t index,
            bool mode )
```

Enables/disables the input with the given index. The scene needs to be updated afterwards.

Parameters

index	the input index
mode	set enabled or disabled

Exceptions

out_of_bounds	if index is incorrect

5.27.3.8 toggle_output()

```
void Scene::toggle_output (
             std::size_t index,
             bool mode )
```

Enables/disables the output with the given index. The scene needs to be updated afterwards.

Parameters

index	the output index
mode	set enabled or disabled

Exceptions

out_of_bounds	if index is incorrect

5.27.3.9 update()

```
void Scene::update (
             void  )
```

Reinitializes all components and redraws the MooViE scene.

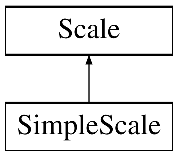
The documentation for this class was generated from the following file:

- include/Scene.h

5.28 SimpleScale Class Reference

`#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="")
- const std::pair< double, double > & get_extremes () const
- std::vector< Label > make_labels (void) const

Additional Inherited Members

5.28.1 Detailed Description

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

5.28.2 Constructor & Destructor Documentation

5.28.2.1 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]
```

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

5.28.3 Member Function Documentation

5.28.3.1 get_extremes()

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

Access function for the Ticks extreme values.

Returns

a reference to the extreme values

5.28.3.2 make_labels()

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

Constructs description labels from the

Returns

the labels

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

- include/Scale.h

5.29 Drawer::TextAlignment Struct Reference

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**

5.29.1 Detailed Description

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

5.30 TextProperties Struct Reference

Properties to modify a MooViE Drawers text style.

```
#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)

Public Attributes

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

5.30.1 Detailed Description

Properties to modify a MooViE Drawers text style.

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

Authors

beyss, stratmann

Date

05.07.2017

5.30.2 Constructor & Destructor Documentation

5.30.2.1 TextProperties()

```
TextProperties::TextProperties (
            const std::string & font_name_,
            double font_size_,
            const Color & color_ = Color::BLACK,
            bool bold_ = false,
            bool italic_ = false )  [inline]
```

Creates a TextProperties instance with the given style information.

Parameters

*font_↩ name_*	
font_size↩ _	
*color_*	
*bold_*	
*italic_*	

5.30.3 Member Data Documentation

5.30.3.1 bold

```
bool TextProperties::bold
```

The boldness of the text

5.30.3.2 color

```
Color TextProperties::color
```

The text color

5.30.3.3 font_name

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

The font name

5.30.3.4 font_size

```
double TextProperties::font_size
```

The font size

5.30.3.5 italic

```
bool TextProperties::italic
```

The skewness of the text

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

- include/TextProperties.h

5.31 Triangle< T, dim > Class Template Reference

Triangular set storage.

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

Public Member Functions

- Triangle ()
- Triangle (const std::vector< T > data)
- const T & at (size_t i, size_t j) const
- T & at (size_t i, size_t j)

5.31.1 Detailed Description

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

Triangular set storage.

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

5.31.2 Constructor & Destructor Documentation

5.31.2.1 Triangle() [1/2]

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

Creates a Triangle with an empty storage.

5.31.2.2 Triangle() [2/2]

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

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

Parameters

data	the data vector

5.31.3 Member Function Documentation

5.31.3.1 at() [1/2]

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

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

5.31.3.2 at() [2/2]

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

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

5.32 DataSet< T >::Variable Struct Reference

Header description.

`#include <DataSet.h>`

Public Member Functions

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

Public Attributes

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

5.32.1 Detailed Description

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

Header description.

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

5.32.2 Constructor & Destructor Documentation

5.32.2.1 Variable()

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

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

Parameters

min	the min value
max	the max value
name	the name

5.32.3 Member Data Documentation

5.32.3.1 max

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

Maximal value

5.32.3.2 min

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

Minimal value

5.32.3.3 name

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

Variable name

5.32.3.4 unit

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

Unit of the Variables values

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

- include/DataSet.h